Assignment 3

ICE191 Software Architecture / Cloud Computing

1.Create a CloudFront distribution for your website and explain each step with great technical detail.

50 points.

Para crear una distribución de Cloudfront en base a lo que tenía hecho en tareas pasadas lo que primero se tiene que hacer es averiguar el comando. Yo me basé en la guía del sitio web LearnAWS (2023), por lo que mis pasos podrían verse un poco distintos que los de mis compañeros. Pero, en fin, este hubiera sido el json de configuración de la distribución en el que me hubiera basado para correr el primer comando:

```
{
    "CallerReference": "cf-cli-distribution",
    "Comment": "LearnAWS Cloudfront Distribution",
    "Origins": {
        "Quantity": 1,
        "Items": [{
            "Id": "learnaws-test-versioning-s3",
            "DomainName": "learnaws-test-versioning-s3.s3.amazonaws.com",
            "S3OriginConfig": {
                "OriginAccessIdentity": ""
            }
        }]
    },
    "DefaultCacheBehavior": {
        "TargetOriginId": "learnaws-test-versioning-s3",
        "ViewerProtocolPolicy": "redirect-to-https",
        "TrustedSigners": {
            "Quantity": 0,
            "Enabled": false
        },
        "ForwardedValues": {
            "Cookies": {"Forward": "all"},
            "Headers": {"Quantity": 0},
            "QueryString": false,
            "QueryStringCacheKeys": {"Quantity": 0}
        },
        "DefaultTTL": 86400,
        "MinTTL": 3600
    },
    "Enabled": true
}
```

Claro que esto no satisfacía completamente mis necesidades, para los que tienen buen ojo verán algo que no es compatible con lo quiero hacer con mi **sitio web estático de s3**, y esa es la situación, de acuerdo con documentación oficial del CLI de Cloudfront (s.f), en la tercera llave de mi json “origins” va otra llave que se llama “ítems”, y dentro de ahí va otra llave que se puede llamar “S3OriginConfig” o “CustomOriginConfig”, el primero es para cubetas de S3 que no son sitios web estáticos, mientras que el segundo

es para los que sí están configurados como sitios web estáticos (también aplica para servidores web corriendo en instancias de EC2 según la misma documentación).

En fin, el json resultante que mostraré no era que el usé originalmente (ese se perdió con el tiempo y los cambios que tuvimos que hacerle), además, este que puse aquí incluye valores predeterminados que te regresa Cloudfront cuando descargas el json de tu distribución. Aun así, es cercano al original:

```
{
    "CallerReference": "6fc3736f-7c45-4e93-9b04-20ff30f330de",
    "Aliases": {
        "Quantity": 1,
        "Items": [
            "omar-duran-cetys.cetystijuana.com"
        ]
    },
    "DefaultRootObject": "index.html",
    "Origins": {
        "Quantity": 1,
        "Items": [
            {
                "Id": "omar-duran-cetys.cetystijuana.com.s3-website.us-east-
1.amazonaws.com",
                "DomainName": "omar-duran-cetys.cetystijuana.com.s3-website.us-
east-1.amazonaws.com",
                "OriginPath": "",
                "CustomHeaders": {
                    "Quantity": 0
                },
                "CustomOriginConfig": {
                    "HTTPPort": 80,
                    "HTTPSPort": 443,
                    "OriginProtocolPolicy": "http-only",
                    "OriginSslProtocols": {
                        "Quantity": 3,
                        "Items": [
                            "TLSv1",
                            "TLSv1.1",
                            "TLSv1.2"
                        ]
                    },
                    "OriginReadTimeout": 30,
                    "OriginKeepaliveTimeout": 5
                },
```

```
                "ConnectionAttempts": 3,
                "ConnectionTimeout": 10,
                "OriginShield": {
                    "Enabled": false
                },
                "OriginAccessControlId": ""
            }
        ]
    },
    "OriginGroups": {
        "Quantity": 0
    },
    "DefaultCacheBehavior": {
        "TargetOriginId": "omar-duran-cetys.cetystijuana.com.s3-website.us-east-
1.amazonaws.com",
        "TrustedSigners": {
            "Enabled": false,
            "Quantity": 0
        },
        "TrustedKeyGroups": {
            "Enabled": false,
            "Quantity": 0
        },
        "ViewerProtocolPolicy": "allow-all",
        "AllowedMethods": {
            "Quantity": 2,
            "Items": [
                "HEAD",
                "GET"
            ],
            "CachedMethods": {
                "Quantity": 2,
                "Items": [
                    "HEAD",
                    "GET"
                ]
            }
        },
        "SmoothStreaming": false,
        "Compress": false,
        "LambdaFunctionAssociations": {
            "Quantity": 0
        },
        "FunctionAssociations": {
            "Quantity": 0
```

```
        },
        "FieldLevelEncryptionId": "",
        "ForwardedValues": {
            "QueryString": false,
            "Cookies": {
                "Forward": "none"
            },
            "Headers": {
                "Quantity": 0
            },
            "QueryStringCacheKeys": {
                "Quantity": 0
            }
        },
        "MinTTL": 0,
        "DefaultTTL": 86400,
        "MaxTTL": 31536000
    },
    "CacheBehaviors": {
        "Quantity": 0
    },
    "CustomErrorResponses": {
        "Quantity": 0
    },
    "Comment": "",
    "Logging": {
        "Enabled": false,
        "IncludeCookies": false,
        "Bucket": "",
        "Prefix": ""
    },
    "PriceClass": "PriceClass_All",
    "Enabled": false,
    "ViewerCertificate": {
        "CloudFrontDefaultCertificate": false,
        "ACMCertificateArn": "arn:aws:acm:us-east-
1:292274580527:certificate/e961006c-fa16-48ce-aeae-1f093f83db26",
        "Certificate": "arn:aws:acm:us-east-1:292274580527:certificate/e961006c-
fa16-48ce-aeae-1f093f83db26",
        "SSLSupportMethod": "vip",
        "MinimumProtocolVersion": "TLSv1",
        "CertificateSource": "acm"
    },
    "Restrictions": {
        "GeoRestriction": {
```

```
            "RestrictionType": "none",
            "Quantity": 0
        }
    },
    "WebACLId": "",
    "HttpVersion": "http2",
    "IsIPV6Enabled": true,
    "ContinuousDeploymentPolicyId": "",
    "Staging": false
}
```

Explicaré línea por línea lo que está pasando, “CallerReference” es una referencia de quien anda haciendo cambios, cada edición a una distribución ya existente debe tener el mismo valor, en mi caso lo genere con un UUID aleatorio, pero se puede poner texto libremente. “Aliases” contiene una estructura que define a donde se redirigirán los recursos (a donde irá el cache”, dentro de ahí va la cantidad y en nuestro caso el subdominio que queremos que sea distribuido.

“DefaultRootObject” es el objeto principal que se le da al usuario al entrar al sitio. En “Origins” describimos los origines, o sea, que es lo que cachearemos, “Id” y “DomainName” en mi caso es igual porque yo corrí el comando de forma distinta (detalles más adelante) pero algunos compañeros al generarlo automáticamente en “Id” tienen unos números al final, en fin, los valores que van ahí son el URL del sitio web de la cubeta de S3. “OriginPath” va vacío porque no cachearemos un directorio en específico, además este valor es opcional de acuerdo con la documentación (recordar que descargar el json de la distribución regresa múltiples valores predeterminados).

“CustomHeaders” es una estructura que indicará un valor de 0 porque no tenía pensado modificar los headers de las solicitudes HTTP al origen (origen siendo la cubeta aquí).

“CustomOriginConfig” lleva algunos datos relacionados a los protocolos HTTP y otras cosas, lo importante aquí es el puerto HTTP y HTTPS, esos escritos son los usados normalmente, por eso los puse, en “OriginProtocolPolicy” va si se usará http, https o lo que quiera el usuario, en mi caso está HTTP y explicaré porque al final. En SSL Protocols va la cantidad de protocolos SSL que se usarán y cuales, esto es relevante si tu sitio

tiene HTTPS. Los otros valores son *timeouts* respecto a respuestas del origen y tiempo que se mantiene la conexión.

De ahí siguen valores predeterminados que no puse en mi JSON original (se ponen automáticamente por Cloudfront y se muestran si visualizas el json de tu distribución al ya estar creada) porque no eran relevantes para la solución del ejercicio hasta llegar a “DefaultCacheBehavior” que como dice el nombre es una estructura para definir el comportamiento de nuestro cache. “TargetOriginId” es otra vez, el URL de la cubeta, “TrustedSigners” y “TrustedKeyGroups” es relacionado a usuarios de AWS que pueden usar cookies firmadas, fueron declarados como falso y cero. “ViewerControlPolicy”, este de aquí si es muy importante, definirá si el usuario puede usar HTTP, redirigirlo a HTTPS o que haga como venga, la diferencia a “OriginProtocolPolicy” es que este último define la conexión entre cache a origen, mientras que la otra es de usuario a cache.

En “AllowedMethods” es los métodos de solicitudes HTTP que se le permiten al usuario hacer a la distribución, no al sitio de la cubeta, en este caso el cache será de solamente lectura.

De aquí seguimos con valores predeterminados hasta “ForwardedValues”, hay 4 valores, “Cookies”, “Headers”, “QueryString”, y “QueryStringCacheKeys”. Los últimos dos son relacionados al cacheo de strings usados para hacer querys en algunas solicitudes web (ej www.lol.com/queso?color=azul). De acuerdo con documentación oficial de cacheo de estos strings, sirve para cachear también resultados de solicitudes así, pero en mi caso no hay nada porque no lo iba a utilizar. Lo mismo también aplica para cookies, indicarle a Cloudfront que hacer cuando le llegan cookies en el header de Cookie, según documentación por defecto no los lee. Headers, misma historia, cachear contenidos dependiendo de los valores de headers de las solicitudes entrantes.

De aquí siguen algunos valores de *Time to Live,* en orden estos son el tiempo que CloudFront mantiene (a lo mínimo) los contenidos del origen, el siguiente es el valor predeterminado, y el siguiente es el máximo.

De nuevo seguimos con valores predeterminados hasta llegar a “Enabled”, este activa o desactiva la distribución, y sí, si tiene sentido porque para borrarla debes poner ese valor como falso, pero si la vas a usar debe de estar verdadero.

En fin, ahora sigue el dolor de cabeza (y de la billetera del profesor) que fue los certificados, un certificado SSL según Cloudflare (s.f) verifica que eres el dueño de un sitio web, y además encripta los contenidos del tráfico del sitio, casi todos los sitios web de hoy en día usan certificados para convertir el tráfico de sus sitios a HTTPS, originalmente nuestro compañero David nos dio un certificado para todos los subdominios de cetystijuana.com, pero resulta que no puedes hacer eso porque muchísimas personas lo estaban usando, o sea que Cloudfront retornaba un error diciendo que no se podía comunicar al origen al intentar cargar la página con HTTPS, y sumémosle que incluso si funcionaba esto iba a costar cientos de dólares.

En fin los valores son así, “CloudFrontDefaultCertificate” especifica si usarás el de CloudFront predeterminado, originalmente diría que sí porque no iba a usar HTTPS, pero hubo cambio de planes y por eso toda la historia, “ACMCertificateArn” lleva el ARN del certificado, “Certificate” también. “CertificateSource” es la fuente del certificado (ACM == AWS Ceritificate Manager), “MinimunProtocolVersión” es la versión mínima de protocolo de encriptación que usaría, en este caso la más básica de TLS (Transport Layer Security).

Por último, más valores predeterminados. Definitivamente muchos valores, pero con eso podemos pasar a describir comandos

aws cloudfront create-distribution --distribution-config file://distroOmar.json

Este comando indica que quieres crear una distribución, y en “distribution-config” van todas las instrucciones y detalles de la distribución, por esto es por lo que mis URLs de origen no tenían esos números extras al final.

Originalmente esto hubiera sido el final, aquí hubiera dicho que me dio un output de tipo JSON con varios valores (incluyendo una copia de la distribución, su ID, y otros relevantes), uno de ellos siendo el URL de la distribución, lo hubiera abierto en el *browser* y todo hubiera salido bien. El que mostraré a continuación no es el de la corrida original del comando, sino el *output* de una actualización, pero en “DomainName” casi al inicio esta lo que digo, otro valor importante por ahí es “Id” siempre será único e identifica nuestra distribución.


```
{
    "ETag": "E2PTCRNQBK4STW",
    "Distribution": {
```

```
        "Id": "ED2BQVM15V4AF",
        "ARN": "arn:aws:cloudfront::292274580527:distribution/ED2BQVM15V4AF",
        "Status": "InProgress",
        "LastModifiedTime": "2023-02-11T03:13:10.699000+00:00",
        "InProgressInvalidationBatches": 0,
        "DomainName": "d3d2fjc6nit6fm.cloudfront.net",
        "ActiveTrustedSigners": {
            "Enabled": false,
            "Quantity": 0
        },
        "ActiveTrustedKeyGroups": {
            "Enabled": false,
            "Quantity": 0
        },
        "DistributionConfig": {
            "CallerReference": "6fc3736f-7c45-4e93-9b04-20ff30f330de",
            "Aliases": {
                "Quantity": 1,
                "Items": [
                    "omar-duran-cetys.cetystijuana.com"
                ]
            },
            "DefaultRootObject": "",
            "Origins": {
                "Quantity": 1,
                "Items": [
                    {
                        "Id": "omar-duran-cetys.cetystijuana.com.s3-website-us-
east-1.amazonaws.com",
                        "DomainName": "omar-duran-cetys.cetystijuana.com.s3-
website-us-east-1.amazonaws.com",
                        "OriginPath": "",
                        "CustomHeaders": {
                            "Quantity": 0
                        },
                        "CustomOriginConfig": {
                            "HTTPPort": 80,
                            "HTTPSPort": 8443,
                            "OriginProtocolPolicy": "match-viewer",
                            "OriginSslProtocols": {
                                "Quantity": 3,
                                "Items": [
                                    "TLSv1",
                                    "TLSv1.1",
                                    "TLSv1.2"
```

```
                            ]
                        },
                        "OriginReadTimeout": 30,
                        "OriginKeepaliveTimeout": 60
                    },
                    "ConnectionAttempts": 3,
                    "ConnectionTimeout": 10,
                    "OriginShield": {
                        "Enabled": false
                    },
                    "OriginAccessControlId": ""
                }
            ]
        },
        "OriginGroups": {
            "Quantity": 0
        },
        "DefaultCacheBehavior": {
            "TargetOriginId": "omar-duran-cetys.cetystijuana.com.s3-website-
us-east-1.amazonaws.com",
            "TrustedSigners": {
                "Enabled": false,
                "Quantity": 0
            },
            "TrustedKeyGroups": {
                "Enabled": false,
                "Quantity": 0
            },
            "ViewerProtocolPolicy": "allow-all",
            "AllowedMethods": {
                "Quantity": 3,
                "Items": [
                    "HEAD",
                    "GET",
                    "OPTIONS"
                ],
                "CachedMethods": {
                    "Quantity": 2,
                    "Items": [
                        "HEAD",
                        "GET"
                    ]
                }
            },
            "SmoothStreaming": false,
```

```
                "Compress": false,
                "LambdaFunctionAssociations": {
                    "Quantity": 0
                },
                "FunctionAssociations": {
                    "Quantity": 0
                },
                "FieldLevelEncryptionId": "",
                "ForwardedValues": {
                    "QueryString": false,
                    "Cookies": {
                        "Forward": "none"
                    },
                    "Headers": {
                        "Quantity": 0
                    },
                    "QueryStringCacheKeys": {
                        "Quantity": 0
                    }
                },
                "MinTTL": 3600,
                "DefaultTTL": 86400,
                "MaxTTL": 31536000
            },
            "CacheBehaviors": {
                "Quantity": 0
            },
            "CustomErrorResponses": {
                "Quantity": 0
            },
            "Comment": "Este json ayudará a crear una distribucion con
certificado de cloudfront para la cubeta de S3 de Omar Duran",
            "Logging": {
                "Enabled": false,
                "IncludeCookies": false,
                "Bucket": "",
                "Prefix": ""
            },
            "PriceClass": "PriceClass_All",
            "Enabled": true,
            "ViewerCertificate": {
                "CloudFrontDefaultCertificate": false,
                "ACMCertificateArn": "arn:aws:acm:us-east-
1:292274580527:certificate/e961006c-fa16-48ce-aeae-1f093f83db26",
                "SSLSupportMethod": "vip",
```

```
                "MinimumProtocolVersion": "TLSv1",
                "Certificate": "arn:aws:acm:us-east-
1:292274580527:certificate/e961006c-fa16-48ce-aeae-1f093f83db26",
                "CertificateSource": "acm"
            },
            "Restrictions": {
                "GeoRestriction": {
                    "RestrictionType": "none",
                    "Quantity": 0
                }
            },
            "WebACLId": "",
            "HttpVersion": "http2",
            "IsIPV6Enabled": true,
            "ContinuousDeploymentPolicyId": "",
            "Staging": false
        },
        "AliasICPRecordals": [
            {
                "CNAME": "omar-duran-cetys.cetystijuana.com",
                "ICPRecordalStatus": "APPROVED"
            }
        ]
    }
}
```


Pero, en una clase nos tomamos tiempo para mejorarlo a HTTPS debido a problemas de autentificación relacionados entre el origen (cubeta) y el cache. Como comentó mi compañero Marcos en clase “debemos verificar que somos los dueños de esos recursos”.

Para actualizar una distribución necesitamos un ETag, valor que también da al crear una distribución, es como un ID, pero referente a X versión, o sea que si le haces tres actualizaciones, habría 3 ETags, 1 original, 2 subsecuentes, y una de la última. La última es la importante para nosotros y se puede conseguir así:

aws cloudfront get-distribution --id ED2BQVM15V4AF | grep "ETag"

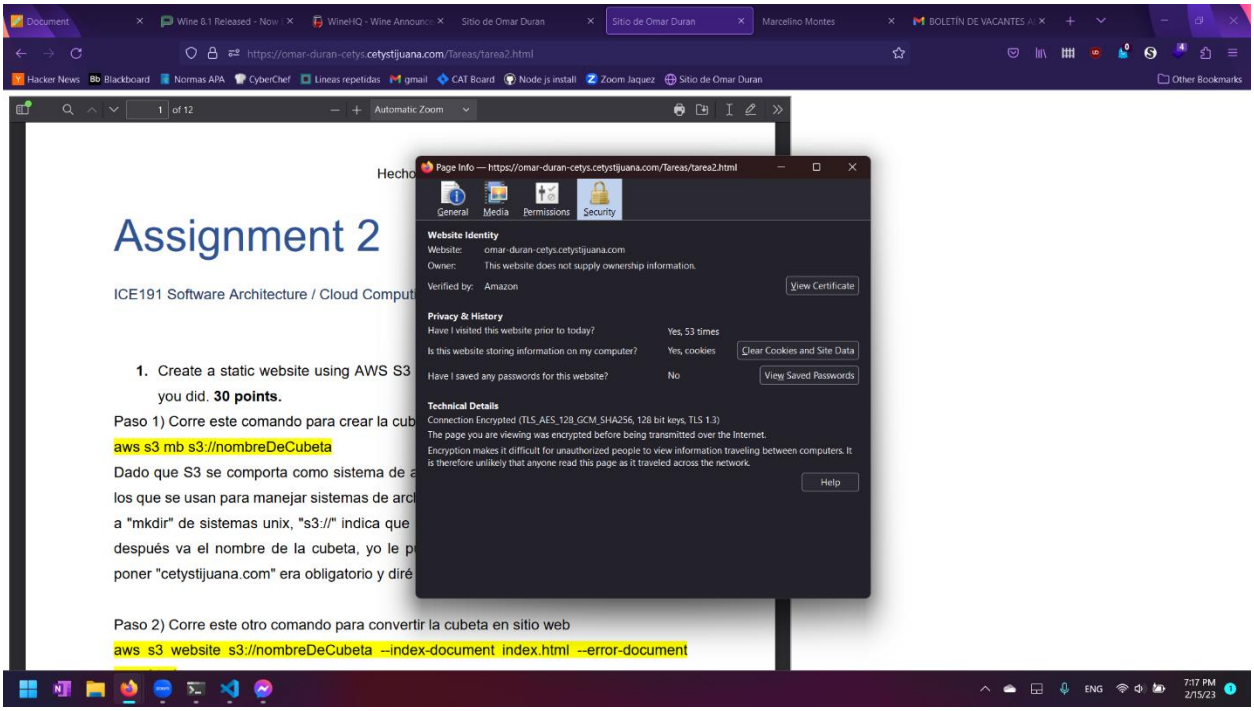
El pipeline de ETag es meramente para solo imprimir el renglón con “ETag”, el comando en sí regresa un JSON con todos los detalles de la distribución, ID, estado, nombre de dominio (URL), ETag, configuración de la distribución, entre muchos otros, los importantes aquí son ETag. Este comando requiere como parámetro la ID de la

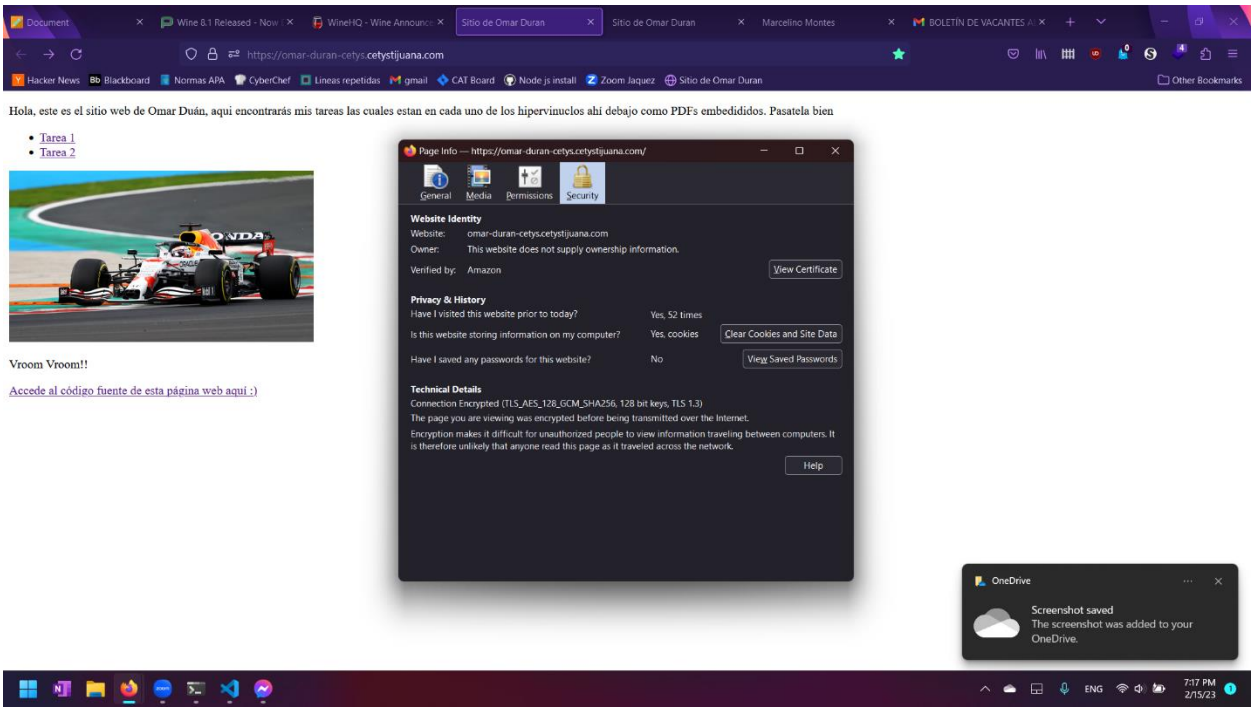
distribución. Nota: El ID se consigue en la corrida del comando create, o viendo todas las distribuciones del usuario, en mi caso lo copie cuando lo cree la distribución

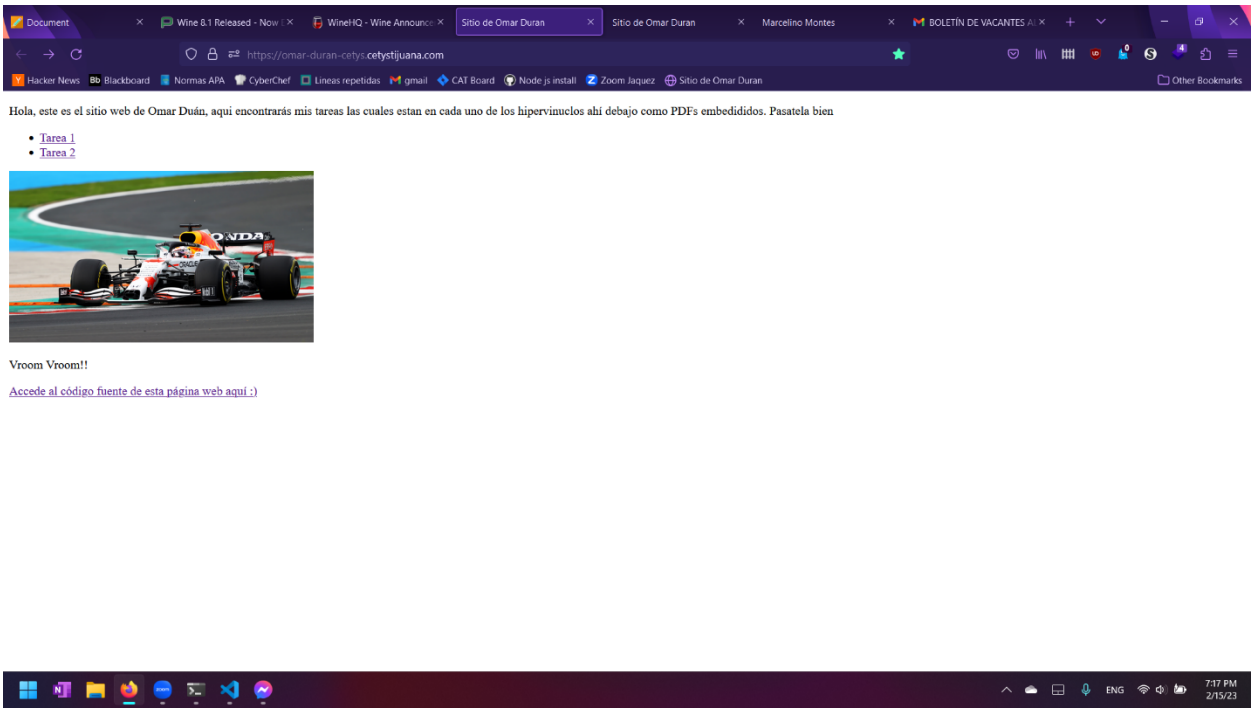
Con esto pudimos actualizar la distribución para tener soporte de HTTPS (originalmente no la tenía, pero repito, el JSON de allá arriba no es el primero que usé):

aws cloudfront update-distribution --id ED2BQVM15V4AF --if-match E27JJF0SJWX57C --distribution-config file://cloudfrontDist.json

El comando tiene sintaxis muy parecida al de “get” y “create” combinados, necesitan archivo de configuración e ID, pero el ETag es para identificar cual versión modificarás, en este caso es la última y se hace con el parámetro “if-match” poniendole el ETag más reciente que se consigue corriendo el comando de get-distribution.







Estas imágenes son evidencia de que sí se logró el sitio (entero) con una distribución de Cloudfront con HTTPS. Misión (dolorosamente) cumplida. Aunque bueno, esas imágenes son de mi sitio web ¿per cuándo cambié el récord de Route53 entonces? Respuestas en la siguiente pregunta

2.Update your DNS Record to route to the CloudFront end point and explain each step with great technical detail.

10 points.

Afortunadamente esta es mucho más corta dado que ya existía un récord para mi sitio web, nada más había que cambiar dos cosas, el TTL para que sea bajo y se refresque rápido, y la dirección de origen, que, en vez de ser la cubeta, ahora era el URL de la distribución de Cloudfront.

aws route53 change-resource-record-sets --hosted-zone-id Z03346142C3RKH191036Y --change-batch file://policyRecord.json

Mismo procedimiento que la tarea pasada, lo que cambia es los contenidos del récord.

```
{
    "Comment": "De vuelta a la cubeta porque los certificados son caros",
    "Changes": [
        {
            "Action": "UPSERT",
            "ResourceRecordSet": {
                "Name": "omar-duran-cetys.cetystijuana.com",
                "Type": "CNAME",
                "TTL": 5,
                "ResourceRecords": [
                    {
                        "Value": " d3d2fjc6nit6fm.cloudfront.net"

                    }
                ]
            }
        }
    ]
}
```

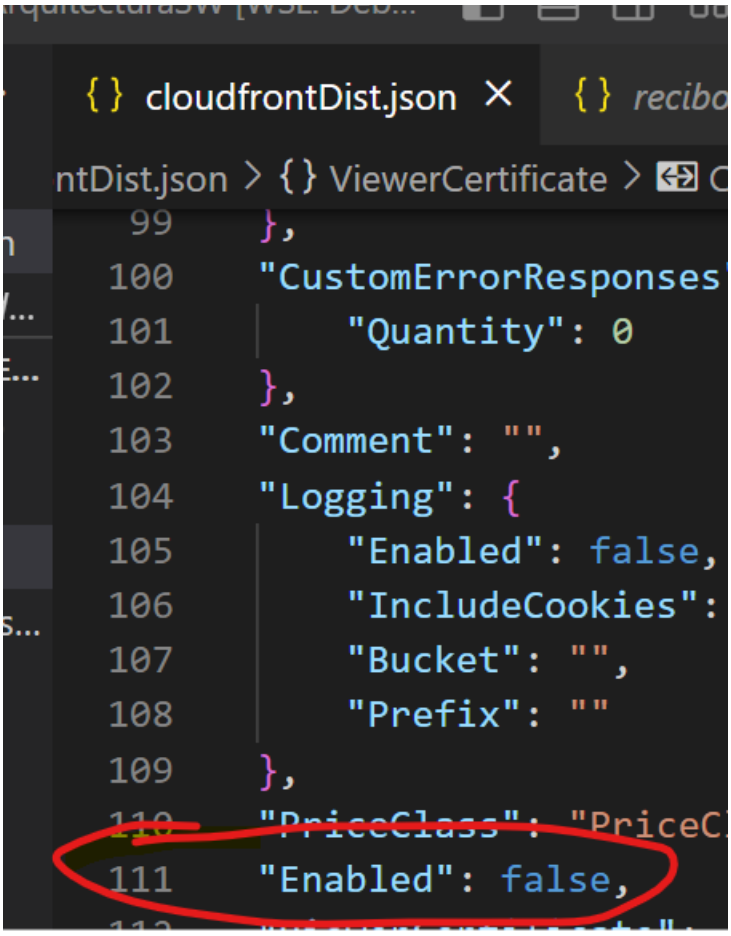
“Action” lleva “UPSERT” que es actualiza si ya existe, y si no existe créalo. Esto se podía hacer a partir desde que se corría el comando de crear la distribución, el URL (el cuál se encuentra en el output corriendo el comando de crear distribución) nunca iba a cambiar siempre y cuando solo se estuviera actualizando y no volviendo a crear.

Como conté en la pregunta previa, debido a los altos costos de certificados, tuvimos que borrar la distribución y cambiar el récord para que apunte a la cubeta. Primero cambié el récord:

aws route53 change-resource-record-set --hosted-zone-id Z03346142C3RKH191036Y --change-batch file://policyRecord.json

Mismo comando, pero ahora en el récord, en “Value” iba de nuevo el URL de mi cubeta. El Zone ID sigue siendo el de la misma tarea porque seguimos trabajando en el dominio de cetystijuana.com

Como siguiente paso fue desactivar la distribución, para eso hay que actualizarla, y la única llave hay que cambiar es “Enabled” con su valor a *false.*



Ahora corremos de nuevo el comando de actualización

aws cloudfront update-distribution --id ED2BQVM15V4AF --if-match {*ETagActual}* --distribution-config file://cloudfrontDist.json

Tomaré un momento para decir que todos estos procesos de actualización, creación, borrado, y otros, son asíncronos (no secuenciales), o sea que pueden tardar segundos, minutos o horas en reflejarse, por eso en los *outputs* de estos comandos tiende a haber una llave de estado indicando “Pending”.

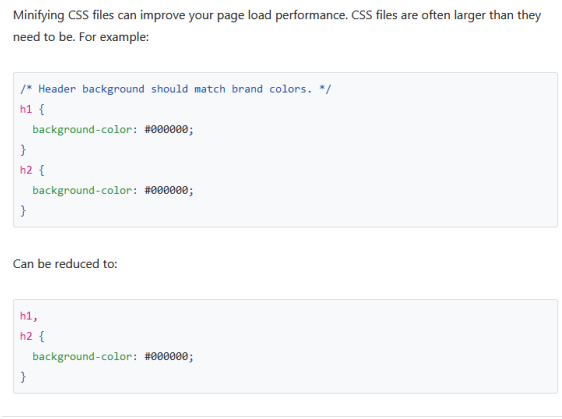
Y para poner el último clavo en el ataúd de este proceso que tomó casi dos semanas, el comando de borrado de distribución:

aws cloudfront delete-distribution --id ED2BQVM15V4AF --if-match E3ILIHONBTO8Y6

Misma sintaxis que el comando de actualización, requiere ID, e ETag de la última versión. En fin, este comando lo que hace es borrar la distribución, puede tardar un rato más que otros comandos porque también debe borrar el cache, pero a lo que nuestro recordó le incumbe, ya no está redirigiendo a este cache, y nunca lo hará jamás hasta que digamos lo contrario en clase.

3.Explain what it means to minify website resources (html, js, css) and the advantages and disadvantages of this process.
10 points.

Por definición formal según la página de desarrolladores de Google (s.f), la “minificación” de recursos web es el acto de modificar recursos de un sitio web para reducir la mayor cantidad posible de datos sin modificar el comportamiento de estos. Eso incluye remover comentarios, variables no utilizadas (solo aplica para JS y clases de CSS), quitar espacios innecesarios, y, simplificar código


Minifying CSS files can improve your page load performance. CSS files are often larger than they need to be. For example:

```
/* Header background should match brand colors. */
h1 {
  background-color: #000000;
}
h2 {
  background-color: #000000;
}
```

Can be reduced to:

```
h1,
h2 {
  background-color: #000000;
}
```

Este es un ejemplo de simplificación (muy simple) de documentación de desarrolladores de Google (s.f) también. Este da el punto que se pueden optimizar ciertos fragmentos de código en el sitio web, como no tener múltiples clases de css haciendo lo mismo.

La razón por la que esto funciona es porque hay menos información que se está transmitiendo al usuario que ve la página web, y en el caso de un sitio web estático donde el cliente hace todo el render, puede ser muy importante si este no tiene una buena conexión a internet, en términos de velocidad.

De acuerdo con documentación oficial de Mozilla (s.f), esto debe ocurrir en tiempo de “*build”*, o sea que estaríamos convirtiendo el sitio web a una versión mínima una vez que hayamos hecho nuestros cambios. No necesariamente se modificaría los archivos originales, sino que habría otros mínimos, los cuales podrían tener resultados difíciles de leer para un humano debido a su meta de simplificación. Esto último es una desventaja porque si la herramienta que utilizamos no funciona a la perfección, nuestro sitio web podría empezar a tener fallos imprevistos e identificarlos sería muy difícil (porque se

remueven comentarios, nombres de variables, y otros identificadores para resolver el problema).

Otros inconvenientes que puede conllevar la minimización de recursos según Cloudflare (s.f) es primero, que no es magia, no hará mágicamente que tu sitio web corra mejor. Puede que si de algunas mejoras de tiempos de carga, pero imágenes grandes, malos algoritmos en términos de rendimiento (en JS), tener tu servidor en un rincón del mundo donde no vive nadie y no tener una CDN, todo esto puede hacer que sea en vano las minificaciones de recursos. Como segundo inconveniente, es que este además de romper y código propio del sitio, también puede romper el uso de variables de ambiente. Si el minificador no tiene contexto de estas puede echar a perder conexiones a bases de datos, usos de plugins, conexiones a APIs, y seguramente otras cosas más

4.Write a python script (and explain each step with great technical detail) to

a.Create or update a record in the Students DynamoDB table based on the id.

b.Delete a record in the Students DynamoDB table based on the id

c.Find a record in the Students DynamoDB table by id.

d.20 points

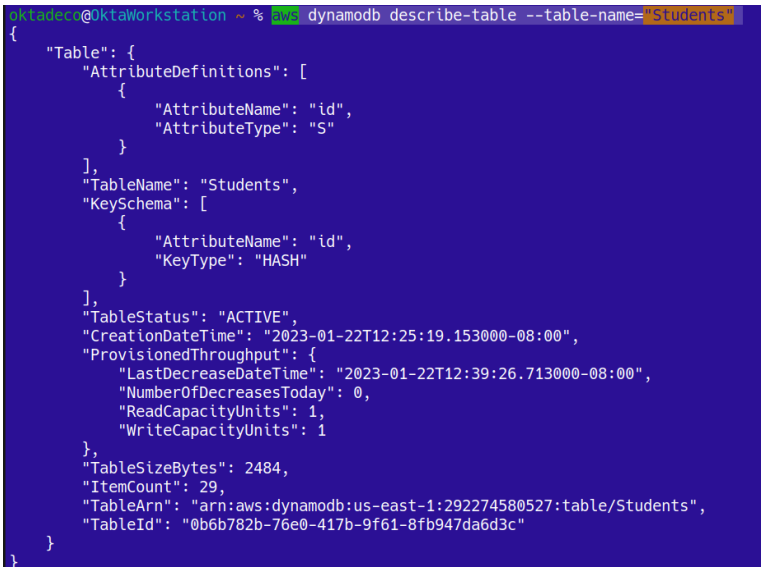
```
#!/usr/bin/env python3
# la primera linea es para declarar que se use python 3 cuando se corre como
ejecutable
#  ej(./archivo.py)

import boto3
import botocore

# Utilizaremos el cliente de DYNAMO DB, este objeto será nuestro medio de
comunicación con dynamo
dynamodb = boto3.client('dynamodb')

# Create or update, delete, o, find. Esas son las opciones
option = input(
   """
   Escriba la opción que desea hacer: \n
   1) Crear/Actualizar Record de estudiante
   2) Borrar record de estudiante
   3) Ver record de estudiante
   """
)
#  Haremos un switch case de 3 opciones (más default) una para
creación/actualización,
# otro para borrado, y otro para lecturas
match option:
   case '1':  # UPSERT
       try:
           # Solicitaremos al usuario matricula, nombre completo y URL del
sitio personal
           matricula = input(
               "Ingrese matricula para el record que actualizará/creará
(escriba ENTER al acabar de escribirla): ")
           fullname = input(
               "Ingrese nombre para el record que actualizará/creará
(escriba ENTER al acabar de escribirla): ")
           personlWebsite = input(
               "Ingrese website para el record que actualizará/creará
(escriba ENTER al acabar de escribirla): ")

           # PUT_ITEM es para meter items en tablas, si ya existe uno lo
sobreescribirá
           # esto lo identifica con la llave primaria (detalles en el
documento)
```

```
            dynamodb.put_item(
                TableName="Students",  # Necesitamos nombre de tabla
                Item={
                    "id": {'S': matricula},  # Llave primaria es obligatoria
                    # Esto y el otro dato son los que ya existian en la tabla
                    "full_name": {'S': fullname},
                    "personal_website": {'S': personlWebsite}
                }
                # En mi tarea 2 explico porque los valores son diccionarios,
pero en resumen deben de indicar
                #  el tipo de valor del valor, en este caso S es STRING, así
esta definida la tabla
            )
            print("proceso de creación/actualización terminado :)")
        except:
            # Excepcion generica para decirle al usuario que si salió mal
            print(
                "Hubo un error inesperado creando/editando el record de la
tabla Students")
    case '2':
        try:
            # Solo necesitamos la llave primaria para borrar
            matricula = input(
                "Ingrese matricula para el record que borrará (escriba ENTER
al acabar de escribirla): ")

            # Para borrar usamos DELETE_ITEM
            response = dynamodb.delete_item(
                TableName="Students",
                Key={
                    "id": {'S': matricula}
                },
                ConditionExpression="attribute_exists(id)"
                # Creamos una condición de expresión, "atrtibute_exists(N)"
                # es para definir si la operación fue un exitó si se cumplió
esa función
                # más detalles en documento
            )
            print("proceso de borrado terminado :)")

        except botocore.exceptions.ClientError as e:
            # En caso de que no se cumpla la condición (no existe ese item)
            # se levantará esta exepción
            if                e.response['Error']['Code']                ==
'ConditionalCheckFailedException':
                print("No existe tal record en la tabla")
        except:
            # Exepción generica si hubo otro tipo de excpeción
            print("Error inesperado borrando record de la tabla Students")

    case '3':
        try:
```

```
            #Solicitar llave primaria (en este caso matricula) para leer el
record en la tabla
            matricula = input(
                "Ingrese matricula porfavor (escriba ENTER al acabar de
escribirla): ")

            #Guardar respuesta en variable porque queremos leer información
            #Esto se hace con GET_ITEM
            response = dynamodb.get_item(
                TableName="Students",
                Key={
                    "id": {'S': matricula}
                }
            )
            #Verificar si existe la llave Item en la respuesta, este contiene
el record en la base de datos
            if ("Item" in response):
                #Imprimirlo tal como está
                print(response["Item"])
                print("proceso de lectura terminado :)")
            else:
                #Levantar excepción si no existe ese record
                print("No hay record para ese alumno")



        except:
            print("Error inesperado intentando leer record de la tabla
Students")

    case _:
        # Si no puso un valor valido entonces
        print("Corre este script denuevo y escoje bien una opcion porfavor")
        pass
```

El script usa dos librerías, boto3 para comunicarse con AWS, y boto3core solamente lo use para el manejo de una excepción.

Este script en sí tiene tres partes, el que actualiza/crea, el que borra, y, el que lee. Para creación/actualización utilicé put_item porque de acuerdo con documentación oficial (s.f) de boto3 este puede crear o actualizar dependiendo de la existencia de esa llave primaria en la tabla ¿pero cómo sé que llave es esa?



aws dynamodb describe-table --table-name="Students"

Este comando me permitió saber los "attributeName" y de acuerdo con documentación oficial, solo atributos de tipo llave primaria se muestran en Dynamo porque no es con esquema (en inglés le dicen "is schemaless").

Para el caso de borrado usé delete_item, hay algo de ciencia aquí, el proceso está bien explicado en los comentarios, pero quiero desarrollar un punto. En cuanto a sus excepciones aquí tuve que ser creativo, por defecto de acuerdo con la documentación, esta operación no dará error si no cumple con lo solicitado, para eso busqué cómo usar el valor de "ConditionExpression" en la llamada a la base de datos. En ese string hay funciones predeterminadas para hacer que mande error o no el uso de la función de borrado, en mi caso quería que hubiera fallo si no existía una entrada con ese ID, para que el usuario tenga esa transparencia.

Para lectura fue el más sencillo, get_item. Da de respuesta un diccionario dentro de su respuesta, y ese diccionario son los valores de ese récord en la tabla, de nuevo, esto lo consigue en base a la llave primaria. Si no retorna ese diccionario, entonces no existen esos datos.

5.Explain the difference between Growth mindset and Fixed mindset according to Carol Dweck
10 points.

La mentalidad de crecimiento según Carol Dweck (2014), es una mentalidad basada en el crecimiento de una persona en base a desafiarse a si mismo. Aquellos poseedores de esta mentalidad afrontan los desafíos de una forma distinta que los fijos. Mientras que los fijos ven el fracaso como una constante en la vida y no afrontan los problemas que les surgen en su vida, los de mentalidad de crecimiento les gustan los retos, aprovechan de los fallos para aprender nuevas cosas, y lo más importante, cuando se afrontan a un problema, su actividad cerebral aumenta.

Es muy importante la forma en la que se le habla a un niño respecto a sus logros y fallos, porque esto puede tener una gran influencia en como es que se va a comportar conforme crece, el celebrar sus logros no debería ser la forma correcta de felicitarlos, sino felicitar su esfuerzo por crecer. No es lo mismo decir, “wow, sacas puros dieces”, a decir “te estas esforzando mucho”).

Productivity Game (2018) da una explicación más a fondo de las ideas de Carol Dweck, la gente con mentalidad fija surge debido a que comete errores, entonces piensan que si los cometen no son inteligentes, porque si lo fueran, entonces no se equivocarían. En otras palabras, la gente de mentalidad fija evita los desafíos para no verse mal.

También explican un ejemplo con niños de 5to grado en un experimento con rompecabezas, los que se les felicitaba por sus habilidades pensaban que eran muy buenos, y reducen sus esfuerzos son el tiempo. Los niños que no redujeron su esfuerzo fue porque los tomaban como desafíos. Con esto se llegó a la definición de una persona de crecimiento, la cual es una que siempre se hace más inteligente y que la inteligencia se gana.

Materiales utilizados:

- https://www.learnaws.org/2023/02/02/aws-cli-cloudfront/#how-to-create-a-distribution
    - Tutorial de como crear una distribución de Cloudfront
- https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/QueryStringParameters.html
    - Documentación de AWS de Cloudfront explicando el uso de Query Strings en distribuciones
- https://awscli.amazonaws.com/v2/documentation/api/latest/reference/cloudfront/create-distribution.html
    - Documentación del CLI de AWS para explicar el comando de creación de distribuciones
- https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/Cookies.html
    - Documentación de AWS de Cloudfront explicando el uso de Cookies en distribuciones
- https://www.cloudflare.com/learning/ssl/what-is-an-ssl-certificate/
    - Explicación de Cloudflare de que es un certificado SSL
- https://developer.chrome.com/en/docs/lighthouse/performance/unminified-css/
    - Documentación de Google explicando que es minificar recursos CSS
- https://developer.mozilla.org/en-US/docs/Glossary/Minification
    - Documentación de Mozilla explicando que es minificar recursos web
- https://www.cloudflare.com/learning/performance/why-minify-javascript-code/
    - Documentación de Cloudflare explicando que es minificar recursos web, pros y contras
- https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/dynamodb.html#DynamoDB.Client.delete_item
    - Documentación de BOTO3 explicando como meter datos en una tabla de DynamoDB

- https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Expressions.ConditionExpressions.html
    - Documentación official de AWS de DynamoDB explicando las expresiones condicionales
- https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/SQLtoNoSQL.GetTableInfo.html
    - Documentación oficial de AWS de DynamoDB explicando como conseguir datos importantes de una tabla
- https://www.youtube.com/watch?v=hiiEeMN7vbQ
    - Presentación de Carol Dweck explicando la mentalidad de crecimiento
- https://www.youtube.com/watch?v=2nF90sAW-Yg
    - Video de Productivity Game explicando las mentalidades fijas y de crecimiento según Carol Dweck