

## Assignment 2

ICE191 Software Architecture / Cloud Computing

### 1. Create a static website using AWS S3 and explain in great technical detail what you did. 30 points.

Para hacer el deploy de un sitio web en Amazon S3, podemos hacer uso de la consola de comandos propia de Amazon, AWS CLI.

Para lograr esto, lo primero que necesitamos hacer es crear algún medio de almacenamiento que sirva como file system para que podamos, subir archivos, los cuales puedan ser usado para mostrar el contenido estático el cual queramos mostrar a nuestros usuarios. Esto lo podemos lograr mediante el servicio s3 de Amazon. El servicio s3 de Amazon, nos permite crear un contenedor o file system denominado “bucket”, este “bucket”, es simplemente un contenedor o unidad de almacenamiento, en el cual podemos subir diferentes archivos a la nube, en este caso Amazon los denomina objetos.

Esto se puede lograr mediante el siguiente comando

```
manmadedarc@Manmade-Laptop:~$ aws s3api create-bucket marcos.cetystijuana.com --profile JaquezAws
```

Este comando lo que hará es crear por defecto un bucket en nuestra cuenta de AWS en la región us-east-1. Por otro lado, si queremos cambiar la región en la que se crea el bucket, podemos hacer uso de los parámetros `--region` para especificar la región, en conjunto con `--create-bucket-configuration LocationConstraint=` Nombre de la región”.

Una vez hecho esto, nuestro bucket estará creado y listo para que podamos subir archivos a él. Sin embargo, antes de eso necesitaremos asegurarnos de que nuestro bucket se haya creado correctamente, esto lo podremos hacer mediante el siguiente comando:

```
manmadedarc@Manmade-Laptop:~$ aws s3api list-buckets --profile JaquezAws
```

El cual nos mostrara la lista de todos los buckets disponibles en nuestra cuenta.

```
CreationDate: 2023-02-04T02:10:23+00:00
},
{
  "Name": "marcos.cetystijuana.com",
  "CreationDate": "2023-02-03T00:56:02+00:00"
},
{
  "Name": "marcos.cetystijuana.com",
  "CreationDate": "2023-02-03T00:56:02+00:00"
},
}
```

Una vez que nos hemos asegurado de crear nuestro bucket, necesitamos poder habilitar el acceso público a nuestros datos, para eso necesitamos cambiar la política de privacidad o acceso del bucket que hemos creado. Esto es debido a que los buckets de Amazon, funcionan como cualquier otro file system, donde un usuario necesita tener permisos para poder acceder a los datos, y verlos. Para lograr esto lo que se hará es el indicarle a nuestro bucket, que queremos habilitar la lectura de estos archivos para cualquiera persona.

Para esto podemos crear un archivo policy.json, que contenga la siguiente estructura:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadGetObject",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::marcos.cetystijuana.com/*"
    }
  ]
}
```

Este archivo json contendrá lo siguiente:

La versión la cual indica el formato de archivo que definiremos para hacer los cambios a la política. En este caso el 2012-10-17.

Después en la llave Statement, podremos las políticas que queremos actualizar como un arreglo, para todos los recursos que queremos modificar sus políticas o permisos. De ahí en adelante los parámetros estarán definidos de la siguiente manera.

“Sid”: Es el nombre de la política y el identificador de dichas políticas.

“Effect”: Es el tipo de política que va a ser aplicada, este tiene 2 valores allow y deny.

“Principal”: Es el usuario al que se le da acceso al recurso, en nuestro caso “\*” hace referencia a cualquier persona/usuario

“Action”: hace referencia al comando o instrucción que va a denegado o aceptado, en función del efecto, en nuestro caso s3:GetObject es la capacidad de tener permisos del lectura del archivo.

“Resource”: Hace referencia a los objetos sobre los cuales se va a aplicar la política. En nuestro caso arn hace referencia a amazon resource name, aws a amazon web services s3 a el servicio s3 marcos.cetystijuana.com al nombre del bucket que hemos creado y además utilizamos /\* para indicar que esta política se va a usar sobre todos los archivos/objetos dentro de este bucket.

Creamos este archivo dentro del directorio en el que estemos ejecutando el aws cli, y lo ejecutamos mediante el siguiente comando.

```
manmadearc@Manmade-Laptop:~$ aws s3api put-bucket-policy --bucket marcos.cetystijuana.com --policy file://policy.json --profile JaquezAws
```

Una vez actualizado esta política, podemos entonces comenzar a subir nuestros archivos a el bucket que hemos creado. Esto lo podemos hacer dirigiéndonos al directorio que contiene nuestros archivos html. Una vez en el directorio solo necesitamos ejecutar el comando sync, el cual nos ayudara a sincronizar los datos, entre los directorios que queramos utilizar, haciendo , donde el primer parámetro es el folder de origen, y el 2 el de destino.

```
manmadearc@Manmade-Laptop:~$ aws s3 sync ./ s3://marcos.cetystijuana.com --profile JaquezAws
```

Donde ./ hace referencia al directorio que queremos subir y s3://marcos.cetystijuana, es el nombre del bucket al que queremos subir nuestros archivos.

Una vez hemos subido los archivos de nuestro sitio web estático, solo no hace falta indicarle a amazon, que los contenidos de nuestro bucket son de hechos archivos de un sitio web. Para esto, usaremos un comando el cual nos ayudara a darle esa estructura a nuestro sitio web.

```
manmadearc@Manmade-Laptop:~$ aws s3 website s3://marcos.cetystijuana.com --index-document index.html --error-document 404.html
```

Este comando se encargara de configurar nuestro bucket como el contenedor de un sitio web, para lograr esto debemos indicarle el documento de index, este documento de index es el utilizado como directorio principal cuando alguien acceda a nuestro recurso o sitio web, por lo tanto debe ser el documento html correspondiente a la página principal de nuestro sitio web estático.

Otro de los documentos a especificar es el documento de error, donde aws se encarga de redirigir a nuestros usuarios cuando la ruta a la que intenten acceder dentro de nuestra página no exista.

En nuestro caso usamos el documento index.html y 404.html ya que son los documentos que hacen nuestra web utiliza para manejar tanto la página principal, como el error mostrado.

Con esto solo quedara acceder al nuestro bucket mediante el enlace de acceso, el cual se compone de la siguiente manera <bucket-name>.s3-website.<AWS-region>.amazonaws.com.

En nuestro caso marcos.cetystijuana.com.s3-website.us-east-1.amazonaws.com/ y podremos ver nuestro sitio web.



## 2. Link your website to a subdomain of cetystijuana.com and explain in great technical detail what you did. 30 points.

Una vez hemos creado nuestro sitio web, solo nos hará falta enlazarlo a una dirección url propia, es decir a un dominio, de esta manera podremos generar una url mas amigable con la cual nuestros usuarios puedan interactuar.

Para poder realizar esto debemos entender que es un DNS. Un DNS es El sistema de nombres de dominio (DNS) es el directorio telefónico de Internet. Es un sistema el cual se encarga de traducir nombres de dominio como lo son Facebook.com o amazon.com a direcciones ip, como lo pueden ser 192.168.20.152 por ejemplos.

Por hacer algún tipo de analogía, supongamos que la internet es una gran biblioteca, y las direcciones ip son los libros a los cuales queremos acceder, un servidor de DNS es un gran índice o base de datos, la cual identifica a un libro en específico, y nos dice donde esta mediante un palabra clave, la cual es el dominio, la cual nos ayuda a determinar de manera más fácil a que libros nos queremos hacer cargo.

Para aprovechar estos, podemos hacer o crear una ruta mediante un servidor de DNS, en este caso podemos hacer uso de Amazon route 53. Amazon route 53 es un servicio de Amazon el cual precisamente nos permite enlazar nuestros recursos de red. Para esto necesitamos tener acceso a un dominio y registrarlo a nuestro nombre.

Podemos buscar la disponibilidad de nuestro dominio, mediante el siguiente comando.

```
aws route53domains check-domain-availability --domain-name example.com
```

Una vez que registramos que nuestro dominio esta disponible podemos registrarlo a nuestro nombre utilizando el siguiente comando

```
aws route53domains register-domain --domain-name example.com --duration-in-years 1 --auto-renew --admin-contact "FirstName LastName, email@example.com, +1.123456789, address" --registrant-contact "FirstName LastName, email@example.com, +1.123456789, address"
```

Una vez hemos registrado nuestro dominio, ya hemos de poder usarlo. Para poder usarlo nosotros hemos de crear una zona de hosteo,

```
aws route53 create-hosted-zone --name example.com --caller-reference reference_string
```

Donde el nombre, será el nombre de dominio que busquemos utilizar y caller reference es un identificador único para nuestra zona de hosteo. Una vez que creemos esto

obtendremos unos nombres de servidor que tendremos que actualizar en nuestro dominio.

Una vez hayamos cambiado esto nos toca crear registros en nuestro dominio para hacer redirección de nuestro dominio a nuestra nuestro bucket. Para hacer esto se harán con registros.

Dentro de los DNS existen diferentes tipos de registros:

1. Registro A: Este registro tiene la dirección ip de un dominio
2. Registro AAAA: Contiene la dirección ipv6 de un dominio.
3. Registro CNAME: reenvía de un dominio o subdominio a otro dominio. (Sin usar ip)
4. Registro MX: redirige a un servidor de correo
5. Registro TXT: Permite a almacenar archivos txt en el registro
6. Registro NS: Almacena nombres de servidor para una entrada de DNS
7. Registro SOA: Almacena información acerca del administrador de un dominio.
8. Registro SRV: Especifica puertos para servicios específicos
9. Registro PTR: Proporciona nombre de dominio en búsquedas por ip.

De estos tipos crearemos un registro de tipo CNAME o nombre canónico, el cual nos dejara enlazar nuestra url de amazon, al dominio que usaremos.

Para esto necesitamos obtener la zona de hosting en la cual se encuentra nuestro DNS, para esto utilizaremos el siguiente comando.

```
manmadearc@Manmade-Laptop:~$ aws route53 list-hosted-zones
{
  "HostedZones": [
    {
      "Id": "/hostedzone/Z03346142C3RKH191036Y",
      "Name": "cetystijuana.com.",
      "CallerReference": "RISWorkflow-RD:fc4e3528-b645-4b45-8c60-ea43b3e4363f",
      "Config": {
        "Comment": "HostedZone created by Route53 Registrar",
        "PrivateZone": false
      },
      "ResourceRecordSetCount": 28
    }
  ]
}
```

Esto nos listara todas nuestras zonas, de esto rescataremos la id de nuestra zona hosteada, en relación con el dominio que queramos utilizar. Para eso crearemos un json con la siguiente estructura:

```
{
  "Changes": [
    {
      "Action": "CREATE",
      "ResourceRecordSet": {
        "Name": "marcos.cetystijuana.com",
        "Type": "CNAME",
        "TTL": 450,
        "ResourceRecords": [
          {
            "Value": "marcos.cetystijuana.com.s3-website.us-east-1.amazonaws.com"
          }
        ]
      }
    }
  ]
}
```

Con esto crearemos cambios al registro de nuestro dominio lo cual esta indicado con la llave changes.

Después con el método ACTION , escribimos create para poder generar un nuevo registro dentro de nuestro dominio

Con “ResourceRecordSet” definiremos la configuración de nuestro servicio de DNS. Dentro de esto encontraremos diferentes parámetros.

- Name: El cual es el dominio/subdominio que utilizaremos para enlazar nuestro bucket
- Type: El tipo de registro que crearemos en este caso es un CNAME, ya que redireccionaremos de un dominio el nuestro, a el dominio de aws
- TTL: Time To Live, es el tiempo de cache. El valor TTL actual de un registro determina cuánto tardará en aplicarse cualquier cambio que realices
- Después resource records, es una lista de a los recursos que se le aplicara este mapeo.
  - Este contiene una propiedad values, donde pondremos el subdominio o dominio al cual nuestro registro DNS CNAME apuntara

Una vez creado este archivo procederemos a hacer los cambios con la consola de aws mediante el comando `aws route 53 change-resource-record-sets`, para esto necesitaremos ingresar los hosted-zone-id a la que queremos hacer cambios, y utilizando el parámetro `–change-batch` procederemos a subir nuestro json con las instrucciones necesarias para actualizar nuestro registro, como mostramos a continuación.

```
manmadearc@Manmade-Laptop:~$ aws route53 change-resource-record-sets --hosted-zone-id Z03346142C3RKH191036Y --change-batch file://cname.json
```

Una vez esto sea realizado, aws procederá a crear el registro y agregarlo a la DNS, con lo que podremos acceder a nuestro sitio web, mediante el subdominio que registramos.



### 3. Add yourself to the list of students in cetystijuana.com. 15 points

Para agregar mi usuario a la lista de estudiantes en la página de cetys Tijuana, lo único que hace falta es editar los datos correspondientes en el index del documento que sirve el bucket de s3. Para realizar ese cambio podemos utilizar el siguiente comando.

```
manmadearc@Manmade-Laptop:~$ aws s3 ls s3://cetystijuana.com/
      PRE .idea/
      PRE assets/
      PRE css/
      PRE img/
      PRE js/
      PRE lib/
      PRE mail/
      PRE scss/
2023-02-04 20:33:40      8196 .DS_Store
2023-02-04 20:18:32     1456 LICENSE.txt
2023-02-04 20:18:32       533 READ-ME.txt
2023-02-04 20:19:16        79 error.html
2023-02-04 21:16:05    29626 index.html
2023-02-03 23:22:11     8658 inner-page.html
2023-02-04 00:57:50     8822 main.css
2023-02-04 00:57:50       501 main.js
2023-02-03 23:22:11    10492 portfolio-details.html
2023-02-04 10:35:48       484 recordPolicy.json
```

Con esto podemos ver los archivos del bucket, de estos podemos ver que existe un archivo llamado index.html. dado que este es el principal de la pagina web, podemos editar el archivo si editamos ese. Procedemos a descargar con el siguiente comando.

```
manmadearc@Manmade-Laptop:~$ aws s3 cp s3://cetystijuana.com/index.html index.html
download: s3://cetystijuana.com/index.html to ./index.html
```

Editamos nuestros datos en el html y volvemos a subirlo con el comando

```
<div class="col-lg-6 mt-4">
  <div class="member d-flex align-items-start" data-aos="zoom-in" data-aos-delay="400">
    <div class="pic"></div>
    <div class="member-info">
      <h4>Marcos Alberto Moroyoqui Olan</h4>
      <span><a href="//marcos.cetystijuana.com"> Marcos - Personal Website </a></span>
      <p>"If you automate a mess, you get an automated mess." – Rod Michael</p>
      <div class="social">
        <a href="https://www.linkedin.com/in/marcos-alberto-moroyoqui-olan-2ba0b8108/"> <i class="ri-linkedin-box-fill"></i> </a>
      </div>
    </div>
  </div>
</div>
```

```
manmadearc@Manmade-Laptop:~$ aws s3 cp index.html s3://cetystijuana.com/index.html
```

Para que la foto se visible públicamente es necesario, subir la imagen en el folder que corresponde por lo que procedemos también a ejecutar el comando



```
manmadearc@Manmade-Laptop:~$ aws s3 cp marco.jpgs 3://cetystijuana.com/assets/img/teams/marcos.jpg
```

Una vez que se subieron ambos archivos podemos acceder a la web y observar los cambios.



**Marcos Alberto Moroyoqui Olan**

[Marcos - Personal Website](#)

"If you automate a mess, you get an automated mess." —  
Rod Michael



#### 4. Create an entry in the Students DynamoDB table using the cli with the following Model: 15 points.

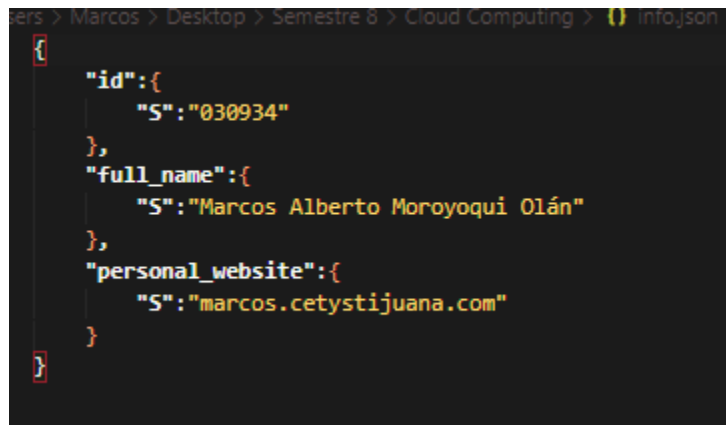
```
{
  "id": "matricula",
  "full_name": "full name",
  "personal_website": "s3 link"
}
```

Para crear una entrada en dynamo db podemos utilizar el siguiente comando.

```
manmadearc@Manmade-Laptop:~$ aws dynamodb put-item --table-name Students -item file:///info.json
```

Donde dynamodb es el servicio, ejecutamos el comando put-item con el parámetro –table-name Students ya que es la tabla o index donde vamos a insertar nuestro elemento. Además, usamos el comando -item para seleccionar el archivo que subiremos.

El archivo debe contener la siguiente estructura.



```
{
  "id": {
    "S": "030934"
  },
  "full_name": {
    "S": "Marcos Alberto Moroyoqui Olán"
  },
  "personal_website": {
    "S": "marcos.cetystijuana.com"
  }
}
```

Como puedes ver este documento no sigue estrictamente la estructura definida en un inicio, y esto es debido al cliente de aws, el cual nos indica que debemos utilizar o definir mediante una llave el tipo de data que introduciremos en al table, en este caso todos son String por lo que utilizamos {"S": "valor"}.

Una vez definido este json, utilizamos el comando anterior y creamos un registro en nuestra tabla.

Una vez definido podemos utilizar el siguiente comando para verificar que nuestro registro existe dentro de la tabla.

```
manmadearc@Manmade-Laptop:/mnt/c/Users/Marcos/Desktop/Semestre 8/Cloud Computing/Personal Website/PersonalWeb$ aws dynamodb get-item --table-name Students --key='{ "id": { "S": "030934" } }' --profile JaquezAws
```

Donde ahora usamos aws dynamodb get-item para obtener el item, especificamos la table y usamos --key="{\"id\":{\"S\":\"030934\"}}" para buscar en la tabla el documento con esa id.

```
aws dynamodb get-item --table-name Students --key="{\"id\":{\"S\":\"030934\"}}"  
{  
  "Item": {  
    "full_name": {  
      "S": "Marcos Alberto Moroyoqui Olán"  
    },  
    "id": {  
      "S": "030934"  
    },  
    "personal_website": {  
      "S": "marcos.cetystijuana.com"  
    }  
  }  
}
```

El cual es desplegado, por lo que existe.

## **5. Watch the Not Just Code Monkeys by Martin Fowler and write your comments in your personal static website. 10 points**

Durante la duración del video, se nos habla de una realidad muy cierta en el campo de desarrollo de software. Los programadores simplemente codifican para ser pagados, sin pensar realmente en la solución que se debe desarrollar, y esto en gran medida es gracias a el modelo ágil, el cual, aunque en sus orígenes pretendía ser una herramienta a o marco para resolver de manera ágil los problemas de los stakeholders, generando un proceso dinámico en el de manera progresiva se generan soluciones a los problemas que plantea el desarrollo. Sin embargo, la realidad es que el proceso, o el marco de metodología ágil, se malentendido y hasta la actualidad, se ha entendido como un proceso dinámico de desarrollo de soluciones.

Lo anterior ha afectado haciendo que la mayoría de los programadores se despegue de la filosofía principal del desarrollo ágil, en la cual, todos los participantes del modelo de desarrollo ágil se involucran en el desarrollo y generación de soluciones e ideas para la problemática que presenta el cliente en el desarrollo de software, reemplazándolo por simplemente desarrolladores que únicamente se preocupan por cumplir con el mayor número de tareas posibles o puntos de historias necesarios para alcanzar su cuota definida, haciendo de los programadores máquinas de programar, en lugar de personas generadoras de soluciones.

Esta parte es hasta cierto punto cierta, y en gran parte la he podido ver reflejada en mi trabajo actual, de una manera similar. En mi trabajo actual, estoy encargado de dar soporte a una herramienta de uso interno para el manejo de inventario y generar cotizaciones de producto como cortinas. Durante mi trabajo para esta empresa, le mayor parte del tiempo he sido el único desarrollador que se ha encargado de mantener y actualizar las funciones necesarias, las cual el cliente necesita por lo que básicamente he sido el encargado del sistema entero alrededor de esta máquina interna del manejo, y si algo he notado que se mostró en el video, fue que en las primeras juntas que tenía con el cliente y el producto poner, lo único que surgió de esas juntas eran nuevas funcionalidades o cosas que hacer en la aplicación, es decir tareas sobre las cuales tenía muy poca incidencia o poder para hacer alguna clase de sugerencia.

Al inicio, el desarrollo estuvo lleno de errores y retrasos, debido a que en conjunto no podamos disponer de un espacio de debate en el cual proponer soluciones al cliente, para poder así elevar la utilidad su sistema, lo cual muchas veces acababa en funcionalidades totalmente inútiles que acababan sin ser usadas por los usuarios de su aplicación. Paso el tiempo, y a raíz de estos problemas se me empezó a dar más voz y surgieron precisamente espacios más amplios de discusión, donde podía dar mi opinión al producto owner y al stakeholder en cuestión de proponer soluciones a ciertos problemas, lo que me llevo a involucrarme más con el producto, y poder entonces tomar decisiones de diseño de la aplicación más fuertes, esto resulto en mi involucrándome

más con la lógica del negocio, entendiendo mejor los flujos, y eliminando procesos que aunque el cliente creía que eran necesarios, el producto owner sobredimensionaba y creaban tareas más grandes y complejas, de lo que realmente se necesitaba.

Mucho de esto dinamizo aún más la dinámica con el cliente y el desarrollo de la aplicación interna, lo que nos llevó a un mejor flujo de trabajo, con menos contratiempos y malentendidos entre cliente y desarrollador principalmente.

Otro de los conceptos que se menciona, es el sentido de responsabilidad a la hora de desarrollar un producto. En este se menciona que cuando uno se siente ligado o atado a esa responsabilidad, es entonces cuando se crea aún más este vínculo entre cliente y desarrollador, porque el desarrollador siente que sus fallos tienen un impacto real, lo que le hace ser consciente de su importancia, y de aquellos a quienes afecta cuando es desarrollador. Volviendo al ejemplo de mi trabajo, esto lo pude observar de manera indirecta cuando un nuevo desarrollador estuvo trabajando conmigo para dar soporte a la aplicación, aunque el producto owner nos asignaba tareas distintas a cada uno, el compromiso del otro desarrollador era menor, por lo que menudo sus cambios estaban incompletos, o no se cercioraba de que la funcionalidad era tal cual el cliente estaba solicitando. Por otro lado, como era yo quien daba la cara al cliente cuando algo salía mal dentro de un deploy, era yo el que tenía que estar asegurándome de que la aplicación funcionara correctamente, y las funcionalidades no se vieran afectadas por lo que sentía más presión a la hora de hacer un deployment, lo que me llevaba a hacer más pruebas de la aplicación e incluso estar revisando y corrigiendo los errores del otro desarrollador.

Algo que me gusta destacar de la plática y es cuando se nos hace llegar la frase “El comportamiento que dejas pasar, es el comportamiento que aceptas”, esta es una frase muy poderosa, porque nos hace ver que en realidad no somos tan rectos como nosotros creemos. Muchas personas en general, decimos que estamos en contra del código espagueti, o el no tener pruebas de automatización, o cosas en general a la hora de realizar código de algún compañero, ignoramos la falta de documentación y pruebas, y no le exigimos a nuestro compañero la misma calidad de código o documentación por su parte, por lo que estamos aceptando ser mediocre y transmitir esa mediocridad a nuestro trabajo.

Finalmente, para concluir como desarrolladores hemos de reflexionar en el impacto que causamos en la sociedad, cualquier producto que generamos , influencia de manera directa el producto que creamos y publicamos en el mundo, y a su vez este influencia a las personas que trabajan en él, debido a que nuestros desarrollos pueden incidir prácticamente en cualquier área debido a la naturaleza de nuestro trabajo, es necesario que empezamos no solo a desarrollar la soluciones, sino también a diseñarlas y aportar ideas de solución que mejoren la experiencia de los usuarios.