PROYECTO 3: TELEMÁTICA

Alejandro Fernández Restrepo
Escuela de Ingeniería
Departamento de Informática y Sistemas
Universidad EAFIT
Medellín, Colombia
afernander@eafit.edu.co

Simón Marín Giraldo
Escuela de Ingeniería
Departamento de Informática y Sistemas
Universidad EAFIT
Medellín, Colombia
smaring1@eafit.edu.co

Miguel Fernando Ramos García
Escuela de Ingeniería
Departamento de Informática y Sistemas
Universidad EAFIT
Medellín, Colombia
mframosg@eafit.edu.co

9 de junio de 2021

RESUMEN

Amazon Web Services (AWS) es una suite completa de servicios en la web de Amazon, para computación basada en la nube, con servicios y APIs web. Amazon le provee servicios a muchas compañías en todo el mundo. Se usó **AWS Educate** para el desarrollo de este proyecto.

1. Introducción

En este informe se muestra de manera detallada la secuencia de pasos que se siguió para desarrollar el **Proyecto 3** del curso de **Telemática**, en el que se proponía el **Despliegue de Servicios y en Red**. Con esto se busca dejar evidencia de la realización de la práctica. **Amazon Web Services (AWS)**.

1.1. Problema

En este problema se busca desplegar una arquitectura de TI robusta y escalable, que permita la operación del sitio web, tanto on-premise como en nube. Esto con el fin de garantizar alta disponibilidad. Este servicio debe soportar múltiples usuarios al igual que la **escalabilidad horizontal**. Es necesario también, la implementación de **balanceadores de carga** con el fin de distribuir las peticiones entrantes entre diferentes **servidores web** y permitir a los usuarios el acceso a la web desplegada. Se debe usar un **CMS**, que en nuestro caso fue **WordPress**.

1.2. Requerimientos

Usando los recursos que provee **Amazon Web Services (AWS)**, se debe diseñar e implementar la solución descrita y se debe garantizar lo siguiente:

- Todos los servicios a desplegar deben ser utilizando el concepto de infraestructura como servicio (IaaS).
- Se hace necesario que usted diseñe e implemente en una nube virtual privada (VPC, por sus siglas en inglés).
- En esta usted debe definir una subred pública y privada. En la pública debe ubicar todos los aspectos relacionados con balanceador de carga, NAT Gateway (en el caso de AWS educate utilice NAT Instance para implementar este servicio), etc. En la privada, debe ubicar el CMS y la capa de persistencia de datos.
- Para efectos del despliegue del CMS se debe utilizar tecnología de contenedores (p.ej., Docker).

- El motor de bases de datos puede ser mysql o maria DB.
- El servicio de balanceador de carga que reciba las peticiones entrantes y las distribuya sobre al menos dos instancias del sitio web (CMS). La solución debe emplear debe ser HAProxy.
- Se requiere que configure e implemente un nombre de dominio para el sitio web a desplegar.

1.3. Arquitectura

La arquitectura a desplegar es la siguiente:

Se plantea la arquitectura anterior para garantizar mayor seguridad, de tal suerte que, para acceder a la capa de persistencia de datos y a los CMS, solamente se pueda hacer desde una instancia al interior de la subnet pública, no directamente a través de internet, pues aumentarían las probabilidades de que nos hackeen o afecten nuestro servicio. Entonces, para darle acceso a Internet a la subred privada aislada, lo hacemos a través del **NAT**, el cual nos va a proveer comunicación entre ambas subredes, para **procesamiento de peticiones**. Debemos tener dos routers, uno para la subred pública y otro para la subred privada. En la subred pública podemos también observar el **bastion**. Este último nos servirá para manipular las máquinas de la subred privada, mediante **SSH**.

2. Creación del VPC

Para la creación del **NAT Gateway**, siguiendo la arquitectura, se procede con la creación de una **VPC** desde **AWS**, dentro de la cual se crean las dos subredes mencionadas anteriormente. Cabe aclarar que las máquinas dentro de la subred privada, no tienen **IP pública**, solamente privada, y que las máquinas dentro de la subred pública tienen **IP tanto pública como privada**.

Para crear el VPC nos vamos a la sección de **VPC** de AWS y lo creamos manualmente de la siguiente forma:

Al finalizar la configuración, tendremos el siguiente **VPC**:

Your VPCs (2) Info					
<input type="text" value="Filter VPCs"/> < 1 >					
☐	Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR (Netw
☐	Entrega-Final ☒	vpc-0827e6934b445715c	Available	172.26.0.0/23	-
☐	-	vpc-08cd4975	Available	172.31.0.0/16	-

Tenemos también, las siguientes subredes, que pertenecen al **VPC Entrega Final**:

☐	pub-subnet	subnet-005fa6c93742ec93b	Available	vpc-0827e6934b445715c En...	172.26.1.0/25
☒	pri-subnet	subnet-0de4b9fd955cf1ab8	Available	vpc-0827e6934b445715c En...	172.26.0.0/24

Debemos activar la asignación automática de IP's elásticas públicas, así:

Auto-assign IPv4 ☒ Enable auto-assign public IPv4 address [?](#)

Es muy importante tener en cuenta que para la creación de cada una de las máquinas, debemos especificar la **VPC Entrega Final** y la **subred**, de acuerdo a donde se vaya a colocar cada una, de la siguiente manera:

Número de instancias [?](#) [Lanzar en grupo de Auto Scaling](#) [?](#)

Opción de compra [?](#) ☐ Solicitar instancias de spot

Red [?](#) vpc-0827e6934b445715c | Entrega-Final [?](#) [Crear nueva VPC](#)

Subred [?](#)

subnet-005fa6c93742ec93b | pub-subnet | us-east-1-
 subnet-005fa6c93742ec93b | pub-subnet | us-east-1-a
 subnet-0de4b9fd955cf1ab8 | pri-subnet | us-east-1-a
 Usar configuración de subred (habilitar)

[Crear nueva subred](#)

Asignar automáticamente IP pública [?](#)

Grupo de ubicación [?](#) ☐ Agregue la instancia a un grupo de ubicación.

Reserva de capacidad [?](#)

Directorio de unión al dominio [?](#) [?](#) [Crear nuevo directorio](#)

Rol de IAM [?](#) [?](#) [Crear un nuevo rol de IAM](#)

Es necesario también, crear la **internet gateway** (salida hacia internet), el cual vamos a adherir a nuestro **VPC**.

3. Subred pública

3.1. NAT Gateway

Para su creación, vamos a tener el siguiente **security group**:

Tipo	Protocolo	Intervalo de puertos	Origen	Descripción: opcional	
HTTP	TCP	80	Person... <input type="text" value="0.0.0.0/0"/>		Eliminar
HTTP	TCP	80	Person... <input type="text" value="::/0"/>		Eliminar
HTTPS	TCP	443	Person... <input type="text" value="0.0.0.0/0"/>		Eliminar
HTTPS	TCP	443	Person... <input type="text" value="::/0"/>		Eliminar
Todos los ICMP IPv4	ICMP	Todo	Person... <input type="text" value="0.0.0.0/0"/>		Eliminar

Debemos ir a la sección de **Route Tables** para crear asociaciones entre las subredes y sus correspondencias, sea pública o privada.

Luego, procedemos a crear una instancia **EC2** con la AMI para **NAT**:

Es necesario crear un **Internet Gateway**. Se puede hacer agregando en las **Route Tables** lo siguiente:

Destination	Target	Status	Propagated	Remove
0.0.0.0/0	local	Active	No	
		No		

En la lista de instancias, es necesario cambiar el ajuste de origen/destino y dejarlo desactivado.

Regresando a **Route Tables**, debo crear una ruta entre el **NAT** y los hosts en la **red privada**. También debo enrutar la subred pública a **Internet**, asociándola con el **internet gateway**.

3.2. Bastion

El Bastion es una máquina que se encuentra en la **subred pública**. Es la única máquina con permisos **SSH** a las máquinas de la **subred privada**. Con este, podemos manipular y configurar estas máquinas. Con esto evitamos el acceso externo de ataques.

Para la configuración del **Bastion**, es de suma importancia habilitar la asignación automática de IP pública:

Red ⓘ vpc-0827e6934b445715c | Entrega-Final ⓘ Crear nueva VPC

Subred ⓘ subnet-005fa6c93742ec93b | pub-subnet | us-east-1 ⓘ Crear nueva subred

120 direcciones IP disponibles

Asignar automáticamente IP pública ⓘ ☒ Habilitar ⓘ

Al bastion le debo enviar la llave para poder acceder a las máquinas de la **subred privada** con el comando `scp -i entregafinal.pem entregafinal.pem ubuntu@54.162.20.97:~`.

3.3. Load Balancer (HAProxy)

La configuración de este es similar a la realizada en la **entrega 1**, con algunos cambios en el archivo `haproxy.cfg`. El **Security Group** para este servicio es el siguiente:

Intervalo de p...	Protocolo	Origen	Grupos de seguridad
80	TCP	0.0.0.0/0	Sg-LoadBalancer
80	TCP	::/0	Sg-LoadBalancer
22	TCP		Sg-LoadBalancer

El **load balancer** se encargará de distribuir el tráfico de usuarios entre nuestras dos instancias de **CMS**. Desde la teoría computacional, el balanceo de carga consiste en distribuir un conjunto de tareas en un conjunto de recursos de manera eficaz.

Para realizar la configuración del **load balancer** montamos un servidor **haproxy** y comenzamos con los siguientes comandos:

```
sudo yum install haproxy
cd /etc/haproxy/
sudo nano haproxy.cfg
```

En nano, debemos configurar varias cosas. Debemos especificar el frontend y el backend. Este último se refiere a nuestras dos instancias de CMS. También es necesario darle todos los permisos al puerto 80:

```

frontend app_servers
  bind *:80
  default_backend apps

backend app
  balance roundrobin
  server app1 127.0.0.1:5001 check
  server app2 127.0.0.1:5002 check
  server app3 127.0.0.1:5003 check
  server app4 127.0.0.1:5004 check

backend apps
  server cms1 [redacted]:80 check
  server cms2 [redacted]:80 check
    
```

Para este **proyecto 3**, tenemos el archivo de configuración del HAProxy así:

```

GNU nano 2.9.8 haproxy.cfg
balance roundrobin
server static 127.0.0.1:4331 check

# round robin balancing between the various backends
backend app
  balance roundrobin
  server app1 127.0.0.1:5001 check
  server app2 127.0.0.1:5002 check
  server app3 127.0.0.1:5003 check
  server app4 127.0.0.1:5004 check

backend apps
  server pri-cms-1 172.26.0.231:80 check
  server pri-cms-2 172.26.0.220:80 check
    
```

Finalmente, debemos iniciar el servicio y configurarlo para que se inicie automáticamente al arrancar la máquina:

```

sudo systemctl start haproxy.service
sudo systemctl enable haproxy.service
    
```

3.4. DNS y nombre de dominio

Al igual que en el **proyecto 1**, se creó un **DNS**, pero esta vez se creó la instancia en la **subred privada**. Pudimos establecer nuestra web en este dominio: <http://telematica.gq/>

El **Security Group** para este servicio es el siguiente:

Intervalo de p...	Protocolo	Origen	Grupos de seguridad
80	TCP	0.0.0.0/0	launch-wizard-4
80	TCP	::/0	launch-wizard-4
22	TCP	[redacted]	launch-wizard-4
53	UDP	0.0.0.0/0	launch-wizard-4
53	UDP	::/0	launch-wizard-4

El **DNS** se encarga de enlazar la **dirección IP** del **load balancer** con la **dirección del dominio**. Tenemos dos instancias de **DNS** en caso de que una presente inconvenientes, se siga con el servicio desde la otra.

Para configurar el **DNS** nos ayudamos del siguiente [tutorial](#).

Para comenzar con la configuración debemos proceder con los siguientes comandos:

```

yum install bind bind-utils -y
nano -w /etc/named.conf
    
```

En este archivo se debe garantizar acceso para cualquiera en el query y permitir transferencia de datos con el **load balancer**, **DNS secundario** y el **DNS actual** en el campo allow-transfer. También se debe crear la **zona**. Este

archivo enlaza al **dominio** con el **load balancer**. También se debe configurar el DNS para los **correos electrónicos**, sin embargo, para esta práctica no se implementó servidor de **correo electrónico**.

```
options {
    listen-on port 53 { 127.0.0.1; };
    listen-on-v6 port 53 { ::1; };
    directory /var/named;
    dump-file /var/named/data/cache_dump.db;
    statistics-file /var/named/data/named_stats.txt;
    memstatistics-file /var/named/data/named_mem_stats.txt;
    recursing-file /var/named/data/named.recursing;
    secroots-file /var/named/data/named.secroots;
    allow-query { any; };
    allow-transfer { localhost; };
}
```

```
GNU nano 2.9.8 vazo.cf.zone
$TTL 86400
$ IN SOA as1.vazo.cf. root.vazo.cf. (
    2013042201 ;Serial
    3600 ;Refresh
    1800 ;Retry
    604800 ;Expire
    86400 ;Minimum TTL
)
; Specify our two nameservers
; Resolve nameserver hostnames to IP, replace with your two droplet IP addresses
; Define hostname -> IP pairs which you wish to resolve
$ IN NS 1 vazo.cf.
$ IN A 1 vazo.cf.
www IN CNAME vazo.cf.
$ IN MX 10 vazo.cf.
```

Posteriormente, entramos a [freenom](#) para configurar el nameserver, para el redireccionamiento a los dos servidores DNS. Al momento de utilizarlo, se busca primero en el **DNS primario**. Si este no funciona, se busca en el **DNS secundario**.

Es importante aclarar que en la zona se debe definir el tipo **slave** e indicar la **IP pública** del **DNS primario** en el campo **masters**.

4. Subred privada

4.1. Base de datos

Para la configuración de la **base de datos** procedimos con estos comandos:

```
1 sudo apt-get update
2 sudo apt-get install apt-transport-https ca-certificates
  curl gnupg lsb-release
3 curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg
  --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
4 echo "deb [arch=amd64
  signed-by=/usr/share/keyrings/docker-archive-keyring.gpg]
  https://download.docker.com/linux/ubuntu \
5 $(lsb_release -cs) stable" | sudo tee
  /etc/apt/sources.list.d/docker.list > /dev/null
6 sudo apt-get update
7 sudo apt-get install docker-ce docker-ce-cli containerd.io
8 sudo docker --version
9 sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2
  /docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
10 sudo chmod +x /usr/local/bin/docker-compose
11 docker-compose --version
12 mkdir mysql
13 cd mysql/
14 nano db.yml
15 sudo docker-compose up -d
16 ls
17 mv db.yml compose.yml
18 sudo docker-compose up -d
19 cd
```

```

20 exit
21 history
22 ls
23 cd mysql/
24 ls
25 nano compose.yml
26 sudo docker ps
27 nano compose.yml
28 sudo docker restart mysql_db_1
29 sudo docker ps
30 exit
31 ls
32 exit
33 ls
34 mysql
35 sudo apt install mysql-client-core-8.0
36 mysql
37 mysql -u root
38 mysql -u root -p
39 sudo service mysql start
40 nano compose.yml
41 cd mysql/
42 nano compose.yml
43 mysql -u root -h my-db
44 mysql -u root -h my-database
45 mysql -u root -h database
46 sudo docker ps
47 sudo docker inspect mysql_db_1
48 mysql -u root -h 172.18.0.2
49 mysql -u user -h 172.18.0.2
50 mysql -u user -h 172.18.0.2 -p
51 exit
52 history
    
```

Para esta configuración, nuestro archivo `compose.yml` contiene lo siguiente:

```

version: '3.3'
services:
  db:
    image: mysql:5.7
    restart: always
    environment:
      MYSQL_DATABASE: 'database'
      MYSQL_USER: 'user'
      MYSQL_PASSWORD: 'Eafit1221*'
      MYSQL_ROOT_PASSWORD: 'Eafit1221*'
    ports:
      - '3306:3306'
    expose:
      - '3306'
    volumes:
      - my-db:/var/lib/mysql
volumes:
  my-db:
    
```

4.2. CMS

Para la configuración de los CMS, se procedió con los siguientes comandos:

```

1  sudo apt-get update
2  sudo apt-get remove docker docker-engine docker.io containerd runc
3  sudo apt-get update
4  sudo apt-get install      apt-transport-https      ca-certificates
   curl      gnupg      lsb-release
5  curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg
   --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
6  echo "deb [arch=amd64
   signed-by=/usr/share/keyrings/docker-archive-keyring.gpg]
   https://download.docker.com/linux/ubuntu \
7  $(lsb_release -cs) stable" | sudo tee
   /etc/apt/sources.list.d/docker.list > /dev/null
8  sudo apt-get update
9  sudo apt-get install docker-ce docker-ce-cli containerd.io
10 sudo docker --version
11 sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2
   /docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
12 sudo chmod +x /usr/local/bin/docker-compose
13 docker-compose --version
14 mkdir cms1
15 cd cms1/
16 nano compose.yml
17 sudo docker-compose up -d
18 sudo docker
19 sudo docker container
20 sudo docker ps -aq
21 sudo docker ps
22 curl localhost:80
23 sudo docker ps
24 curl localhost:80
25 nano compose.yml
26 curl localhost:80
27 exit
28 curl localhost:80
29 ls
30 cd cms1/
31 curl localhost:80
32 sudo apt-get install lynx
33 lynx localhost:80
34 history
35 nano compose.yml
36 cd cms1/
37 sudo nano compose.yml
38 curl localhost:80
39 sudo docker container inspect
40 sudo docker container ls
41 sudo docker container inspect cms1_wordpress_1
42 find / -name wp-config.php
43 sudo find / -name wp-config.php
44 cd /var/lib/docker
45 ls
46 sudo ls
47 sudo cd volumes
48 cd volumes
49 cd cms1_wordpress
50 cd _data
51 sudo nano wp-config.php
52 cd

```



```

53 sudo docker-compose config
54 sudo nano wp-config.php
55 sudo nano compose.yml
56 cd cms1/
57 nano compose.yml
58 cd
59 cd /var
60 ls
61 cd www
62 cd
63 cd cms1/
64 nano compose.yml
65 sudo docker ps
66 sudo docker stop cms1_wordpress_1
67 sudo docker start cms1_wordpress_1
68 exit
69 history
70 exit

```

Para esta configuración, nuestro archivo `compose.yml` contiene lo siguiente:

```

version: '3.1'
services:
  wordpress:
    image: wordpress
    restart: always
    ports:
      - 80:80
    environment:
      WORDPRESS_DB_HOST: '172.26.0.131'
      WORDPRESS_DB_USER: 'user'
      WORDPRESS_DB_PASSWORD: 'Eafit1221*'
      WORDPRESS_DB_NAME: 'database'
    volumes:
      - wordpress:/var/www/html
volumes:
  wordpress:

```

5. Resultados finales

Finalmente pudimos montar nuestro **VPC** con subred **pública** y **privada**. Pudimos tener un nuevo dominio, el cual será <http://telematica.gq/> el cual está funcionando.