**Práctica 12: *Despliegue de un clúster escalable, altamente disponible y seguro de un servicio Proxy web en una VPC***

**Objetivo:**

Los servidores Proxy web permiten que los clientes de una red puedan navegar por Internet, mediante los protocolos HTTP y HTTPS de forma segura, permitiendo el enmascaramiento de las direcciones IP de los clientes, el filtrado de solicitudes y el cacheo de respuestas.

El objetivo de esta práctica es diseñar y desplegar la infraestructura necesaria para un clúster de servidores proxy balanceados y escalable, atendiendo a las mejores prácticas de seguridad y fiabilidad.

Para ello, se hará uso de los siguientes servicios:

* Un servicio de Amazon ECS, desplegado como una tarea de AWS Fargate. Cada tarea ejecutará un contenedor Docker personalizado con un servicio Squid (Proxy web).
* Un balanceador de carga de red (NLB) que permitirá distribuir la carga de solicitudes HTTP y HTTPS entre los diferentes contenedores Docker que componen el servicio Proxy. Además, se utilizarán puntos de enlace de servicio para exponer el servicio Proxy en diferentes VPCs de la misma región (podrían encontrarse en la misma o diferentes cuentas de AWS)
* Un repositorio privado de Amazon ECR donde se cargará la imagen del contenedor Docker que implementa el servicio Proxy.
* Una instancia EC2 con sistema operativo Linux que simulará solicitudes HTTP y HTTPS mediante *Lynx* (navegador web de consola).
* AWS CloudFormation, que permitirá simplificar la creación de la infraestructura de red de las VPCs utilizadas.
* Dos Gateway NAT, que permitirán el acceso a Internet de los contenedores Docker que implementan el servicio Proxy
* AWS Systems Manager, que se utilizará para el acceso seguro y privado a la instancia cliente web.

**Requerimientos:**

* Disponer de acceso a los recursos de AWS a través de un *sandbox* de AWS Academy
* Disponer de acceso a una máquina Linux con AWS CLI y el motor de contenerización Docker instalados.

**Arquitectura propuesta:**

**Diagrama, Escala de tiempo

Descripción generada automáticamente**

**Realización:**

**CREACIÓN DE LA INFRAESTRUCTURA DE RED**

1. Para simplificar la creación de la infraestructura de red, se proporciona una plantilla de AWS CloudFormation, llamada *vpc.yaml* (dentro del directorio *vpc* del repositorio), que utilizaremos para desplegar tanto la VPC que aloja el servicio como la VPC donde se encuentran los clientes que consumirán el servicio Proxy. La configuración de las VPCs quedará como sigue:

* **VPC Servicio (10.0.0.0/16)**
  + Subred Pública 1 (10.0.0.0/24)
  + Subred Pública 2 (10.0.1.0/24)
  + Subred Privada 1 (10.0.2.0/24)
  + Subred Privada 2 (10.0.3.0/24)
  + 1 Internet Gateway
  + 2 NAT Gateway (uno por cada zona de disponibilidad)
  + Sin bastión
* **VPC Cliente (172.16.0.0/16)**
  + Subred Pública 1 (172.16.0.0/24)
  + Subred Pública 2 (172.16.1.0/24)
  + Subred Privada 1 (172.16.2.0/24)
  + Subred Privada 2 (172.16.3.0/24)
  + Sin Internet Gateway
  + Sin Nat Gateways
  + Sin bastión

Las infraestructuras anteriores están parametrizadas en dos archivos denominados *service-vpc.json* y *client-vpc.json* (dentro del directorio *vpc* del repositorio). Cada uno de estos archivos contiene los parámetros de la plantilla de AWS CloudFormation especificada en el archivo *vpc.yaml* necesarios para desplegar sendas VPCs.

Para crear la infraestructura de red de la VPC Servicio, introducimos la siguiente orden desde la AWS CLI:

$ aws cloudformation deploy --template-file vpc.yaml --stack-name service --parameter-overrides file://service-vpc.json --region us-east-1

Tras unos minutos, la infraestructura ya estará creada. Para crear la infraestructura de red de la VPC Cliente, introducimos una orden similar:

$ aws cloudformation deploy --template-file vpc.yaml --stack-name client --parameter-overrides <file://client-vpc.json> --region us-east-1

1. Una vez creadas las VPCs, podremos visualizarlas desde la consola de Amazon VPC:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

**CREACIÓN DEL SERVICIO PROXY WEB**

1. El servicio Proxy Web se implementará contenerizado mediante Docker y se desplegará utilizando el servicio de orquestación de contenedores Amazon Elastic Container Service (ECS). Previamente, personalizaremos un contenedor Docker con un servicio *Squid* (Proxy Web) a partir de un archivo *Dockerfile* que se encuentra dentro del directorio *squid* del repositorio. Este directorio, además, contiene el archivo de configuración de *Squid* (*squid.conf*) y el script que se ejecutará al levantar el contenedor Docker.
2. Para generar la imagen del contenedor Docker con el servicio Proxy, nos ubicamos en el directorio *squid* e introducimos la siguiente orden:

$ docker build -t proxy-squid:latest .

Se obtendrá una salida similar a la siguiente:

[+] Building 1.7s (11/11) FINISHED

=> [internal] load build definition from Dockerfile 0.0s

=> => transferring dockerfile: 484B 0.0s

=> [internal] load .dockerignore 0.0s

=> => transferring context: 2B 0.0s

=> [internal] load metadata for docker.io/library/ubuntu:latest 1.2s

=> [1/6] FROM docker.io/library/ubuntu:latest@sha256:27cb6e6ccef575a4698b66f5de06c7ecd61589132d5a91d098f7f3f9285415a9 0.0s

=> [internal] load build context 0.0s

=> => transferring context: 4.45kB 0.0s

=> CACHED [2/6] RUN apt-get update && DEBIAN\_FRONTEND=noninteractive apt-get install -y squid && rm -rf /var/lib/apt/lists/\* 0.0s

=> CACHED [3/6] RUN usermod -a -G tty proxy 0.0s

=> [4/6] COPY entrypoint.sh /sbin/entrypoint.sh 0.0s

=> [5/6] RUN chmod 755 /sbin/entrypoint.sh 0.4s

=> [6/6] COPY squid.conf /etc/squid/squid.conf 0.0s

=> exporting to image 0.1s

=> => exporting layers 0.0s

=> => writing image sha256:6a5847cae50d589d8f3a0dea994159456e38bbd71aa79634e810a341995eece9 0.0s

=> => naming to docker.io/library/proxy-squid:latest 0.0s

Tras ello, ya habremos creado la imagen del contenedor Docker con el servicio *Squid*.

1. A continuación, vamos a crear un repositorio privado de imágenes de contenedores Docker mediante el servicio de Amazon Elastic Container Registry (ECR), para lo que introducimos la siguiente orden desde la AWS CLI:

$ repo=$(aws ecr create-repository --repository-name proxy-squid --region us-east-1 --query repository.repositoryUri --output text)

1. Para poder enviar (*push*) nuestra imagen del repositorio Docker local al repositorio en Amazon ECR, es necesaria la autenticación previa. Para ello introducimos la siguiente orden:

$ docker login -u AWS -p $(aws ecr get-login-password --region us-east-1) $repo

Si todo va bien, obtendremos un mensaje *Login Succedeed* desde la consola.

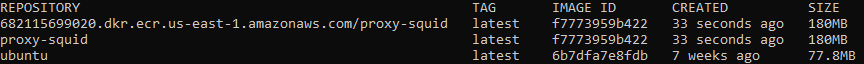
1. Para poder enviar la imagen de nuestro contenedor Docker, debemos asegurarnos que nuestra imagen se llama igual que el repositorio de Amazon ECR que hemos creado. Para ello, debemos renombrar nuestra imagen de contenedor Docker mediante la siguiente instrucción:

$ docker tag proxy-squid:latest $repo:latest

Podemos visualizar las imágenes locales mediante la orden:

$ docker images

Obtenemos la siguiente salida:



1. Por último, sólo nos resta cargar el repositorio local en Amazon ECR, para lo que ejecutamos la orden:

$ docker push $repo:latest

Obtendremos una salida similar a la siguiente:

Texto

Descripción generada automáticamente

1. El siguiente paso por realizar es crear una definición de tarea en Amazon ECS, donde especificaremos un único contenedor que obtendrá la imagen a partir del repositorio de Amazon ECR. Para ello, previamente debemos obtener el rol de ejecución de la tarea (LabRole, debido a las restricciones de los *AWS Academy Learner Labs*):

$ LabRole=$(aws iam get-role --role-name LabRole --query Role.Arn --output text)

Para facilitar la práctica hay creado un archivo denominado *definicion-tarea* en el directorio *task* del repositorio de la práctica. El contenido de dicho archivo es el siguiente:

{

    "containerDefinitions": [

       {

          "essential": true,

          "image": "<imagen>:latest",

          "logConfiguration": {

             "logDriver": "awslogs",

             "options": {

                "awslogs-group" : "/ecs/squid-task",

                "awslogs-region": "us-east-1",

                "awslogs-stream-prefix": "ecs"

             }

          },

          "name": "squid",

          "portMappings": [

             {

                "containerPort": 3128,

                "hostPort": 3128,

                "protocol": "tcp"

             }

          ]

       }

    ],

    "cpu": "256",

    "executionRoleArn": "<LabRole>",

    "family": "proxy-web",

    "memory": "512",

    "networkMode": "awsvpc",

    "runtimePlatform": {

         "operatingSystemFamily": "LINUX"

     },

    "requiresCompatibilities": [

        "FARGATE"

     ]

 }

El archivo anterior define una tarea llamada *proxy-web* con un único contenedor basado en la imagen creada anteriormente, con una asignación de 0.25 vCPU (256) y 512 MiB de RAM.

En dicho archivo, hay que sustituir los campos “*image*” y “*executionRoleArn*” del JSON anterior por los valores obtenidos anteriormente, por lo que introducimos las siguientes órdenes:

$ sed -i 's|<imagen>|'$repo'|g' definicion-tarea.json

$ sed -i 's|<LabRole>|'$LabRole'|g' definicion-tarea.json

Por último, y para crear la definición de la tarea ejecutamos la orden siguiente desde la AWS CLI:

$ aws ecs register-task-definition --cli-input-json file://definicion-tarea.json --region us-east-1

Si todo marcha bien, obtendremos un documento JSON con toda la información de la tarea registrada.

1. A continuación, crearemos un clúster de Amazon ECS donde desplegaremos nuestro servicio *Proxy*. Para ello, desde la CLI, introducimos la siguiente orden:

$ aws ecs create-cluster --cluster-name proxy-cluster --region us-east-1

Obtendremos como salida un documento JSON con la información sobre la creación del clúster de ECS.

1. Una vez creado el clúster, es necesario crear un balanceador de carga de red (*Network Load Balancer*, *NLB*) privado (no expuesto a Internet) en la VPC del servicio que abarque las dos subredes privadas. El balanceador de carga distribuirá el tráfico entre las diferentes tareas *proxy-web* desplegadas en el clúster.

Previamente, debemos obtener los IDs de las subredes privadas creadas en el apartado 1 mediante las plantillas de AWS CloudFormation. Para ello ejecutamos las siguientes órdenes:

$ privada1=$(aws cloudformation describe-stacks --stack-name service --query 'Stacks[0].Outputs[?OutputKey==`Privada1`].OutputValue' --output text --region us-east-1)

$ privada2=$(aws cloudformation describe-stacks --stack-name service --query 'Stacks[0].Outputs[?OutputKey==`Privada2`].OutputValue' --output text --region us-east-1)

A continuación, creamos el NLB mediante la orden de la CLI:

$ aws elbv2 create-load-balancer --name proxy-nlb --type network --subnets $privada1 $privada2 --scheme internal

1. Tras crear el balanceador de carga, crearemos un grupo de seguridad necesario para el servicio proxy. En este caso, deberemos permitir el tráfico de entrada por el puerto 3128 TCP. Para ello, debemos conocer el ID de la VPC del servicio, por lo que ejecutamos la orden:

$ vpc=$(aws cloudformation describe-stacks --stack-name service --query 'Stacks[0].Outputs[?OutputKey==`VPC`].OutputValue' --output text --region us-east-1)

A continuación, creamos el grupo de seguridad:

$ proxysg=$(aws ec2 create-security-group --group-name service-proxy-sg --description "Trafico 3128 TCP" --vpc-id $vpc --output text --query GroupId)

Asignamos una regla de entrada que permita el tráfico por el puerto 3128 TCP:

$ aws ec2 authorize-security-group-ingress --group-id $proxysg --protocol tcp --port 3128 --cidr 0.0.0.0/0 --region us-east-1

1. es necesario crear un servicio basado en la definición de tarea anterior. El servicio determina, entre otras configuraciones las cantidades mínimas, máximas y deseadas de la tarea *proxy-web* que se ejecutarán, si el servicio estará o no balanceado, las subredes donde se desplegará el servicio, etc.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. Una vez creada la VPC es necesario activar los nombres DNS de host. Para ello, desde el menú **Actions** seleccionamos la opción **Edit DNS hostnames**

Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente

1. Marcamos la casilla de verificación **Enable** y presionamos el botón **Save changes**.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. Vamos a crear ahora un Gateway de Internet, que será necesario para el acceso a la Internet pública desde nuestra VPC. Para ello, seleccionamos la opción **Internet Gateways** desde la consola de Amazon VPC y presionamos el botón **Create Internet Gateway**:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

1. A continuación, introducimos en el campo **Name tag** el valor *mi-IGW* y presionamos el botón **Create Internet Gateway**.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

1. Ahora es necesario asociar el Gateway de Internet a nuestra VPC; para ello, desde el menú **Actions** seleccionamos la opción **Attach to VPC**:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

1. Seleccionamos nuestra VPC y presionamos el botón **Attach Internet Gateway**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. A continuación, vamos a crear la estructura de subredes necesaria. Necesitaremos 6 subredes, de las cuales 2 serán públicas y 4 privadas, distribuidas en dos zonas de disponibilidad diferentes. Se utilizará la siguiente asignación de bloques CIDR, tal y como muestra la siguiente tabla:

|  |  |  |  |
| --- | --- | --- | --- |
| **Nombre** | **AZ** | **Bloque CIDR** | **Utilización** |
| publica-01 | *us-east-1a* | 10.0.0.0/24 | Gateway NAT. Acceso a Internet |
| publica-02 | *us-east-1b* | 10.0.1.0/24 | Gateway NAT. Acceso a Internet |
| privada-01-proxy | *us-east-1a* | 10.0.2.0/24 | Servidores Proxy Web |
| privada-02-proxy | *us-east-1b* | 10.0.3.0/24 | Servidores Proxy Web |
| privada-01-clientess | *us-east-1a* | 10.0.4.0/24 | Clientes web |
| privada-02-clientes | *us-east-1a* | 10.0.5.0/24 | Clientes web |

Para crear las subredes, pulsamos el botón **Create subnte** desde la opción **Subnets** en la consola de Amazon VPC:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. A continuación, seleccionamos en el campo **VPC ID** el ID de nuestra VPC e introducimos los datos de todas las subredes en el formulario a partir de los proporcionados en la tabla del apartado anterior. Por último, presionamos el botón **Create subnet**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. Para permitir que las instancias en las subredes públicas adquieran automáticamente una IP pública y un nombre DNS público es necesario activar la autoasignación de IPv4. Para ello seleccionamos, desde las subredes, la subred *publica-01* y presionamos el botón **Actions** y seleccionamos la opción **Edit subnet settings**:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

1. En la siguiente ventana, marcamos la opción **Enable auto-assign public IPv4 address** y presionamos el botón **Save**:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

1. Repetimos el procedimiento del apartado 13 para la subred *publica-02****.***
2. Ahora vamos a crear las tablas de encaminamiento para las subredes púbicas y privadas. Necesitaremos 3 tablas de encaminamiento, tal y como se detalla:

Subredes públicas

|  |  |
| --- | --- |
| **Destino** | **Objetivo** |
| 10.0.0.0/16 | Local |
| 0.0.0.0/0 | Internet Gateway |

Subredes privadas *us-east-1a*

|  |  |
| --- | --- |
| **Destino** | **Objetivo** |
| 10.0.0.0/16 | Local |
| 0.0.0.0/0 | NAT Gateway 01 |

Subredes privadas *us-east-1b*

|  |  |
| --- | --- |
| **Destino** | **Objetivo** |
| 10.0.0.0/16 | Local |
| 0.0.0.0/0 | NAT Gateway 02 |

Para crea la tabla de encaminamiento de las subredes públicas, seleccionamos la opción **Route Tables** desde la consola de Amazon VPC y presionamos el botón **Create Route Table**:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. Introducimos en el campo **Name** el valor *publicas*, seleccionamos en el campo **VPC** la VPC etiquetada como *mi-VPC* y presionamos el botón **Create Route Table**.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

1. Ahora, desde la opción **Routes** presionamos el botón **Edit Routes**:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

1. Se añade la ruta correspondiente indicando como objetivo el Gateway de Internet. Por último se presiona el botón **Save changes**:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

1. Ahora sólo resta asociar la tabla de encaminamiento recién creada con las subredes públicas de ambas zonas de disponibilidad. Para ello desde la pestaña de **Subnet associations** presionamos el botón **Edit subnet associations**:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

1. Desde la tabla siguiente seleccionamos ambas subredes públicas y presionamos el botón **Save associations**:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

1. Las tablas de encaminamiento para las subredes privadas deben apuntar en su ruta por defecto (0.0.0.0/0) al Gateway NAT correspondiente (recuerda que un Gateway NAT es altamente disponible dentro de una zona de disponibilidad; al abarcar nuestra infraestructura dos zonas de disponibilidad se hace necesario crear un segundo Gateway NAT para garantizar la alta disponibilidad).Para ello, previamente crearemos dichos Gateways NAT. Desde la consola de Amazon VPC, seleccionamos la opción **NAT Gateways** y presionamos el botón **Create NAT Gateway**:

Interfaz de usuario gráfica, Texto, Correo electrónico

Descripción generada automáticamente

1. En la siguiente ventana, introducimos en el campo **Name** el valor *NAT-1* y seleccionamos en el campo **Subnet** la subred etiquetada como *publica-01*. Presionamos el botón **Allocate Elastic IP** (todos los Gateway NAT necesitan de una IP Elástica) y por último, presionamos el botón **Create NAT Gateway**:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. Repetimos el proceso anterior para crear un segundo Gateway NAT en la subred *publica-02*.
2. Tras un par de minutos, podremos comprobar que los Gateway NAT se encuentra operativos:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. A continuación, se creará una tabla de encaminamiento para las subredes privadas de la zona de disponibilidad *us-east-1a*. Para ello, desde la consola de Amazon VPC seleccionamos la opción **Route Tables** y creamos una tabla de encaminamiento como en los apartados anteriores:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. Editamos la tabla de encaminamiento y hacemos que la ruta por defecto apunte al Gateway NAT *NAT-1*.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

1. Asociamos la tabla de encaminamiento *privadas-01* con las subredes etiquetadas como *privada-01-proxy* y *privada-01-clientes*:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

1. Repetimos de forma análoga los pasos 25 al 27, pero para la tabla de encaminamiento de las subredes privadas de la zona de disponibilidad *us-east-1b*.

**CREACIÓN DEL CLUSTER DE SERVIDORES PROXY WEB**

1. Previamente a la creación del clúster, es necesario crear grupos de seguridad tanto para el clúster de las instancias EC2 que actuarán como Proxy web, como para las instancias EC2 que se comportarán como clientes HTTP/HTTPS. Siguiendo las mejores prácticas de seguridad, los grupos de seguridad deben configurarse como sigue:

**Grupo de seguridad *proxy-sg***

Reglas de Entrada

|  |  |  |
| --- | --- | --- |
| **Protocolo** | **Puertos** | **Origen** |
| TCP | 3128 | 10.0.0.0/16 |

Reglas de Salida

|  |  |  |
| --- | --- | --- |
| **Protocolo** | **Puertos** | **Destino** |
| TCP y UDP | Todos | 0.0.0.0/0 |

**Grupo de seguridad *web-sg***

Reglas de Entrada

|  |  |  |
| --- | --- | --- |
| **Protocolo** | **Puertos** | **Origen** |

Reglas de Salida

|  |  |  |
| --- | --- | --- |
| **Protocolo** | **Puertos** | **Destino** |
| TCP y UDP | Todos | 0.0.0.0/0 |

Para crear el primer grupo de seguridad, desde la consola de Amazon VPC seleccionamos la opción **Security Groups** y presionamos el botón **Create Security Group**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. Introducimos en el campo **Security Group name** el valor *proxy-sg*. A continuación, introducimos una descripción y seleccionamos la VPC etiquetada como *mi-vpc*. Añadimos las reglas de entrada y salida correspondientes a partir de los datos de la tabla del apartado anterior. Por último, presionamos el botón **Create Security Group.**

Una captura de pantalla de una red social

Descripción generada automáticamente

1. Repetimos el paso anterior para crear el grupo de seguridad *web-sg*, tal y como muestra la siguiente figura:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. A continuación, vamos a crear una plantilla de lanzamiento donde se indicarán las características de las instancias EC2 del clúster de servidores Proxy y se hará un *bootstrap* para instalar y configurar el software *Apache Squid*. Para ello, desde la consola de Amazon EC2, seleccionamos la opción **Launch Templates** y presionamos el botón **Create Launch Template.**

Captura de pantalla de un celular

Descripción generada automáticamente

1. Desde la ventana siguiente completamos con las siguientes opciones de configuración:
   * **Launch Template Name**: *proxy-template*
   * **Template Version Description**: *Versión inicial*
   * **Application and OS Images**:
     1. **QuickStart:** Seleccionamos *Amazon Linux*
     2. **Architecture**: *64-bit (Arm)*
   * **Instance type**
     1. **Instance type**: *t4g.small*
   * **Key pair (login)**
     1. **Key pair name**: *Don’t include in launch template*
   * **Network settings**:
     1. **Firewall (security group)**: Seleccionar *proxy-sg*
   * **Advanced details:**
     1. **IAM instance profile**: *LabInstanceProfile* (lo necesitaremos para poder administrar las instancias EC2 desde AWS Systems Manager)
     2. **User data:** Introduciremos el siguiente script:

#!/bin/bash

yum update -y

yum install squid -y

systemctl enable squid

systemctl start squid

Por último, se presiona el botón **Create Launch Template**.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

1. Ahora deberemos crear un balanceador de carga de red (NLB). Sin embargo, previamente necesitaremos un grupo de destinos donde se encontrarán asociadas nuestras instancias EC2 del clúster. Para ello, desde la consola de Amazon EC2, seleccionamos la opción **Target Groups** y presionamos el botón **Create Target Group**.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

1. En el primer paso, especificaremos los siguientes parámetros y presionamos el botón **Next**:
   * **Target type**: *Instances*
   * **Target group name**: *proxy-tg*
   * **Protocol**: TCP
   * **Port**: 3128
   * **VPC**: *mi-vpc*

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

1. Al no tener aún instancias EC2 en el clúster, la siguiente ventana la omitimos y presionamos el botón **Create Target Group**.
2. Para crear el NLB, desde la consola de Amazon EC2 seleccionamos la opción **Load Balancers** y presionamos el botón **Create Load Balancer**.

Interfaz de usuario gráfica, Texto, Correo electrónico

Descripción generada automáticamente

1. En la siguiente ventana, seleccionamos el balanceador de carga de red, presionando en el botón **Create** correspondiente.

Interfaz de usuario gráfica

Descripción generada automáticamente

1. En la ventana de configuración el NLB, configuramos los siguientes parámetros:
   * **Basic configuration**
     1. **Load balancer name**: *nlb-proxy*
     2. **Scheme**: *Internal*
     3. **IP address type**: *IPv4*

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

* + **Network mapping**
    1. **VPC**: *mi-vpc*
    2. **Mappings*:*** Se marcan las zonas de disponibilidad *us-east-1a* y *us-east-1b*, seleccionando las subredes *privada-01-proxy* y *privada-02-proxy*, respectivamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* + **Listeners and routing**
    1. **Protocol**: *TCP*
    2. **Port**:*3128*
    3. **Default action**: *Forward to proxy-tg*

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Por último, se presiona el botón **Create Load Balancer**. El NLB tardará unos pocos minutos en aprovisionarse y estar operativo.

1. A continuación, vamos a crear un grupo autoescalable de instancias EC2 que formarán el clúster de servidores proxy, a partir de la plantilla de lanzamiento creada anteriormente. Para ello, desde la consola de Amazon EC2, seleccionamos la opción **Auto Scaling Groups** y presionamos el botón **Create Auto Scaling Group**.

Interfaz de usuario gráfica

Descripción generada automáticamente

1. En el primer paso, introduciremos en el campo **Auto Scaling group name** el valor *proxy-asg* y seleccionaremos como plantilla de lanzamiento la plantilla *proxy-template* creada anteriormente. A continuación, presionamos el botón **Next**.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

1. En el paso 2, en el campo **VPC** introducimos la red etiquetada como *mi-vpc* y en el campo **Availability Zones and subnets**, seleccionamos las subredes etiquetadas como *privada-01-proxy* y *privada-02-proxy*. A continuación, presionamos el botón **Next**.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. En el paso 3, seleccionamos la opción *Attach to an existing load balancer* en el apartado **Load Balancing**. Además, seleccionamos el grupo de destinos *proxy-tg* del NLB creado en los apartados anteriores:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Además, en el apartado **Health checks** **type** marcamos la opción *ELB*. Esto nos permitirá que el grupo de autoescalado considere las comprobaciones de estado del NLB y, en caso de que una instancia EC2 no pase con éxito dichas comprobaciones, será reemplazada por una nueva. Además, también habilitaremos la obtención de métricas agregadas activando la opción *Enable group metrics collection within CloudWatch*, en el apartado **Additional settings**.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Por último, se presiona el botón **Next**

1. En el paso 4, establecemos el tamaño del grupo de autoescalado en número de instancias EC2, introduciendo los siguientes valores:
   * **Group size**
     1. **Desired capacity**: *2*
     2. **Minimum capacity**: *2*
     3. **Maximum capacity**: 6
   * **Scaling policies**: *Target tracking scaling policy*
     1. **Scaling policy name**: *Target tracking Policy*
     2. **Metric type**:*Average CPU utilization* (quizás no sea la más adecuada para este caso de uso, sería más interesante una métrica basada en tráfico de red, pero por simplicidad se define una más sencilla)
     3. **Target value**: *50*

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Por último, presionamos el botón **Next**.

1. En el paso 5, permite definir notificaciones cuando se lleve a cabo una acción de escalado o desescalado. Para simplificar esta práctica, no se tendrá en cuenta. Presionamos el botón **Next**.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

1. En el apartado 6, introduciremos una etiqueta **Name** con un valor *proxy*. Presionamos el botón **Next**.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. En el paso 7, revisamos la configuración y presionamos el botón **Create Auto Scaling Group**. En unos minutos, comprobaremos que el grupo de autoescalado ha lanzado dos instancias EC2 nuevas, una en cada zona de disponibilidad, en las subredes *privada-01-proxy* y *privada-02-proxy*.

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

**LANZAMIENTO DE INSTANCIAS EC2 CON CLIENTES HTTP/HTTPS**

1. Para simplificar el lanzamiento de clientes HTTP/HTTPS se va a utilizar instancias EC2 con Amazon Linux y se utilizará el navegador web *Lynx* basado en consola. Crearemos una plantilla de lanzamiento desde la opción **Launch templates** de la consola de Amazon EC2. Presionamos el botón **Create launch template**:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

1. Desde la ventana siguiente completamos con las siguientes opciones de configuración:
   * **Launch Template Name**: *http-client-template*
   * **Template Version Description**: *Versión inicial*
   * **Application and OS Images**:
     1. **QuickStart:** Seleccionamos *Amazon Linux*
     2. **Architecture**: *64-bit (Arm)*
   * **Instance type**
     1. **Instance type**: *t4g.microt4*
   * **Key pair (login)**
     1. **Key pair name**: *Don’t include in launch template*
   * **Network settings**:
     1. **Firewall (security group)**: Seleccionar *web-sg*
   * **Advanced details:**
     1. **IAM instance profile**: *LabInstanceProfile* (lo necesitaremos para poder administrar las instancias EC2 desde AWS Systems Manager)
     2. **User data:** Introduciremos el siguiente script:

#!/bin/bash

yum install lynx -y

NLB\_ID=$(aws elbv2 describe-load-balancers --names nlb-proxy --query "LoadBalancers[].DNSName" --output text --region us-east-1)

NLB\_NAME=${NLB\_ID: -16}

IP=$(aws ec2 describe-network-interfaces --filters Name=description,Values="ELB net/nlb-proxy/$NLB\_NAME" --query 'NetworkInterfaces[\*].PrivateIpAddresses[\*].PrivateIpAddress' --output text --region us-east-1)

AZ=$(aws ec2 describe-network-interfaces --filters Name=description,Values="ELB net/nlb-proxy/$NLB\_NAME" --query 'NetworkInterfaces[\*].AvailabilityZone' --output text --region us-east-1)

IP1=$(echo $IP|cut -d " " -f 1)

IP2=$(echo $IP|cut -d " " -f 2)

AZ1=$(echo $AZ|cut -d " " -f 1)

AZ2=$(echo $AZ|cut -d " " -f 2)

AZEC2=$(curl http://169.254.169.254/latest/meta-data/placement/availability-zone)

if [ "$AZEC2" = "$AZ1" ]; then

IP=$IP1

else

IP=$IP2

fi

echo -e "http\_proxy=http://$IP:3128\nhttps\_proxy=http://$IP:3128" > /etc/environment

reboot

El script anterior, instala el navegador *Lynx* y configura el archivo */etc/environment* con las interfaces correspondientes del NLB, en función de la zona de disponibilidad donde se vaya a lanzar la instancia EC2.

Por último, se presiona el botón **Create Launch Template**.

1. A continuación, lanzaremos un cliente HTTP/HTTPS en una de las subredes habilitadas para ello, desde la opción **Launch Templates** de la consola de Amazon EC2, seleccionamos la plantilla *http-client-template* y presionamos el botón **Actions** y activamos la opción **Launch instance from template**.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

1. En el siguiente formulario dejamos todas las opciones por defecto, excepto:
   * **Key pair (login)**: *Proceed without a key (Not recommended*)
   * **Network settings**
     1. **Subnet**: Seleccionamos *privada-01-clientes* o *privada­02-clientes*

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Por último, presionamos el botón **Launch instance**

1. Tras esperar de 2 a 3 minutos, nuestra instancia EC2 configurada ya estará preparada. Para poder acceder a ella de forma sencilla y segura, vamos a utilizar la característica *Session Manager* del servicio AWS Systems Manager. Para ello, vamos a la consola de AWS Systems Manager (AWS SSM) y seleccionamos la opción **Session Manager**.

Interfaz de usuario gráfica

Descripción generada automáticamente

1. Desde la siguiente ventana, presionamos el botón **Start session.**

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

1. En la siguiente ventana, podremos ver todas nuestras instancias EC2 que están siendo administradas por AWS SSM. Seleccionamos la instancia EC2 con el cliente HTTP/HTTPS y presionamos el botón **Start session**:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

1. A continuación, se nos abrirá una ventana con un *Shell* desde la que podremos trabajar con la instancia cliente. Para comprobar que se ha configurado automáticamente el proxy definiendo como punto de enlace la IP del NLB correspondiente, visualizamos el archivo */etc/environment*:

Forma, Rectángulo

Descripción generada automáticamente

1. Para comprobar que la navegación funciona correctamente podemos ejecutar el comando

$ lynx <https://www.iestetuan.es>

Podemos comprobar que la página web se carga correctamente:

Texto

Descripción generada automáticamente

1. Por último, y para comprobar que el tráfico está pasando por la flota de servidores proxy, abrimos una sesión de consola de AWS SSM contra una de las dos instancias EC2 proxy (con un poco de suerte acertamos a cuál la ha enviado el NLB) y ejecutamos el comando:

$ sudo cat /var/log/squid/acces.log

Podremos ver que efectivamente, las solicitudes pasan a través de una instancia EC2 del clúster con el código HTTP 200, indicando que todo ha ido correctamente:

Texto

Descripción generada automáticamente

**Limpieza de la Práctica:**

Para eliminar la práctica y que no consuma recursos de AWS, debemos:

1. Eliminar el NLB, desde la consola de Amazon EC2
2. Eliminar los NAT Gateway, desde la consola de Amazon VPC
3. Liberar las direcciones IP Elásticas, desde la consola de Amazon VPC
4. Eliminar el grupo de autoescalado, desde la consola de Amazon EC2 (con esto se eliminarán todas las instancias del clúster del servicio proxy)
5. Terminar la instancia EC2 que actúa como cliente HTTP/HTTPS

El resto de recursos creados pueden también eliminarse, pero no suponen ningún coste económico adicional.

Recuerda también que el laboratorio de AWS Academy debe cerrarse presionando el botón *End Lab* desde la plataforma.