

ALTA DISPONIBILIDAD MEDIANTE LOAD BALANCER, GRUPOS DE DESTINO Y AUTOESCALADO EN AWS

En esta práctica prepararemos el despliegue de un servidor web para que detecte automáticamente que es necesario aumentar la capacidad del servidor y cree nuevas réplicas si es necesario. Cuando la necesidad de rendimiento del sistema baje automáticamente se eliminarán réplicas de servidores volviendo a la configuración inicial de servidores. Mediante este sistema evitaremos sobre dimensionar el sistema y su rendimiento ira en acorde con las necesidades obteniendo una buena respuesta y adaptando los costes. Esta características se denomina escalabilidad elástica. Para obtener esta característica utilizaremos balanceadores de carga junto con grupos de escalabilidad.

Además de mejorar el rendimiento también ofreceremos un servicio web de alta disponibilidad ubicando servidores. Para la alta disponibilidad crearemos grupos de autoescalado en zonas de disponibilidad diferentes.

En esta práctica podéis utilizar el wordpress junto con RDS o un sencillo servidor web de prueba simplemente. Eso sí, recordar que si queremos ofrecer este tipo de sistema de alta disponibilidad la base de datos también debe de ir replicada mediante un cluster independiente a los servidores WEB, para ello el servicio RDS es perfecto debido a que ofrece la posibilidad de despliegue de réplicas en varias zonas de disponibilidad AZ.

Desplegando un servidor web con balanceador de carga mejorando la disponibilidad

0- En este apartado crearemos 4 servidores idénticos, réplicas, en dos subredes públicas diferentes y en zonas de disponibilidad diferentes . La idea es que por ejemplo fueran 4 wordpress con la misma base de datos RDS. Para simplificar utilizaremos un pequeño servidor web con una única página.

1- En este paso crearemos una máquina que utilizaremos como base en la práctica. Si quereis tambien podeis utilizar el AMI del wordpress. Lo que prefiráis.

2- Crear una máquina EC2 con AMI de AMAZON LINUX en una subred pública con el siguiente USER_DATA y con IP pública , simplemente para probar la máquina. No olvideis abrir el puerto 80.

```
#!/bin/sh
yum -y install httpd php
chkconfig httpd on
systemctl start httpd.service
cd /var/www/html
wget https://practica-s3-macc.s3.amazonaws.com/fotos/index.php
```

3- 'index.php' es una sencilla página que indica la IP del servidor en el cual se ha ejecutado la página php. Aquí teneis el código

```
<html>
<body>

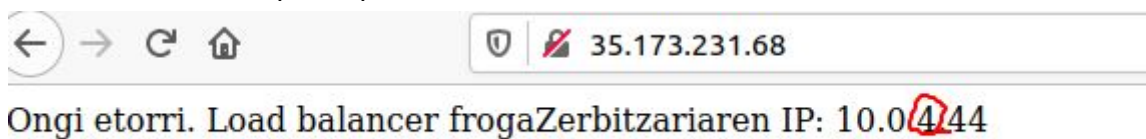
<?php
    echo "Ongi etorri. Load balancer froga. ";
    $localIP = getHostByName(getHostName());
    echo "Zerbitzariaren IP: $localIP";
?>

</body>
</html>
```

4- Una vez arrancada la página probar que funciona.



5- Ahora de momento crearemos uno por uno otras 3 máquinas iguales, la idea es desplegar 2 máquinas en una subred y otras 2 en otra subred. De forma que podamos ofrecer un servicio con alta disponibilidad (servicios replicas en diferentes AZ) y escalado horizontal con dos réplicas por subred.



6- Ahora el objetivo es poner un Load balancer de AWS que distribuya el tráfico a los diferentes servidores. El servicio de load balancer de AWS que se ofrece en una región es un cluster de servidores distribuido por las zonas de disponibilidad. Los Load balancer reparten el tráfico dentro de las máquinas que conforman un Target group. Así que creamos un Target Group con nuestras máquinas que representan un mismo servicio. Se pueden crear tres tipos de target groups: 1) por instancias 2) Por Ips 3) para Servicios Lambda

serverless de AWS. En nuestro caso utilizaremos las instancias , así ofreceremos alta disponibilidad utilizando dos subredes ;-). Al crear el Target Group por ejemplo seleccionar el nombre 'Servicio1', elegir la VPC correspondiente y el protocolo Http1, y podéis dejar las demás opciones por defecto. Es interesante ver las opciones de healthcheck que después permitirá al load balancer no utilizar los servidores que no responden. Finalmente seleccionar las instancias que quereis que conformen el Target Group. Seleccionarlas e incluirlas. Y ya tenemos el servicio creado ;-)

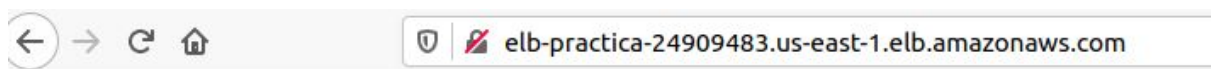
7- Ahora el siguiente paso es crear un load balancer que redirija el tráfico al cluster del servicio creado (Target Group). Para ello crear el ELB (Elastic Load Balancer). Se pueden crear 4 tipos de ELB: 1) Aplicaciones HTTP 2) Balanceador a nivel de red 3) Balanceador clásico (instancias) 4) Balanceador Gateway. En nuestro caso crearemos un balanceador de carga de aplicación el cual utilizaremos para reenviar al Target Group todas las peticiones HTTP recibidas. Este tipo de load balancer permite también definir reglas de redireccionamiento con las cuales podemos redirigir PATH a Target Groups , es decir Path por servicios ;-)

Al crear el ELB indicareis la VPC y las dos subredes a utilizar. Después os pedirá que indiquéis el grupo de servidores , es decir el Target Group, y automáticamente detectará vuestros servidores. Y ya tendréis el ELB creado.

8- Una vez creado el ELB , AWS os dará un nombre de dominio el cual podéis utilizar para probar el Load balancer. Realizar varias peticiones y veréis como cambia la IP del servidor.



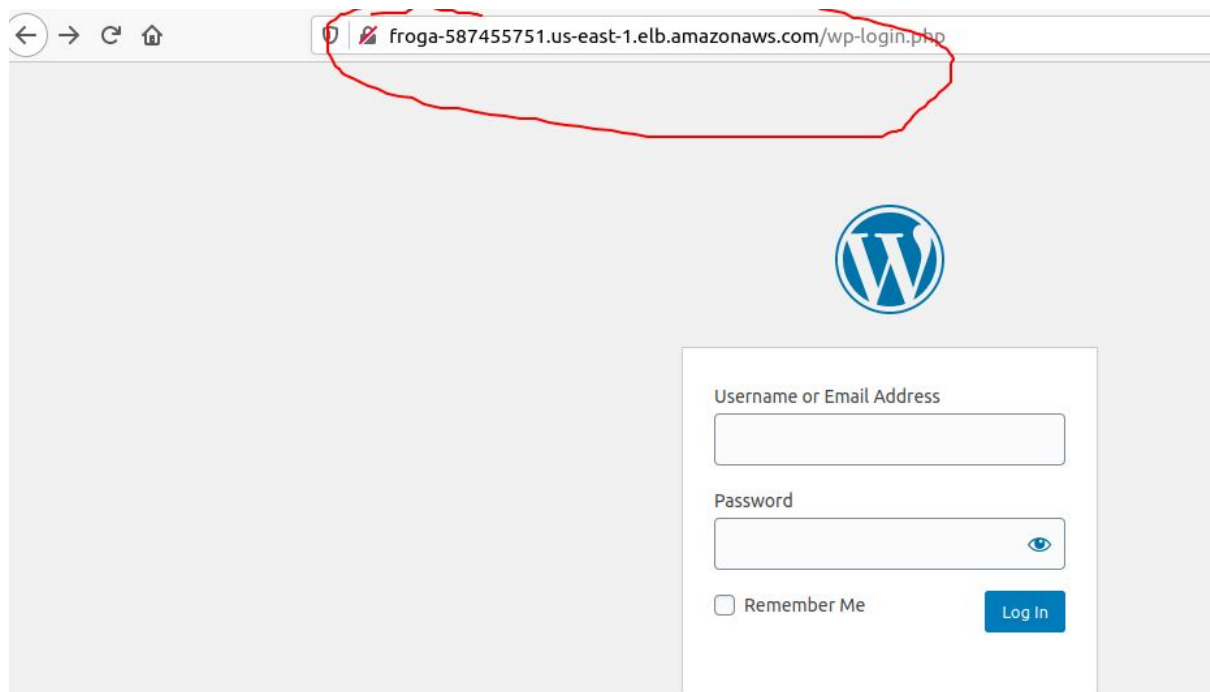
Ongi etorri. Load balancer frogazarbitzariaren IP: 10.0.4.44



Ongi etorri. Load balancer frogazarbitzariaren IP: 10.0.0.85

9- Otra característica interesante de los ELB de Aplicación es que se pueden crear reglas de enrutado a diferentes Target groups. Por ejemplo en el siguiente ejemplo tras crear un Target Group con un WP hemos creado una regla para que todo lo que comience con /wp* se redirige a otro grupo de servidores. Esto hacerlo con wordpress no tiene mucho sentido , pero en un sistema de microservicios es perfecto para crear réplicas de servicios ;-)

Froga HTTP:80 (2 rules)		
▶ Rule limits for condition values, wildcards, and total rules.		
1	arn...600a2	<div> <div>IF</div> <div>✓ Path is /wp*</div> </div> <div> <div>THEN</div> <div>Forward to</div> <div>wp: 100 (100%)</div> <div>Group-level stickiness: 1 hour (3600 seconds)</div> </div>
last	HTTP 80: default action <i>This rule cannot be moved or deleted</i>	<div> <div>IF</div> <div>✓ Requests otherwise not routed</div> </div> <div> <div>THEN</div> <div>Forward to</div> <div>Servicio1: 1 (100%)</div> <div>Group-level stickiness: Off</div> </div>



10- Al utilizar ELB no tiene sentido utilizar máquinas con IP públicas. Lo normal es poner las máquinas que conforman los target groups con IP privadas y despues poner una entrada DNS que redirija el tráfico de un dominio al load balancer;-) Aquí lo hemos hecho con IPs publicas únicamente con un objetivo didáctico y de facilitar la práctica.

Autoescalando un servicio en función de la carga de trabajo

11- En esta parte de la práctica especificaremos una configuración de lanzamiento (una AMI base para crear un Target Group Elástico) y después mediante una configuración de autoescalado se definirá cómo se irán creando y destruyendo máquinas de ese target group en base al tráfico;-)

12- Primero creamos una configuración de lanzamiento de nuestro servicio. Por ejemplo si quisiéramos desplegar una web basada en wordpress la configuración de lanzamiento sería la AMI de nuestro servidor. En este caso la configuración de lanzamiento será la misma que las 4 máquinas de la parte anterior. Crear la configuración de lanzamiento utilizando el AMI de Amazon (ami-04d29b6f966df1537) y el siguiente USER_DATA. También indicar que no hace falta utilizar ip pública y abrir el puerto 80 (Por un error en la consola web , mejor seleccionar un SG ya existente que se adapte a nuestras necesidades). seleccionar un tag para que después en las instancias podamos saber cuál es la instancia creada por defecto.

```
#!/bin/sh
yum -y install httpd php
chkconfig httpd on
systemctl start httpd.service
cd /var/www/html
wget https://practica-s3-macc.s3.amazonaws.com/fotos/index.php
```

13- Ahora crearemos un grupo de autoescalado que cree máquinas del tipo que acabamos de definir. Para ello tenemos varias opciones: ir al menu de grupo de autoescalado o desde la misma configuración de lanzamiento en acciones tenemos la posibilidad de crear el grupo de autoescalado directamente. Seleccionamos el grupo de lanzamiento y elegimos crear grupo de autoescalado. Le damos un nombre y elegimos el VPC y las subredes que queramos (mejor de dos zonas de disponibilidad diferentes y privadas , no olvidéis el NAT;-) Para este grupo y esta práctica crearemos un nuevo load balancer para facilitar la comprensión, pero podríamos utilizar uno ya existente. Al crear el nuevo balanceador indicareis las mismas características que en la práctica anterior, no olvidéis hacer público el ELB indicando Internet-facing y ubicar el load balancer en las subredes públicas (*Recordad ubicar el grupo de autoescalado en las subredes privada y el load balancer en la pública. Esto depende de la arquitectura que busquéis, en esta práctica queremos hacer publica los servidores privados poniendo en la red pública el load balancer. Esto permite incluso utilizar WAF en el load balancer y securizar más los servidores. ¿Que es un WAF de AWS?*).


Después os pedirá el número mínimo y máximo de servidores que queréis (en Educate como máximo solo podéis indicar 3) .Configurar, por ejemplo, como mínimo 1 y como máximo 2. Después crear una regla que cree máquinas nuevas cuando las CPUs están al

40%, y después si queréis crear alarmas que os indicarán que se está realizando un escalado o desescalado en el target group. Y le damos a crear.

14- Ahora comprobar que el sistema funciona correctamente. Para ello chequear primero en el grupo de autoescalado cuantas instancias están preparadas. También comprobar en el menú de EC2 que se han creado las nuevas máquinas automáticamente. Podéis probar a terminar una máquina y se creará otra automáticamente. A continuación ir al menú de load balancer obtener el nombre del nuevo ELB y probar que funciona.

<input type="checkbox"/>	bastion	i-02c3558456c0e6e26	✓ En ejecu...	t2.micro	✓ 2/2 compro...	Sin alar...	+
<input type="checkbox"/>	NatFroga	i-0f0f5696afd8fd23a	⊖ Detenida	t2.micro	–	Sin alar...	+
<input type="checkbox"/>	wordpress	i-0e140b66689741a79	⊖ Detenida	t2.micro	–	Sin alar...	+
<input type="checkbox"/>	wordpress-re...	i-0f404ce37879ceb56	✓ En ejecu...	t2.micro	✓ 2/2 compro...	Sin alar...	+
<input type="checkbox"/>	elb	i-0643fe7c932776153	✓ En ejecu...	t2.micro	✓ 2/2 compro...	Sin alar...	+
<input type="checkbox"/>	–	i-03dec0f4d0dabf33a	⊖ Terminada	t2.micro	–	Sin alar...	+
<input type="checkbox"/>	–	i-0d47cbab7c10a495e	✓ En ejecu...	t2.micro	✓ 2/2 compro...	Sin alar...	+
<input type="checkbox"/>	bigarrena	i-021590e3b8b054f7a	✓ En ejecu...	t2.micro	✓ 2/2 compro...	Sin alar...	+

Seleccione una instancia anterior

← → ↺ 🏠  adibidea-privado-1-816052168.us-east-1.elb.amazonaws.com

Ongi etorri. Load balancer frogaZerbitzariaren IP: 10.0.3.150

15- Por último probaremos el autoescalado y desescalado del sistema mediante JMETTER.