Geolocalizador de IPs

Cloud Computing

Ramón Palacios Rodríguez
José Manuel Lozano Dominguez

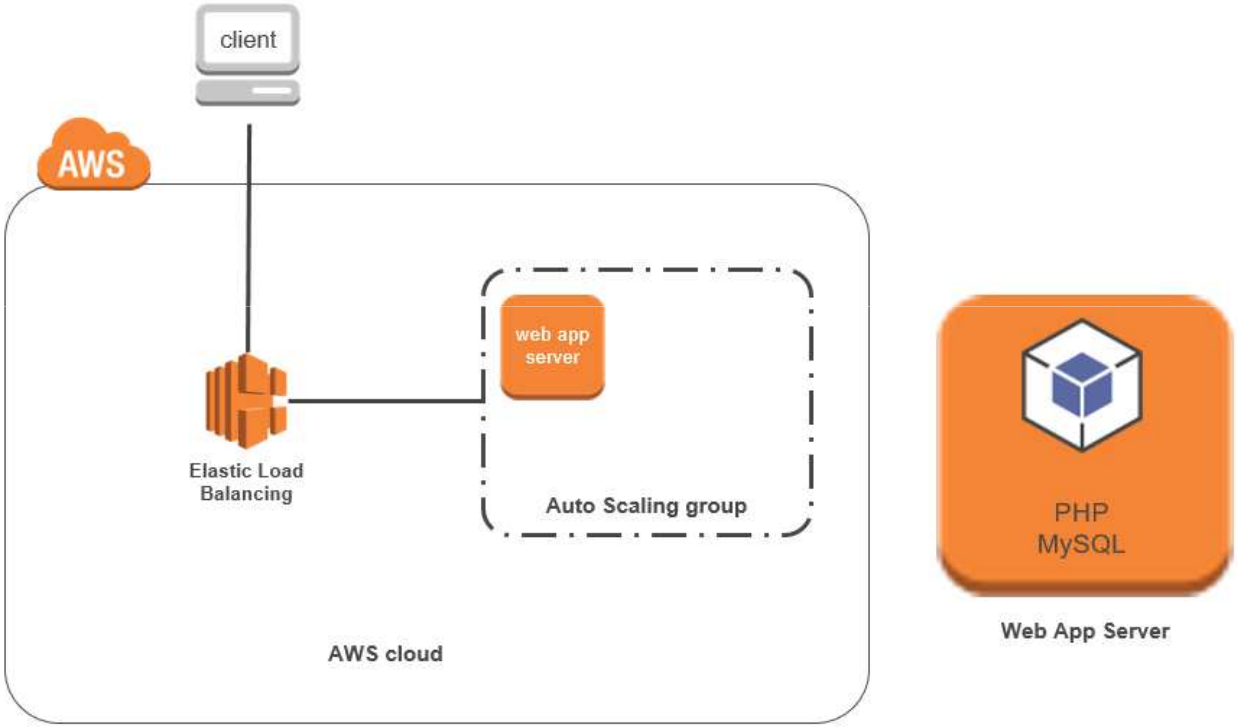
Introducción

- El desarrollo de esta práctica consiste en realizar un **servicio de Geolocalización de IPs auto escalable de alta disponibilidad**.

- Este servicio debe recibir por URL el formato en el que quiere recibir los datos, y la dirección IP. Los formatos que debe aceptar son CSV, XML, y Json. El servicio debe usar elementos de Amazon Web Services (AWS).

- Este servicio hará uso de elementos de Amazon Web Services (AWS), como instancias y Elastic Block Store (EBS) entre otros. Como gestor de base de datos se ha usado phpmyadmin, el cual almacena los datos relacionados con las direcciones IPs.

Infraestructura desarrollada



Instancias, Balanceador de Carga y Grupo de auto escalado

- Las instancias son t2.micro tanto para servidores como para los clientes
- El balanceador de carga junto con el grupo de auto escalado son los encargados de ajustar el número de instancias activas en función de la carga de los servidores
- Para optimizar el funcionamiento del balanceador de carga es necesario activar la opción *"Health Check"* y configurar la ruta del servicio ofrecido
- Para que el grupo de auto escalado funcione correctamente se ofrezca de manera adecuada las instancias se lanzan desde una AMI previamente configurada

Diseño del servidor Web

- Se usará una distribución *Amazon Linux AMI* con la siguiente configuración
  - Gestor de base de datos: phpMyAdmin
  - Base de datos: realacional MySQL
  - Servidor Web: Apache

Diseño del servidor Web (II)

- Una vez han sido instaladas las herramientas anteriores en el servidor se procede a configurar a las mismas

- La base de datos se cargo mediante un fichero .csv. Dicho fichero se tuvo que cargar a partir de la consola propia de MySQL debido al gran tamaño del fichero

- El servicio que ofrece el servidor se encuentra en el fichero *index.php*

Diseño de los clientes

- Son instancias del mismo tipo que el servidor, con la salvedad de la configuración.

- En este caso se utiliza la herramienta *Siege,* dicha herramienta permite medir la capacidad de una aplicación web para mantenerse activa, y mantener sus servicios.

- Se utilizaron varias instancias clientes para comprobar el correcto funcionamiento del servicio

Toma de decisiones

- Para seleccionar el tipo de instancia se opto por el de menor precio posible para abaratar costes. En este caso se usará t2.micro
- Se opto por el lenguaje PHP para la programación web debido a que es el más conocido por los componentes del grupo
- Se opto por una base de datos relacional debido a que era necesaria una base de datos con muchas lecturas y pocas escrituras. Por ello se opto por MySQL y phpMyAdmin como gestor.
- También se instaló el framework SLIM para la creación de API REST en PHP pero por motivos de tiempo no se termino de desarrollar la API REST.

Problemas encontrados

- Creación de la BD a partir de un fichero .csv con un gran número de registros. Se solucionó con la importación del fichero .csv a través de la consola de MySQL

- Balanceador de carga no detectaba el fallo al conectar el servidor Web con la BD. Se solucionó con línea que enviará el código 500 HTTP. Además se estableció como ruta de *"Health Check"* con la ruta del fichero *index.php*

- Fallos al crear o eliminar instancias. Se debía a la necesidad de establecer un correo y validar el correo en las alarmas asociadas al crear o eliminar una instancia.

Problemas encontrados

- Pruebas de esfuerzo Siege. Provocaba un gran número de fallos de segmentación de memoria. Se solucionó con un mayor número de clientes y con un menor número de peticiones concurrentes

- Limitación en el número de peticiones concurrentes en MySQL. Se resuelve modificando un parámetro. Actualmente está limitado a 150 peticiones concurrentes.

- Tiempo necesario para conocer como implementar una API REST con SLIM en PHP

- Retraso en el estado de las instancias . En el modo administrador no se observa el estado de la instancia en tiempo real

Pruebas de esfuerzo

root@ip-172-31-0-33:~/siege-4.0.2/bin

```
^C
Lifting the server siege...
Transactions:                  503188 hits
Availability:                   97.35 %
Elapsed time:                  899.06 secs
Data transferred:               47.06 MB
Response time:                   0.01 secs
Transaction rate:              559.68 trans/sec
Throughput:                      0.05 MB/sec
Concurrency:                     6.10
Successful transactions:       503188
Failed transactions:            13685
Longest transaction:             9.00
Shortest transaction:            0.00

[root@ip-172-31-0-33 bin]#
```

root@ip-172-31-0-29:~/siege-4.0.2/bin

```
^C
Lifting the server siege...
Transactions:                   76036 hits
Availability:                  100.00 %
Elapsed time:                  199.09 secs
Data transferred:                7.09 MB
Response time:                   0.01 secs
Transaction rate:              381.92 trans/sec
Throughput:                      0.04 MB/sec
Concurrency:                     4.63
Successful transactions:        76036
Failed transactions:                0
Longest transaction:             8.14
Shortest transaction:            0.00

[root@ip-172-31-0-29 bin]#
```

root@ip-172-31-0-31:~/siege-4.0.2/bin

```
^C
Lifting the server siege...
Transactions:                  170901 hits
Availability:                  100.00 %
Elapsed time:                  295.97 secs
Data transferred:               15.92 MB
Response time:                   0.01 secs
Transaction rate:              577.43 trans/sec
Throughput:                      0.05 MB/sec
Concurrency:                     5.27
Successful transactions:       170901
Failed transactions:                0
Longest transaction:             8.99
Shortest transaction:            0.00
```

root@ip-172-31-0-30:~/siege-4.0.2/bin

```
^C
Lifting the server siege...
Transactions:                   71934 hits
Availability:                  100.00 %
Elapsed time:                  189.20 secs
Data transferred:                6.70 MB
Response time:                   0.01 secs
Transaction rate:              380.20 trans/sec
Throughput:                      0.04 MB/sec
Concurrency:                     4.80
Successful transactions:        71934
Failed transactions:                1
Longest transaction:            10.00
Shortest transaction:            0.00

[root@ip-172-31-0-30 bin]#
```

root@ip-172-31-0-32:~/siege-4.0.2/bin

```
^C
Lifting the server siege...
Transactions:                  468437 hits
Availability:                   99.53 %
Elapsed time:                  816.61 secs
Data transferred:               43.65 MB
Response time:                   0.01 secs
Transaction rate:              573.64 trans/sec
Throughput:                      0.05 MB/sec
Concurrency:                     5.55
Successful transactions:       468437
Failed transactions:             2227
Longest transaction:             9.39
Shortest transaction:            0.00

[root@ip-172-31-0-32 bin]#
```

Pruebas de esfuerzo (II)

rmjm-Server-aeg	i-046aa76f6e757ecef	t2.micro	eu-central-1a	running	2/2 checks ...	None	ec2-52-59-214-187.eu-c...	52.59.214.187	ramon_cc16	enabled	Decemb
rmjm-Server-aeg	i-04f72ae95d6f1e583	t2.micro	eu-central-1b	running	2/2 checks ...	None	ec2-54-93-95-84.eu-cen...	54.93.95.84	ramon_cc16	enabled	Decemb
rmjm-Server-aeg	i-0834d56755082810b	t2.micro	eu-central-1a	running	2/2 checks ...	None	ec2-54-93-62-211.eu-ce...	54.93.62.211	ramon_cc16	enabled	Decemb
rmjm-Server-aeg	i-0f1637584537dc630	t2.micro	eu-central-1b	running	2/2 checks ...	None	ec2-54-93-211-48.eu-ce...	54.93.211.48	ramon_cc16	enabled	Decemb
server-rmjm	i-086fc0e894d7bb9e2	t2.micro	eu-central-1b	running	2/2 checks ...	None	ec2-52-59-216-12.eu-ce...	52.59.216.12	ramon_cc16	disabled	Decemb

Description | Status Checks | **Monitoring** | Tags

CloudWatch alarms: No alarms configured — Create Alarm

CloudWatch metrics: Showing data for: Last Hour

Below are your CloudWatch metrics for the selected resources (a maximum of 10). Click on a graph to see an expanded view. All times shown are in UTC. › View all CloudWatch metrics

CPU Utilization (Percent) — 100, 75, 50, 25, 0 — 12/15 16:00 — 12/15 16:30

Disk Reads (Bytes) — 1, 0.75, 0.5, 0.25, 0 — 12/15 16:00 — 12/15 16:30

Disk Read Operations (Operations) — 1, 0.75, 0.5, 0.25, 0 — 12/15 16:00 — 12/15 16:30

Disk Writes (Bytes) — 1, 0.75, 0.5, 0.25, 0 — 12/15 16:00 — 12/15 16:30

Disk Write Operations (Operations) — 1, 0.75, 0.5, 0.25, 0 — 12/15 16:00 — 12/15 16:30

Pruebas de esfuerzo (III)

Auto Scaling Group: rmjm-aeg

Details | Activity History | Scaling Policies | **Instances** | Monitoring | Notifications | Tags | Scheduled Actions

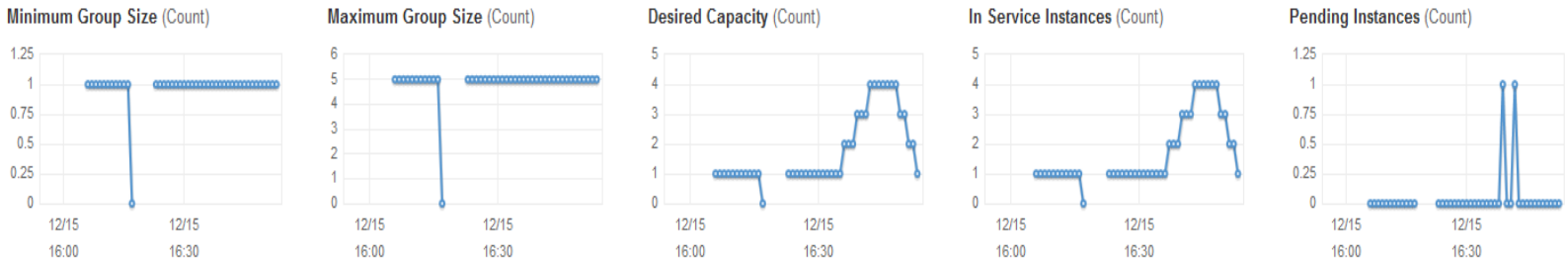
Actions

Filter: Any Health Status | Any Lifecycle State | Filter instances...

Instance ID	Lifecycle	Launch Configuration Name	Availability Zone	Health Status
i-046aa76f6e757ecef	Terminating	rmjm-lc	eu-central-1a	Healthy
i-04f72ae95d6f1e583	InService	rmjm-lc	eu-central-1b	Healthy
i-0834d56755082810b	InService	rmjm-lc	eu-central-1a	Healthy
i-0f1637584537dc630	InService	rmjm-lc	eu-central-1b	Healthy

Below are your CloudWatch metrics for the selected resources (a maximum of 10). Click on a graph to see an expanded view. All times shown are in UTC. › View all CloudWatch metrics

rmjm-aeg



Conclusiones

- Se ha descubierto la posibilidad de tener máquinas alojadas en cualquier lugar del planeta con las cuales se puede trabajar para hacer llegar un servicio a la población global

- La práctica desarrollada ha permitido obtener un gran conocimiento sobre cómo funciona AWS y cómo poder desplegar una aplicación en su ecosistema. Al finalizar la práctica se ha conseguido un buen grado de manejabilidad y compresión de los sistemas utilizados de AWS.

- Los datos obtenidos en las pruebas de estrés confirman que el sistema soporta hasta 1000 peticiones concurrentes, siendo el tiempo medio de respuesta de 0.01 segundos y la petición más lenta de 10 segundos. Con ello se observa que el servicio en su peor momento ha sufrido un retraso de 10 segundos. También se ha obtenido una media de 494.57 transacciones/segundo y una disponibilidad del 99.47%.

Posibles mejoras

- El servicio tenido en cuenta genera una base de datos no es modificable. Por tanto, se ha seguido un modelo en el que las instancias tengan cargadas la base de datos localmente, y no requieran de un servicio externo. Si los registros subrieran muchas escrituras sería mejor opción una base de datos tipo RDS de AWS

- Por otro lado, si lo que se desea es un menor tiempo de respuesta, se debería optar por un diseño con una base de datos no relacional como MongoDB.

- Otra posible mejora, sería el diseño del servicio como una API REST como se pedía en la parte opcional para realizar la práctica. Esto permitiría incorporar este servicio a cualquier aplicación que responda al servicio HTTP

Referencias

- https://www.youtube.com/watch?list=PL_d-XKRO_5G82DPjw4r5xEbdhVngzC9Ju&v=sNkAgCdBxo4
- https://aws.amazon.com/es/documentation/
- https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-ug.pdf
- http://howto.gandasoftwarefactory.com/desarrollo-software/2014/como-instalar-phpmyadmin-ubuntu-linux-20140929/
- http://lite.ip2location.com/database/ip-country-region-city-latitude-longitude-zipcode-timezone
- http://www.konnichiwamundo.com/2013/08/importar-un-fichero-csv-grande-a-mysql.html
- https://www.youtube.com/watch?v=V0J2ULF0A-c&list=PL_d-XKRO_5G82DPjw4r5xEbdhVngzC9Ju&index=2
- http://helloit.es/2011/07/limite-de-conexiones-en-mysql/