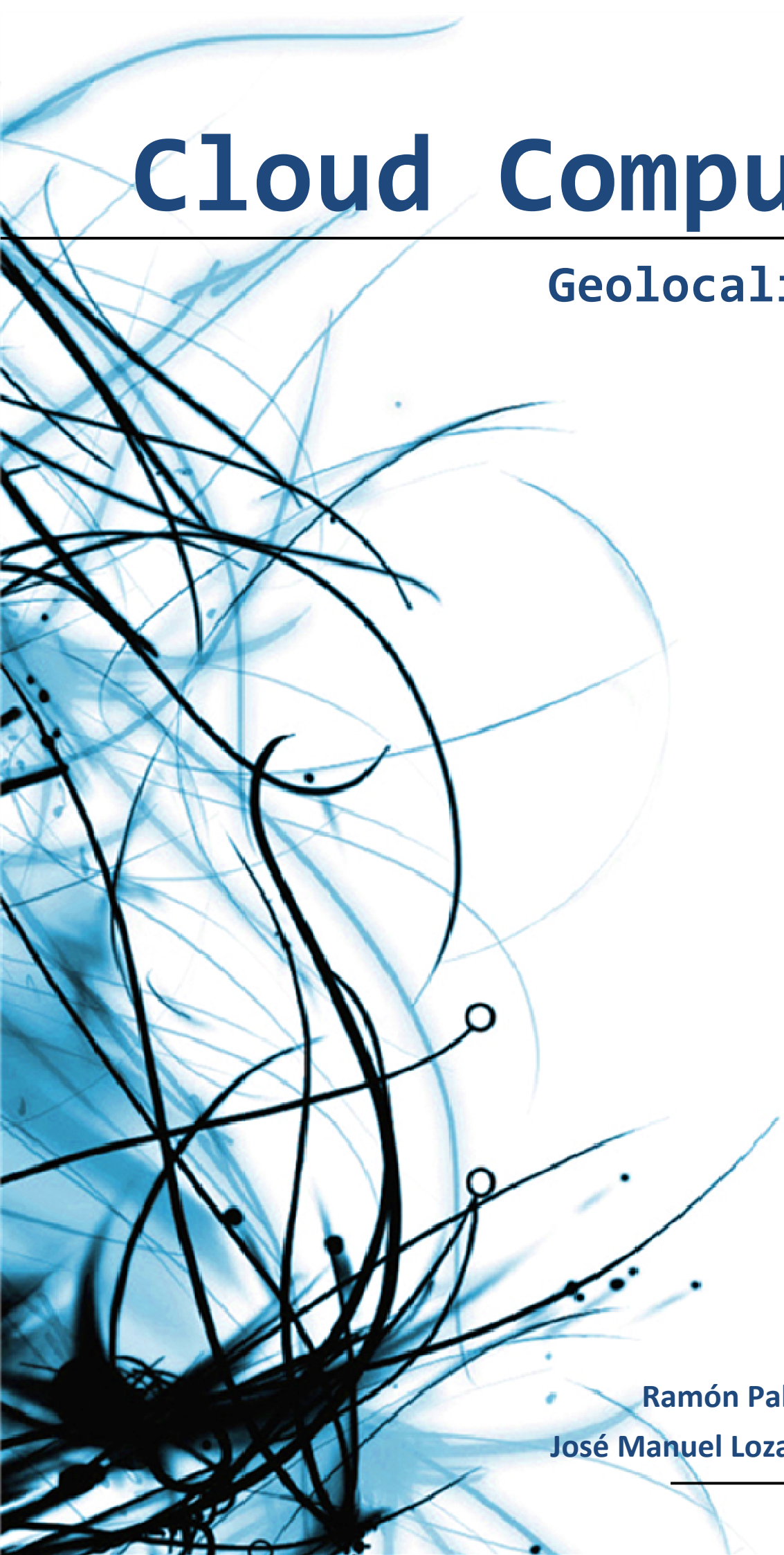
Cloud Computing

Geolocalizador de IPs



Ramón Palacios Rodríguez

José Manuel Lozano Domínguez

CURSO 2016-2017

Contenido

1. Introducción

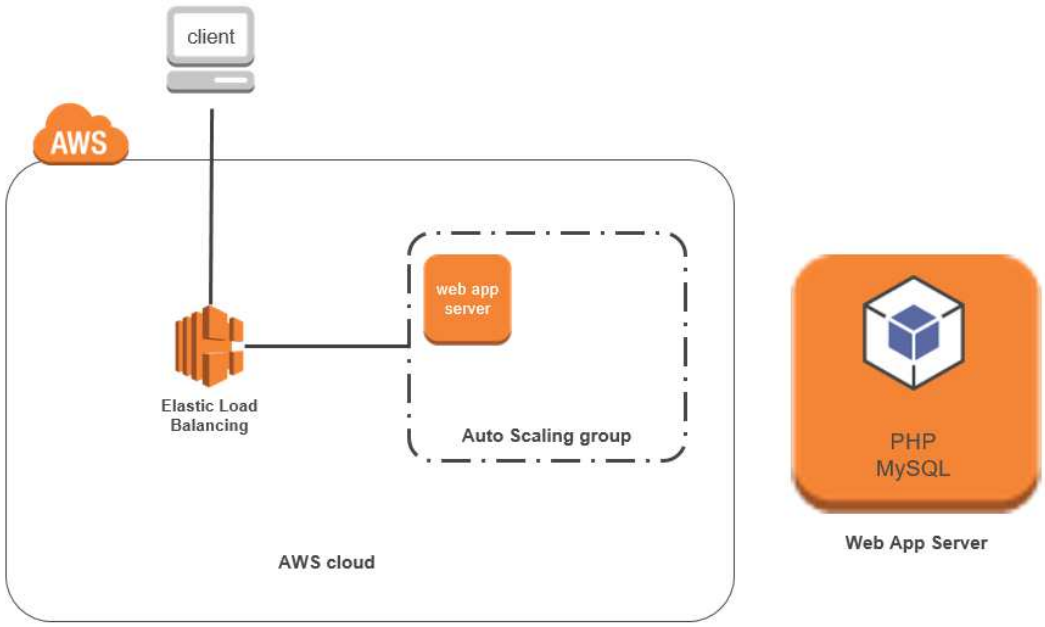
El desarrollo de esta práctica consiste en realizar un**servicio de Geolocalización de IPs auto escalable de alta disponibilidad**. Este servicio debe recibir por URL el formato en el que quiere recibir los datos, y la dirección IP. Los formatos que debe aceptar son CSV, XML, y Json. El servicio debe 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.

2. Infraestructura desarrollada

Para el desarrollo del servicio se ha usado la tecnología facilitada por Amazon Web Services (AWS), como son Elastic Block Store (EBS), instancias, balanceadores de cargas y grupos de autoescalado.

El servicio requiere una base de datos donde almacenar los datos sobre las direcciones IPs. Dichos datos son IP, código de país, nombre del país, región, ciudad, latitud, longitud, código postal y zona horaria. Para ello se usa como gestor de base de datos phpmyadmin con una base de datos relacional MySQL, y Apache como servidor de Web.



2.1. Instancias

Las instancias creadas en AWS son de tipo t2.micro puesto que es el tipo que permite la cuenta que se está usando para el desarrollo de las prácticas, y que no tienen ningún coste.

Las características de estas instancias son las siguientes:

- 1 CPU virtual – Procesador Intel Xeon de alta frecuencia
- 1 Gb RAM
- Almacenamiento EBS 8 Gb – Amazon EBS General Purpose SSD (gp2)

Para crear correctamente las instancias, se requiere haber configurado previamente el grupo de seguridad (*Security Group*). Esto permite configurar un firewall virtual, con ello conseguimos habilitar los puertos necesarios para el servicio que se quiera ofrecer.

2.2. Balanceador de Carga, Auto escalado

Para permitir que el servicio tenga la característica de auto escalado, se debe definir el balanceador de carga, y el auto escalado. Esto permitirá al servicio ir añadiendo o eliminando instancias en función de la carga que tenga el/los servidor/es. Para ello se puede seguir el video tutorial proporcionado en la sección de referencias donde se explica detalladamente los pasos a seguir.

Para mejorar el funcionamiento el balanceador de carga, hay que configurar la opción de "*Health Check*". Esto permite deshabilitar instancias que dejen de dar el servicio correctamente mediante pings. Si el servicio excede un intervalo de tiempo con respuestas diferentes al código 200 HTTP, el balanceador de carga detiene la instancia hasta que vuelva a enviar códigos 200 HTTP, y redirige las peticiones hacia el servicio a otras instancias que funcionen correctamente. Al configurar esta opción es interesante colocar la ruta del servicio ofrecido, consiguiendo de este modo además de conocer que la máquina está activa también lo está el servicio.

En el segundo paso del tutorial se crea el "*Launch Configuration*", para ello se debe haber creado previamente una imagen de la instancia preparada para funcionar como "servidor". Esta imagen (AMI) será usada para crear las instancias del grupo de auto escalado que son las instancias que se irán lanzando y deteniendo en función de la necesidad para dar el servicio.

3. Diseño del Servidor Web

Las características hardware de la instancia son las proporcionadas en el apartado Instancias.

La distribución de esta instancia es "*Amazon Linux AMI*" puesto que está disponible de manera gratuita en AWS (durante el primer año a partir del registro del usuario).

A esta instancia se le instaló como gestor de base de datos "*phpMyAdmin*", basado en base de datos "*MySQL*", y como servidor web "*Apache*". Para poder usar el script que se proporciona en esta documentación para el servicio web requiere una versión de PHP 5.3 o superior. Se optó por esta configuración puesto que es la que actualmente conocemos los integrantes del grupo. En el apartado de posibles mejoras se tratan posibles mejoras sobre la configuración del servicio.

Para poder realizar la instalación de los servicios de esta instancia sobre una distribución Linux puede seguirse el tutorial facilitado en el apartado de Instalación LAMP y phpMyAdmin.

Tras haber realizado la instalación de todos los servicios correctamente, puesto que es una aplicación web lo que se quiere realizar, se crea el fichero "*index.php*" con el código proporcionado en el Anexo 1. Este código contiene el funcionamiento del servicio *Geolocalizador de IPs*.

Para la correcta ejecución del código propuesto (Anexo 1) se ha tenido que realizar una comprobación sobre el estado en que se encuentra la base de datos. En caso de que no funcionase como debiera la base de datos, se enviaráel código 500 HTTP para indicar que el servicio no se encuentra disponible. Esto es necesario para que el balanceador de carga detecte esta instancia como fuera de servicio, así puede detenerla hasta que funcione correctamente y redirige el tráfico hacia otra instancia para continuar dando servicio.

El servicio requiere una base de datos donde se almacenen los datos proporcionados. Para cargar la base de datos se usa el enlace proporcionado(permite la descarga con formato *.csv)* en el apartado Base de Datos de IPs. En dicho apartado se tiene el código necesario para importar el fichero. Para importar este fichero se debe acceder mediante comandos a la base de datos MySQL en phpMyAdmin, tras esto, se debe ejecutar el código proporcionado. Para poder importar la base de datos se debe seguir el procedimiento indicado en importación de .CSV con gran cantidad de datos en MySQL.

Tras la instalación de phpMyAdmin es recomendable cambiar el usuario y contraseña de la cuenta root para hacer login. Esto puede hacerse siguiendo el tutorial de cambiar cuenta root phpMyAdmin.

4. Diseño de los Clientes

Para realizar pruebas de esfuerzo, se crean instancias con la función de clientes. Estas instancias son creadas con la misma metodología en la que fueron creadas las instancias de servidor. Sin embargo, estas instancias no tienen instalados los servicios que tiene el servidor, debe tener instalado el software "*Siege*". Este software puede obtenerse en Siege Software.

Este software permite medir la capacidad de una aplicación web para mantenerse activa, y dando los servicios que ofrece. En este caso se va a instalar en las instancias con el rol de cliente para realizar múltiples conexiones al servicio creado. Se crearán distintas instancias para simular mayor número de conexiones y así poder demostrar la capacidad de auto escalado por parte del servidor.

En el tutorial proporcionado en el apartado Uso de Siege Software puede verse un video tutorial de cómo instalar y usar este software para simular múltiples conexiones sobre el servicio creado previamente.

5. Toma de decisiones

En este apartado se habla sobre las decisiones que se han tomado para llevar a cabo esta práctica, argumentando cada una de las decisiones.

Las instancias que se han creado durante el desarrollo de la práctica se han tomado en función del menor coste de los servicios AWS, así como las opciones que da para realizar pruebas gratuitas durante el período de un año.

El lenguaje de programación web que se ha usado ha sido PHP puesto que es el más conocido por los integrantes del grupo. Por la misma razón, se ha usado el gestor de base datos phpMyAdmin, puesto que el principal motivo de la realización de esta práctica es aprender a desplegar un sistema auto escalable, y no el software con el que se implementan los servicios.

Esta práctica tiene como parte opcional el funcionamiento del servicio mediante API REST. Para ello, se ha instalado el framework SLIM, que permite la creación de una API REST en PHP. Sin embargo, esto requería un tiempo de aprendizaje que ninguno delque los integrantes del grupo no disponían, por lo tanto, se encuentra disponible en la instancia aunque no se ha usado.

Como base de datos se ha tomado MySQL puesto que es el tipo de base de datos que acepta phpMyAdmin, y es la más conocida por los miembros del grupo.

6. Problemas encontrados

Durante el desarrollo de esta práctica han aparecido diversos problemas, los cuales han ido resolviéndose convenientemente.

El primer problema que surgió fue al querer cargar la base de datos a phpMyAdmin. Este problema es debido a que el fichero .csv que se quería importar es demasiado grande, y al hacerlo mediante la forma convencional usando la interfaz gráfica, no subía el fichero por completo. Para subsanar este error, se debió subir al servidor el fichero .csv, y mediante el acceso manual a la base de datos de phpMyAdmin, introducir una línea de comandos para poder importar el fichero .csv a la tabla correspondiente de la base de datos.

Otro problema que se dio fue que el balanceador de carga no detectaba el fallo al conectar con la base de datos debido a la cantidad de conexiones que se estaban simulando con Siege. Para ello se tuvo que modificar el fichero "*index.php*" y añadir la línea que envía un código 500 HTTP, además de establecer como ruta de *"Health Check"* la ruta al fichero *"index.php"* en lugar del la / (raiz). Con estas mejoras el balanceador de carga ya detectaba que se producía una interrupción del servicio en los casos de sobre carga en la base de datos o en servidor.

Durante la configuración de la característica de auto escalado del servicio, surgía el problema de que las alarmas que se crean para gestionar el número de instancias que se encuentran activas para dar servicio no era el correcto. Esto era debido a varios errores, uno de ellos era que requería confirmación para la creación de las alarmas. Otro de los errores era la incorrecta creación de dichas alarmas, puesto que necesitan un correo para poder crear las alarmas correspondientes.

Para realizar las pruebas de esfuerzo del servicio, se ha usado el software Siege. Sin embargo, este software tiene un problema en cuanto al número límite de conexiones simultáneas que puede realizar. Para solucionar esto, se han creado más instancias con el rol de cliente ejecutándose simultáneamente, y así conseguir un número mayor de conexiones sobre el servidor. El principal problema que provoca era errores por segmentación de memoria.

A la vez que el caso anterior, la base de datos MySQL de phpMyAdmin tiene una limitación en cuanto al número de conexiones simultáneas que permite. Para solventar este error, se debe ir al fichero de configuración de MySQL y cambiar el parámetro que

limita el número de conexiones. Esto puede verse en el apartado Aumentar conexiones máximas a MySQL.

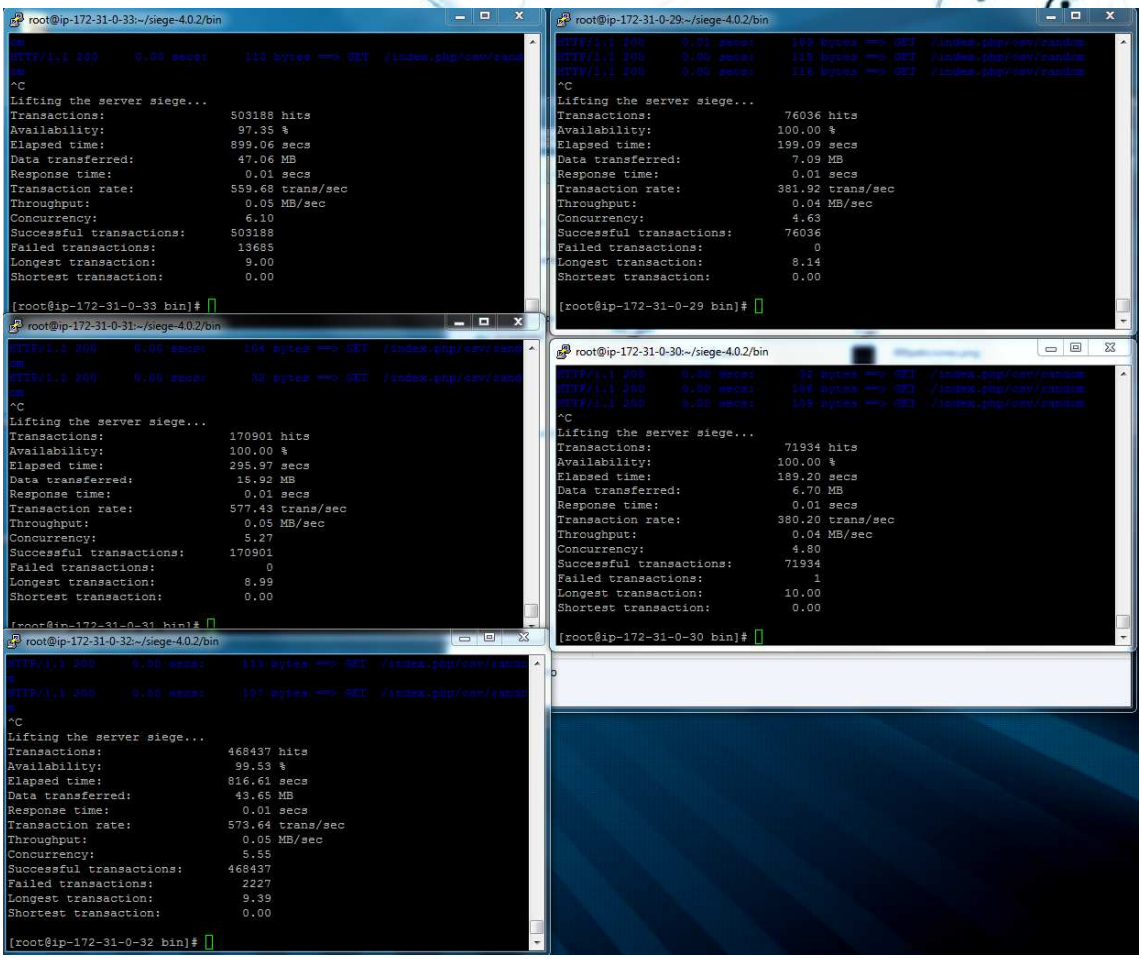
Se quería realizar la parte opcional de la práctica con la cual se consigue que el servicio funcione como una API REST. Sin embargo, debido al tiempo del que se disponía en clase para realizar la práctica, y el problema de compatibilidad de php y el framework SLIM, esta parte no ha podido implementarse. Esto se ha solucionado simulando el funcionamiento de API REST mediante la captura de la URL y "rompiéndola" en función del formato deseado, y de la IP de la cual se quiere obtener información, ya sea aleatoria o específica.

Por último, hay un problema en ver el estado de las instancias con el rol de servidor en AWS. Esto es debido a que los datos que se proporciona en la interfaz no es en tiempo real, tiene una demora aproximada de 2 minutos. Por tanto, no es posible determinar con exactitud en qué momento se lanza una nueva instancia, y en qué momento se detiene por falta de necesidad, o porque ha quedado inservible.

7. Pruebas de esfuerzo

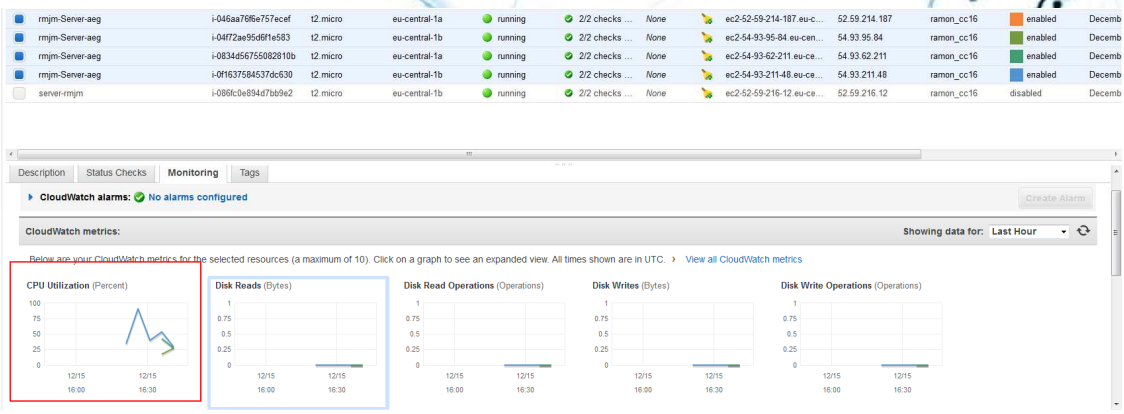
En el presente apartado se muestra el resultado del servicio, así como de datos que se pueden obtener que representan la carga de trabajo que tienen las instancias, y la función de auto escalado.

Una vez el servicio está en funcionamiento, se realizan las distintas pruebas con las instancias con el rol de cliente para realizar conexiones múltiples hacia el servicio.

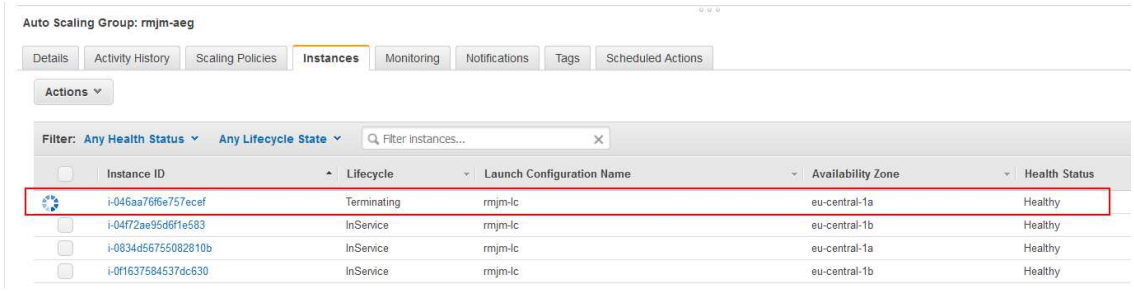


En la figura anterior se está realizando una prueba de estrés que implica 200 peticiones por parte de 5 clientes diferentes. En el resumen puede verse cuál es el tiempo de respuesta medio, las transacciones correctas, y las fallidas.

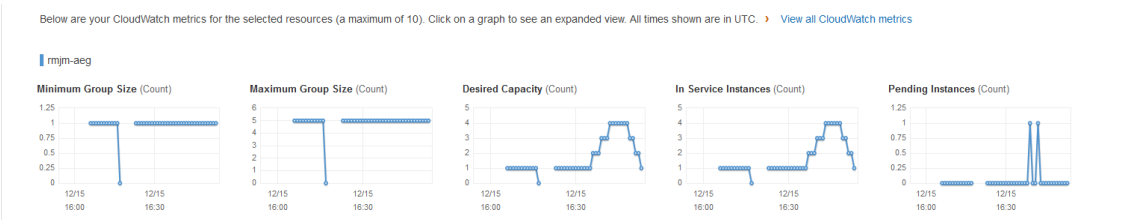
Por parte del servicio, en la figura que aparece a continuación puede observarse como se tiene el grupo de auto escalado que son las instancias seleccionadas en la parte superior de la imagen, y la carga de utilización que tienen las distintas instancias. Esto puede verse en la sección enmarcada en rojo, con los picos de subida y bajadas que se ven. Un pico de bajada hace referencia al lanzamiento de una instancia nueva, y repartición de carga de trabajo entre las distintas instancias. Por el contrario, cuando se ve un pico de subida, representa a la carga de trabajo que está teniendo la instancia por falta de recursos del servicio, lo cual supondrá el lanzamiento de una nueva instancia.



Cuando una instancia no es necesaria, ya sea por falta de carga de trabajo, o por que ha quedado inservible o bloquedad, esta instancia es automáticamente eliminada. Esto puede verse en la siguiente imagen mediante el atributo Lifecycle como “Terminating”.



Para terminar, a continuación, se ve un resumen total tras la prueba de esfuerzo en el servicio. En este caso, se puede ver como el número total de instancias que permite este auto escalado es de cinco. Esto es configurable cuando se está creando el grupo de auto escalado en función de las necesidades que tiene que cubrir. Las gráficas situadas más a la derecha representan el número de instancias que son necesarias en cada instante, el número de instancias que están en funcionamiento en cada momento, y las instancias que están pendientes para comenzar a funcionar.



8. Conclusiones

Al realizar la presente práctica se han obtenido y afianzado conceptos relacionados con la computación en la nube o cloud computing. 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. En primer lugar, se obtuvo el conocimiento para poder lanzar una instancia y las diferencias entre los tipos de instancias. Así como también se ha alcanzado un buen grado de manejabilidad y comprensión de los balanceadores de carga y sistemas de auto escalado.

El principal escollo como se comento en el apartado *"Problemas encontrados"* ha sido el intento de utilizar una API REST que la utilización de los servicios de AWS. Por otro lado, la utilización de AWS no ha provocado grandes problemas, con la configuración adecuada el sistema trabajaba de la forma esperada. 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%.

9. Posibles mejoras

Para realizar este servicio hay que tener en cuenta la configuración que se ha tomado.

Primeramente, hay que tener en cuenta que se ha realizado un diseño del servicio teniendo en cuenta que la 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 se requiriese un modelo en el que se permitiera cambios en la base de datos, este no sería un buen sistema puesto que habría que ir modificando todas las instancias individualmente. Esto podría solventarse usando los recursos de 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.

10. Referencias

- Tutorial para crear y configurar balanceador de carga y grupo de auto escalado. https://www.youtube.com/watch?list=PL_d-XKRO_5G82DPjw4r5xEbdhVngzC9Ju&v=sNkAgCdBxo4
- Documentación Amazon Web Services. https://aws.amazon.com/es/documentation/
- Documentación para creación de instancias EC2-Linux. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-ug.pdf
- Instalación LAMP (Linux , Apache, MySQL, PHP) y phpMyAdmin. http://howto.gandasoftwarefactory.com/desarrollo-software/2014/como-instalar-phpmyadmin-ubuntu-linux-20140929/
- Base de Datos de IPs. Creación de tabla e importación a base de datos.http://lite.ip2location.com/database/ip-country-region-city-latitude-longitude-zipcode-timezone
- Importación de .CSV con gran cantidad de datos a MySQL.

  http://www.konnichiwamundo.com/2013/08/importar-un-fichero-csv-grande-a-mysql.html

- Cambiar cuenta root en phpMyAdmin.

  https://www.youtube.com/watch?v=b_WbtrUF8pw

- Siege Software. https://www.joedog.org/siege-home/
- Uso de Software Siege. https://www.youtube.com/watch?v=V0J2ULF0A-c&list=PL_d-XKRO_5G82DPjw4r5xEbdhVngzC9Ju&index=2
- Aumentar conexiones máximas a MySQL. http://helloit.es/2011/07/limite-de-conexiones-en-mysql/