PONTIFICIA UNIVERSIDAD CATOLICA MADRE Y MAESTRA
FACULTAD DE INGENIERÍA Y CIENCIAS EXACTAS
DEPARTAMENTO DE INGENIERÍA EN CIENCIAS DE LA COMPUTACIÓN

PRÁCTICA #6
-- --
Balanceadores de Carga y Sesiones Distribuidas

REALIZADO POR:

Eduardo E. Martínez **ID** 1014-2001
Gabriel Cepeda **ID** 1014-1803

ASIGNATURA:
Programación Web Avanzada
ICC-354

PROFESOR:
Ing. Carlos Camacho

Santiago de los Caballeros, República Dominicana

Introducción

El panorama del desarrollo web moderno evoluciona continuamente, impulsado por la necesidad de mejorar el rendimiento, garantizar la escalabilidad y priorizar las experiencias del usuario. En este contexto dinámico, esta práctica 6 de la asignatura Ingeniería en Ciencias de la Computación representa una exploración y acercamiento a tecnologías de vanguardia que permitan lograr el balance de cargas y demandas de aplicación, tolerancia a fallas y una comunicación segura. Como fundamento de la aplicación, Java y el framework Spring son utilizados dada su versatilidad y eficiencia que permiten satisfacer las necesidades del desarrollo web moderno. A su vez, este fundamento se ve acompañado de la implementación del balanceo de cargas como enfoque principal de esta práctica.

La integración de tecnologías de equilibrio de carga en aplicaciones web modernas es fundamental para abordar los desafíos planteados por el aumento del tráfico y las diversas interacciones de los usuarios. El equilibrio de carga mejora el rendimiento general y la disponibilidad de la aplicación al distribuir las solicitudes entrantes en múltiples instancias. Esto no solo mejora los tiempos de respuesta y reduce la latencia, sino que también garantiza una utilización eficiente de los recursos, allanando el camino para la escalabilidad y la tolerancia a fallos. En una era en la que las expectativas de los usuarios en cuanto a capacidad de respuesta y confiabilidad son primordiales, el equilibrio de carga emerge como una práctica fundamental que da forma al panorama del desarrollo web moderno.

La incorporación de HAProxy como equilibrador de carga marca un paso significativo hacia la optimización de la utilización de recursos y la mejora de la capacidad de respuesta de la aplicación. El equilibrio de carga, una práctica fundamental en el desarrollo web moderno, aborda los desafíos planteados por el creciente tráfico y las diversas interacciones de los usuarios. HAProxy, con su algoritmo Round-robin, distribuye sistemáticamente las solicitudes entrantes entre múltiples instancias, garantizando un uso equitativo de los recursos. Esto no sólo mejora los tiempos de respuesta sino que también sienta las bases para la escalabilidad y la tolerancia a fallos.

Además de esto, la adopción de sesiones, facilitadas por cookies, refleja un enfoque del desarrollo web centrado en el usuario. Las aplicaciones web modernas se esfuerzan por ofrecer experiencias fluidas y personalizadas. Las sesiones permiten la preservación de datos específicos del usuario a través de diversas interacciones, fomentando un viaje coherente y personalizado independientemente de las solicitudes de atención de la instancia de backend. Esto contribuye significativamente a la participación y satisfacción del usuario.

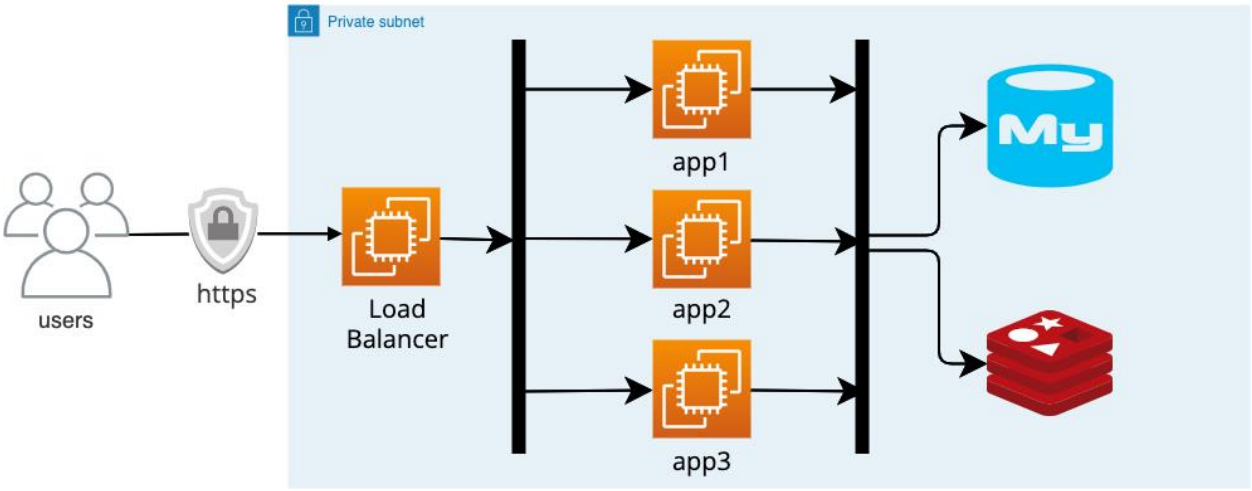
Finalmente, se evidencia como se puede observar el puerto o la identificación de la instancia que es ejecutada, información que no solo ayuda a solucionar problemas y monitorear, sino que también permite validar el funcionamiento adecuado y consistente del balanceador de carga.

Las siguientes secciones de este informe profundizarán en las complejidades técnicas, las configuraciones y las funciones colaborativas de estas tecnologías en la entrega de nuestra sólida aplicación web Mocky API.

Metodología de Desarrollo

La metodología de desarrollo para implementar balanceo de cargas con HAProxy en nuestra práctica siguió un enfoque sistemático y por fases para garantizar una integración perfecta de las tecnologías y lograr los objetivos del proyecto. El proceso comenzó con la implementación de HAProxy dentro de la arquitectura de la aplicación, proporcionando una base sólida para el equilibrio de carga y la tolerancia a fallos.

Esta primera fase implicó la integración de HAProxy a través de una configuración meticulosa, especificando el algoritmo Round-robin para la distribución de carga y los mecanismos de tolerancia a fallas para manejar con elegancia posibles instancias de indisponibilidad. Esta fase sentó las bases para optimizar la utilización de recursos y mejorar la capacidad de respuesta general de la aplicación, a la vez que se aseguraba el cumplimiento de la arquitectura solicitada para satisfacer esta práctica.



Implementamos, una navegación segura vía https desde el cliente hacia la instancia que estaría funcionando como balanceadora de carga. Posteriormente, el proyecto pasó a un entorno de AWS, aprovechando la escalabilidad de la plataforma en la nube y la facilidad de aprovisionamiento de máquinas virtuales. Se crearon tres instancias adicionales de máquinas virtuales, cada una de las cuales aloja la aplicación web del mockup. Este uso estratégico de múltiples instancias se alineó con los objetivos de balanceo de carga, lo que permitió la distribución de solicitudes entrantes entre estas instancias. Adicional, se creó un clúster de Redis en ElastiCache con el fin de almacenar las sesiones de las distintas instancias en un solo lugar y así poder realizar el balanceo de la carga manteniendo las sesiones de los usuarios independientes de las instancias.

Un aspecto fundamental del proyecto fue la configuración de certificados SSL para el acceso público de la instancia que estaría manejando el balanceador de la carga. La integración de Let's

Encrypt jugó un papel crucial en la automatización de este proceso, facilitando la generación de certificados seguros y verificados para esta instancia.

Con los certificados SSL configurados, el proyecto pasó a la fase de ejecución. Se eligió una de las instancias para ejecutar la aplicación, proporcionando una demostración tangible de la gestión de carga lograda a través del algoritmo Round-robin implementado. La recarga de la aplicación mostró la distribución dinámica de solicitudes entre las instancias disponibles, ofreciendo una ilustración práctica de las capacidades de equilibrio de carga.

Adicional, se controló la seguridad configurando un grupo de seguridad para la instancia que estaría manejando el balanceo. Esta sería la única que tendría acceso desde cualquier IP a los puertos de acceso de la aplicación (80 para http y 443 para https). Las demás instancias e incluyendo el clúster de Redis se trabajó dentro de una subred privada con el fin de que solo tuvieran acceso los servicios que estén dentro de la misma. Excluyendo así el acceso a dichos servicios desde fuera de la subred privada de AWS.

Repositorio de Proyecto

https://github.com/gabrielcepedag/web-avanzada/tree/main/practica-6

Enlace a Video de Práctica

https://www.youtube.com/watch?v=tn6czJBcb9A