Universidad Tecnológica de Panamá
Sistemas Operativos I
Proyecto por Examen Semestral
Informe Técnico

Prof. Aris Castillo de Valencia

Estudiantes: Jaime Acosta (8-971-833) Luis Villanueva (E-8-172113) Manuel Villanueva (E-8-172111)

Objetivos:

- Demostrar el funcionamiento adecuado y de las configuraciones tanto en el escenario contenerizado como el de máquinas virtuales.

- Probar el desempeño que demuestren el balanceo de carga.

Selección de Herramientas:

Version Control

GitHub es la herramienta de Version Control escogida, debido a que es actualmente la tecnología de este campo más utilizada y con mayor soporte, alguno de los puntos que hacen a GitHub una opción viable son los siguientes:

- Control de versiones distribuido: GitHub utiliza Git como su sistema de control de versiones subyacente. Git es un sistema distribuido, lo que significa que cada desarrollador tiene una copia completa del historial del proyecto en su máquina local. Esto permite a los desarrolladores trabajar de manera independiente sin depender de un servidor central. Además, facilita la fusión de cambios y la gestión de ramas para implementar nuevas características o arreglar errores de manera eficiente.

- Integración con herramientas y servicios: GitHub ofrece una amplia gama de integraciones con otras herramientas y servicios de desarrollo, como sistemas de construcción continua, servicios de revisión de código, servicios de seguimiento de problemas y mucho más. Esto facilita la configuración de un flujo de trabajo de desarrollo completo y permite una automatización más sencilla de tareas repetitivas.

- Seguridad y confiabilidad: GitHub cuenta con sólidas medidas de seguridad para proteger los repositorios y los datos del usuario. Además, la plataforma es confiable y escalable, lo que garantiza un acceso rápido y consistente a los proyectos en todo momento.

Una de las ventajas clave de GitHub es su integración con GitHub Actions. GitHub Actions es un poderoso sistema de automatización y CI/CD (Continuous Integration/Continuous Deployment) integrado directamente en la plataforma de GitHub. Aquí están algunas de las ventajas de GitHub Actions:

- Automatización de flujos de trabajo: GitHub Actions permite a los desarrolladores automatizar flujos de trabajo completos en sus repositorios. Puedes definir tareas específicas que se ejecutan automáticamente en respuesta a eventos, como confirmaciones de código, creación de solicitudes de extracción o etiquetas específicas.

- CI/CD integrado: Con GitHub Actions, puedes configurar fácilmente un flujo de integración continua (CI) y despliegue continuo (CD) para tu proyecto. Esto te permite compilar, probar y desplegar automáticamente tu código en entornos de desarrollo, pruebas y producción cada vez que realizas cambios en el repositorio.

Todo esto permite a GitHub ser una herramienta esencial para la aplicación, debido a que mediante este, se facilita los procesos de producción en cuánto a pruebas y replicación tanto de Contenedores como VMs, facilitando la operación de la aplicación.

Selección de Load Balancer

<u>¿Porqué NGINX Reverse Proxy?</u>

- Balanceo de carga: Nginx actúa como un balanceador de carga distribuyendo las solicitudes entrantes entre los servicios backend (app1 y app2 en este caso). Esto ayuda a evitar la sobrecarga en un solo servicio y mejora la eficiencia y la capacidad de respuesta de toda la infraestructura.

- Seguridad: Al utilizar un reverse proxy, los servicios backend se mantienen ocultos del tráfico externo, lo que agrega una capa adicional de seguridad. Nginx actúa como punto de entrada, lo que permite filtrar y bloquear solicitudes maliciosas antes de que lleguen a los servicios internos.

- Redirecciones y reescritura de URL: Nginx proporciona capacidades flexibles para realizar redirecciones y reescrituras de URL, lo que permite la manipulación y modificación de las rutas de las solicitudes para adaptarse a los requisitos de la aplicación.

- Caché de contenido estático: Nginx puede almacenar en caché contenido estático, como imágenes, CSS y JavaScript, reduciendo así la carga en los servicios backend y mejorando el tiempo de respuesta para los usuarios.

- Compresión de respuesta: Nginx puede comprimir las respuestas antes de enviarlas al cliente, lo que reduce el ancho de banda y mejora la velocidad de carga de las páginas.

- Mantenimiento y despliegue simplificado: El uso de un reverse proxy permite aislar los servicios backend y proporciona un punto central para gestionar cambios en la infraestructura sin afectar a los servicios internos directamente.

- Monitoreo y registro: Nginx ofrece opciones avanzadas de registro y supervisión de tráfico, lo que facilita el diagnóstico de problemas y el seguimiento del rendimiento del sistema.

El uso de Nginx como reverse proxy puede mejorar significativamente la seguridad, el rendimiento y la escalabilidad de la infraestructura de aplicaciones. Además, proporciona una capa de abstracción que facilita el mantenimiento y el despliegue de servicios backend, lo que lo convierte en una herramienta valiosa en arquitecturas modernas basadas en contenedores como la que se describe en el archivo Docker Compose proporcionado en el repositorio del proyecto.

Ambiente de Contenedores

¿Porqué Docker como solución de Contenedores?

- Facilidad de configuración: Docker Compose permite definir y configurar múltiples servicios, sus dependencias, volúmenes, redes y otras opciones en un solo archivo YAML. Esta simplicidad facilita la gestión y el mantenimiento del entorno de desarrollo o producción.

- Orquestación y escalabilidad: Con Docker Compose, se pueden definir dependencias entre servicios (mediante la opción depends_on) y escalar réplicas de servicios de manera sencilla para satisfacer las necesidades de rendimiento y escalabilidad.

- Portabilidad y consistencia: Docker Compose asegura que todos los servicios se ejecuten de la misma manera en diferentes entornos, evitando problemas de configuración y dependencias que pueden surgir en diferentes sistemas.

- Aislamiento de servicios: Cada servicio se ejecuta en su propio contenedor, lo que proporciona un alto nivel de aislamiento, lo que significa que los problemas en una aplicación no afectarán a otras.

- Versionado y control de cambios: El archivo docker-compose.yml es un recurso de control de versiones y permite mantener un registro histórico de cambios en la configuración de los servicios.

- Interoperabilidad: Docker Compose es compatible con otras herramientas de Docker y se integra fácilmente con otras tecnologías, lo que facilita la implementación y la creación de soluciones más complejas.

- Comunidad y soporte: Docker y Docker Compose tienen una gran comunidad de desarrolladores y empresas que brindan soporte y documentación. Esto garantiza que la tecnología esté en constante desarrollo y mejora.

En comparación con otras herramientas, Docker y Docker Compose destaca por su sencillez y su enfoque centrado en contenedores, lo que lo hace ideal para entornos que requieren implementaciones con contenedores. También es ampliamente adoptado en la industria, lo que significa que hay una gran cantidad de recursos, tutoriales y ejemplos disponibles en línea. En general, Docker es una elección profesional sólida para implementar aplicaciones con contenedores debido a su facilidad de uso, flexibilidad, escalabilidad y soporte de la comunidad. Su capacidad para orquestar y gestionar múltiples servicios con facilidad lo convierte en una herramienta valiosa para simplificar el proceso de desarrollo y despliegue de aplicaciones basadas en contenedores.

Ambiente de Máquinas Virtuales:

¿Porqué Fedora 38 Server Edition como solución de Máquinas Virtuales?

- Eficiencia y ligereza: Fedora Server Edition está diseñada para ser una distribución de Linux eficiente y ligera, lo que significa que consume menos recursos en comparación con otras distribuciones más pesadas. Esto es crucial en un entorno con limitaciones de recursos como el descrito.

- Rendimiento: Fedora es conocida por su enfoque en la última tecnología y actualizaciones frecuentes, lo que garantiza un buen rendimiento. Esto es especialmente importante para un servidor que puede estar manejando múltiples tareas y servicios.

- Comunidad activa y soporte técnico: Fedora cuenta con una comunidad activa de desarrolladores y usuarios, lo que proporciona un sólido soporte técnico y una amplia base de conocimientos. Si surge algún problema o se necesita ayuda, es más probable que encuentres soluciones y respuestas rápidamente.

- Seguridad: Fedora tiene una buena reputación en términos de seguridad y actualizaciones regulares para abordar vulnerabilidades. Para un servidor, es esencial mantener un alto nivel de seguridad para proteger los datos y servicios alojados en él.

- Herramientas y paquetes actualizados: Fedora incluye una amplia selección de herramientas y paquetes actualizados, lo que facilita la configuración y gestión del servidor. Esto es especialmente útil cuando se trabaja con software más nuevo y se necesita acceso a las últimas características y mejoras.

- Administración y configuración sencillas: Fedora Server Edition ofrece herramientas y configuraciones fáciles de usar que facilitan la administración del servidor, incluso para aquellos con menos experiencia en administración de servidores.

- Estabilidad y confiabilidad: Aunque Fedora es conocida por su enfoque en la innovación y las últimas tecnologías, también ha mejorado su enfoque en la estabilidad en las últimas versiones. Esto hace que sea una opción confiable para ejecutar servicios críticos en un entorno de servidor.

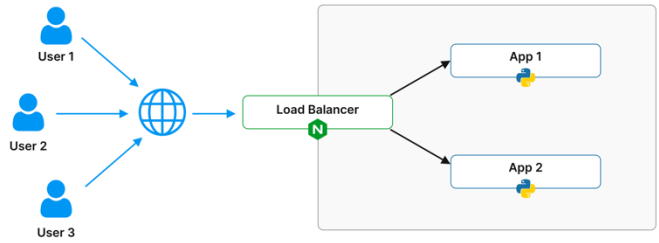
Estos punto agregados a que Fedora es un Distro que frecuenta más actualizaciones y un alto soporte, lo vuelve más atractivo como tecnología de VMs para manejar la aplicación en comparación con otros distros como Ubuntu.

Modelo de la Solución:

El sistema sirve de forma casi igual para el ambiente manejado mediantes Contenedores y el manejado mediante VMs. Ambos modelos en su base son un sistema compuesto por 3 elementos principales: 1 Reverse Proxy como punto de ingreso que manejará la carga hacia las instancias donde se encuentra la aplicación (2 instancias).

El objetivo de esta arquitectura es aislar las instancias de la aplicación de otras conexiones, y haciendo uso es del Load Balancer de NGINX para manejar las conexiones a las instancias.

FUNCIONAMIENTO DEL REVERSE PROXY EN LA SOLUCIÓN



Un servidor proxy inverso (reverse proxy en inglés) es un tipo de servidor intermediario que actúa en nombre de otros servidores, ocultando la infraestructura detrás de él y proporcionando una interfaz única para los clientes. En el caso específico de Nginx, se refiere a la funcionalidad proporcionada por el servidor web y proxy Nginx cuando actúa como un reverse proxy. Un reverse proxy funciona de la siguiente manera:

- **Cliente envía una solicitud:** Un cliente (como un navegador web) envía una solicitud HTTP a un servidor reverse proxy, generalmente identificado por una URL o una dirección IP específica.
- **El reverse proxy evalúa la solicitud:** El reverse proxy analiza la solicitud entrante y determina a qué servidor final o aplicación debe enviarla según ciertos criterios, como la dirección URL, cabeceras HTTP u otros factores.

- **Comunicación con el servidor de destino:** El reverse proxy actúa en nombre del cliente y establece una nueva solicitud al servidor de destino (por ejemplo, una aplicación web, un servidor de aplicaciones o una API) según la evaluación realizada previamente.
- **Respuesta del servidor de destino:** El servidor de destino responde al reverse proxy con el contenido solicitado.
- **Respuesta al cliente:** El reverse proxy recibe la respuesta del servidor de destino y la reenvía al cliente original que realizó la solicitud.

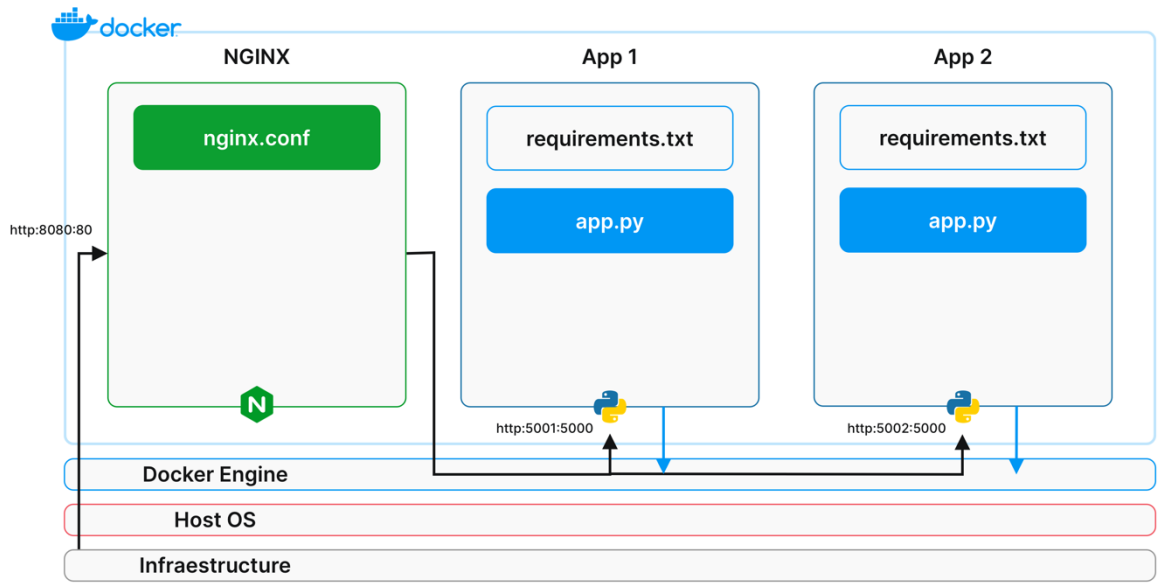
En este caso, el NGINX que es el servidor proxy, oculta detrás de él la infraestructura, la cuál en este caso son las instancias de la aplicación de Python. Por lo que el usuario hace conexión mediante http por el puerto 8080, el Reverse Proxy recibe la petición por su puerto 80, y procede a mandar hacia el puerto 5000 de una de las 2 instancias de la aplicación la petición http, por lo que el cliente nunca conecta directamente con la aplicación.

Diferencias entre la solución manejada con Contenedores vs. Máquinas Virtuales

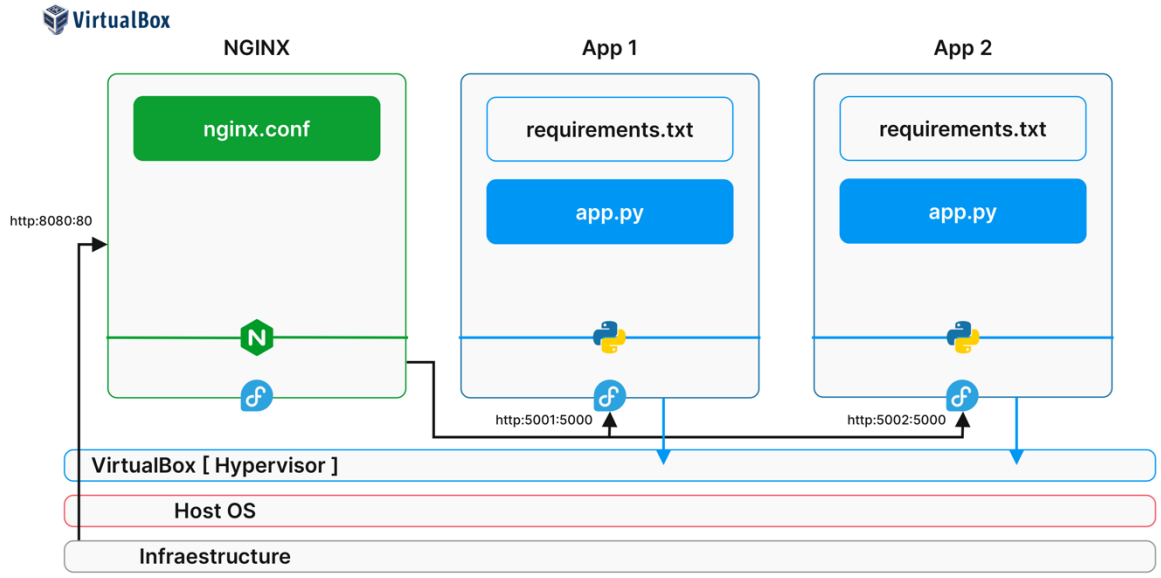
La principal diferencia entre implementar la aplicación con contenedores (usando un reverse proxy Nginx y dos instancias de Python) y con máquinas virtuales (VMs) radica en el nivel de aislamiento y el uso de recursos del sistema.

Aislamiento y Overhead:

- Contenedores: Los contenedores proporcionan un nivel de aislamiento más ligero en comparación con las máquinas virtuales. Cada contenedor comparte el kernel del host y utiliza sus propios espacios de usuario, lo que resulta en un menor overhead y mayor eficiencia en el uso de recursos. El reverse proxy Nginx y las dos instancias de Python se ejecutan dentro de contenedores separados pero comparten el mismo kernel del sistema operativo.

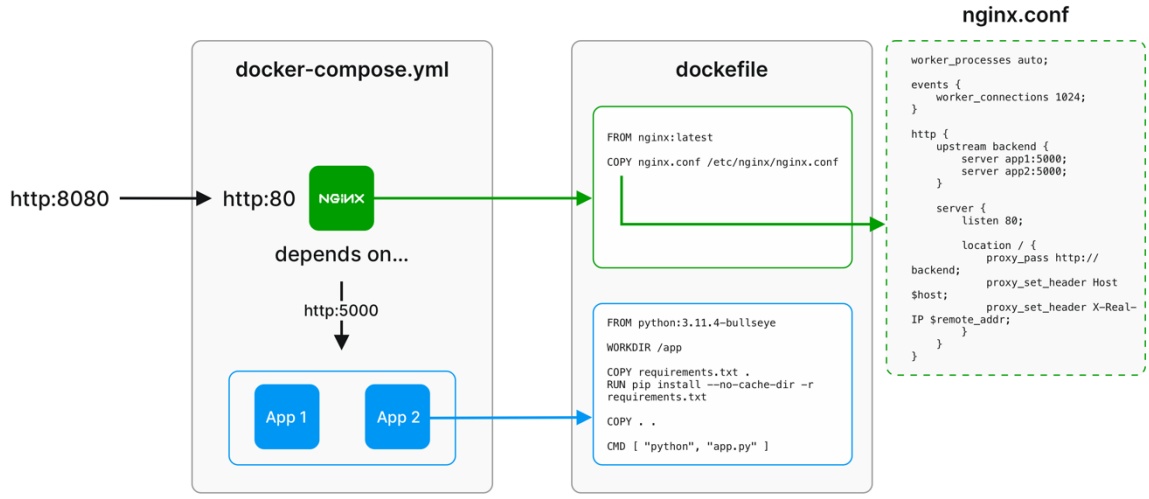


- Máquinas Virtuales: Cada máquina virtual se comporta como un sistema operativo completo y aísla completamente su propio kernel y sistema operativo del host. Esto crea un mayor overhead, ya que cada VM debe tener su propio sistema operativo y recursos asignados, lo que puede requerir más memoria y CPU en comparación con los contenedores.



Escalabilidad y Tiempo de Despliegue:

- Contenedores: Los contenedores son más rápidos de desplegar y escalar, ya que solo contienen la aplicación y sus dependencias, sin necesidad de un sistema operativo completo. Esto permite una implementación más rápida y una mayor flexibilidad para escalar la aplicación horizontalmente.



- Máquinas Virtuales: Las VMs pueden requerir más tiempo para desplegar debido a la necesidad de instalar y configurar el sistema operativo completo. Además, la escalabilidad vertical (aumentar los recursos de la VM) puede ser más sencilla que la escalabilidad horizontal.

Requisitos de Recursos:

- Contenedores: Los contenedores suelen requerir menos recursos en comparación con las máquinas virtuales, ya que comparten el kernel del host y evitan la duplicación de sistemas operativos.

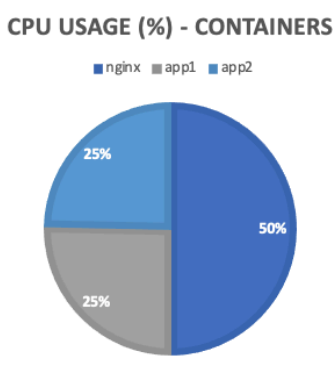
- Máquinas Virtuales: Las VMs pueden consumir más recursos, ya que cada una tiene su propio sistema operativo completo y debe reservar memoria y CPU para ello.

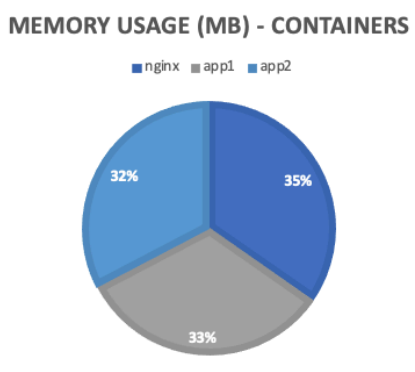
Rendimiento y Consumo de Recursos de las Soluciones:

El rendimiento comparado entre Contenedores y Máquinas Virtuales es esencial, ya que nos permitirá analizar cuanto es el gasto de recursos de cada solución. A la hora de ver el rendimiento, nos basamos en métricas de RAM y CPU, ya que estos son los recursos clave que se utilizan por la infraestructura para proveer el servicio. A continuación las métricas de rendimiento de ambas soluciones:

Uso de Recursos de Contenedores

Infrastructure Solution [Containers]	nginx	app1	app2
CPU usage (%)	0.73	0.37	0.36
Memory Usage (MB)	34.93	32.94	32.9



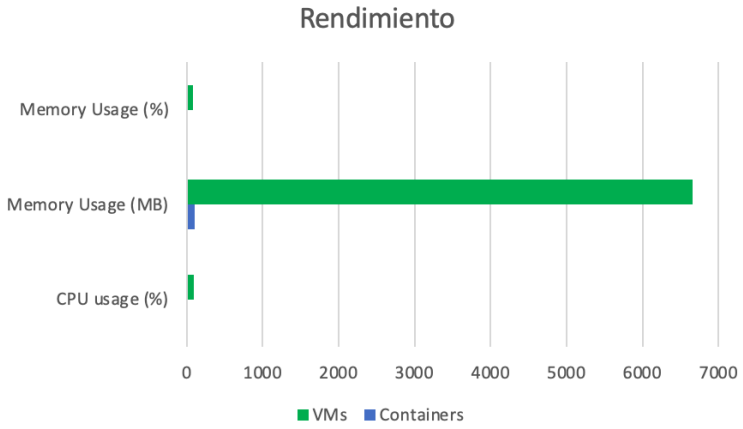


Uso de Recursos de Máquinas Virtuales

Infrastructure Solution [VMs]	nginx	app1	app2
CPU usage (%)	7.3	5.3	5.2
Memory Usage (MB)	661.8	612.7	662.8

Comparación de Rendimiento General de ambas soluciones

Infrastructure Solution	Containers	VMs
CPU usage (%)	0.73	90
Memory Usage (MB)	100.08	6656
Memory Usage (%)	2.61	82



Rendimiento de Load Balancers de ambas soluciones

Parte clave a evaluar del sistema son los Load Balancer, ya que evaluar el rendimiento del Load Balancer asegura que se distribuyan adecuadamente las cargas de trabajo, evitando la sobrecarga en nodos específicos y mejorando la disponibilidad y respuesta del sistema. Además, los Load Balancers desempeñan un papel fundamental en la alta disponibilidad del sistema, ya que redirigen el tráfico a instancias replicadas. Al evaluar su rendimiento, podemos detectar posibles cuellos de botella o fallos en el balanceo de carga, lo que garantiza una mayor resiliencia del sistema frente a fallos individuales. Otra razón importante es la optimización de recursos. Un Load Balancer eficiente aprovechará al máximo los recursos disponibles, optimizando la utilización de los contenedores y las máquinas virtuales. Esto se traduce en una mayor eficiencia operativa y, en última instancia, en menores costos de infraestructura.

A continuación, se muestran las métricas del monitoreo del funcionamento de los Load Balancers tanto en el ambiente de Contenedores como Máquinas Virtuales. Para el benchmarking y monitoreo se hizo suo de ApacheBench (ab), la cuál es una herramienta de línea de comandos diseñada para realizar pruebas de carga y evaluar el rendimiento de servidores web, incluidos los servidores proxy y balanceadores de carga, como nginx reverse proxy load balancer. Esta herramienta es especialmente útil para verificar el rendimiento de nginx reverse proxy load balancer debido a que provee:

- **Facilidad de uso:** ApacheBench es una herramienta simple y fácil de usar, lo que la convierte en una opción accesible para realizar pruebas rápidas y obtener resultados inmediatos.
- **Simulación de múltiples clientes:** ApacheBench puede simular múltiples clientes concurrentes enviando solicitudes HTTP al servidor proxy o balanceador de carga. Esto permite probar cómo el sistema se comporta bajo diferentes cargas y evaluar su capacidad para manejar solicitudes simultáneas.
- **Medición de rendimiento:** La herramienta proporciona métricas como el tiempo de respuesta, el número de solicitudes por segundo y la velocidad de transferencia. Estos datos permiten evaluar el rendimiento del nginx reverse proxy load balancer y obtener información valiosa sobre su capacidad para distribuir la carga de manera efectiva.
- **Análisis comparativo:** ApacheBench permite realizar comparaciones entre diferentes configuraciones o versiones del nginx reverse proxy load balancer. Al modificar parámetros y ejecutar pruebas, es posible determinar qué configuración ofrece el mejor rendimiento en diferentes escenarios.

Por lo que se vuelve una herramienta altamente efectiva a la hora de comparar a detalle la funcionalidad de los LBs en ambos ambientes en uso, permitiéndonos profundizar en el Benchmarking y rendimiento general según la infraestructura de cada solución.

Los benchmark se corrieron mediante el comando: **ab -n 1000 -c 50 nginx.local/**

- **ab:** Este es el comando para ejecutar la herramienta de pruebas de carga del servidor web Apache.
- -**n 1000:** Especifica el número de solicitudes que se enviarán al servidor. En este caso, se ha establecido en 1000, lo que significa que se enviarán 1000 solicitudes al servidor durante el proceso de prueba.
- **-c 50:** Esta opción establece el número de solicitudes concurrentes que se realizarán al mismo tiempo. En este caso, se ha establecido en 50, lo que significa que habrá 50 solicitudes simultáneas enviadas al servidor en cualquier momento durante la prueba.
- **nginx.local/:** La URL del servidor que se va a probar. La herramienta de pruebas de carga está apuntando al nginx.local donde esta el NGINX Reverse Proxy en funcionamiento.

Resultados de la Prueba de Balance

Infrastructure Solution	**Containers**	**VMs**
Document Length (bytes)	2807	2804
Concurrency Level	50	50
Time Taken for Tests (seconds)	13.887	53.273
Complete Requests	1000	1000
Failed Requests	553	826
Connect	0	0
Receive	0	0
Length	553	826
Exceptions	0	0
Total Transferred (bytes)	2965151	2964458
HTML Transferred (bytes)	2806151	2805458
Requests per Second	72.01	18.77
Time per request (ms)	694.342	2663.644

Time per request (ms) [across all concurrent requests]	13.447	53.273
Transfer Rate (Kbytes/sec)	208.52	54.34
Connection Times (ms)		
Connect		
min	0	0
mean	0	2
median	0.5	2.9
max	4	73
Processing		
min	180	618
mean	673	2495
median	626	2368
max	4035	5545
Waiting		
min	177	481
mean	672	2495
median	625	2368
max	4035	5545
Total		
min	180	621
mean	672	2496
median	626	2370
max	4035	5546
% of Requests served within a certain time (ms)		

50%	626	2370
66%	695	2619
75%	753	2844
80%	790	3006
90%	947	3485
95%	1094	3862
98%	1493	4211
99%	1602	4613
100% [Longest request]	4035	5546

Para analizar los resultados y determinar cuál tiene mejor rendimiento entre máquinas virtuales (VMs) y contenedores, examinemos las métricas clave proporcionadas en el informe de pruebas:

Requests per Second (Peticiones por segundo):

- Contenedores: 72.01 peticiones por segundo
- VMs: 18.77 peticiones por segundo

Time per request (Tiempo por petición) - Time per request (across all concurrent requests) (Tiempo por petición, considerando todas las peticiones simultáneas):

- Contenedores: 13.447 ms
- VMs: 53.273 ms

Transfer Rate (Tasa de transferencia) (Kbytes/sec):

- Contenedores: 208.52 Kbytes/sec
- VMs: 54.34 Kbytes/sec

% of Requests served within a certain time (Porcentaje de peticiones servidas dentro de cierto tiempo):

- Contenedores (50%): 626 ms
- VMs (50%): 2370 ms

Tabla de Resultados

Solution	Containers	VMs
Requests per Second	72.01	18.77
Time per request (ms) [across all concurrent requests]	13.447	53.273
Transfer Rate (Kbytes/sec)	208.52	54.34
50% of Requests served within a certain time (ms)	626	2370

Análisis de Resultados y Discusión:

Rendimiento General y Consumo de Recursos

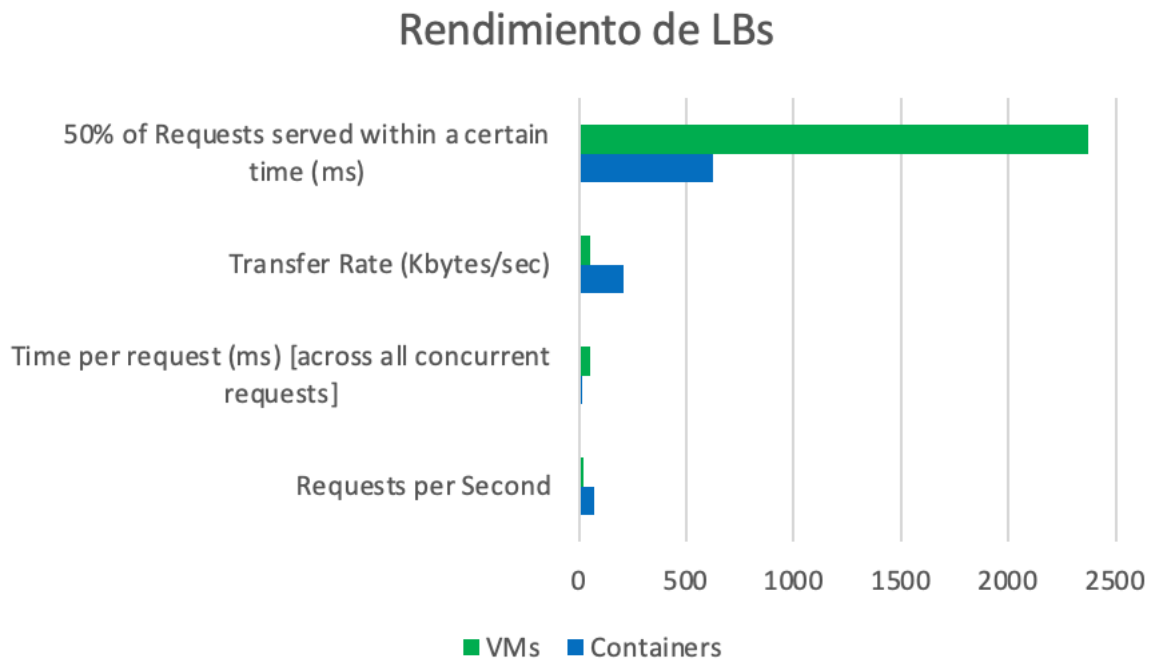
- **CPU Usage (Uso de CPU):** En la infraestructura de VMs, el uso de la CPU es significativamente más alto (90%) en comparación con el uso de CPU en contenedores (0.73%). Esto indica que los contenedores están utilizando la CPU de manera mucho más eficiente, ya que están ejecutando procesos con una carga de trabajo más ligera y liberando recursos para otros procesos en el sistema. En este aspecto, los contenedores presentan un mejor rendimiento.

- **Memory Usage (Uso de memoria):** En la infraestructura de VMs, el uso de memoria es más alto que en contenedores. Las VMs utilizan un total de 6656 MB de memoria, mientras que los contenedores utilizan solo 100.08 MB en total. Nuevamente, esto indica que los contenedores son más eficientes en el uso de memoria, ya que tienen una huella de memoria mucho más ligera en comparación con las VMs.

- **Memory Usage (%) (Uso de memoria en porcentaje):** Al calcular el uso de memoria como porcentaje del total disponible, vemos que los contenedores tienen un uso del 2.61%, mientras que las VMs tienen un uso del 82%. Esto confirma que los contenedores tienen una sobrecarga de memoria mucho menor y utilizan la memoria de manera más eficiente que las VMs.

- En resumen, según la información proporcionada, los contenedores tienen un mejor rendimiento en comparación con las VMs. Tienen un uso mucho más eficiente de los recursos de CPU y memoria, lo que resulta en una menor sobrecarga del sistema y en la capacidad de ejecutar más procesos en la misma infraestructura.

Rendimiento de Load Balancers



En general, los resultados indican que los contenedores superan significativamente a las máquinas virtuales en términos de rendimiento en esta prueba específica. Aquí hay algunas observaciones clave:

Peticiones por segundo: Los contenedores manejan aproximadamente cuatro veces más peticiones por segundo que las máquinas virtuales. Esto indica una mayor eficiencia en la capacidad de respuesta de los contenedores.

Tiempo por petición: Los contenedores tienen un tiempo de respuesta mucho más bajo que las máquinas virtuales. La diferencia es de aproximadamente 4 veces. Esto significa que los contenedores pueden responder a las solicitudes más rápidamente.

Tasa de transferencia: Los contenedores también superan ampliamente a las máquinas virtuales en términos de tasa de transferencia. Nuevamente, esta diferencia es de aproximadamente 4 veces.

Porcentaje de peticiones servidas dentro de cierto tiempo: Los contenedores tienen tiempos de respuesta más consistentes, con un tiempo de mediana (50%) de 626 ms, en comparación con 2370 ms para las máquinas virtuales. Esto sugiere que los contenedores son más predecibles y estables en su rendimiento.

En resumen, según los resultados proporcionados, los contenedores exhiben un rendimiento considerablemente mejor que las máquinas virtuales en esta prueba.

Conclusión

Los resultados de las pruebas muestran claramente que la implementación del sistema utilizando contenedores ha demostrado ser significativamente más eficiente y confiable en comparación con el enfoque basado en máquinas virtuales. Los contenedores han superado a las máquinas virtuales en todas las métricas clave, incluido el número de peticiones por segundo, el tiempo de respuesta por petición, la tasa de transferencia y el porcentaje de peticiones servidas dentro de ciertos tiempos. Esta diferencia en el rendimiento entre ambos enfoques destaca la utilidad y ventajas de adoptar tecnologías de contenedores en el desarrollo y despliegue de aplicaciones. La capacidad de los contenedores para proporcionar un entorno aislado y ligero ha sido crucial para lograr una mayor eficiencia y tiempos de respuesta más rápidos, además, la rápida escalabilidad horizontal de los contenedores ha permitido un manejo efectivo de altos niveles de concurrencia y carga, lo que ha resultado en una mayor cantidad de peticiones atendidas satisfactoriamente en un tiempo más corto. Por su rendimiento mejorado, la utilización de contenedores también ha demostrado ser ventajosa en términos de administración y mantenimiento del sistema, y la capacidad de orquestar y administrar contenedores mediante herramientas como Kubernetes podría facilitar aún mas la gestión de la infraestructura y el despliegue de actualizaciones de forma más sencilla y segura. Los entornos de desarrollo y producción se vuelven más coherentes, lo que reduce la posibilidad de errores causados por diferencias entre ambientes. Si bien las máquinas virtuales también ofrecen ventajas en términos de aislamiento y flexibilidad, los resultados de las pruebas sugieren que en este caso específico, los contenedores se han destacado como la mejor opción para implementar el sistema. La elección de tecnología adecuada para cada caso depende de los requisitos específicos del proyecto, por lo cuál este trabajo nos permitió como equipo no solo entender como es el proceso de desarrollar y levantar a producción sistemas, sino también como las diferentes elecciones que hay a la hora de hacer infraestructura pueden impactar fuertemente los resultados y el rendimiento del producto. También pudimos entender el funcionamiento de partes de infraestructura, como Web Servers, Contenedores, VMs y Load Balancers y como ciertas en herramientas como el caso de NGINX, pueden ser acomodadas de múltiples formas para adaptarse a los requisitos del sistema.

Bibliografía

[1]

S. Brownlee, “Visualizing Docker Compose,” Nashvillesoftwareschool.com, Jun. 13, 2019. https://learn.nashvillesoftwareschool.com/blog/2019/06/13/visualizing-docker-compose (accessed Jul. 21, 2023).

[2]

“NGINX Reverse Proxy | NGINX Documentation,” Nginx.com, 2023. https://docs.nginx.com/nginx/admin-guide/web-server/reverse-proxy/ (accessed Jul. 21, 2023).

[3]

G. Thomas, “A beginner’s guide to Docker — how to create your first Docker application,” freeCodeCamp.org, Apr. 02, 2019. https://www.freecodecamp.org/news/a-beginners-guide-to-docker-how-to-create-your-first-docker-application-cc03de9b639f/ (accessed Jul. 21, 2023).

[4]

“Apache Bench Tutorial,” Tutorialspoint.com, 2023. https://www.tutorialspoint.com/apache_bench/index.htm (accessed Jul. 21, 2023).

[5]

“Deploying Multiple Applications to VMs with NGINX as a Reverse Proxy,” Engineering Education (EngEd) Program | Section, 2020. https://www.section.io/engineering-education/nginx-reverse-proxy/ (accessed Jul. 21, 2023).