Universidad Latina de Costa Rica

BISOF-18 Sistemas Operativos II

Análisis de caso #1 – Jaime Fung Delgado

**Plataforma Web Institucional Escalable con Contenedores**

## 1. Contexto actualizado

La Universidad Latina está implementado una infraestructura moderna para hospedar sitios web institucionales en Java, NodeJS, Drupal y WordPress, basada en contenedores Docker, balanceo de carga con HAProxy y automatización CI/CD. Este entorno garantiza alta disponibilidad, seguridad y escalabilidad.

## 2. Objetivo del caso

Diseñar la plataforma web basada en contenedores Docker, balanceo de carga con HAProxy y automatización CI/CD. Este entorno garantiza alta disponibilidad, seguridad y escalabilidad.

Sugerencia:

<https://www.haproxy.com/blog/haproxy-on-docker-swarm-load-balancing-and-dns-service-discovery>

Simular y analizar el comportamiento de una aplicación web básica dentro de la plataforma, permitiendo a los estudiantes comprender la integración de conceptos de sistemas operativos, redes y arquitectura distribuida.

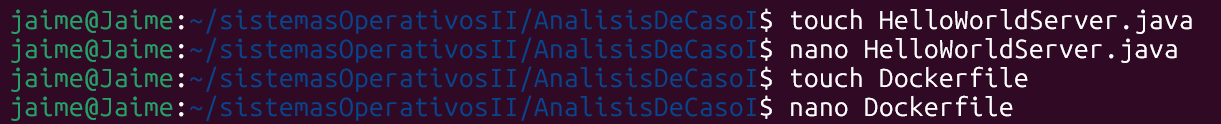
Sugerencia:

Apache JMeter

## 3. Actividades y análisis técnicos

### 1. Simulación de un servicio en contenedor

Creación de archivos necesarios



Containerizar el servidor Java que sirve una página 'Hola mundo'.

A computer screen shot of a program

AI-generated content may be incorrect.

Configurar un Dockerfile básico.

A screenshot of a computer

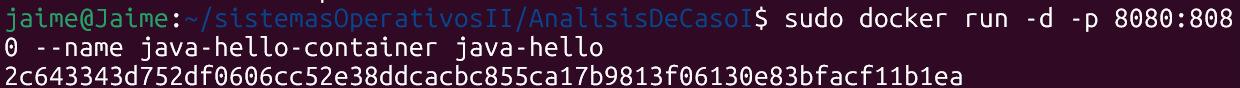
AI-generated content may be incorrect.

Construir la imagen

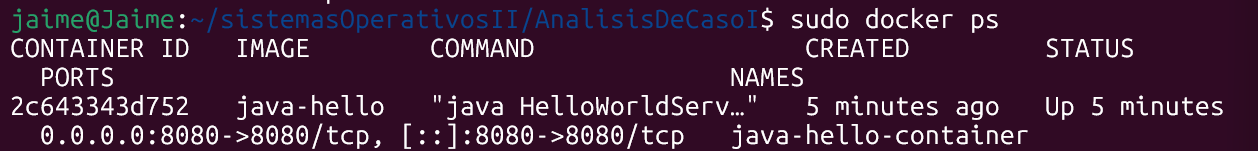
A screenshot of a computer screen

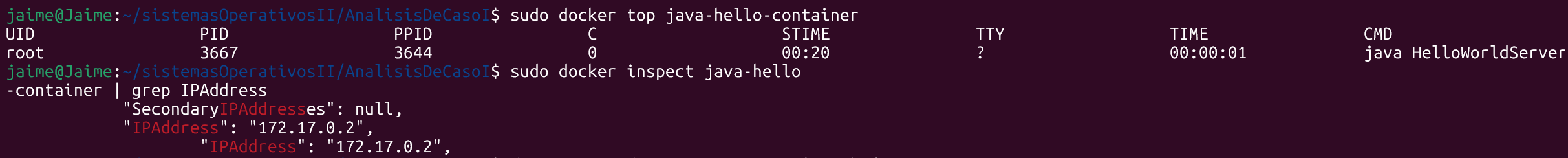
AI-generated content may be incorrect.

Ejecutar el contenedor de forma local.



Evaluar el aislamiento de procesos, red y persistencia.





Ejecutamos el localhost en el navegador con el puerto que especificamos (8080), verificamos que si funciona.

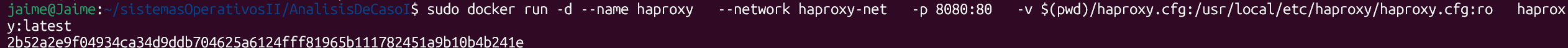
A screenshot of a computer

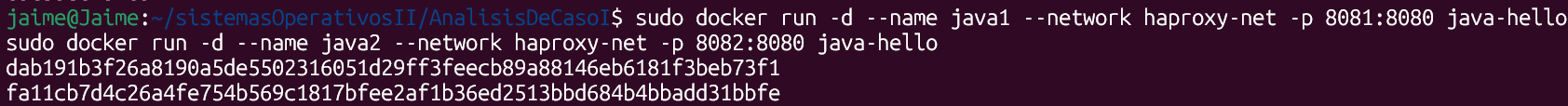
AI-generated content may be incorrect.

### 2. Balanceo de carga

· Levantar dos instancias del contenedor Java.







Aquí podemos ver como se puede acceder a la pagina con cualquiera de los dos puertos

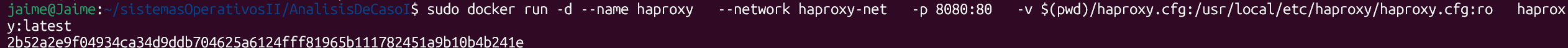
A screenshot of a computer

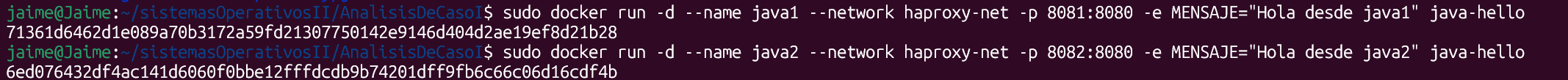
AI-generated content may be incorrect.

Tambien se puede hacer con mensajes personalizados para cada instancia, modificando el Server

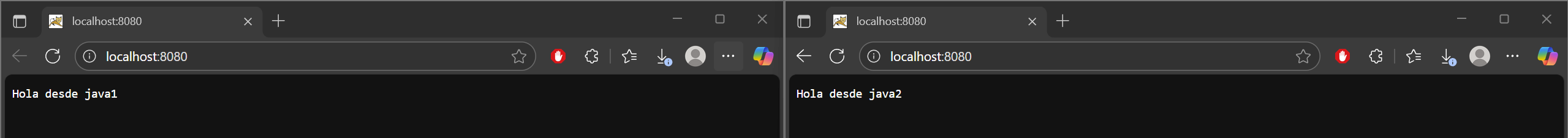
A computer screen shot of a program

AI-generated content may be incorrect.

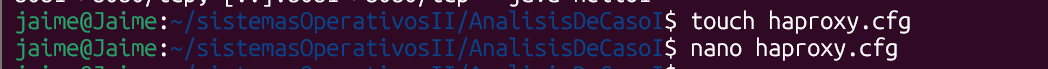




Junto al haproxy podemos ingresar a la pagina y nos redirige a cualquiera de las instancias activas, esto significa que el haproxy esta funcionando correctamente, ya que controla a que instancia entras, en este caso modificamos el server para poder comprobar que si esta funcionando correctamente, y vemos que hay veces que entras a la primera y hay veces que entras en la segunda, este balance la podemos indicar en la configuración de haproxy, mas adelante se muestra



· Configurar HAProxy para distribuir el tráfico.



A screenshot of a computer

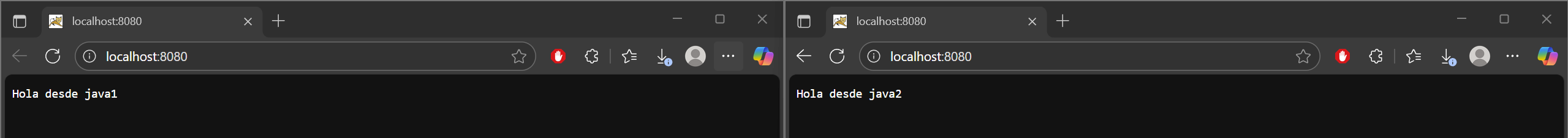
AI-generated content may be incorrect.

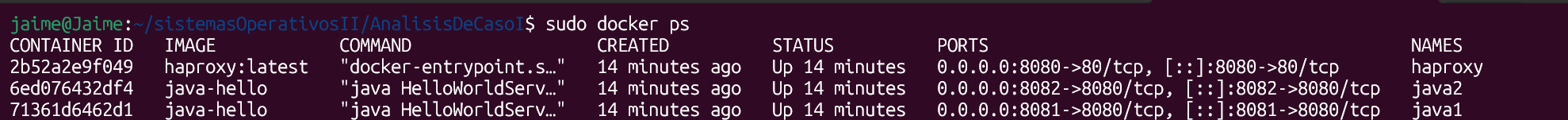
· Verificar comportamiento con navegador o curl.

Aquí comprobamos que al hacer curl nos lleva responde a veces con java1 o java2

A screenshot of a computer screen

AI-generated content may be incorrect.





· Analizar tipos de balanceo y su configuración.

El que utilizamos anteriormente fue roundrobin si se ve en la imagen va a java1 luego java2 y se repite el ciclo

Si cambiamos el roundrobin por el leastconn, manda al servidor con menos conexiones activas

A computer screen with numbers

AI-generated content may be incorrect.

Tambien existe:

source: fija al mismo backend según la IP origen útil para sesiones que requieren afinidad.

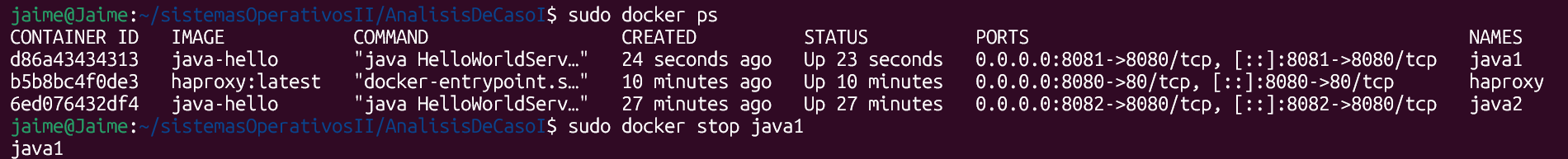
random: selecciona servidores al azar.

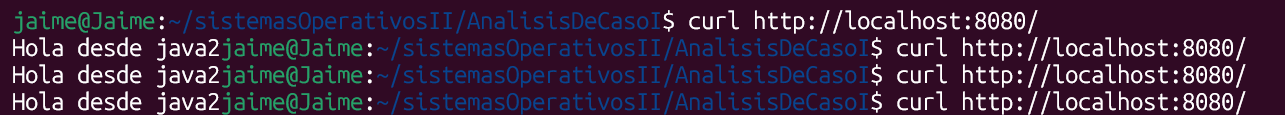
### 3. Alta disponibilidad

· Simular falla de una instancia del servidor.

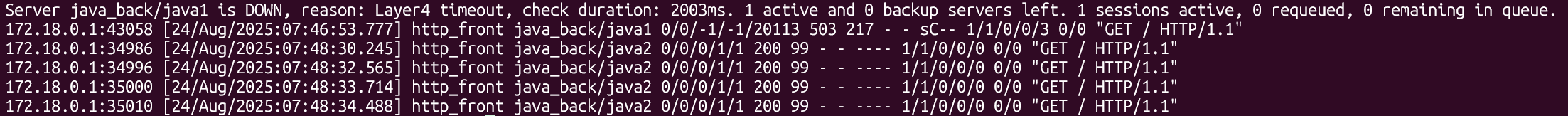
· Observar cómo HAProxy mantiene el servicio activo.

Aquí quitamos el java1 para ver si responde java2, como se puede ver el server empieza a responder solo con java2 significa que no se cayó, el usuario podrá seguir viendo la página, también se puede ver que el 8081 ya no funciona debido a paramos java1 que esta ubicado en el puerto 8081, en este caso el 8080 te dirige al que este disponible









A screenshot of a computer

AI-generated content may be incorrect.

· Explorar replicación con Docker Swarm (opcional).

# Iniciar swarm

docker swarm init

# Crear servicio con 3 réplicas

docker service create --name java-hello --replicas 3 -p 8080:8080 java-hello

### 4. Seguridad

· Implementar HTTPS (auto-firmado o Let's Encrypt).

A white dots on a black background

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

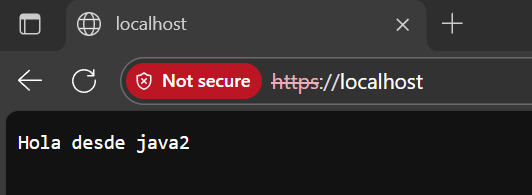
Aquí podemos ver que al correr nuevamente el haproxy con la nueva configuración ya funciona con el https ya que devuelve la página y el haproxy decide cual mostrarte si java1 o java2

A computer screen with white text

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.



· Analizar riesgos de HTTP plano.

* **Intercepción (sniffing):** Todo viaja en texto plano usuarios y contraseñas pueden ser robados.
* **Manipulación (MITM):** Un atacante puede modificar respuestas.
* **Falsificación:** Los clientes no pueden verificar que el servidor es legítimo.
* **Políticas modernas:** Navegadores y APIs bloquean funciones en HTTP.
* **HTTPS es obligatorio** para cualquier sistema serio (autenticación, datos personales, académicos).

· Examinar mecanismos de control de acceso y firewalls.

Modificamos la configuración de haproxy para que en la ruta / es decir que cuando ingresa a la página te pida credenciales y las definimos en la lista de usuarios, le ponemos insecure para que el haproxy lo valide por ahora.

A screenshot of a computer

AI-generated content may be incorrect.

Reiniciamos la configuración de haproxy



Aquí se puede ver que efectivamente nos pide la autenticación y solo se puede entrar con los usuarios definidos en la configuración del haproxy

A screenshot of a computer screen

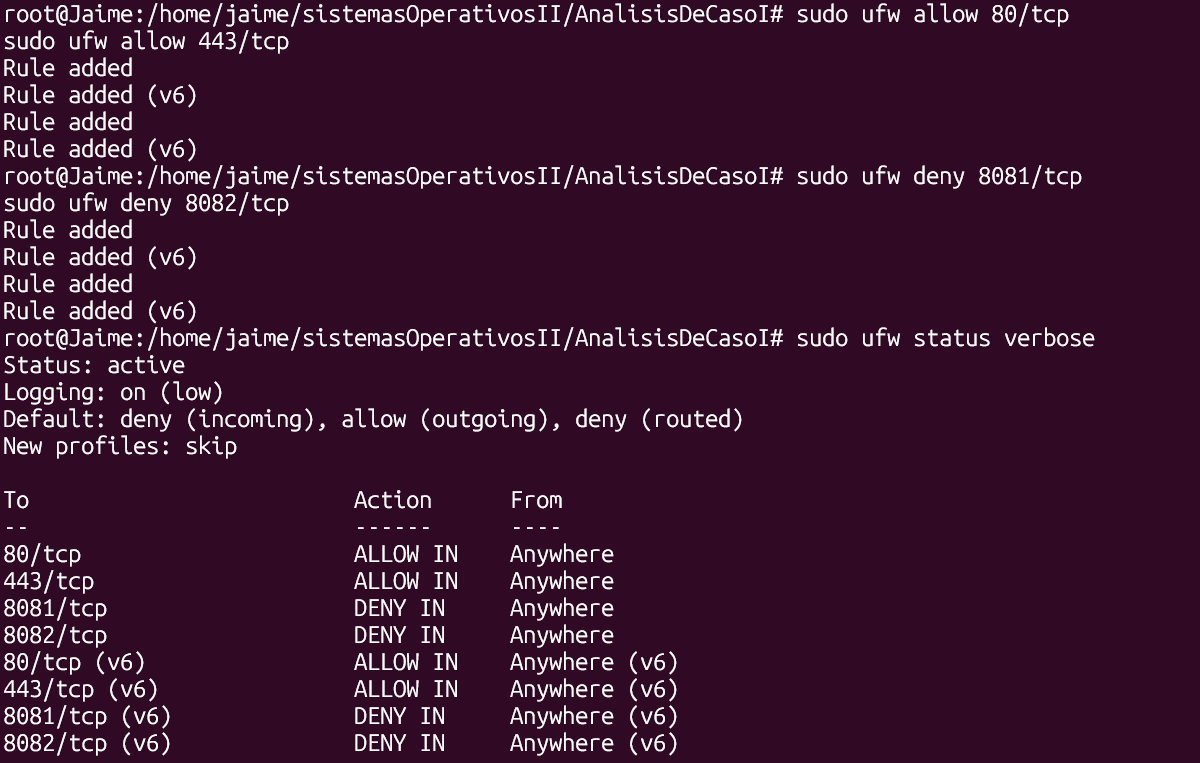
AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.



Desde otra máquina, ya no son accesibles con esos puertos, al deshabilitarlos con el ufw hacemos que no pueda acceder a la información con el puerto de java1 y java2, entonces el cliente solo tiene una entrada, es decir, que si o si solo puede acceder a la información ingresando con el usuario y la contraseña

A computer screen with white text

AI-generated content may be incorrect.

### 5. Escalabilidad y mantenimiento

· Comparar con sitios reales en Drupal/WordPress.

Java en Docker:

* Contenedores: varias instancias
* Base de datos: No requiere
* Cache: Opcional
* Balanceo: HAProxy simple
* Persistencia de datos: No necesaria
* Actualizaciones: Manual Rebuild
* Escalabilidad horizontal: Facil con Docker Swarm

Drupal / WordPress:

* Contenedores: 1 por app + DB + Cache + Storage
* Base de datos: MySQL/MariaDB/PostgreSQL
* Cache: Redis / Memcached
* Balanceo: HAProxy/Traefik
* Persistencia de datos: Obligatoria
* Actualizaciones: CI/CD
* Escalabilidad horizontal: Muy útil para sitios de alta demanda

· Discutir crecimiento horizontal en Docker Swarm.

Crecimiento horizontal significa agregar más instancias para manejar mayor tráfico.

* Docker Swarm mantiene automáticamente las réplicas activas.
* Si una falla, Swarm la reemplaza.
* HAProxy puede balancear tráfico entre todas las réplicas.

En Drupal/WordPress:

* Se puede tener múltiples contenedores php-fpm detrás de HAProxy.
* La base de datos debe ser replicada o en clúster.
* Los uploads y contenido multimedia requieren almacenamiento compartido (GlusterFS).

· Opcional: usar docker-compose.yml y diagrama de arquitectura.

Docker Compose es una herramienta de Docker que permite definir y ejecutar aplicaciones multi-contenedor usando un archivo de configuración llamado docker-compose.yml.

En lugar de levantar cada contenedor con un comando docker run separado, Compose te permite describir toda tu arquitectura en un solo archivo y ejecutarla con un solo comando.

**Ventajas de Docker Compose**

1. **Sencillez**: todo tu proyecto, incluyendo varios contenedores y redes, se define en un solo archivo.
2. **Redes automáticas**: Compose crea redes internas para que los contenedores se comuniquen entre sí sin exponer puertos innecesarios.
3. **Escalabilidad**: puedes levantar múltiples instancias de un servicio con docker-compose up --scale.
4. **Persistencia de datos**: fácil de mapear volúmenes para bases de datos u otros servicios.
5. **Reproducibilidad**: cualquier persona con Docker y el docker-compose.yml puede levantar todo el entorno igual.

## 4. Objetivos de aprendizaje ampliados

|  |  |  |
| --- | --- | --- |
| Categoría | Antes | Ahora |
| Sockets/HTTP | Conceptual | Ejecutado en contenedor, expuesto a internet |
| Puertos | Teoría | Mapas reales Docker (8080:80), firewalls |
| Concurrencia | Threads | Balanceadores con múltiples instancias |
| OS y procesos | local | Namespaces, cgroups, PID en contenedores |
| Seguridad | Conceptual | VPN, roles, SSL, CI/CD, backups |
| Disponibilidad | Un solo punto de fallo | Alta disponibilidad real (HAProxy, Keepalived) |
| Escalabilidad | No aplicable | Horizontal, Docker Swarm, orquestación |
| Persistencia | Inexistente | Volúmenes NFS, backups automáticos |

## 5. Entregables

· Informe con análisis de rendimiento, escalabilidad, seguridad y alta disponibilidad.

· Capturas de prueba (github, curl, navegador, logs de HAProxy, configuraciones, entre otros).