
Servidores Web de Altas Prestaciones



Práctica 2: Balanceo de carga



ICAR

INGENIERÍA DE COMPUTADORES,
AUTOMÁTICA Y ROBÓTICA



UNIVERSIDAD
DE GRANADA



Índice





Objetivos

Esta práctica se centrará en crear un escenario multicontenedor utilizando contenedores Docker y estrategias de balanceo de carga.

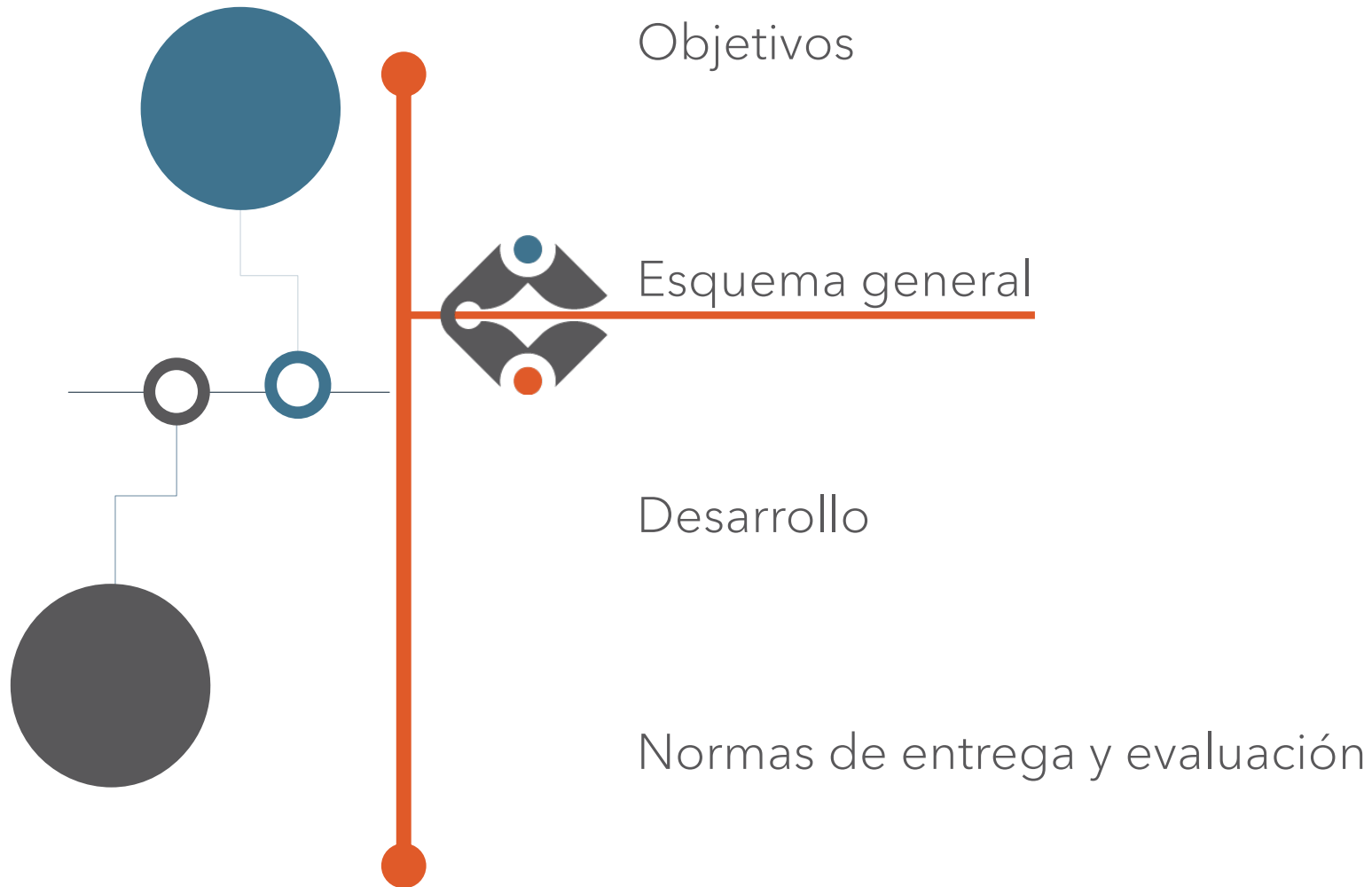
1. Aprender la configuración y uso de **nginx y HAProxy** como balanceadores de carga.
2. Implementar distintas estrategias de balanceo de carga, incluyendo round-robin, entre otras.
3. Configurar y analizar las estadísticas de balanceo de carga para nginx y HAProxy.

La práctica se realizará de manera individual. Tiene un peso del **30%** del total de prácticas.





Índice





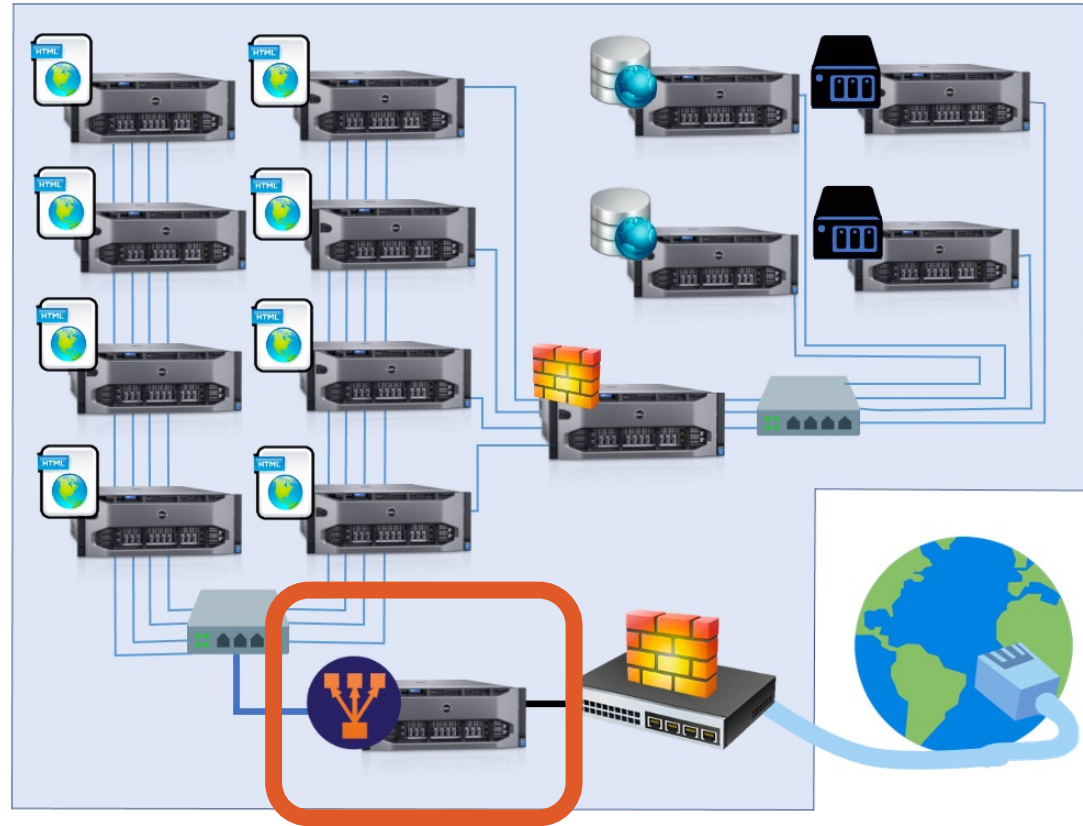
Práctica 2. Balanceo de carga



- Creación imágenes de balanceadores de carga (DockerFile)
- Configuración de balanceadores
- Habilidad de estadísticas de balanceo



2 sesiones



Requisitos Previos:

- Haber completado satisfactoriamente la Práctica 1 o tener experiencia equivalente configurando servidores web con Docker.
- Conocimientos básicos sobre estrategias de balanceo de carga.

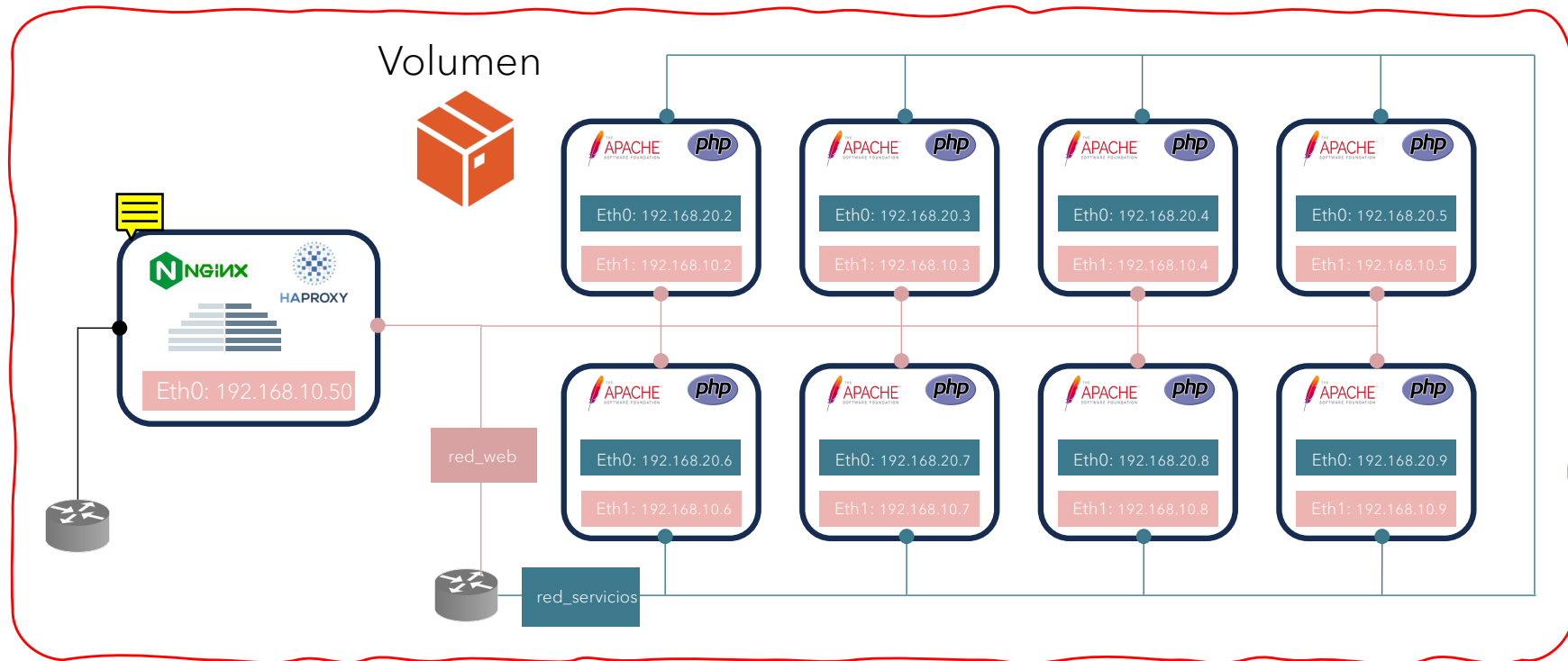




Práctica 2. Balanceo de carga



Esquema general de la práctica





Índice





Se pretende desarrollar y configurar un entorno de balanceo de carga utilizando dos de los balanceadores más populares: nginx y HAProxy.

Se crearán al menos 2 contenedores, uno para cada balanceador de carga, los cuales estarán conectados a una red específica denominada `red_web`, con la dirección IP 192.168.10.50. A través de esta configuración, se establecerán dependencias con los 8 contenedores de servidores web desplegados en la Práctica 1, permitiendo experimentar con diversas estrategias de balanceo, como round-robin, entre otras. Además, se configurarán y analizarán las estadísticas de balanceo de carga de cada balanceador.





Parte 0: Creación del directorio y espacio de trabajo para cada balanceador

En esta parte se establecerá el espacio de trabajo a través de directorios específicos donde se crearán los archivos de configuración, así como la web del escenario.

- Crea un directorio en tu máquina local llamado **P2-tuusuariougr-nginx** para trabajar con los archivos de configuración de Nginx y otro directorio llamado **P2-tuusuariougr-haproxy** para trabajar con los archivos de configuración de HAProxy.
- Copia en cada directorio de los balanceadores, es decir en **P2-tuusuariougr-nginx** y **P2-tuusuariougr-haproxy**, el directorio que creaste en la práctica 1 llamado **web_tuusuariougr** para que los servidores web sirvan el index.php que creaste en la práctica 1.





Parte 1: Configuración de Nginx como Balanceador de Carga

En esta parte configuraremos los archivos de configuración de nginx así como crearemos imágenes personalizadas con Dockerfile para la imagen de nginx.

- **Dockerfile para Nginx:**

- En el directorio **P2-tuusuariougr-nginx**, crea un archivo llamado **DockerfileNginx** para crear una imagen a partir de la imagen oficial de nginx: `FROM nginx:latest`

- **Archivo de Configuración de Nginx (nginx.conf):**

- Crea un archivo **nginx.conf** en el directorio **P2-tuusuariougr-nginx** con la configuración de balanceo de carga. A continuación, se detalla un ejemplo de configuración básica aplicando round-robin como algoritmo de balanceo y registro de accesos y errores, y habilitadas las estadísticas.






Parte 1: Configuración de Nginx como Balanceador de Carga

En esta parte configuraremos los archivos de configuración de nginx así como crearemos imágenes personalizadas con Dockerfile para la imagen de nginx.

- **Archivo de Configuración de Nginx (nginx.conf):**

```
http {  
    upstream backend_tuusuarioUGR {  
        #algoritmo balanceo (least_conn; round-robin; etc.)  
        round-robin  
        server ip_web1;  
        server ip_web2;  
        etc...  
    }  
    server{  
        ...  
    }  
}
```



```
server{  
    listen 80;  
    server_name nginx_tuusuarioUGR;  
    access_log /var/log/nginx/nginx_tuusuarioUGR.access.log;  
    error_log /var/log/nginx/nginx_tuusuarioUGR.error.log;  
    location / {  
        proxy_pass http://backend_tuusuarioUGR;  
        proxy_set_header Cookie $http_cookie;  
        proxy_hide_header Set-Cookie;  
    }  
    location /estadisticas_usuarioUGR {  
        stub_status on;  
    }  
}
```





Parte 1.1: Configuración de Nginx con Docker Compose

En esta parte se configurará Nginx como balanceador de carga para los servidores web Apache utilizando Docker Compose, con conexiones a las redes red_web y red_servicios y se establecerán dependencias entre los servicios.

- **Construcción de Servidores Web Apache:**
 - Utilizar el DockerfileApache de la Práctica 1 para construir los contenedores de Apache con el nuevo tag p2.
- **Definición de Servicios en docker-compose.yml:**
 - Definir un servicio para cada instancia de Apache con las siguientes características:
 - Imagen construida a partir del DockerfileApache de la Práctica 1.
 - Nombre de la imagen personalizada como **tuusuarioUGR-apache-image:p2**.
 - Nombre del contendedor: webX donde X es un número del 1 al 8.
 - Volumen para montar el directorio local **web_tuusuarioUGR** en la ruta por defecto de Apache para servir el index.php.
 - Conexión a las redes red_web y red_servicios.





Parte 1.1: Configuración de Nginx con Docker Compose

En esta parte se configurará Nginx como balanceador de carga para los servidores web Apache utilizando Docker Compose, con conexiones a las redes `red_web` y `red_servicios` y se establecerán dependencias entre los servicios.

- **Definir un servicio llamado `balanceador-nginx` que incluya:**
 - Construcción de la imagen a partir del `DockerFileNginx`.
 - Nombre de la imagen personalizada como `tuusuarioUGR-nginx-image:p2`.
 - Volumen para montar el archivo `nginx.conf` en el contenedor en `/etc/nginx/nginx.conf`.
 - Asignación de una dirección IP estática `192.168.10.50` en la red `red_web`.
 - Dependencia establecida con los servicios de Apache para garantizar el orden correcto de despliegue.

NOTA: Si declaras una red como "external" en el archivo `docker-compose.yml`, le estás indicando a Docker Compose que no gestione la red (ni la creación ni la eliminación) porque es una red creada fuera de Docker Compose, y deberías haberla creado manualmente con anterioridad usando "docker network create" o debería estar siendo gestionada por otro proyecto de Docker Compose.





Parte 1.2: Verificación y pruebas del balanceador Nginx

En esta parte realizaremos el despliegue del escenario con el balanceador nginx y los servidores web finales, así como verificaremos el correcto funcionamiento del escenario.

- **Despliegue y Ejecución:**

- Ejecutar `docker-compose up -d` para iniciar todos los servicios definidos en el archivo.
- Verificar que los servicios están corriendo y que Nginx está balanceando carga.

- **Verificación y Pruebas:**

- Comprobar que Nginx distribuye correctamente las solicitudes entre los distintos contenedores de Apache.



Práctica SWAP - José Manuel Soto Hidalgo



La dirección IP del servidor es: 192.168.10.2





Parte 1.2: Verificación y pruebas del balanceador Nginx

En esta parte realizaremos el despliegue del escenario con el balanceador nginx y los servidores web finales, así como verificaremos el correcto funcionamiento del escenario.

- **Verificación y Pruebas:**

- Acceder a la página de estadísticas de Nginx que hemos configurado previamente (/estadisticas_**tuusuarioUGR**) para observar el rendimiento del balanceador.

```
← → ↻ ⓘ localhost/estadisticas_jmsoto 📄 ☆  
  
Active connections: 2  
server accepts handled requests  
2 2 22  
Reading: 0 Writing: 1 Waiting: 1
```





Parte 2: Configuración de HAProxy como Balanceador de Carga

En esta sección, configuraremos HAProxy para que funcione como balanceador de carga, estableciendo su configuración y preparando una imagen personalizada con Dockerfile. Recuerda que debes tener parado el escenario con Nginx

- **Dockerfile para HAProxy:**
 - Dentro del directorio **P2-tuusuariougr-haproxy**, crea un archivo llamado **DockerfileHAProxy**.
 - Usa la imagen oficial de HAProxy como punto de partida: **FROM haproxy:latest**.
 - Agrega instrucciones para copiar la configuración personalizada de HAProxy al contenedor.
- **Archivo de Configuración de HAProxy (haproxy.cfg):**
 - Crea un archivo **haproxy.cfg** en el directorio **P2-tuusuariougr-haproxy** donde añadir las configuraciones necesarias.
 - Configura HAProxy para utilizar algoritmos de balanceo de carga, como round-robin.





Parte 2: Configuración de HAProxy como Balanceador de Carga

En esta sección, configuraremos HAProxy para que funcione como balanceador de carga, estableciendo su configuración y preparando una imagen personalizada con Dockerfile.

- **Archivo de Configuración de HAProxy (haproxy.cfg):**

frontend frontend **usuarioUGR**

bind *:80

default_backend **backend_usuarioUGR**

backend **backend_usuarioUGR**

server web1 192.168.10.2:80 **maxconn 32**

server web2 192.168.10.3:80 maxconn 32

etc...





Parte 2: Configuración de HAProxy como Balanceador de Carga

En esta sección, configuraremos HAProxy para que funcione como balanceador de carga, estableciendo su configuración y preparando una imagen personalizada con Dockerfile.

- **Archivo de Configuración de HAProxy (haproxy.cfg):**

- Activa las estadísticas de HAProxy para facilitar el monitoreo de su rendimiento. Ejemplo básico de configuración de estadísticas en el puerto 9000:

```
global
stats socket /var/lib/haproxy/stats
listen stats
    bind *:9000
    mode http
    stats enable
    stats uri /estadisticas_tuusuarioUGR
    stats realm HAProxy\ Statistics
    stats auth tuusuarioUGR:SWAP1234
```





Parte 2.1: Configuración de HAProxy con Docker Compose

En esta sección, estableceremos HAProxy como balanceador de carga para los servidores web con Docker Compose, definiendo las conexiones de red y dependencias de servicios.

- **Construcción de Servidores Web Apache:**
 - Utiliza el **DockerfileApache** de la Práctica 1 para construir las imágenes de los contenedores Apache con el nuevo tag **p2**.
- **Definición de Servicios en docker-compose.yml:**
 - Definir un servicio para cada instancia de Apache con las siguientes características:
 - Imagen construida a partir del DockerfileApache de la Práctica 1.
 - Nombre de la imagen personalizada como **tuusuarioUGR-apache-image:p2**.
 - Nombre del contendedor: **webX** donde X es un número del 1 al 8.
 - Volumen para montar el directorio local **web_tuusuarioUGR** en la ruta por defecto de Apache para servir el index.php.
 - Conexión a las redes **red_web** y **red_servicios**.





Parte 2.1: Configuración de HAProxy con Docker Compose

En esta sección, estableceremos HAProxy como balanceador de carga para los servidores web con Docker Compose, definiendo las conexiones de red y dependencias de servicios.

- **Definición de Servicios en docker-compose.yml:**
 - Definir un servicio para el balanceador llamado balanceador-haproxy:
 - Construcción de la imagen a partir del DockerfileHAProxy.
 - Personaliza el nombre de la imagen como **tuusuarioUGR-haproxy-image:p2**.
 - Monta el archivo **haproxy.cfg** en el contenedor en **/usr/local/etc/haproxy/haproxy.cfg**.
 - Asigna la dirección IP estática **192.168.10.50** en la red **red_web**.
 - Establece la dependencia con los servicios de Apache para garantizar el correcto despliegue secuencial.

*Nota sobre las redes Docker Compose: Como en la configuración de Nginx, si las redes ya están creadas externamente, marca las redes **red_web** y **red_servicios** como externas en tu **docker-compose.yml** para evitar conflictos o intentos de recreación.*





Parte 2.2: Verificación y pruebas del balanceador HAProxy

En esta parte realizaremos el despliegue del escenario con el balanceador HAProxy y los servidores web finales, así como verificaremos el correcto funcionamiento del escenario.

- **Despliegue y Ejecución:**

- Ejecutar `docker-compose up -d` para iniciar todos los servicios definidos en el archivo.
- Verificar que los servicios están corriendo y que HAProxy está balanceando carga.

- **Verificación y Pruebas:**

- Comprobar que HAProxy distribuye correctamente las solicitudes entre los distintos contenedores de Apache.



Práctica SWAP - José Manuel Soto Hidalgo

La dirección IP del servidor es: 192.168.10.2





Parte 2.2: Verificación y pruebas del balanceador HAProxy

En esta parte realizaremos el despliegue del escenario con el balanceador HAProxy y los servidores web finales, así como verificaremos el correcto funcionamiento del escenario.

- **Verificación y Pruebas:**

- Acceder a la página de estadísticas de HAProxy (/estadisticas_**tuusuarioUGR**) para observar el rendimiento del balanceador.

localhost:9000/estadisticas_jmsoto

Iniciar sesión
http://localhost:9000

Nombre de usuario

Contraseña

localhost:9000/estadisticas_jmsoto

HAProxy version 2.3.21-3ce4ee0, released 2022/07/27

Statistics Report for pid 8

> General process information

pid = 8 (process #1, nbproc = 1, nbthread = 10)
uptime = 0d 0h01m34s
system limits: memmax = unlimited; ulimit-n = 1048576
maxsock = 1048576; maxconn = 524257; maxpipes = 0
current conns = 1; current pipes = 0/0; conn rate = 1/sec; bit rate = 0.000 kbps
Running tasks: 0/24; idle = 100 %

active UP

active UP, going down

active DOWN, going up

active or backup DOWN

active or backup DOWN for maintenance (MAINT)

active or backup SOFT STOPPED for maintenance

backup UP

backup UP, going down

backup DOWN, going up

not checked

Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

Display option:
• Scope :
• [Hide "DOWN" servers](#)
• [Disable refresh](#)
• [Refresh now](#)
• [CSV export](#)
• [JSON export \(schema\)](#)

External resources:
• [Primary site](#)
• [Updates \(v2.3\)](#)
• [Online manual](#)

state																					
	Queue			Session rate			Sessions				Bytes		Denied	Errors		Warnings		Server			
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis
Frontend				1	2	-	1	2	524 257	13			5 103	196 916	0	0	6				
																					OPEN

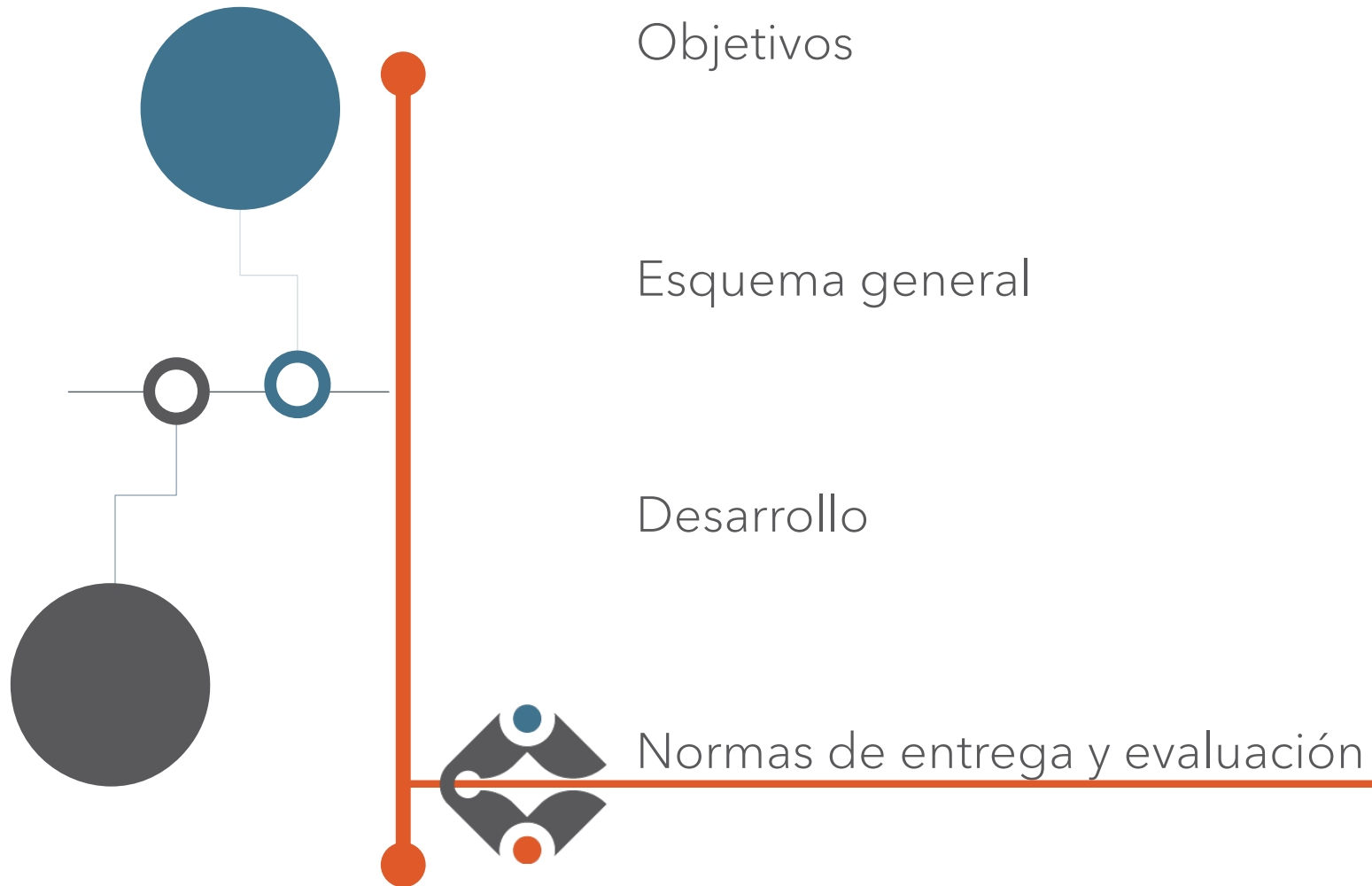
frontend_jmsoto																					
	Queue			Session rate			Sessions				Bytes		Denied	Errors		Warnings		Server			
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis
Frontend				0	2	-	0	2	524 257	2			11 185	8 032	0	0	1				
																					OPEN

backend_jmsoto																					
	Queue			Session rate			Sessions				Bytes		Denied	Errors		Warnings		Server			
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis
web1	0	0	-	0	1		0	1	-	3	3	11s	1 862	1 299	0	0	0	0	0	0	1m34s UP
web2	0	0	-	0	1		0	1	-	3	3	11s	1 787	1 302	0	0	0	0	0	0	1m34s UP
web3	0	0	-	0	1		0	1	-	2	2	12s	1 256	866	0	0	0	0	0	0	1m34s UP
web4	0	0	-	0	1		0	1	-	2	2	12s	1 256	866	0	0	0	0	0	0	1m34s UP
web5	0	0	-	0	1		0	1	-	2	2	12s	1 256	866	0	0	0	0	0	0	1m34s UP
web6	0	0	-	0	1		0	1	-	2	2	12s	1 256	866	0	0	0	0	0	0	1m34s UP
web7	0	0	-	0	1		0	1	-	2	2	12s	1 256	864	0	0	0	0	0	0	1m34s UP
web8	0	0	-	0	1		0	1	-	2	2	12s	1 256	866	0	0	0	0	0	0	1m34s UP
Backend	0	0		0	6		0	1	52 426	18	18	11s	11 185	7 795	0	0	0	0	0	0	1m34s UP





Índice





Normas de entrega y evaluación

Para superar la práctica se deben realizar las siguientes tareas básicas:

B1: Preparación del Entorno de Trabajo

- Crear directorios específicos para los archivos de configuración de los balanceadores:
 - P2-tuusuariougr-nginx para nginx.
 - P2-tuusuariougr-haproxy para HAProxy.

B2: Configuración de Nginx como Balanceador de Carga

- Redactar el Dockerfile para crear una imagen personalizada de Nginx a partir de la imagen oficial.
- Escribir el archivo de configuración nginx.conf con la estrategia de balanceo round-robin y configuraciones de registro de accesos y errores.

B3: Implementación del escenario de Ninx con Docker Compose

- Reutilizar o adaptar el DockerfileApache de la Práctica 1 para los contenedores de Apache con el nuevo tag p2.
- Desarrollar el docker-compose.yml para configurar el servicio para cada contenedor Apache, el volumen para el directorio web_tuusuariougr, y para el balanceador de carga Nginx con las características correspondientes.





Normas de entrega y evaluación

Para superar la práctica se deben realizar las siguientes tareas básicas:

B4: Verificación y Pruebas del escenario de Nginx

- Desplegar los servicios con docker-compose up -d.
- Verificar que los servicios están activos y que Nginx distribuye correctamente las solicitudes.
- Acceder a la página de estadísticas de Nginx para observar el rendimiento del balanceador.

B5: Configuración de HAProxy como Balanceador de Carga

- Redactar el Dockerfile para crear una imagen de HAProxy a partir de la imagen oficial.
- Crear el archivo de configuración haproxy.cfg incluyendo las estrategias de balanceo de carga y la configuración de las estadísticas.

B6: Implementación del escenario de HAProxy con Docker Compose

- Reutilizar o adaptar el DockerfileApache de la Práctica 1 para los contenedores de Apache con el nuevo tag p2.
- Detallar en docker-compose.yml la configuración de cada servicio Apache, el volumen para el directorio web_tuusuariougr, y el servicio para el balanceador de carga HAProxy.





Normas de entrega y evaluación

Para superar la práctica se deben realizar las siguientes tareas básicas:

B7: Verificación y Pruebas del escenario de HAProxy

- Iniciar los servicios con `docker-compose up -d` asegurando que HAProxy esté operativo.
- Comprobar la correcta distribución de solicitudes por parte de HAProxy.
- Observar el rendimiento a través de la interfaz de estadísticas configurada.





Normas de entrega y evaluación

Se proponen, opcionalmente, las siguientes tareas avanzadas:

A1: Configuraciones Avanzadas de Nginx



- Implementar estrategias de balanceo de carga avanzadas en `nginx.conf`, como el balanceo basado en menor tiempo de respuesta o ponderado y analiza el impacto de esas configuraciones en el escenario de balanceo.

A2: Configuraciones Avanzadas de HAProxy

- Implementar estrategias de balanceo de carga avanzadas en `haproxy.cfg`, como el balanceo basado en menor tiempo de respuesta o ponderado y analiza el impacto de esas configuraciones en el escenario de balanceo.

A3: Experimentación con Diferentes Balanceadores de Carga

- Configurar y desplegar otros balanceadores de carga disponibles en el ecosistema Docker, como Traefik, gobetween o balanceadores personalizados.



A4: Investigación y Pruebas de Tolerancia a Fallos

- Realizar pruebas de tolerancia a fallos apagando intencionadamente instancias de servidor web para observar la reacción y la reasignación de carga de los balanceadores.





Normas de entrega y evaluación

Se proponen, opcionalmente, las siguientes tareas avanzadas:

A5: Automatización de escalado del escenario

- Implementar una lógica de escalado automático para añadir o eliminar instancias de servidor web basadas en la carga, horarios específicos o consumo de recursos utilizando Docker Compose. Esta idea podría realizarse mediante:

1. Monitorización de la Carga de los Servidores:

- Configura métricas de monitoreo (CPU, memoria, conexiones activas, etc.). para ver el estado de cada contenedor de servidor web. Puedes usar herramientas como cAdvisor, Node Exporter o agentes personalizados que expongan las métricas a un sistema de monitoreo.

2. Script para escalar servicios:

- Escribe un script que utilice la API de Docker para obtener las métricas y ajuste la configuración del balanceador de carga en función de estas métricas. Este script podría ajustar el archivo **nginx.conf** o **haproxy.cfg** para añadir o eliminar servidores del backend en el balanceador y luego recargar la configuración del balanceador sin interrumpir el tráfico.

3. Automatización del script:

- Configura que el script se ejecute a intervalos regulares, por ejemplo, cada 2 minutos.





Normas de entrega y evaluación

Se proponen, opcionalmente, las siguientes tareas avanzadas:

A5: Automatización de escalado del escenario

- Implementar una lógica de escalado automático para añadir o eliminar instancias de servidor web basadas en la carga, horarios específicos o consumo de recursos utilizando Docker Compose. Esta idea podría realizarse mediante:



Recuerda que la implementación real de este sistema sería bastante compleja y requeriría pruebas exhaustivas. En entornos de producción, es recomendado utilizar plataformas como Kubernetes o Docker Swarm diseñadas para orquestación que ofrecen funcionalidades de autoescalado y autohealing integradas.





Normas de entrega y evaluación

Se desarrollará un documento siguiendo el guion de la práctica y **detallando** e indicando, en su caso, los **aspectos básicos y avanzados realizados**, comandos de terminal ejecutados, resultados de ejecución, etc.

- Por ejemplo, si se ha realizado la tarea básica de configuración del entorno, el documento .pdf con la memoria de prácticas debe aparecer una sección titulada: *Tareas Básicas - B1. Configuración del Entorno* donde aparezcan detalladas las configuraciones. De igual forma, si por ejemplo, se han realizado tareas avanzadas sobre automatizaciones con Scripts, debe aparecer *Tareas Avanzadas - A1. Configuraciones Avanzadas con Nginx*, detalles de las configuraciones, explicaciones sobre ellas y explicaciones sobre ellas.

Se recomienda utilizar herramientas de control de Tiempo (por ejemplo, clockify) para contabilizar el tiempo de dedicado a la realización de la práctica.

Se deja a **libre elección** la **estructura y formato** del documento el cual reflejará el correcto desarrollo de la práctica a modo de diario/tutorial siguiendo los puntos descritos anteriormente. Asimismo, se recomienda incluir capturas de pantalla que reflejen el correcto desarrollo de los distintos apartados de la práctica. La **primera página** del documento debe incluir, al menos, **nombre, apellidos y tiempo dedicado a la práctica** medido con herramientas de control de tiempo.





Normas de entrega y evaluación

Para la entrega se habilitará una tarea en PRADO cuya entrega debe seguir **OBLIGATORIAMENTE** el formato especificado.

1. Un archivo **.pdf** con el documento desarrollado siguiendo el formato **ApellidosNombreP2.pdf**
2. Un archivo **.zip** con los distintos archivos de configuraciones, carpetas, etc. necesarios para la ejecución de la práctica siguiendo el formato **ApellidosNombreP2.zip**

Uso de Inteligencia Artificial Generativa

Para cada práctica es **OBLIGATORIO** usar herramientas de IA generativa (ChatGPT, Copilot u otras) e incluir enlace al chat/prompt utilizado. También se debe analizar y justificar el resultado que proporciona la herramienta con el resultado final que opta el estudiante para la práctica.

Es **OBLIGATORIO** incluir en el guion una sección titulada: **"Análisis propuesta IA"** donde se incluya enlace al chat/prompt con las consulta/as realizada/as, resultado que proporciona la IA y un párrafo con un análisis crítico y detallado del resultado proporcionado.





Normas de entrega y evaluación

La práctica se realizará de manera individual. Tiene un peso del **30%** del total de prácticas.

La práctica se evaluará mediante el uso de rúbrica específica (accesible por el estudiante en la tarea de entrega) y una defensa final de prácticas.

Cuestiones sobre la calificación obtenida en cada práctica se realizarán **UNICAMENTE** en la sesión dedicada a recuperación/defensa al final de curso.

La detección de prácticas copiadas implicará el suspenso inmediato de todos los implicados en la copia (tanto del autor del original como de quien las copió). **OBLIGATORIO ACEPTAR LICENCIA EULA DE TURNITIN** en la entrega. Si la memoria supera un 40% de copia Turnitin implicará el suspenso automáticamente.



Servidores Web de Altas Prestaciones



Práctica 2: Balanceo de carga



ICAR

INGENIERÍA DE COMPUTADORES,
AUTOMÁTICA Y ROBÓTICA



UNIVERSIDAD
DE GRANADA