CDPS:
Práctica creativa 2

Despliegue de una aplicación escalable

Ángel Domínguez, Javier Velázquez, Lucía Gutiérrez
Grupo 27

18 de enero de 2021

Índice

Índice de figuras

1. Introducción

El objetivo de este proyecto es el despliegue de una arquitectura completa de una aplicación de un juego de quizzes con el fin de afianzar y complementar los conocimientos teóricos que se han impartido a lo largo de la asignatura.

La arquitectura básica del proyecto cuenta con un firewall, un balanceador de tráfico, tres servidores web, una base de datos y tres sistemas de almacenamiento NAS.

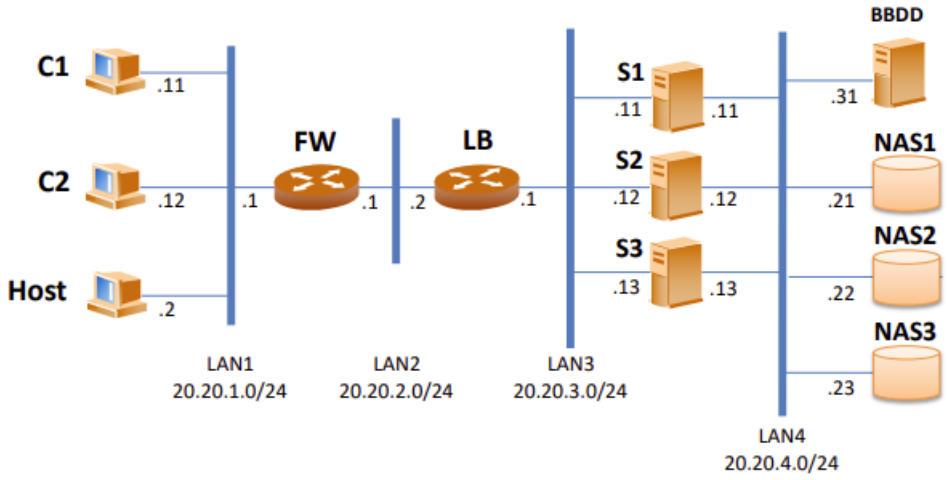


Figura 1: Arquitectura básica de la aplicación

El funcionamiento mínimo y necesario es el siguiente:

- **Firewall:** configurado a través de la aplicación FWbuilder, el firewall debe filtrar todo el tráfico proveniente de Internet y dejar pasar únicamente el destinado a la aplicación.
- **Balanceador de trafico:** debe balancear el tráfico entre los tres servidores utilizando round-robin. Además utiliza la función de mapear el puerto 3000 donde corren los servidores web al puerto 80.
- **Servidores web:** deben estar accesibles desde el cliente por el puerto http (80).
- **Base de datos:** debe almacenar la información que maneja el juego del quiz, se utilizará MariaDB como gestor. Es externa al servidor web.
- **Sistemas de almacenamiento NAS:** debe almacenar las imágenes utilizadas por el juego del quiz, se debe utilizar GlusterFS y que se replique la información en los tres.

2. Funcionamiento

En primer lugar, el proyecto está pensado para desplegarlo en el laboratorio de la universidad, se recomienda descomprimir el ZIP con todos los ficheros en /mnt/tmp/pc2 después de haber descomprimido el escenario que se proporcionaba y reemplazando todos los ficheros del ZIP por los que hay en esa carpeta, debe tener en cuenta que una vez apagado el ordenador se borra el contenido de este directorio. Se tomó esta decisión por falta de recursos en los ordenadores personales de los integrantes.

Debe tener en cuenta que si intenta desplegar el proyecto en una máquina Ubuntu puede tener problemas con algún comando adaptados al entorno del laboratorio. A continuación, se describe la manera de uso:

- **Crear el escenario:**

  Se debe tener en cuenta que a la hora de crear el escenario se ha utilizado una arquitectura distinta a la básica (Ver apartado de mejoras) por lo que es necesario revisar que se está utilizando pc2.xml que se incluye en el archivo ZIP.

  - Tres servidores de aplicación: python3 pc2.py crear

  - Cuatro servidores de aplicación (La mejora se explica en el apartado de mejoras): python3 pc2.py crear s4

- **Arrancar el escenario:** python3 pc2.py arrancar
- **Parar el escenario:** python3 pc2.py parar
- **Destruir el escenario:** python3 pc2.py destruir
- **Configurar el escenario básico:** python3 pc2.py configuracionbasica

  Por comodidad se configuran también las mejoras del firewall, servidor 4 (opcional) y mejora del algoritmo round robin.

- **Configurar Nagios:** python3 pc2.py nagios

  Ver apartado mejoras.

- **Configurar servidor de gestión:** python3 pc2.py gestion

  Ver apartado mejoras.

- **Configurar servidor de logs:** python3 pc2.py logs

  Ver apartado mejoras.

3. Mejoras

Cabe mencionar que se ha modificado la pc2.xml proporcionada para realizar algunas de las siguientes mejoras y se utilizará en todos los casos del escenario. En ella se han añadido tres máquinas nuevas que se corresponden con el servidor de gestión, nagios y el servidor de consolidación de logs. A continuación se muestra la arquitectura final del escenario (falta incluir el servidor número cuatro pero al ser opcional lo hemos definido en un xml a parte).

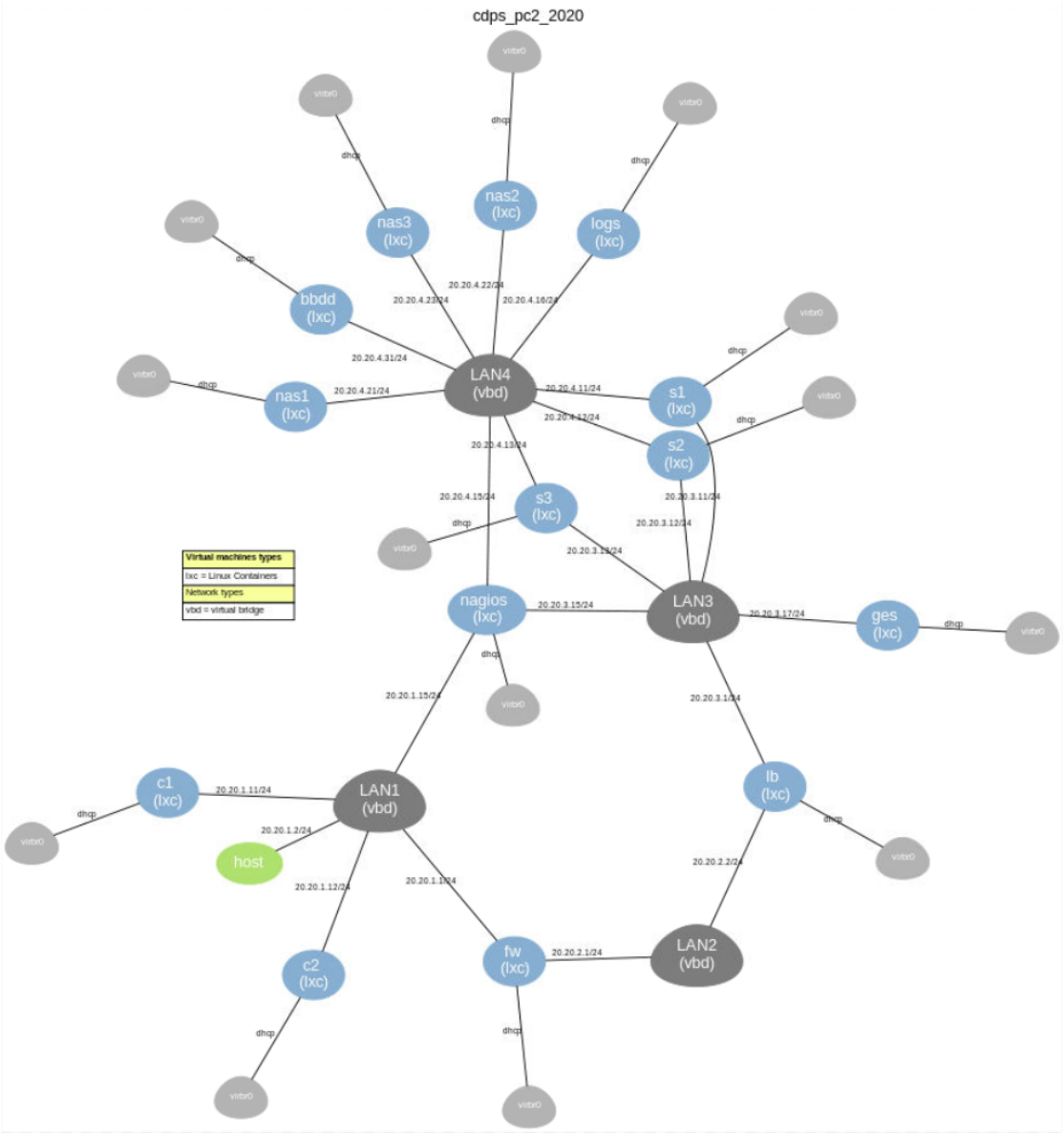


Figura 2: Arquitectura final de la aplicación

3.1. Firewall

El firewall funcionará como cortafuegos y solo va permitir el siguiente tráfico:

- Se permite el ping desde cualquier lugar al puerto 80 del balanceador.
- Se permite el acceso al puerto 80 del balanceador.
- Se permite el acceso al puerto 8001 del balanceador. En un escenario real este puerto debería estar filtrado pero lo hemos habilitado para demostrar el correcto funcionamiento de este a través de la interfaz gráfica.

- Únicamente se permite el acceso por SSH desde los clientes al servidor de gestión, quedando bloqueadas el resto. Para ello hemos configurado las siguientes reglas:

	Source	Destination	Service	Interface	Direction	Action	Time	Options	Comment
0	LAN1	ges	ssh	Any	Both	Accept	Any	log	Permite el acceso mediante ssh a la direccion del servidor de g
1	Any	lb	any ICMP	Any	Both	Accept	Any	log	Deja pasar los mensaje ICMP desde Internet al balanceador
2	Any	lb	http HaProxy	Any	Both	Accept	Any	log	Permite acceso al servidor a traves del balanceador y al puerto
3	fw Redes_seguras	fw	ssh X11	Any	Both	Accept	Any	log	Permite el acceso a la gestión del firewall desde las redes segu
4	fw	Any	Any	Any	Both	Accept	Any	log	Permite el acceso desde el fw a cualquier otro sistema
5	Red Interna	Any	ssh http	Any	Both	Accept	Any	log	Permite el acceso desde la red interna a la DMZ por ssh y http
6	Any	Any	Any	Any	Both	Deny	Any	log	Prohibe cualquier otro tráfico

Figura 3: Configuración de reglas firewall

Además se muestra un escaneo de puertos para demostrar el correcto funcionamiento. Vemos que al no tener acceso mediante ping reconoce los servidores como apagados.

```
root@c1:~# nmap -F 20.20.3.1-20

Starting Nmap 7.60 ( https://nmap.org ) at 2021-01-09 12:56 UTC
Nmap done: 20 IP addresses (0 hosts up) scanned in 17.15 seconds
```

Figura 4: Mapeo de puertos servidores

Vemos de nuevo el bloqueo del ping en el servidor numero uno.

```
root@c1:~# nmap -F 20.20.3.11

Starting Nmap 7.60 ( https://nmap.org ) at 2021-01-09 12:57 UTC
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 3.11 seconds
```

Figura 5: Mapeo de puertos servidor numero uno

Sin embargo, el balanceador tiene todos los puertos filtrados menos los especificados anteriormente.

```
root@c1:~# nmap 20.20.2.2

Starting Nmap 7.60 ( https://nmap.org ) at 2021-01-09 12:54 UTC
Nmap scan report for lb (20.20.2.2)
Host is up (0.00017s latency).
Not shown: 998 filtered ports
PORT     STATE SERVICE
80/tcp   open  http
8001/tcp open  vcom-tunnel

Nmap done: 1 IP address (1 host up) scanned in 18.78 seconds
```

Figura 6: Mapeo de puertos LB

Por último, hemos habilitado el ping momentaneamente para poder comprobar el correcto filtrado de los servidores y a la vez observar que la conexión SSH al servidor de gestión está habilitada.

```
root@c1:~# nmap -F 20.20.3.11-17

Starting Nmap 7.60 ( https://nmap.org ) at 2021-01-09 19:06 UTC
Nmap scan report for s1 (20.20.3.11)
Host is up (0.000034s latency).
All 100 scanned ports on s1 (20.20.3.11) are filtered

Nmap scan report for s2 (20.20.3.12)
Host is up (0.00013s latency).
All 100 scanned ports on s2 (20.20.3.12) are filtered

Nmap scan report for s3 (20.20.3.13)
Host is up (0.000067s latency).
All 100 scanned ports on s3 (20.20.3.13) are filtered

Nmap scan report for 20.20.3.17
Host is up (0.000028s latency).
Not shown: 99 filtered ports
PORT   STATE SERVICE
22/tcp open  ssh
```

Figura 7: Mapeo de puertos SSH

En definitiva, observamos que se corresponde con el tráfico descrito en las reglas.

3.2. Nagios

Para la monitorización en tiempo real del escenario hemos desplegado un servidor Nagios. Para ello, en primer lugar hemos instalado el paquete principal y el los plugins, a continuación hemos configurado el fichero nagios.cfg para incorporar las máquinas que queremos monitorizar, por último, hemos creado una carpeta servers dentro de las carpetas de configuración de Nagios donde configuramos los servicios que queremos monitorizar de cada máquina. A continuación se muestra el mapa que obtenemos de la interfaz gráfica de Nagios donde vemos las máquinas que estamos monitorizando.

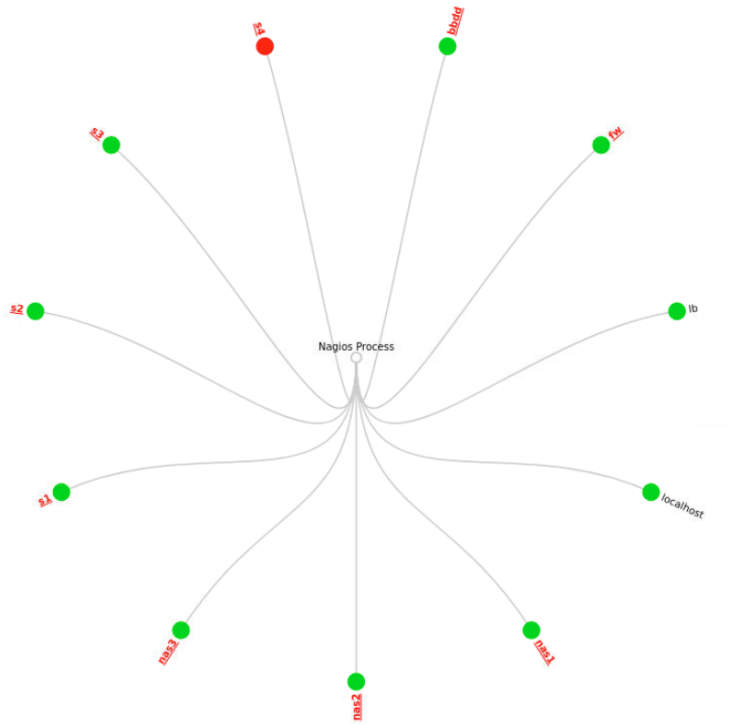


Figura 8: Mapa monitorización Nagios

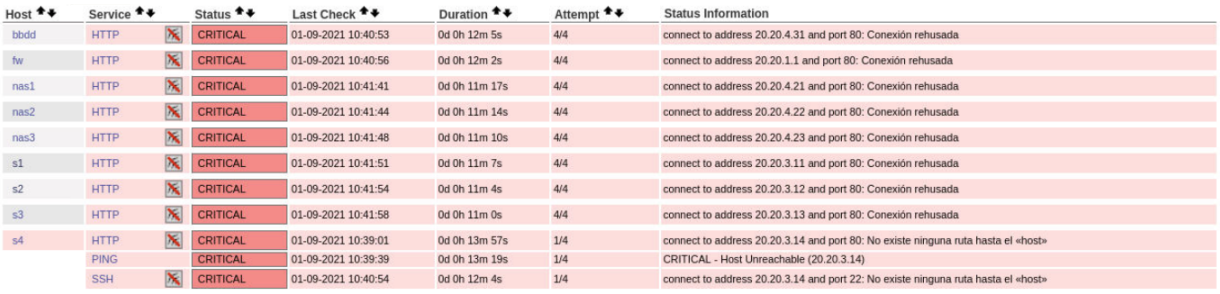
Podemos ver que la única máquina que se encuentra en verde es el localhost y el balanceador esto se debe a que hemos configurado 8 servicios en los cuales encontramos entre ellos el servicio http en el puerto 80 y la única máquina que está utilizando este puerto es el balanceador como se ha comentado anteriormente. También podemos ver que el servidor numero cuatro se encuentra en rojo ya que la máquina se encuentra apagada en el momento de la captura, demostrando el correcto funcionamiento de la monitorización.

En la siguiente imagen podemos ver los servicios que están disponibles y están funcionando correctamente en algunas de las máquinas de nuestro escenario, como se comentó, están disponibles todo menos el acceso HTTP exceptuando la máquina del balanceador.

nas1	Current Load	OK	01-09-2021 10:37:26	0d 0h 14m 40s	1/4	OK - load average: 0.37, 0.75, 1.08
	Current Users	OK	01-09-2021 10:38:03	0d 0h 14m 3s	1/4	USERS OK - 0 users currently logged in
	PING	OK	01-09-2021 10:39:18	0d 0h 12m 48s	1/4	PING OK - Packet loss = 0%, RTA = 0.07 ms
	Root Partition	OK	01-09-2021 10:39:56	0d 0h 12m 10s	1/4	DISK OK - free space: / 421365 MB (97,53% inode=99%):
	SSH	OK	01-09-2021 10:40:33	0d 0h 11m 33s	1/4	SSH OK - OpenSSH_7.6p1 Ubuntu-4ubuntu0.3 (protocol 2.0)
	Swap Usage	OK	01-09-2021 10:41:01	0d 0h 11m 5s	1/4	SWAP OK - 100% free (3813 MB out of 3813 MB)
	Total Processes	OK	01-09-2021 10:41:52	0d 0h 15m 14s	1/4	PROCS OK: 19 processes with STATE = RSZDT
nas2	Current Load	OK	01-09-2021 10:37:29	0d 0h 14m 37s	1/4	OK - load average: 0.37, 0.75, 1.08
	Current Users	OK	01-09-2021 10:38:07	0d 0h 13m 59s	1/4	USERS OK - 0 users currently logged in
	PING	OK	01-09-2021 10:39:22	0d 0h 12m 44s	1/4	PING OK - Packet loss = 0%, RTA = 0.10 ms
	Root Partition	OK	01-09-2021 10:39:59	0d 0h 12m 7s	1/4	DISK OK - free space: / 421365 MB (97,53% inode=99%):
	SSH	OK	01-09-2021 10:40:37	0d 0h 11m 29s	1/4	SSH OK - OpenSSH_7.6p1 Ubuntu-4ubuntu0.3 (protocol 2.0)
	Swap Usage	OK	01-09-2021 10:41:01	0d 0h 11m 5s	1/4	SWAP OK - 100% free (3813 MB out of 3813 MB)
	Total Processes	OK	01-09-2021 10:41:55	0d 0h 15m 11s	1/4	PROCS OK: 19 processes with STATE = RSZDT
nas3	Current Load	OK	01-09-2021 10:37:33	0d 0h 14m 33s	1/4	OK - load average: 0.34, 0.74, 1.07
	Current Users	OK	01-09-2021 10:38:10	0d 0h 13m 56s	1/4	USERS OK - 0 users currently logged in
	PING	OK	01-09-2021 10:39:25	0d 0h 12m 41s	1/4	PING OK - Packet loss = 0%, RTA = 0.11 ms
	Root Partition	OK	01-09-2021 10:40:03	0d 0h 12m 3s	1/4	DISK OK - free space: / 421365 MB (97,53% inode=99%):
	SSH	OK	01-09-2021 10:40:40	0d 0h 11m 26s	1/4	SSH OK - OpenSSH_7.6p1 Ubuntu-4ubuntu0.3 (protocol 2.0)
	Swap Usage	OK	01-09-2021 10:41:01	0d 0h 11m 5s	1/4	SWAP OK - 100% free (3813 MB out of 3813 MB)
	Total Processes	OK	01-09-2021 10:41:58	0d 0h 15m 8s	1/4	PROCS OK: 19 processes with STATE = RSZDT
s1	Current Load	OK	01-09-2021 10:37:36	0d 0h 14m 30s	1/4	OK - load average: 0.31, 0.73, 1.07
	Current Users	OK	01-09-2021 10:38:13	0d 0h 13m 53s	1/4	USERS OK - 0 users currently logged in
	PING	OK	01-09-2021 10:39:28	0d 0h 12m 38s	1/4	PING OK - Packet loss = 0%, RTA = 0.07 ms
	Root Partition	OK	01-09-2021 10:40:06	0d 0h 12m 0s	1/4	DISK OK - free space: / 421365 MB (97,53% inode=99%):
	SSH	OK	01-09-2021 10:40:43	0d 0h 11m 23s	1/4	SSH OK - OpenSSH_7.6p1 Ubuntu-4ubuntu0.3 (protocol 2.0)
	Swap Usage	OK	01-09-2021 10:41:01	0d 0h 11m 5s	1/4	SWAP OK - 100% free (3813 MB out of 3813 MB)
	Total Processes	OK	01-09-2021 10:37:02	0d 0h 15m 4s	1/4	PROCS OK: 19 processes with STATE = RSZDT
s2	Current Load	OK	01-09-2021 10:37:39	0d 0h 14m 27s	1/4	OK - load average: 0.31, 0.73, 1.07
	Current Users	OK	01-09-2021 10:38:17	0d 0h 13m 49s	1/4	USERS OK - 0 users currently logged in
	PING	OK	01-09-2021 10:39:32	0d 0h 12m 34s	1/4	PING OK - Packet loss = 0%, RTA = 0.09 ms
	Root Partition	OK	01-09-2021 10:40:09	0d 0h 11m 57s	1/4	DISK OK - free space: / 421365 MB (97,53% inode=99%):
	SSH	OK	01-09-2021 10:40:47	0d 0h 11m 19s	1/4	SSH OK - OpenSSH_7.6p1 Ubuntu-4ubuntu0.3 (protocol 2.0)
	Swap Usage	OK	01-09-2021 10:41:01	0d 0h 11m 5s	1/4	SWAP OK - 100% free (3813 MB out of 3813 MB)
	Total Processes	OK	01-09-2021 10:37:05	0d 0h 15m 1s	1/4	PROCS OK: 19 processes with STATE = RSZDT

Figura 9: Servicios disponibles

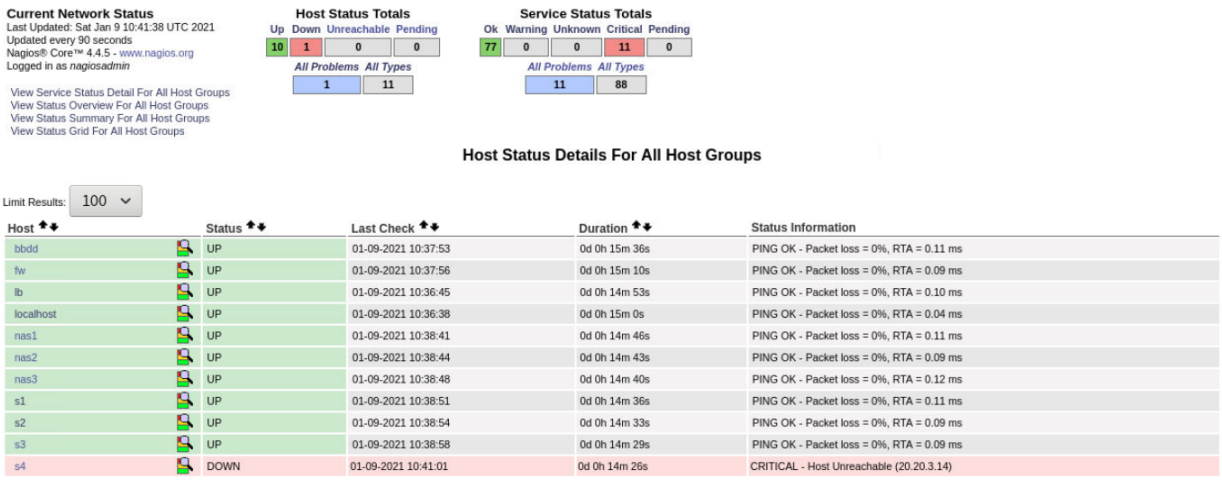
En esta imagen demostramos lo comentado anteriormente, los servicios que no están funcionando.



Host	Service	Status	Last Check	Duration	Attempt	Status Information
bbdd	HTTP	CRITICAL	01-09-2021 10:40:53	0d 0h 12m 5s	4/4	connect to address 20.20.4.31 and port 80: Conexión rehusada
fw	HTTP	CRITICAL	01-09-2021 10:40:56	0d 0h 12m 2s	4/4	connect to address 20.20.1.1 and port 80: Conexión rehusada
nas1	HTTP	CRITICAL	01-09-2021 10:41:41	0d 0h 11m 17s	4/4	connect to address 20.20.4.21 and port 80: Conexión rehusada
nas2	HTTP	CRITICAL	01-09-2021 10:41:44	0d 0h 11m 14s	4/4	connect to address 20.20.4.22 and port 80: Conexión rehusada
nas3	HTTP	CRITICAL	01-09-2021 10:41:48	0d 0h 11m 10s	4/4	connect to address 20.20.4.23 and port 80: Conexión rehusada
s1	HTTP	CRITICAL	01-09-2021 10:41:51	0d 0h 11m 7s	4/4	connect to address 20.20.3.11 and port 80: Conexión rehusada
s2	HTTP	CRITICAL	01-09-2021 10:41:54	0d 0h 11m 4s	4/4	connect to address 20.20.3.12 and port 80: Conexión rehusada
s3	HTTP	CRITICAL	01-09-2021 10:41:58	0d 0h 11m 0s	4/4	connect to address 20.20.3.13 and port 80: Conexión rehusada
s4	HTTP	CRITICAL	01-09-2021 10:39:01	0d 0h 13m 57s	1/4	connect to address 20.20.3.14 and port 80: No existe ninguna ruta hasta el «host»
	PING	CRITICAL	01-09-2021 10:39:39	0d 0h 13m 19s	1/4	CRITICAL - Host Unreachable (20.20.3.14)
	SSH	CRITICAL	01-09-2021 10:40:54	0d 0h 12m 4s	1/4	connect to address 20.20.3.14 and port 22: No existe ninguna ruta hasta el «host»

Figura 10: Servicios no disponibles

Por último, mostramos los hosts que hemos configurando. Destacamos que el servidor cuatro está apagado.



Current Network Status
Last Updated: Sat Jan 9 10:41:38 UTC 2021
Updated every 90 seconds
Nagios® Core™ 4.4.5 - www.nagios.org
Logged in as *nagiosadmin*

View Service Status Detail For All Host Groups
View Status Overview For All Host Groups
View Status Summary For All Host Groups
View Status Grid For All Host Groups

Host Status Totals

Up	Down	Unreachable	Pending
10	1	0	0

All Problems	All Types
1	11

Service Status Totals

Ok	Warning	Unknown	Critical	Pending
77	0	0	11	0

All Problems	All Types
11	88

Host Status Details For All Host Groups

Limit Results: 100

Host	Status	Last Check	Duration	Status Information
bbdd	UP	01-09-2021 10:37:53	0d 0h 15m 36s	PING OK - Packet loss = 0%, RTA = 0.11 ms
fw	UP	01-09-2021 10:37:56	0d 0h 15m 10s	PING OK - Packet loss = 0%, RTA = 0.09 ms
lb	UP	01-09-2021 10:36:45	0d 0h 14m 53s	PING OK - Packet loss = 0%, RTA = 0.10 ms
localhost	UP	01-09-2021 10:36:38	0d 0h 15m 0s	PING OK - Packet loss = 0%, RTA = 0.04 ms
nas1	UP	01-09-2021 10:38:41	0d 0h 14m 46s	PING OK - Packet loss = 0%, RTA = 0.11 ms
nas2	UP	01-09-2021 10:38:44	0d 0h 14m 43s	PING OK - Packet loss = 0%, RTA = 0.09 ms
nas3	UP	01-09-2021 10:38:48	0d 0h 14m 40s	PING OK - Packet loss = 0%, RTA = 0.12 ms
s1	UP	01-09-2021 10:38:51	0d 0h 14m 36s	PING OK - Packet loss = 0%, RTA = 0.11 ms
s2	UP	01-09-2021 10:38:54	0d 0h 14m 33s	PING OK - Packet loss = 0%, RTA = 0.09 ms
s3	UP	01-09-2021 10:38:58	0d 0h 14m 29s	PING OK - Packet loss = 0%, RTA = 0.09 ms
s4	DOWN	01-09-2021 10:41:01	0d 0h 14m 26s	CRITICAL - Host Unreachable (20.20.3.14)

Figura 11: Hosts configurados

Por otro lado, si activamos el firewall con las reglas comentadas anteriormente Nagios no tiene acceso a este y lo marca como máquina apagada.

Host	Status	Last Check	Duration	Status Information
bbdd	UP	01-10-2021 18:13:38	0d 0h 7m 35s	PING OK - Packet loss = 0%, RTA = 0.08 ms
fw	DOWN	01-10-2021 18:13:04	0d 0h 7m 8s	(Host check timed out after 30.03 seconds)
lb	UP	01-10-2021 18:12:30	0d 0h 6m 51s	PING OK - Packet loss = 0%, RTA = 0.09 ms
localhost	UP	01-10-2021 18:12:22	0d 0h 6m 59s	PING OK - Packet loss = 0%, RTA = 0.07 ms
nas1	UP	01-10-2021 18:13:48	0d 0h 6m 48s	PING OK - Packet loss = 0%, RTA = 0.09 ms
nas2	UP	01-10-2021 18:13:52	0d 0h 6m 44s	PING OK - Packet loss = 0%, RTA = 0.10 ms
nas3	UP	01-10-2021 18:13:55	0d 0h 6m 41s	PING OK - Packet loss = 0%, RTA = 0.09 ms
s1	UP	01-10-2021 18:13:58	0d 0h 6m 38s	PING OK - Packet loss = 0%, RTA = 0.10 ms
s2	UP	01-10-2021 18:14:02	0d 0h 6m 34s	PING OK - Packet loss = 0%, RTA = 0.10 ms
s3	UP	01-10-2021 18:14:05	0d 0h 6m 31s	PING OK - Packet loss = 0%, RTA = 0.08 ms
s4	UP	01-10-2021 18:14:09	0d 0h 6m 27s	PING OK - Packet loss = 0%, RTA = 0.11 ms

Figura 12: Hosts con reglas firewall activadas

fw	Current Load	OK	01-10-2021 18:12:26	0d 0h 7m 56s	1/4	OK - load average: 1.03, 1.01, 1.00
	Current Users	OK	01-10-2021 18:13:04	0d 0h 7m 18s	1/4	USERS OK - 0 users currently logged in
	HTTP	CRITICAL	01-10-2021 18:13:41	0d 0h 6m 41s	1/4	CRITICAL - Socket timeout
	PING	CRITICAL	01-10-2021 18:14:19	0d 0h 6m 3s	1/4	CRITICAL - Plugin timed out
	Root Partition	OK	01-10-2021 18:14:56	0d 0h 5m 26s	1/4	DISK OK - free space: / 202788 MB (94,50% inode=99%):
	SSH	CRITICAL	01-10-2021 18:10:34	0d 0h 4m 48s	1/4	CRITICAL - Socket timeout
	Swap Usage	OK	01-10-2021 18:11:11	0d 0h 8m 36s+	1/4	SWAP OK - 100% free (3812 MB out of 3813 MB)
	Total Processes	OK	01-10-2021 18:11:46	0d 0h 8m 36s+	1/4	PROCS OK: 33 processes with STATE = RSZDT

Figura 13: Servicios del firewall con este activado

Es posible acceder a dicha interfaz gráfica a través de 20.20.1.15/nagios , las credenciales de acceso son usuario nagiosadmin y contraseña hola.

3.3. Servidor front-end adicional

Hemos incorporado la posibilidad añadir un servidor web adicional al escenario, para su uso váyase al apartado de 'Utilización'. En esta mejora nos hemos apoyado en el archivo s4.xml que se proporcionaba y hemos utilizado distintas maneras de balancear el tráfico en función de si este servidor se crea o no (Mejora del algoritmo de balanceo round-robin).

3.4. Mejora el algoritmo de balanceo round-robin

Hemos realizado una configuración mejorada del algoritmo round-robin estableciendo pesos. De esta configuración surgen dos posibilidades, que se incluya el servidor añadido número cuatro o no. En ambos casos los servidores número uno y dos soportan una cargan del 40 porciento del tráfico, si solamente hay tres servidores este último se quedaría con el 20 porciento restante, sino los dos restantes se repartirían el tráfico correspondiendo un 10 por ciento a cada uno. Además en todos los servidores se habilita el modo check para evitar el uso de servidores caídos.

3.5. Servidor de gestión

Como se ha comentado en la mejora del firewall, los clientes no van a poder acceder a los servidores si no que tendrán que acceder a través del balanceador. Es por esto que se ha configurado un servidor de gestión denominado GES para que los clientes puedan acceder de manera remota a los servidores. Este acceso se realizará mediante una acceso remoto con SSH y utilizando claves RSA, para una mayor seguridad, evitando el acceso mediante nombre y clave de usuario.

Para acceder a este servidor de gestión, después de haber ejecutado el comando de configuración ya descrito anteriormente, se debe escribir el siguiente comando en nuestra consola:

ssh root@ges

En caso de que nos de error, podemos escribir la ruta donde se ha generado la clave pública con -i:

ssh -i /mnt/tmp/pc2/ges_rsa root@ges

Una vez hemos accedido remotamente podemos comprobar que tenemos accesibilidad directa a los servidores si ejecutamos una orden ping a la dirección de estos.

3.6. Consolidación de logs

En esta mejora hemos realizado dos maneras de consolidar distintos logs. En primer lugar y como se recomendaba, un servidor centralizado que recoge los logs de las distintas máquinas a través de rsyslog. En segundo lugar, hemos hecho una consolidación de logs en la máquina host a través de la librería de python logging.

3.6.1. Servidor de logs

Para realizar esta mejora hemos configurado en el servidor logs el archivo rsyslog.conf en el cual hemos habilitado que pueda recibir el tráfico TCP de distintos clientes. Además, hemos creado una carpeta llamada mislogs en el directorio /var/log donde se creará de manera automática una carpeta por cliente que dispondrá de sus logs. En esta máquina también hemos instalado el paquete lsof para poder comprobar su correcto funcionamiento.

Por otro lado, hemos configurado también el archivo rsyslog.conf de las máquinas clientes, en este caso nos hemos decantado por s1,s2,s3,s4,bbdd,nas1,nas2 y nas3.

```
root@logs:/var/log/mislogs# lsof -i :514
COMMAND   PID   USER   FD   TYPE  DEVICE SIZE/OFF NODE NAME
rsyslogd 1248 syslog    5u  IPv4 2781535      0t0  UDP *:syslog
rsyslogd 1248 syslog    6u  IPv6 2781536      0t0  UDP *:syslog
rsyslogd 1248 syslog    7u  IPv4 2781539      0t0  TCP *:shell (LISTEN)
rsyslogd 1248 syslog    8u  IPv6 2781540      0t0  TCP *:shell (LISTEN)
rsyslogd 1248 syslog   15u  IPv4 2781108      0t0  TCP logs:shell->20.20.3.13:44822 (ESTABLISHED)
rsyslogd 1248 syslog   17u  IPv4 2782598      0t0  TCP logs:shell->20.20.4.23:34522 (ESTABLISHED)
rsyslogd 1248 syslog   20u  IPv4 2782386      0t0  TCP logs:shell->20.20.3.12:58492 (ESTABLISHED)
rsyslogd 1248 syslog   22u  IPv4 2783745      0t0  TCP logs:shell->20.20.4.22:48872 (ESTABLISHED)
rsyslogd 1248 syslog   23u  IPv4 2784781      0t0  TCP logs:shell->20.20.4.21:39266 (ESTABLISHED)
rsyslogd 1248 syslog   24u  IPv4 2784372      0t0  TCP logs:shell->20.20.3.11:43196 (ESTABLISHED)
rsyslogd 1248 syslog   26u  IPv4 2784458      0t0  TCP logs:shell->20.20.4.31:39098 (ESTABLISHED)
```

Figura 14: Comando lsof en el puerto 514 de TCP

```
root@logs:/var/log/mislogs# ls
bbdd  logs  nas1  nas2  nas3  s1  s2  s3
```

Figura 15: Carpeta con logs de cada cliente

3.6.2. Librería python

Para ello hemos importado la librería logging y hemos configurado los atributos que queremos que queden constancia, asimismo su nivel. Los logs quedan registrados en el fichero archivo.log con un formato configurado por nosotros.

```
01/09/2021 12:29:48 PM - DEBUG - ESCENARIO - Escenario base creado
01/09/2021 12:29:48 PM - DEBUG - ESCENARIO - Se ha optado por la configuracion por defecto
01/09/2021 12:30:11 PM - DEBUG - ESCENARIO - Maquinas del escenario base destruidas
01/09/2021 12:32:25 PM - DEBUG - ESCENARIO - Escenario base creado
01/09/2021 12:32:25 PM - DEBUG - ESCENARIO - Se ha optado por la configuracion por defecto
01/09/2021 12:32:46 PM - DEBUG - ESCENARIO - Maquinas del escenario base destruidas
01/09/2021 12:33:34 PM - DEBUG - ESCENARIO - Escenario base creado
01/09/2021 12:33:34 PM - DEBUG - ESCENARIO - Se ha optado por la configuracion por defecto
01/09/2021 12:33:54 PM - DEBUG - ESCENARIO - Maquinas del escenario base destruidas
01/09/2021 12:34:32 PM - DEBUG - ESCENARIO - Escenario base creado
01/09/2021 12:34:32 PM - DEBUG - ESCENARIO - Se ha optado por la configuracion por defecto
01/09/2021 12:34:52 PM - DEBUG - ESCENARIO - Maquinas del escenario base destruidas
01/09/2021 12:35:57 PM - DEBUG - ESCENARIO - Escenario base creado
01/09/2021 12:35:57 PM - DEBUG - ESCENARIO - Se ha optado por la configuracion por defecto
01/09/2021 12:36:24 PM - DEBUG - ESCENARIO - Maquinas del escenario base destruidas
01/09/2021 12:37:11 PM - DEBUG - ESCENARIO - Escenario base creado
01/09/2021 12:37:11 PM - DEBUG - ESCENARIO - Se ha optado por la configuracion por defecto
01/09/2021 12:39:08 PM - DEBUG - BBDD - Comenzando la configuración de las bases de datos
01/09/2021 12:40:56 PM - DEBUG - BBDD - Bases de datos configuradas
01/09/2021 12:40:56 PM - DEBUG - NAS - Configurando el nas
01/09/2021 12:41:12 PM - DEBUG - NAS - NAS configurado
01/09/2021 12:41:12 PM - DEBUG - SERV - Configurando los servidores
01/09/2021 12:43:18 PM - DEBUG - SERV - Servidor s1 configurado
01/09/2021 12:45:36 PM - DEBUG - SERV - Servidor s2 configurado
01/09/2021 12:47:41 PM - DEBUG - SERV - Servidor s3 configurado
01/09/2021 12:47:42 PM - DEBUG - LB - Configurando balanceador
01/09/2021 12:47:42 PM - DEBUG - LB - Configurando el fichero HAproxy
01/09/2021 12:47:57 PM - DEBUG - LB - Balanceador configurado
01/09/2021 12:47:57 PM - DEBUG - ESCENARIO - Escenario basico configurado
01/09/2021 12:49:03 PM - DEBUG - Gestion - Nuevas claves generadas para conectarse al servidor de gestión
01/09/2021 12:49:03 PM - DEBUG - Gestion - Servidor configurado
01/09/2021 12:49:10 PM - DEBUG - LOGS - Configurando servidor de logs
01/09/2021 12:49:18 PM - DEBUG - LOGS - Servidor de logs configurado
01/09/2021 12:49:25 PM - DEBUG - Nagios - Configurando servidor Nagios
01/09/2021 12:52:47 PM - DEBUG - Nagios - Se han configurado las dependencias
01/09/2021 12:53:02 PM - DEBUG - Nagios - Se ha establecido el certificado SSL con github
01/09/2021 12:53:05 PM - DEBUG - Nagios - Se ha descargado el paquete de Nagios
01/09/2021 12:53:42 PM - DEBUG - Nagios - Se han descargado los plugins de Nagios
01/09/2021 12:54:31 PM - DEBUG - Nagios - Servidor Nagios configurado
```

Figura 16: Archivo logs con python

3.7. Despliegue con Amazon Web Services

Con este mismo escenario cabe la posibilidad de desplegarlo en la nube, donde podemos encontrar distintos servicios estudiados en la asignatura como Amazon Web Services, OpenStack o Google Cloud. Cabe mencionar que tanto en la nube como en local tenemos la opción de utilizar Docker en lugar de contenedores LXC como hemos realizado en la práctica.

En nuestro caso vamos a explicar como podríamos desplegar con exactitud este mismo escenario con Amazon Web Services.

En Amazon Web Services (AWS) tenemos a nuestra disposición numerosos servicios y productos que podemos adquirir para llevar a cabo un despliegue lo más eficiente posible.

Primero nos centraremos en los servidores que utilizamos (s1, s2, s3 y s4), para ello AWS tiene **Amazon EC2**, son máquinas a las que podemos asignarle la configuración de hardware que queramos (desde el número de núcleos hasta la capacidad de almacenaje) e instalarle el software necesario (el sistema operativo y dependencias que utilice nuestra aplicación). Además AWS nos permite crear instancias EC2 a partir de imágenes o réplicas de otras máquinas existentes. En nuestro caso crearemos 4 instancias pero que podrán ser ampliadas o reducidas en función de la demanda que necesitemos.

Para la base de datos de los quizzes estamos usando MariaDB. AWS ofrece este tipo de servicio de base de datos en **Amazon Relational Database Service**. El servicio proporciona una capacidad escalable a la vez de rentable al mismo tiempo que automatiza varias tareas administrativas como la creación de copias de seguridad.

Al igual que en nuestro escenario, necesitaremos un balanceador de tráfico para que el cliente no tenga que utilizar una dirección por cada instancia, que se reparta dicho tráfico de manera equitativa entre nuestros servidores o que debamos desviar el tráfico que iba hacia un servidor que se ha caído. Para ello utilizaremos **AWS Elastic Load Balancing**, que se encarga de distribuir el tráfico a través de las instancias EC2 configuradas, ocupándose además de determinar el estado de las mismas y enviar solo pedidos a las que se presenten operativas.

Para el firewall ofrece **AWS Firewall Manger**, un servicio de administración de seguridad que permite la configuración y administración centralizadas de reglas de firewall en todas las aplicaciones que tengamos. Es un servicio único para generar reglas de firewall, crear políticas de seguridad y ponerlas en práctica de manera coherente y jerárquica en toda nuestra infraestructura, desde una cuenta de administrador central.

Nas, para ello contamos con dos opciones: utilizar **CloudFusion NAS for AWS**, que es un servicio ofrecido por Avere Systems pero que se encuentra en la tienda de AWS o utilizar **Amazon Elastic File System** que suministra un almacenamiento de archivos simple, escalable y elástico para utilizar con los servicios en la nube de AWS y los recursos locales. Se montan en instancias Amazon EC2, que serán los NAS necesarios.

En lugar del Nagios, AWS ofrece algo parecido: **Elastic Beanstalk**. Es el servicio que une todos los mencionados anteriormente proveyendo un entorno de monitorización y administración donde orquestar los diferentes componentes, así como controlar las versiones y entornos de liberación. Para ello solo necesitaremos configurar la pila de nuestro sistema y AWS se encargará del resto.

Como extra para tener una mayor escalabilidad y rendimiento podríamos utilizar también los servicios:

Auto Scaling que se encarga de mantener el número de instancias mínimo para cumplir con la capacidad exigida por nuestro sistema. Para lograrlo, se configuran determinadas métricas que funcionan como condiciones para añadir o eliminar instancias del servidor, como por ejemplo uso de CPU y consumo de la red entre otras. De esta forma se utilizan unas pocas instancias cuando los servidores están ociosos y aumentan el número en demanda de procesamiento.

Availability zones: más que un servicio es un concepto que nos es de gran utilidad ya que representa que Amazon tiene servidores en varios países. De esta manera nuestros servidores y componentes de base de datos pueden ser replicados a lo largo del mundo en caso de sufrir un crecimiento de clientes ubicados en países diferentes al de despliegue. Consiguiendo así que no se vea afectada la experiencia del usuario con nuestro sistema.

Amazon Cloudfront, que complementa la capacidad de replicación geográfica. Este servicio agiliza la distribución de contenido web estático y dinámico a los usuarios finales funcionando a modo de middleware entre la solicitud de un usuario por determinado archivo y el archivo final, redirigiendo al usuario a la ubicación de menor latencia posible garantizando la entrega óptima.

Cloud watch, para el monitoreo interno del sistema. Revela información acerca del funcionamiento de cada servicio. Provee métricas adecuadas a cada situación que permiten al usuario determinar las acciones a tomar o simplemente para mantener un control de estados. Por ejemplo, para las instancias EC2 se puede visualizar el uso de CPU y el consumo de red en cada instante de tiempo y con esta información podremos adaptar mejor el tipo de instancia que necesitamos para resolver las peticiones.

4. Puntos débiles del sistema

- **Seguridad:** la seguridad de la arquitectura es excasa ya que si por algún motivo un atacante consigue acceder a un servicio que no permite el firewall realmente puede acceder al resto de servicios también ya que no tenemos instalados cortafuegos entre ellos. Una manera de mejorar esta prestación sería delimitar nuestra arquitectura en distintas subredes estableciendo cortafuegos y reglas entre ellas, de esta manera mejoraríamos la seguridad de nuestra arquitectura de manera significativa.
- **Despliegue:** al ser una aplicación pequeña en principio no deberíamos tener problemas de escabilidad, pero en el momento que nuestro proyecto crece durante el tiempo los recursos podrían quedarse escasos y habría que plantearse el despliegue en un servicio en la nube como AWS.
- **Disponibilidad base de datos** al no haber realizado la mejora de replicar la base de datos nuestro servicio podría quedar fuera de servicio si fallase la base de datos.
- **Balanceador:** no se ha tenido en cuenta nada más que si el servidor está caído o no y el reparto del tráfico en función de pesos, podría mejorarse aún mas el balanceo estudiando los distintos tipos de tráfico que llegan.
- **Nagios:** hemos establecido una monitorización básica de distintos servicios de nuestros servidores pero en un escenario real se establecerían distintos protocolos de monitorización y más complejos. Por ejemplo, se deberían instalar distintos plugins en las máquinas a monitorizar.
- **Logs:** hemos recopilado distintos logs y los hemos almacenado en un servidor centralizado y en una carpeta del host. En un escenario real la cantidad de información crecería de una manera muy rápida y habría que establecer alguna política de que hacer con estos logs como hemos visto en clase. Además, lo más común sería juntar ambas fuentes de información y a continuación podríamos almacenarlas de nuevo para luego explotarlas o utilizarlas para análisis con técnicas de Big Data en tiempo real.

5. Bibliografía

Bibliografía utilizada:

1. `http://www.haproxy.org//`
2. `https://github.com/NagiosEnterprises`
3. `https://aws.amazon.com/es/`
4. `https://github.com/rsyslog/rsyslog/`

5. https://www.openstack.org/software/project-navigator/openstack-components
6. https://es.wikipedia.org/wiki/Protocolo_simple_de_administraci%C3%B3n_de_red#:~:text=El%20Protocolo%20simple%20de%20administraci%C3%B3n,administraci%C3%B3n%20entre%20dispositivos%20de%20red.
7. https://github.com/mariadb/
8. https://mariadb.com/kb/en/what-is-mariadb-galera-cluster/
9. https://hub.docker.com/
10. https://docs.python.org/es/3/tutorial/index.html
11. https://aws.amazon.com/marketplace/pp/Avere-Systems-Inc-CloudFusion-NAS-for-AWS/B015NKVAS2
12. https://aws.amazon.com/es/products/storage/
13. https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/Introduction.html
14. https://aws.amazon.com/es/about-aws/global-infrastructure/?nc1=h_ls
15. https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/Welcome.html
16. https://docs.aws.amazon.com/elasticloadbalancing/latest/userguide/what-is-load-balancing.html
17. https://medium.com/@liams_o/escalando-en-amazon-web-services-8f213c8f7e18
18. https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Welcome.html
19. https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_MariaDB.html
20. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html
21. https://aws.amazon.com/es/firewall-manager/
22. https://docs.aws.amazon.com/es_es/elasticbeanstalk/latest/dg/concepts.concepts.design.html