
Práctica 1.11

Implantación de Aplicaciones Web

José Juan Sánchez Hernández

Curso 2023/2024

Índice

1	Implantación de Wordpress en AWS utilizando una arquitectura de tres niveles	1
1.1	Arquitectura	1
1.2	Fases de la práctica	3
1.3	Tareas a realizar	4
1.3.1	<i>Balanceador de carga</i>	4
1.3.2	<i>NFS Server (Capa de Frontend)</i>	4
1.3.3	<i>Servidores web (Capa de Frontend)</i>	4
1.3.4	<i>Servidor de base de datos (Capa de Backend)</i>	5
1.4	Sincronización del contenido estático en la capa de <i>Front-End</i>	5
1.4.1	Opción 1: NFS (Network File System)	5
1.4.1.1	Paso 1: Instalación de paquetes	6
1.4.1.2	Paso 2: Exportamos el directorio en el servidor NFS	6
1.4.1.3	Paso 3: Reiniciamos el servicio NFS	6
1.4.1.4	Paso 4: Creamos el punto de montaje en el cliente NFS	7
1.4.1.5	Paso 5: Editamos el archivo <i>/etc/fstab</i> en el cliente NFS	7
1.4.2	Opción 2: GlusterFS	7
1.4.3	Opción 3: CephFS	8
1.5	Instalación de WordPress sobre un directorio que no es el raíz	9
1.5.1	Dirección de WordPress (WP_SITEURL) y Dirección del sitio (WP_HOME)	9
1.5.2	Configuración de WordPress en un directorio que no es el raíz	10
1.6	Configurar los enlaces permanentes	10
1.7	Configuración de la variable <code>\$_SERVER['HTTPS']</code>	11
1.8	Configuración de las <i>Security Keys</i>	11
1.9	Tutorial de referencia	11
1.10	Entregables	11
2	Referencias	13
3	Licencia	14

Índice de figuras

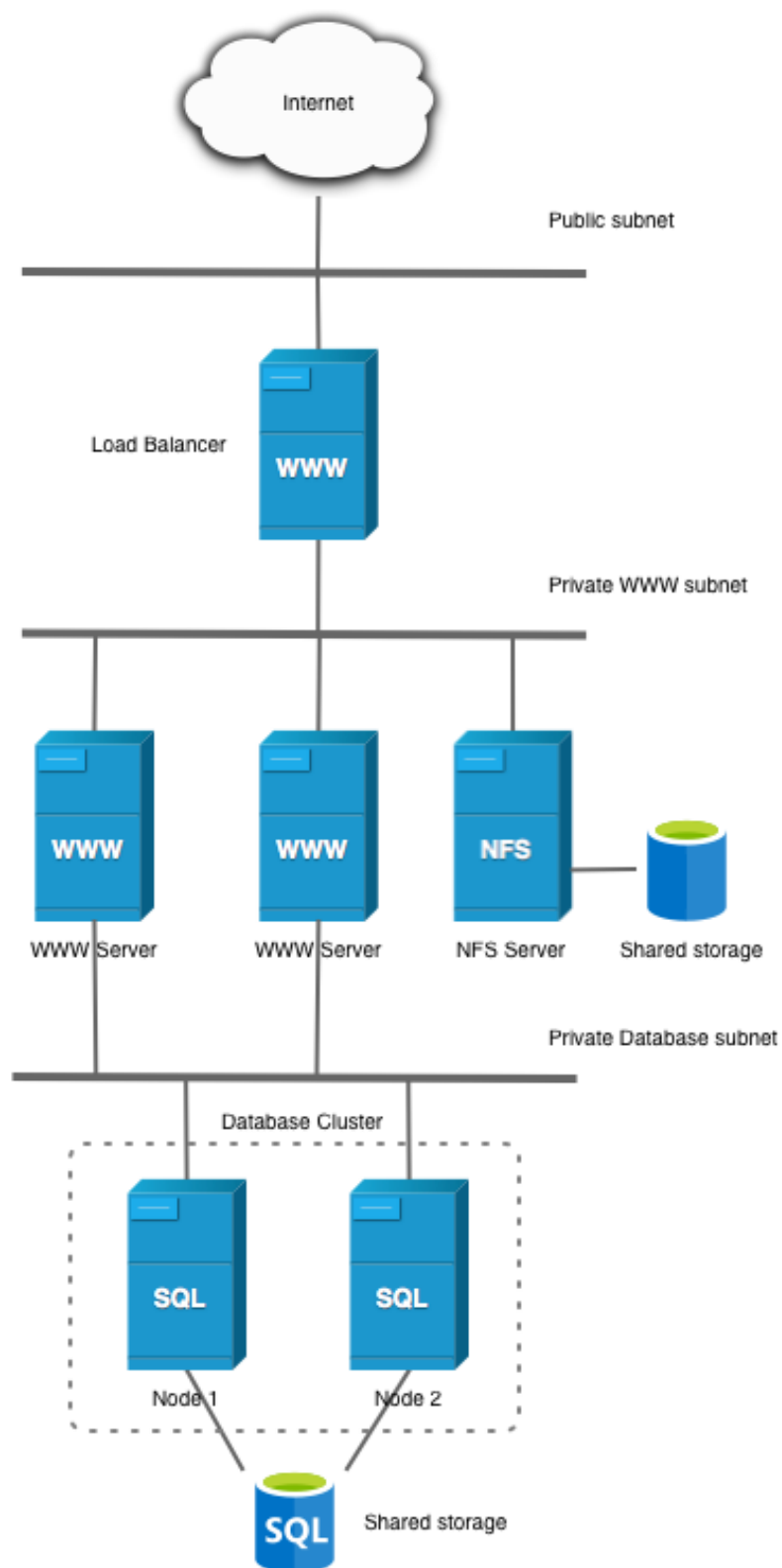
Índice de cuadros

1 Implantación de Wordpress en AWS utilizando una arquitectura de tres niveles

En esta práctica tendrá que realizar la instalación de un sitio [WordPress](#) haciendo uso de los servicios de [Amazon Web Services \(AWS\)](#).

1.1 Arquitectura

Deberá desplegar la última versión de [Worpress](#) utilizando la siguiente arquitectura de tres niveles.



La arquitectura estará formada por:

- Un balanceador de carga, implementado con un [Apache HTTP Server](#) configurado como [proxy inverso](#).
- Una capa de *front-end*, formada por dos servidores web con [Apache HTTP Server](#) y un [servidor NFS](#).
- Una capa de *back-end*, formada por un servidor [MySQL](#).

Necesitará crear las siguientes máquinas virtuales:

- Balanceador.
- Frontal Web 1.
- Frontal Web 2.
- Servidor NFS.
- Servidor de Base de Datos MySQL.

1.2 Fases de la práctica

Tendrá que resolver la práctica en diferentes fases, documentando en cada fase todos los pasos que ha ido realizando. El repositorio final tiene que contener un directorio para cada una de las fases donde se incluyan los scripts y archivos de configuración utilizados para su resolución.

```

1  práctica-wordpress-lamp
2  .├─
3    fase00├─
4    fase01├─
5    fase02├─

```

Las fases que tendrá que resolver son las siguientes:

- **Fase 0.** Instalación de Wordpress en un nivel (Un único servidor con todo lo necesario).
- **Fase 1.** Instalación de Wordpress en dos niveles (Un servidor web y un servidor MySQL).
- **Fase 2.** Instalación de Wordpress en tres niveles (Balanceador, 2 Servidores webs, Servidor NFS y Servidor MySQL).

Cada una de las fases debe incluir dos directorios:

- [infrastructure](#): Contiene los scripts de bash que se han utilizado para crear la infraestructura con la utilidad [AWS CLI](#).
- [software](#): Contiene los scripts de bash que se han utilizado para realizar la configuración y el despliegue del sitio web.

A continuación se muestra un ejemplo del contenido final que debería tener el directorio [fase-02](#).

```

1  └─
2  fase-02├─
3    infrastructure├─
4    └─ 00-terminate_all_instances.sh├─
5    └─ 01-delete_all_security_groups.sh├─
6    └─ 02-delete_all_elastic_ips.sh├─

```

7		03-create_security_groups.sh	
8		04-create_instances.sh	
9		05-create_elastic_ip.sh	
10	software		
11	conf		
12		000-default-frontend.conf	
13		000-default-loadbalancer.conf	
14	config.sh		
15	install_backend.sh		
16	install_frontend.sh		
17	install_loadbalancer.sh		
18	install_nfs_client.sh		
19	install_nfs_server.sh		
20	install_wordpress.sh		

1.3 Tareas a realizar

A continuación se describen **muy brevemente** algunas de las tareas que tendrá que realizar sobre cada una de las máquinas.

1.3.1 Balanceador de carga

- Instalar el software necesario.
- Habilitar los módulos necesarios y configurar [Apache HTTP Server](#) como [proxy inverso](#).
- Instalar y configurar [Certbot](#) para solicitar un certificado HTTPS.

1.3.2 NFS Server (Capa de Frontend)

- Instalar el software necesario.
- Crear el directorio que utilizará para compartir el contenido con los servidores web.
- Configurar el archivo `/etc/exports` para permitir el acceso al directorio compartido solo a los servidores web.

1.3.3 Servidores web (Capa de Frontend)

- Instalar el software necesario.
- Configurar el archivo de Apache para incluir la directiva `AllowOverride All`.
- Habilitar el módulo `rewrite`.
- Sincronizar el contenido estático en la capa de *Front-End*.
 - Crear un punto de montaje con el directorio compartido del servidor NFS.
 - Configurar el archivo `/etc/fstab` para montar automáticamente el directorio al iniciar el sistema.

- Descargar la última versión de [WordPress](#) y descomprimir en el directorio apropiado.
- [Configurar WordPress para que pueda conectar con MySQL](#).
- Configuración de las *Security Keys*.

1.3.4 Servidor de base de datos (Capa de Backend)

- Instalar el software necesario.
- Configurar [MySQL](#) para que acepte conexiones que no sean de *localhost*.
- Crear una base de datos para [WordPress](#).
- Crear un usuario para la base de datos de [WordPress](#) y asignarle los permisos apropiados.

1.4 Sincronización del contenido estático en la capa de *Front-End*

Al tener varias máquinas en la capa de *Front-End* tenemos que tener en cuenta que podemos tener algunos problemas a la hora de guardar contenido estático en el directorio **uploads**, instalar nuevos **themes** o instalar nuevos **plugins**, ya que estos contenidos se guardarán sobre el sistema de ficheros del frontal web que esté atendiendo nuestra petición. El contenido estático se almacena en el directorio **wp-content**.

Por ejemplo, puede ocurrir que hayamos instalado un nuevo **plugin** en uno de los frontales web y que el resto de frontales no tengan constancia de que este nuevo **plugin** ha sido instalado. También puede ocurrir que cuando uno de los frontales web esté fuera de servicio todo el contenido del directorio **uploads** estará inaccesible.

Para resolver este problema tenemos varias opciones:

1. Utilizar almacenamiento compartido por [NFS](#) del directorio `/var/www/html` entre todos los servidores de la capa de *front-end*.
2. Utilizar un sistema de almacenamiento distribuido seguro con [GlusterFS](#).
3. Utilizar un sistema de almacenamiento distribuido seguro con [CephFS](#).

1.4.1 Opción 1: NFS (Network File System)

Inconveniente: La máquina que actúa como servidor NFS es un [SPOF \(Single Point of Failure\)](#).

Podemos utilizar [NFS](#) para que los servidores de la capa de *front-end* compartan el directorio `/var/www/html`. Si utilizamos esta opción podemos hacerlo de dos formas:

- Podemos hacer que un frontal haga de servidor [NFS](#) y el otro de cliente [NFS](#). En este caso, el servidor [NFS](#) compartirá el directorio `/var/www/html/` y el cliente podrá montar este directorio en su sistema de ficheros.
- La otra posibilidad es utilizar un servidor [NFS](#) dedicado donde se almacenará el directorio compartido `/var/www/html` y los servidores web serán los clientes que utilizarán el directorio compartido.

Ejemplo de configuración de un cliente/servidor NFS

Vamos a suponer que tenemos dos máquinas con la siguientes IPs:

- **Servidor NFS:** 192.168.33.11
- **Cliente NFS:** 192.168.33.12

1.4.1.1 Paso 1: Instalación de paquetes

Instalación de paquetes necesarios en el **servidor NFS**:

```
1 sudo apt update
2 sudo apt install nfs-kernel-server -y
```

Instalación de paquetes necesarios en el **cliente NFS**:

```
1 sudo apt update
2 sudo apt install nfs-common -y
```

1.4.1.2 Paso 2: Exportamos el directorio en el servidor NFS

Cambiamos los permisos al directorio que vamos a compartir:

```
1 sudo chown nobody:nogroup /var/www/html
```

Editamos el archivo `/etc/exports`:

```
1 sudo nano /etc/exports
```

Solución para compartir el directorio con una dirección IP

Añadimos la siguiente línea:

```
1 /var/www/html 192.168.33.12(rw,sync,no_root_squash,no_subtree_check)
```

Donde **192.168.33.12** es la IP del cliente NFS con el que queremos compartir el directorio.

Solución para compartir el directorio con un rango de IPs

Si quisiéramos compartir el directorio con todos los equipos de la subred **192.168.33.0/24** tendríamos que añadir la siguiente línea:

```
1 /var/www/html 192.168.33.0/24(rw,sync,no_root_squash,no_subtree_check)
```

En la [documentación oficial](#) podemos consultar una descripción detallada de cada uno de los parámetros utilizados en el archivo `/etc/exports`.

1.4.1.3 Paso 3: Reiniciamos el servicio NFS

```
1 sudo systemctl restart nfs-kernel-server
```

NOTA: Tenga en cuenta que para que el servicio de NFS pueda funcionar tendrá que abrir el puerto 2049 para poder aceptar conexiones TCP.

1.4.1.4 Paso 4: Creamos el punto de montaje en el cliente NFS

```
1 sudo mount 192.168.33.11:/var/www/html /var/www/html
```

Donde **192.168.33.11** es la IP del servidor NFS que está compartiendo el directorio.

Una vez hecho esto comprobamos con `df -h` que el punto de montaje aparece en el listado.

```
1 $ df -h
2
3 udev                    490M    0  490M    0% /dev
4 tmpfs                  100M    3.1M   97M    4% /run
5 /dev/sda1              9.7G    1.1G   8.6G   12% /
6 tmpfs                  497M    0  497M    0% /dev/shm
7 tmpfs                  5.0M    0   5.0M    0% /run/lock
8 tmpfs                  497M    0  497M    0% /sys/fs/cgroup
9 192.168.33.11:/var/www/html 9.7G    1.1G   8.6G   12% /var/www/html
10 tmpfs                 100M    0   100M    0% /run/user/1000
```

1.4.1.5 Paso 5: Editamos el archivo `/etc/fstab` en el cliente NFS

Editamos el archivo `/etc/fstab` para que al iniciar la máquina se monte automáticamente el directorio compartido por NFS.

```
1 sudo nano /etc/fstab
```

Añadimos la siguiente línea:

```
1 192.168.33.11:/var/www/html /var/www/html nfs auto,nofail,noatime,nolock,intr,tcp,actimeo=1800 0 0
```

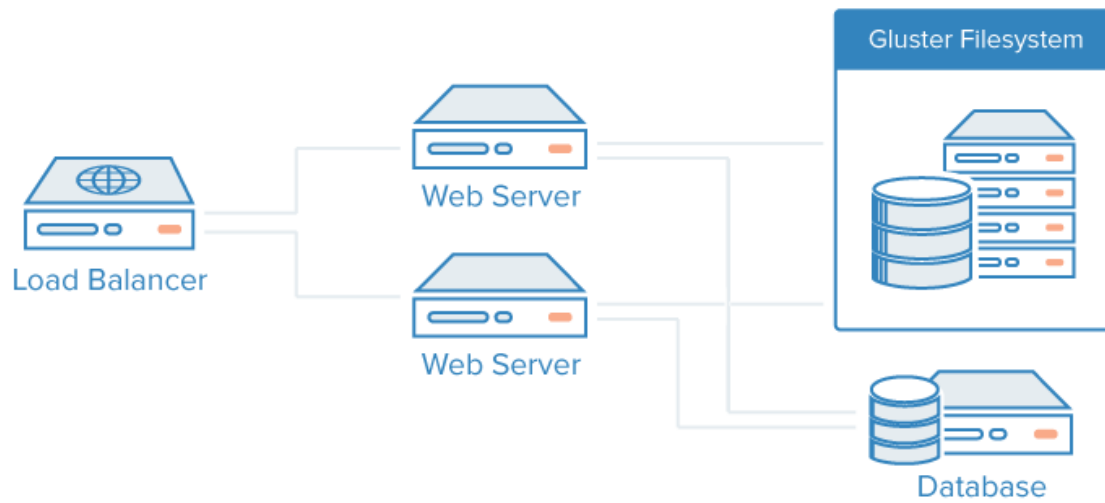
Donde **192.168.33.11** es la IP del servidor NFS que está compartiendo el directorio.

En la [documentación oficial](#) podemos consultar una descripción detallada de cada uno de los parámetros utilizados en el archivo `/etc/fstab`.

1.4.2 Opción 2: GlusterFS

Otra opción es hacer uso de [GlusterFS](#), que es un sistema de archivos multiescalable para [NAS](#).

WordPress Cluster



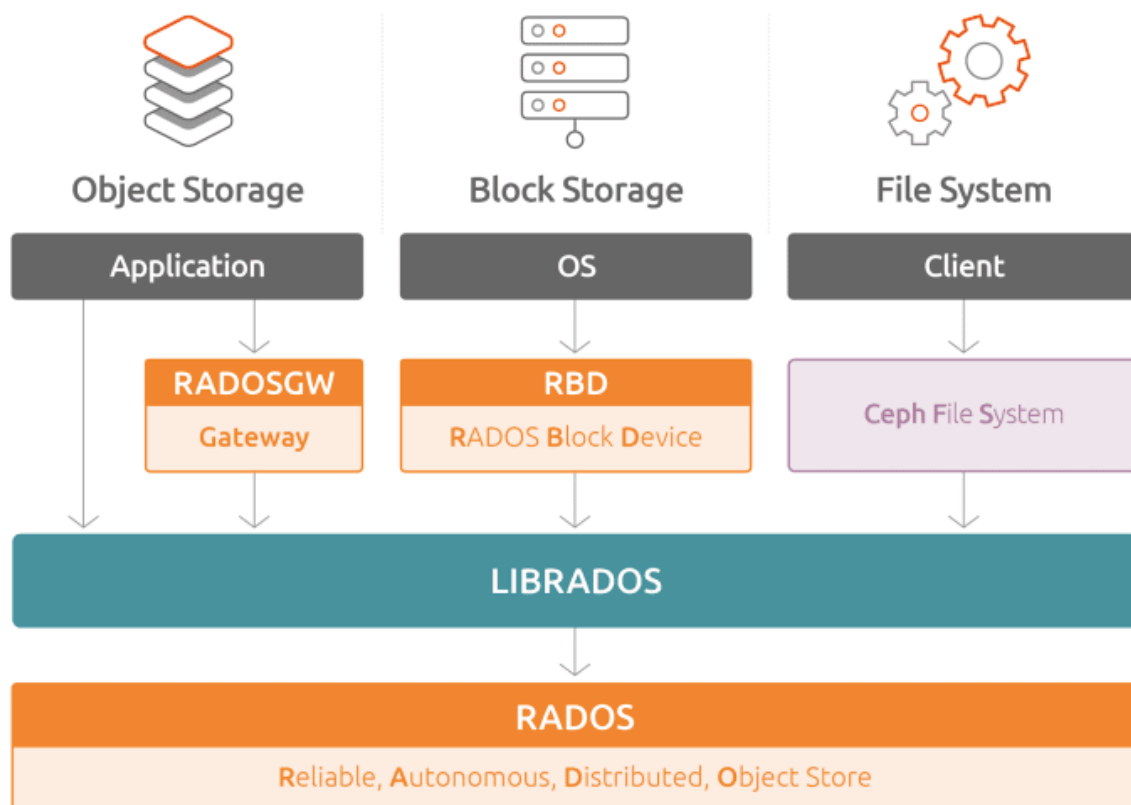
Puede encontrar más información sobre cómo crear un grupo de almacenamiento redundante con GlusterFS en [la guía de Mark Drake publicada en DigitalOcean](#).

Fuente de la imagen: [DigitalOcean](#).

1.4.3 Opción 3: CephFS

[Ceph](#) es un sistema de almacenamiento distribuido de código abierto.

Puede encontrar más información en la [documentación oficial](#).



Fuente de la imagen: [Ceph storage on Ubuntu: An overview](#).

1.5 Instalación de WordPress sobre un directorio que no es el raíz

Si hemos realizado la instalación de [WordPress](#) sobre un directorio que no es el raíz tendremos que realizar dos pasos adicionales.

Por ejemplo, si tenemos los archivos de [WordPress](#) en el directorio `/var/www/html/wordpress` en lugar de tenerlos en el directorio `/var/www/html` tendremos que configurar la dirección de WordPress (`WP_SITEURL`) y la dirección del sitio (`WP_HOME`).

1.5.1 Dirección de WordPress (`WP_SITEURL`) y Dirección del sitio (`WP_HOME`)

- **Dirección de WordPress (`WP_SITEURL`):** Es la URL que incluye el directorio donde está instalado WordPress.
- **Dirección del sitio (`WP_HOME`):** Es la URL que queremos que usen los usuarios para acceder a WordPress.

Las variables de **Dirección de WordPress (`WP_SITEURL`)** y **Dirección del sitio (`WP_HOME`)** se pueden configurar:

- desde la sección de **Ajustes -> Generales** del **panel de administración** de WordPress.

- desde el archivo de configuración `wp-config.php`.

Si hemos realizado la instalación de [WordPress](#) sobre el directorio `wordpress` tendremos que asignarles los siguientes valores:

- **Dirección de WordPress (`WP_SITEURL`):** `https://NOMBRE_DE_DOMINIO/wordpress`
- **Dirección del sitio (`WP_HOME`):** `https://NOMBRE_DE_DOMINIO`

Donde `NOMBRE_DE_DOMINIO` será el nombre de dominio que ha reservado para su sitio web. Observe que hemos utilizado `https`, porque su sitio web tendrá que utilizar un certificado HTTPS.

Si quiere realizar pruebas sin tener un nombre de dominio, puede hacerlas utilizando en su lugar la dirección IP pública del balanceador de carga.

- **Dirección de WordPress (`WP_SITEURL`):** `http://IP_PUBLICA_BALANCEADOR/wordpress`
- **Dirección del sitio (`WP_HOME`):** `http://IP_PUBLICA_BALANCEADOR`

1.5.2 Configuración de WordPress en un directorio que no es el raíz

Realiza las siguientes acciones en cada uno de los frontales web:

- **Copia** el archivo `/var/www/html/wordpress/index.php` a `/var/www/html/index.php`.
- Edita el archivo `/var/www/html/index.php` y modifica la siguiente línea de código:

```
1 /** Loads the WordPress Environment and Template */
2 require( dirname( __FILE__ ) . '/wp-blog-header.php' );
```

Por esta línea de código:

```
1 /** Loads the WordPress Environment and Template */
2 require( dirname( __FILE__ ) . '/wordpress/wp-blog-header.php' );
```

Donde `wordpress` es el directorio donde se encuentra el código fuente de [WordPress](#) que hemos descomprimido en pasos anteriores.

1.6 Configurar los enlaces permanentes

Compruebe que existe un archivo `.htaccess` en el directorio `/var/www/html/` con un contenido similar a este:

```
1 # BEGIN WordPress
2 <IfModule mod_rewrite.c>
3 RewriteEngine On
4 RewriteBase /
5 RewriteRule ^index\.php$ - [L]
6 RewriteCond %{REQUEST_FILENAME} !-f
7 RewriteCond %{REQUEST_FILENAME} !-d
8 RewriteRule . /index.php [L]
9 </IfModule>
10 # END WordPress
```

Para que el contenido de los archivos `.htaccess` sea interpretado por el servidor web Apache tendrá que incluir la directiva `AllowOverride All` en el archivo de configuración de Apache.

También deberá habilitar el módulo `rewrite` en cada uno de los servidores web de la capa de *FrontEnd*. Puede hacerlo ejecutando el siguiente comando.

```
1 sudo a2enmod rewrite
```

Después de habilitar el módulo deberá reiniciar el servicio de Apache.

```
1 sudo systemctl restart apache2
```

1.7 Configuración de la variable `$_SERVER['HTTPS']`

Como el certificado HTTPS solo se ha instalado en el servidor Apache que está haciendo de proxy inverso, vamos a tener algunos problemas con las respuestas que le devuelven los servidores de la capa de *Frontend* al proxy inverso (balanceador de carga). Puede ocurrir que la respuesta incluya referencias al contenido del sitio web con el protocolo `http` y `https`. Para forzar que las referencias de los servidores web de la capa de *Frontend* siempre sean para el protocolo `https` tenemos que incluir la variable `$_SERVER['HTTPS']` en el archivo `wp-config.php` y configurarla como `on`.

```
1 $_SERVER['HTTPS'] = 'on';
```

Referencia:

- [Administration Over SSL. Using a Reverse Proxy](#)

1.8 Configuración de las *Security Keys*

Podemos mejorar la seguridad de WordPress configurando las *security keys* que aparecen en el archivo `wp-config.php`.

En la [documentación oficial](#) podemos encontrar el motivo por el que nos recomiendan configurar estas *security keys* y cómo podemos hacerlo.

1.9 Tutorial de referencia

- [Tutorial oficial de instalación de WordPress.](#)

1.10 Entregables

En esta práctica habrá que entregar un **documento técnico** con la descripción de los pasos que se han llevado a cabo durante todo el proceso.

El documento debe incluir **como mínimo** lo siguientes contenidos:

- URL del repositorio de GitHub donde se ha alojado el documento técnico escrito en [Markdown](#).
- *Scripts* de bash utilizados para crear la infraestructura necesaria con [AWS CLI](#).
- *Scripts* de bash utilizados para realizar el aprovisionamiento de las máquinas virtuales.
- Tenga en cuenta que el aprovisionamiento de las máquinas virtuales se realizará mediante un *script* de *bash*. Cada máquina usará su propio *script*. El contenido de cada uno de los *scripts* deberá ser incluido en el documento y **deberá describir qué acciones se han ido realizando en cada uno de ellos**.
- URL del sitio web con HTTPS habilitado.

2 Referencias

- [Building for Production: Web Applications](#)
- [Building for Production: Web Applications — Deploying](#)
- [WordPress.org](#)
- [How To Set Up an NFS Mount on Ubuntu 16.04](#)
- [Generador automático de archivos `wp-config.php` para WordPress](#)
- [WordPress Codex. Manual online para WordPress](#)
- [Installing WordPress](#)
- [Advanced topics about WordPress](#)
- [Automating the deployment of a scalable WordPress site](#)
- [Cómo crear un grupo de almacenamiento redundante con GlusterFS en Ubuntu 20.04](#). Mark Drake.
- **Ebook: 21 Trucos para tener tu WordPress seguro**. SiteGround.
- [Cómo crear una máquina virtual con Amazon EC2](#).
- [Cómo crear una máquina virtual con WordPress en con Amazon EC2](#).
- [Cómo crear sitios WordPress escalables en Amazon EC2](#).
- [Cómo crear un sitio WordPress con AWS Elastic Beanstalk y Amazon Relational Database Service \(RDS\)](#).

3 Licencia

Esta página forma parte del curso Implantación de Aplicaciones Web de José Juan Sánchez Hernández y su contenido se distribuye bajo una licencia Creative Commons Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional.