Universidad Tecnológica de Panamá

Facultad de Ingeniería en Sistemas de Computación

Carrera:

Licenciatura en Ingeniería de Sistemas de Información

Asignatura:

Seguridad informática

Parcial N°1

Presentado por:  
Fernando Cutire (8-972-906)

Díaz, Gabriel (20-53-5198)

Jonathan Gamero (8-982-2008)

Grupo: 1IF141

A consideración de:

Mgtr. José Moreno

I Semestre

2022

# ÍNDICE

[ÍNDICE ii](#_Toc107967192)

[INTRODUCCIÓN 1](#_Toc107967193)

[MARCO TEÓRICO 2](#_Toc107967194)

[1. Arquitectura web 2](#_Toc107967195)

[1.1. Microsoft Azure 3](#_Toc107967196)

[1.2. Docker 4](#_Toc107967197)

[1.3. Nginx Proxy Inverso 4](#_Toc107967198)

[1.4. Let´s Encrypt 5](#_Toc107967199)

[1.5. WordPress 5](#_Toc107967200)

[1.6. Ghost 6](#_Toc107967201)

[1.7. Redis 6](#_Toc107967202)

[2. Implementación y configarión del Web Server 11](#_Toc107967203)

[2.1. Parte 1: Creando VPS en Microsoft Azure 11](#_Toc107967204)

[2.2. Parte 2: Instalación de Docker y configuración de directorios 13](#_Toc107967205)

[2.3. Parte 3: Creación y ejecución del docker-compose.yml 17](#_Toc107967206)

[2.4. Parte 4: Configuración de variables de entorno de Wordpress 22](#_Toc107967207)

[2.5. Parte 5: Instalación de Wordpress 24](#_Toc107967208)

[2.6. Parte 6: Anadir Redis en WordPress y plugins 27](#_Toc107967209)

[3. Verificacion y Resultados 30](#_Toc107967210)

[3.1. Redireccion del balanceador de carga 30](#_Toc107967211)

[3.2. Certificados de Let’s Encrypt 35](#_Toc107967212)

[3.3. Parte 6: Anadir Redis en WordPress y plugins 38](#_Toc107967213)

[CONCLUSIONES 39](#_Toc107967214)

[INFOGRAFÍA 41](#_Toc107967215)

# INTRODUCCIÓN

El siguiente trabajo presentado es una implementación de aplicación de practicas de seguridad para los sistemas informáticos a una infraestructura de servicio web multisitio gestionado en contenedores Docker, la cual contendrá un servidor balanceador de carga el cual administrará las peticiones hacia los nodos de las aplicaciones correspondientes en la red interna, una plataforma de blog conocida como Ghost y un el gestor de contenido WordPress con su debida base de datos (Mariadb) y un cache gestionado por Redis. Adicionalmente la red contara con una autoridad de certificación gestionada por Let´s Encrypt que generara certificados de confianza para las aplicaciones.

**Objetivo**

Desarrollar un proyecto de seguridad en las aplicaciones, del mundo real, asegurando los tres pilares de la seguridad la confidencialidad, integridad y disponibilidad de la información protegida.

**Perfil del Proyecto**

Deberán implementar una metodología que salvaguarde un servidor web y su respectiva aplicación. Aplicar los conceptos de seguridad en las aplicaciones como integración y desarrollo continuos, alta disponibilidad, infraestructura inmutable, Devops.

# MARCO TEÓRICO

## Arquitectura web

La arquitectura web es la planificación y el diseño de los componentes técnicos, funcionales y visuales de un sitio web, antes de que sea diseñado, desarrollado e implementado. Una buena arquitectura de la información del sitio web garantiza que los usuarios y los rastreadores de los motores de búsqueda puedan moverse fácilmente por un portal web.

Cuando decidimos que arquitectura de servidor utilizar para nuestro entorno, hay muchos factores a considerar, como el rendimiento, escalabilidad, disponibilidad, costo y facilidad de administración. Para efectos de esta investigación se estará empleando una arquitectura multiservidor. [1]

Una plataforma multisitio es útil para portales con alto tráfico y de misión crítica, sitios con estructuras complejas, con alto volumen de carga de contenido editorial y múltiples equipos editoriales, o portales que requieren integración y mejoramiento continuo.

La arquitectura empleada contendrá múltiples aplicaciones que responderán a través de un balanceador de carga o proxy inverso como se puede observar en la ilustración 1. La infraestructura empleara en una máquina virtual de Ubuntu 20.04 LTS y contenedores Docker que estará alojada los servicios de Azure de Microsoft cuales se explicara con más detalle en los puntos siguientes.

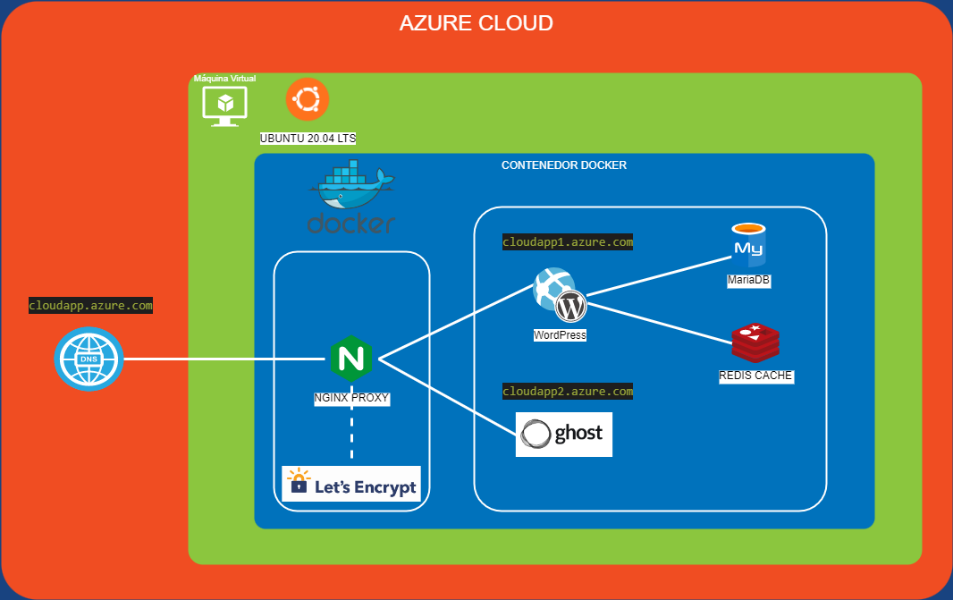


Ilustración 1 Arquitectura multisitio-Elaboración propia

Luego de definir la arquitectura de servidores a emplear, se describen a continuación cada uno de los componentes empleados para la implementación.

### Microsoft Azure

Azure es una nube pública de pago por uso que te permite compilar, implementar y administrar rápidamente aplicaciones en una red global de datacenters (centros de datos) de Microsoft. [2]

En el portal Microsoft Azure existen diferentes servicios de infraestructura y de plataforma para que puedas montar los servicios que necesites de manera sencilla. Entre los servicios, dispones de infraestructuras (IaaS: almacenamiento, redes, máquinas virtuales) y plataformas (PaaS: bases de datos de alta disponibilidad SQL, CMS para desarrollo de web, backend para aplicaciones móviles).

### Docker

Docker es una plataforma de software que le permite crear, probar e implementar aplicaciones rápidamente. Docker empaqueta software en unidades estandarizadas llamadas contenedores que incluyen todo lo necesario para que el software se ejecute, incluidas bibliotecas, herramientas de sistema, código y tiempo de ejecución. Con Docker, puede implementar y ajustar la escala de aplicaciones rápidamente en cualquier entorno con la certeza de saber que su código se ejecutará. [3]

* + 1. **Docker Compose**

Es una herramienta para definir y ejecutar aplicaciones Docker multicontenedor que permite simplificar el uso de Docker a partir de archivos YAML, de esta forma es más sencillo crear contendores que se relacionen entre sí, conectarlos, habilitar puertos, volúmenes, etc. Nos permite lanzar un solo comando para crear e iniciar todos los servicios desde su configuración (YAML), esto significa que puedes crear diferentes contenedores y al mismo tiempo diferentes servicios en cada contenedor, integrarlos a un volumen común e iniciarlos y/o apagarlos, etc. Este es un componente fundamental para poder construir aplicaciones y microservicios. [4]

### Nginx Proxy Inverso

Nginx es un servidor web que también puede ser usado como proxy inverso, balanceador de carga y proxy para protocolos de correo.

Un proxy inverso se ubica frente a un servidor web y recibe todas las solicitudes antes de que lleguen al servidor de origen. Funciona de manera similar a un proxy de reenvío, con la excepción de que en este caso es el servidor web el que utiliza el proxy y no el usuario o el cliente. Los proxies inversos se utilizan normalmente para mejorar el rendimiento, la seguridad y la fiabilidad del servidor web, como se describe en la ilustración 2. [5]

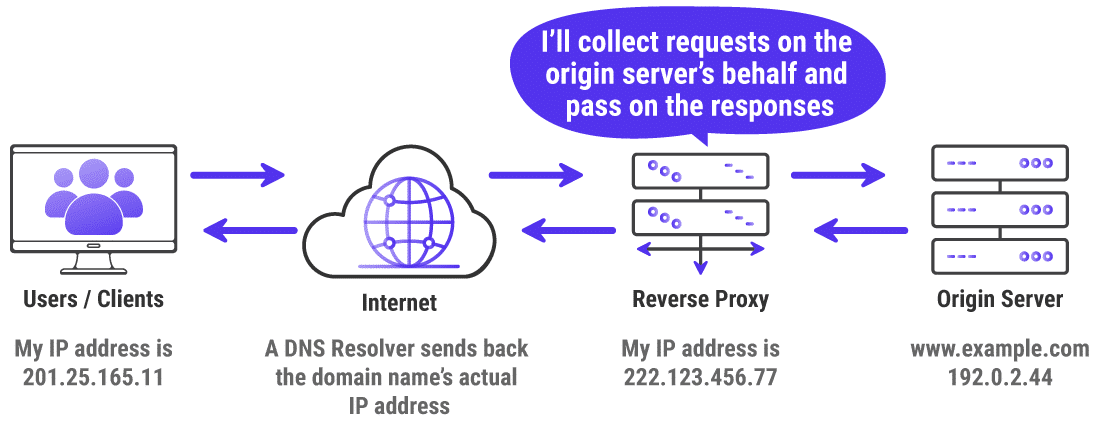


Ilustración 2 Proxy Inverso -https://kinsta.com/es/blog/proxy-inverso/

### Let´s Encrypt

Let’s Encrypt es una autoridad de certificación (AC, o CA por sus siglas en inglés) gratuita, automatizada, y abierta, que existe para el beneficio del público. Es un servicio provisto por el Internet Security Research Group (ISRG). [6]

Permite generar certificados digitales para poder habilitar HTTPS (SSL/TLS) en sitios web, de la forma más amigable al usuario posible. Cabe destacar que solo es posible generar certificados a nombres de dominio registrados y válidos.

El objetivo de Let’s Encrypt y el protocolo ACME es hacer posible la configuración de un servidor HTTPS y hacer que obtenga automáticamente un certificado confiado por el navegador, sin ninguna intervención humana. Esto se logra ejecutando un agente de maneja miento de certificados en un servidor de web.

### WordPress

WordPress es un sistema de gestión de contenidos web (CMS o content management system), que en pocas palabras es un sistema para publicar contenido en la web de forma sencilla.

Es un software de código abierto (se puede tener acceso a todo el código) que además podemos tratar de mejorar dentro de su comunidad. Por lo tanto, WordPress es gratuito y podemos descargarlo desde sus servidores https://wordpress.org/ para distintos fines.

### Ghost

Ghost es una plataforma de blogs gratuita y de código abierto escrita en JavaScript y distribuida bajo la Licencia MIT, diseñada para simplificar el proceso de publicación en línea para bloggers individuales, así como publicaciones en línea. [7]

### Redis

Redis, que significa Remote Dictionary Server, es un rápido almacén de datos clave-valor en memoria de código abierto. El proyecto se inició cuando Salvatore Sanfilippo, el desarrollador original de Redis trataba de mejorar la escalabilidad de su empresa emergente italiana. A partir de ahí, desarrolló Redis, que ahora se utiliza como base de datos, caché, agente de mensajes y cola. [8]

Redis ofrece tiempos de respuesta inferiores al milisegundo, lo que permite que se realicen millones de solicitudes por segundo para aplicaciones en tiempo real de la industria, como videojuegos, tecnología publicitaria, servicios financieros, sanidad e IoT. Por su rápido rendimiento, Redis es una opción muy habitual en aplicaciones de almacenamiento en caché, administración de sesiones, videojuegos, tablas de clasificación, análisis en tiempo real, datos geoespaciales, servicios de vehículos compartidos, chat/mensajería, streaming de contenido multimedia y publicación/suscripción.

## Implementación y configuración del Web Server

A continuación, se describen ordenadamente cada uno de los pasos utilizados para la implementación y configuración de un servidor web en la plataforma de Microsoft Azure, seguida de la creación de una máquina virtual para alojar un servido privado virtual basado en Ubuntu Server 20.04 LTS y su configuración de nombres de dominio que se requerirán para los certificados digitales.

### Parte 1: Creando VPS en Microsoft Azure

Desde el portal de Microsoft Azure se procede a crear una nueva máquina virtual de Ubuntu 20.04, la cual contara que requerimientos básicos como se muestra en la siguiente ilustración.

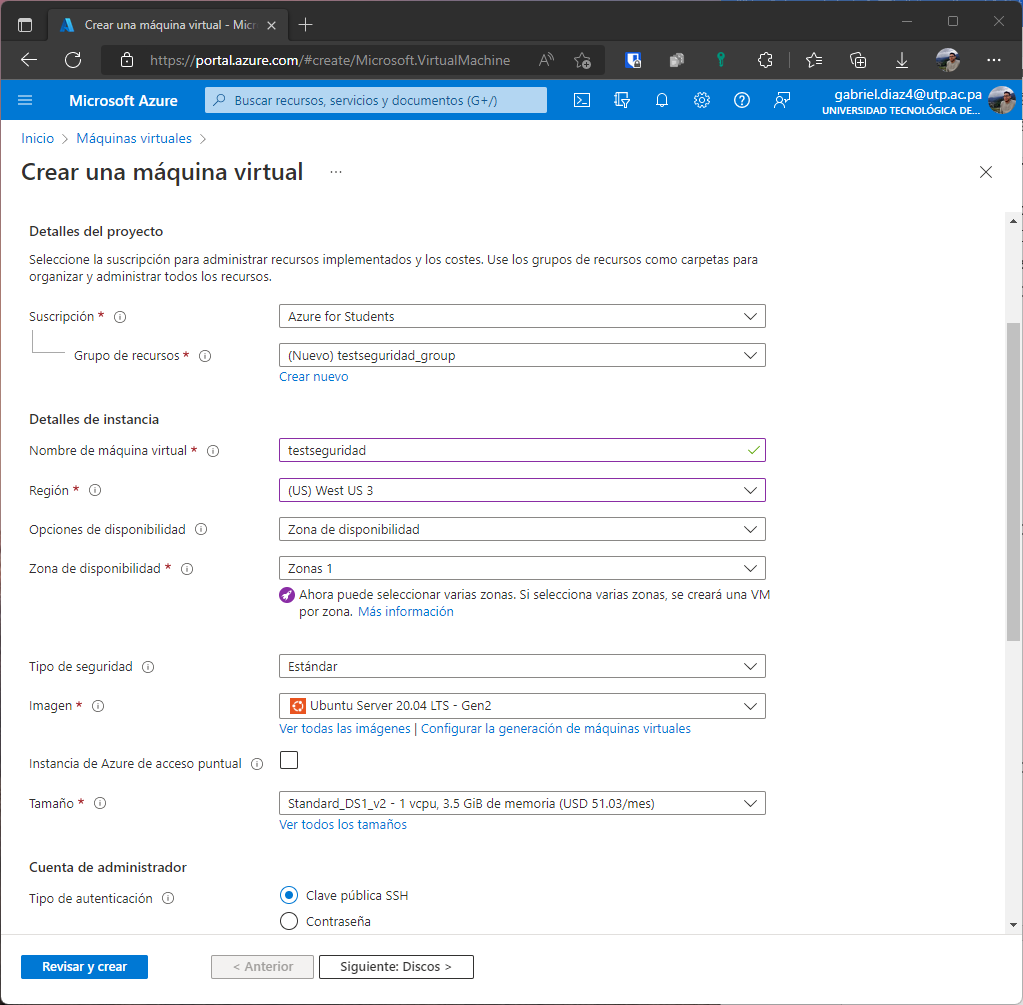


Ilustración 3 Configuración de Máquina Virtual Azure

Para el control de acceso se utilizará claves públicas SSH que explicará en la parte 2 de esta guía. También en la misma pantalla se habilitarán los puertos 22 para ssh, el puerto 80 para HTTP y el puerto 443 para HTTPS.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ilustración 4 Habilitar puerto 80 y 443 para la máquina virtual

Seleccionados estos requisitos se ejecuta la opción de Revisar y Crear. Terminado la creación nos dirigiremos a las opciones de configuración de DNS para agregar un nombre de dominio a la dirección IP publica suministrada por la plataforma de Azure.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ilustración 5 Configuración del nombre de dominio

Se procede a guardar la configuración con el nombre de dominio ingresado.

### Parte 2: Instalación de Docker y configuración de directorios

El primer paso para realizar la configuración de la máquina virtual será conectarnos a través de una conexión por SSH por medio de una terminal, para luego realizar la instalación de los paquetes necesarios y la configuración de los directorios.

Para la conexión SSH necesitaremos los valores ingresados en la creación de la máquina virtual los cuales son el nombre del usuario y la dirección IP suministrada por la plataforma de Azure. Desde una terminal ingresaremos el siguiente comando:

1. ssh -i c:\user\.ssh\azurekey parcial1@20.150.217.123

El tag -i es la ubicación del directorio que contiene la llave SSH pública y privada para acceder a la máquina virtual en Azure. Al conectarnos tendríamos un resultado como la ilustración 6.

Texto

Descripción generada automáticamente

Ilustración 6 Conexión por ssh hacia la máquina virtual

Una vez conectados procederemos a ejecutar la siguiente lista de comandos:

1. #Actualización de los repositorios
3. sudo apt-update

Luego de actualizados los repositorios procedemos a la instalación de Docker y Docker-Compose. Para efectos de esta guía se realizaron los pasos propuestos por la guía del portal web de DigitalOcean [9] donde se emplean la siguiente lista de comandos:

1. #Agregar paquetes previos apt:
2. sudo apt install apt-transport-https ca-certificates curl software-properties-common
3. #Agregar clave gpg:
4. curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
5. #Agregar el repositorio docker a las fuentes APT:
6. sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"
7. #Actualizar los paquetes:
8. sudo apt update
9. #Verificación del repositorio
10. apt-cache policy docker-ce
11. #Instalación de docker:
12. sudo apt install docker-ce
13. # Verificación del deamon de Docker en ejecución:
14. sudo systemctl status docker

Luego de ejecutar los comandos tendremos el siguiente resultado de la ilustración 7.

Texto

Descripción generada automáticamente

Ilustración 7 Daemon de docker

Se procede a instalar Docker-Compose, que utilizaremos para la configuración automática de los contenedores y sus respectivas imágenes. Usaremos los siguientes comandos:

1. # El siguiente comando descargará la versión 1.25.0 y guardará el archivo ejecutable en /usr/local/bin/docker-compose, que hará que este software esté globalmente accesible como docker-compose:
2. sudo curl -L "https://github.com/docker/compose/releases/download/1.25.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
3. # estableceremos los permisos correctos para que el comando docker-compose sea ejecutable:
4. sudo chmod +x /usr/local/bin/docker-compose
5. #Para verificar que la instalación se realizó correctamente, puede ejecutar:
6. docker-compose --version

Luego de ejecutar los comandos tendremos el siguiente resultado de la ilustración 8.

Texto

Descripción generada automáticamente

Ilustración 8 Versión de docker-compose

Se procede a crear la carpeta que contendrá el archivo de configuración docker-compose.yml de los contenedores para nuestras aplicaciones web y servidor web.

Nos ubicaremos en el directorio home del usuario y ejecutamos los siguientes comandos:

1. #verificamos el directorio:
2. pwd
3. #creamos la carpeta parcial1 y accedemos a ella:
4. mkdir parcial && cd parcial1
5. #creamos un nuevo archivo con el nombre Docker-compose y la extensión. yml:
6. touch docker-compose.yml

El resultado debe ser el siguiente como la ilustración 9.

Texto

Descripción generada automáticamente

Ilustración 9 Creación del directorio raíz

### Parte 3: Creación y ejecución del docker-compose.yml

Para esta parte nos basamos en el archivo de configuración compartido por el profesor José Moreno disponible desde su blog personal en Backtrack Academy [10], al cual realizamos adecuaciones para la incorporación de Redis.

A continuación, presentamos el archivo docker-compose.yml que se estará ejecutando para la configuración de nuestra arquitectura multisitio:

1. version: '3.1'
2. services:
3. nginx-proxy:
4. image: jwilder/nginx-proxy
5. ports:
6. - "80:80"
7. - "443:443"
8. volumes:
9. - /var/run/docker.sock:/tmp/docker.sock:ro
10. - ./certs:/etc/nginx/certs:ro
11. - ./vhostd:/etc/nginx/vhost.d
12. - ./html:/usr/share/nginx/html
13. labels:
14. - com.github.jrcs.letsencrypt\_nginx\_proxy\_companion.nginx\_proxy
16. letsencrypt:
17. image: jrcs/letsencrypt-nginx-proxy-companion:v1.13
18. restart: always
19. environment:
20. - NGINX\_PROXY\_CONTAINER=nginx-proxy
21. volumes:
22. - ./certs:/etc/nginx/certs:rw
23. - ./vhostd:/etc/nginx/vhost.d
24. - ./html:/usr/share/nginx/html
25. - /var/run/docker.sock:/var/run/docker.sock:ro
27. mariadb:
28. image: mariadb
29. environment:
30. - MYSQL\_ROOT\_PASSWORD=qwerty123456789
31. - MYSQL\_DATABASE=wpdb
32. - MYSQL\_USER=userwp
33. - MYSQL\_PASSWORD=qwertyqwerty
34. volumes:
35. - ./database:/var/lib/mysql
37. wordpress:
38. image: wordpress
39. container\_name: wordpress\_1
40. links:
41. - mariadb:mysql
42. expose:
43. - 80
44. environment:
45. - WORDPRESS\_DB\_PASSWORD=qwertyqwerty
46. - WORDPRESS\_DB\_USER=userwp
47. - WORDPRESS\_DB\_NAME=wpdb
48. - VIRTUAL\_HOST=parcialdeseguridad.westus3.cloudapp.azure.com
49. - LETSENCRYPT\_HOST=parcialdeseguridad.westus3.cloudapp.azure.com
50. - LETSENCRYPT\_EMAIL="gabriel.diaz@utp.ac.pa"
51. volumes:
52. - ./web1/html:/var/www/html
54. redis:
55. image: redis
56. expose:
57. - 6379
58. environment:
59. - ALLOW\_EMPTY\_PASSWORD=yes
61. ghost:
62. image: ghost
63. expose:
64. - 80
65. environment:
66. - VIRTUAL\_HOST=parcialdeseguridad.westus2.cloudapp.azure.com
67. - VIRTUAL\_PORT=2368
68. - LETSENCRYPT\_HOST=parcialdeseguridad.westus2.cloudapp.azure.com
69. - LETSENCRYPT\_EMAIL="gabriel.diaz@utp.ac.pa"

Presentado el codigo del archivo docker-compose.yml pasaremos a realizar la descripción de las partes que lo integran y su función que cada uno desempeña.

En la línea número 2, se describen los servicios que contendrá el contenedor global donde se estará ejecutando todos los procesos, las imágenes. Para ello fue necesario crear la carpeta descrita en la parte 2 de esta guía, debido a que por default Docker crea una red interna donde todos los contenedores puedan comunicarse entre si.

En las líneas 3-14, se estable la imagen que se utilizara para el balanceador de carga, siendo este una imagen de Nginx, a su vez se declaran los puertos 80 y 443 que recibirán las peticiones de los clientes. Se crearán los volúmenes para persistencia de datos que contendrá el directorio de los certificados y el directorio de nuestras aplicaciones web para poder accesar y realizar cambios en tiempo real.

En las líneas 16-25, se establece la imagen que se utilizara para la gestión de los certificados, siendo una imagen de Let´s Encrypt, que a su vez está vinculada con la imagen del servidor proxy Ngnix.

En las líneas 27-35, se establece la imagen de base de datos que utilizara Worpress, siendo esta una imagen de Mariadb. También se establecen los valores de seguridad como el nombre del host, la contraseña y el un nombre de base de datos, entre las líneas 30-33.

En las líneas 37-52, se estable la imagen que se utilizara para el gestor de contenido Wordpress, se establece una conexión hacia la base de datos Mariadb y a su vez las respetivas acreditaciones de usuario y contraseña para la conexión de la base de datos.

En las líneas 54-59, se estable la imagen que se utilizara para redis, se expone el puerto 6379 dentro de la red default del contenedor para que sea accesible por Wordpress.

En las líneas 61-69, se estable la imagen que se utilizara para gestor de blogs Ghost, se expone el puerto 80 para que sea accedido dentro de la misma red.

**Configuración de variables de entorno en los contenedores para la generación automática de certificados por let´s encrypt:**

* + VIRTUAL\_HOST: en este parámetro establece el dominio de la aplicación. Tiene que ser un dominio oficial valido y registrado.
  + LETSENCRYPT\_HOST: el parámetro toma el host que vamos a generar los certificados.
  + LETSENCRYPT\_EMAIL: correo a utilizar para el certificado.
  + NGINX\_PROXY\_CONTAINER: definimos el proxy

Establecidos los parámetros de los contenedores se procede a escribir la configuración en el archivo docker-compose.yml con los siguientes pasos:

1. #Editar el archivo con el editor de texto nano en el terminal con los siguientes comandos:
2. nano Docker-compose.yml
3. #Copiar el contenido en el archivo
4. CRTL + U
6. #Guardar cambios
7. CRTL + O + ENTER
8. Salir
9. CRTL + X

Obteniendo el resultado de la ilustración 10.

Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamente

Ilustración 10 Edición de docker-compose.yml

Completada la edición del archivo de configuración se procede a ejecutar los siguiente comado:

1. docker-compose up -d

Donde el tag -d significa que cuando se inicien los contenedores estos se ejecuten en segundo plano.

Terminado la descarga de todos los contenedores y que no haya errores visualizamos los contenedores corriendo en segundo plano con el comando:

1. docker-compose ps

Obteniendo el resultado de la ilustración 11.

Texto

Descripción generada automáticamente

Ilustración 11 - procesos del docker-compose

También podemos verificar la red creada por default con el comando:

1. docker network ls

Obteniendo el resultado siguiente:

Texto

Descripción generada automáticamente

Ilustración 12-red interna de docker

Verificamos que en el directorio parcial1 se hayan creado las subcarpetas de los volúmenes de cada contenedor y certificados. Estos accesos nos dan la posibilidad de realizar cambios en los archivos de configuración de las aplicaciones como Wordpress, la cual en la siguiente parte de esta guía se estará editando las variables de entorno.

Texto

Descripción generada automáticamente

Ilustración 13 - valumenes de aplicación

### Parte 4: Configuración de variables de entorno de Wordpress

Para realizar cambios en Wordpress se tienen que realizar en el archivo de configuración wp-config.php este lo encontramos en el siguiente directorio con los comandos:

1. cd web1/html/ && ls -la

Obteniendo el siguiente resultado:

Texto

Descripción generada automáticamente

Ilustración 14 - ubicación de wp-config.php

Ubicado el archivo de configuración de Wordpress se procede a realizar la edición, para ello utilizamos el siguiente comando:

1. sudo nano wp-config.php

Dentro del editor de código veremos todas las configuraciones disponibles de Wordpres. Como requerimiento de esta guía se necesitan agregar las siguientes variables:

1. #Variable para la conexión de redis:
2. define(‘WP\_REDIS\_HOST’, ‘redis’);
3. #Forzar que las conexiones sean a través del protocolo ssl
4. define(‘FORCE\_SSL\_ADMIN’, true);
5. #Deshabilitación del wordpress editor:
6. define(‘DISALLOW\_FILE\_EDIT', true);

Obteniendo un resultado como el siguiente:

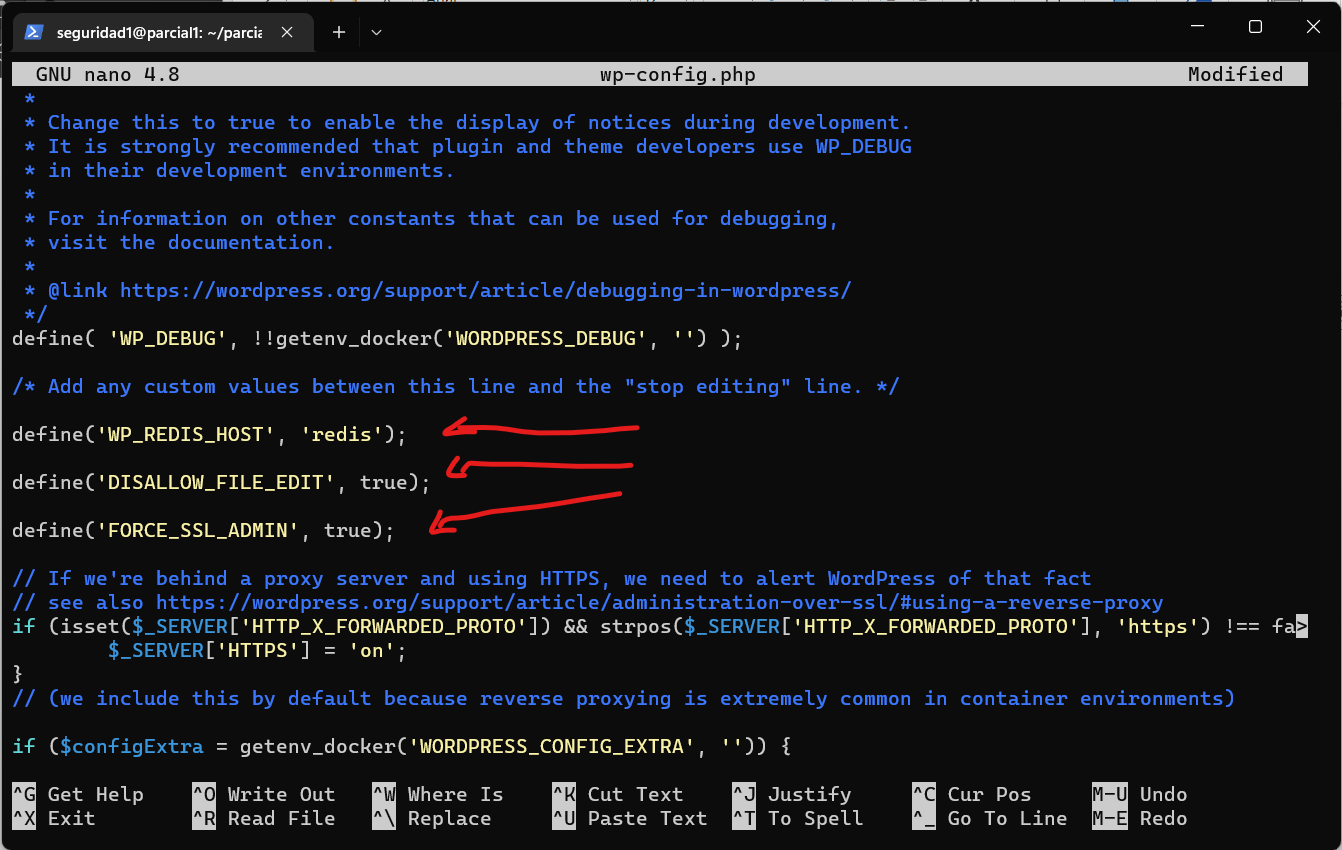


Ilustración 15 - editando el wp-config.php

Agregadas las variables, se guarda y cierra el archivo.

También, otra practica de seguridad recomendada es la de proteger en si el archivo wp-config.php. Esto se consigue agregando una configuración extra en el archivo .htaccess ubicado en el mismo directorio. El código es el siguiente:

1. <Files wp-config.php>
2. order allow, deny
3. deny from all
4. </Files>

Editamos el archive .htaccess con el siguiente comando:

1. sudo nano .htaccess

Copiamos el código y el resultado es el siguiente:

Texto

Descripción generada automáticamente

Ilustración 16 - Editando el .htaccess

Con estas modificaciones tendremos una capa extra de seguridad.

### Parte 5: Instalación de Wordpress

Ahora toca realizar la instalación del wordpress accediendo a la dirección de dominio registrada y realizar la instalación del plugin de conexión de redis.

Accediendo a la URL registrada obtenemos la pantalla de instalación de wordpress:

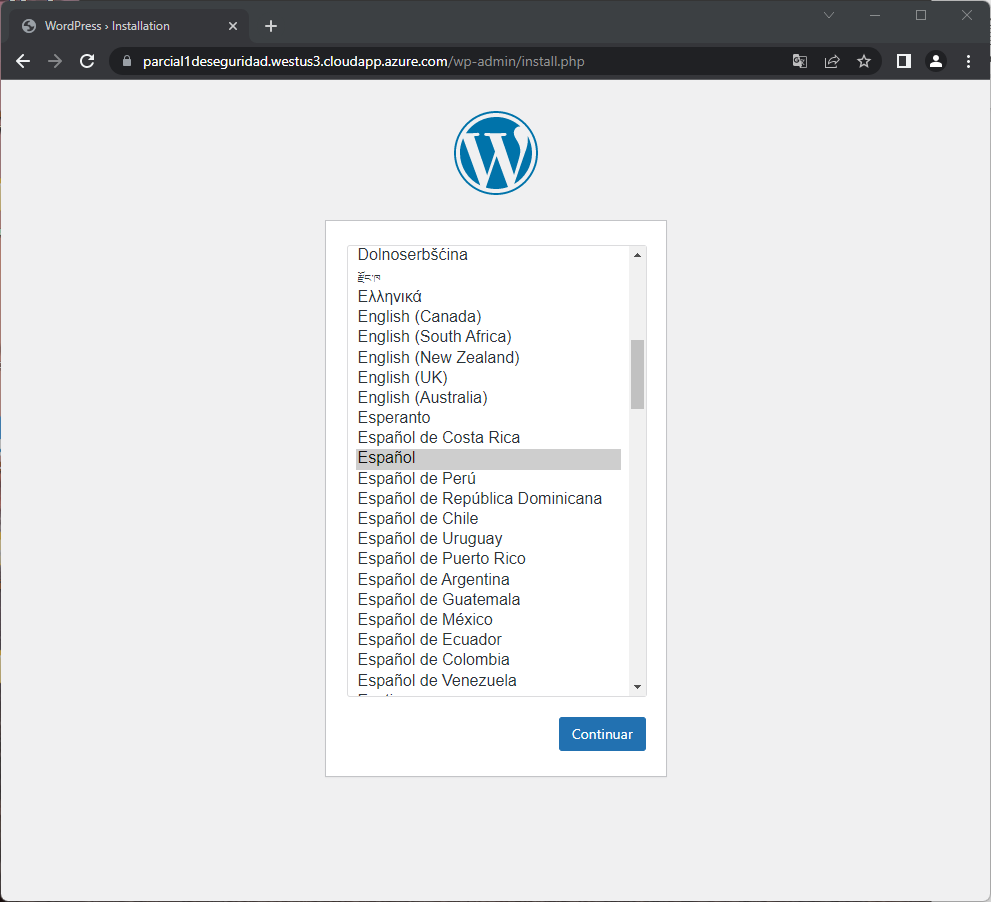


Ilustración 17 - Instalación de wordpress

Completamos el formulario de registro de usuario y establecemos una contraseña segura:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración 18 - Formulario de registro

Damos a la opción de instalar wordpress, luego nos logearemos con los datos de usuario y tendremos nuestra aplicación de wordpress instalada:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ilustración 19 - Pantalla de bienvenida de Wordpress

### Parte 6: Anadir Redis en WordPress y plugins

Para finalizar la configuración de wordpress y los requisitos de esta guía se pide que wordpress funcione con un cache de Redis, para esto nos ubicaremos en el panel de administrador de wordpress, en la opción de plugins, luego añadir plugins y realizar la instalación:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ilustración 20 - Instalación de Redis

Una vez instalado, tenemos que activarlo, si la configuración de nuestro Docker-composer.yml se ejecutó si errores, wordpress reconocerá que hay una instancia de redis en la misma red y lo agregará:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ilustración 21 - Activación de redis

Procedemos a ejecutar la opción de activar cache de objetos obtendremos el siguiente resultado:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ilustración 22 - Activacion de Redis

Finalizamos con la activación de las actualizaciones automáticas para tener nuestros plugins con las ultimas actualizaciones:

Una captura de pantalla de una computadora

Descripción generada automáticamente

Ilustración 23 - Actualizar los plugins de Wordpress

## Verificación y Resultados

Detallaremos a continuación que el balanceador de carga funcione correctamente redirigiendo el tráfico por HTTPS y que las aplicaciones poseas su respectivo certificado de confianza.

### Redirección del balanceador de carga

Cabe destacar que para lograr la redirección del balanceador de carga se tuvo que dar de baja a una de las dos aplicaciones, debido a que solo se posee un solo dominio registrado, esto lo podemos ver con la aplicación de los siguientes comandos:

1. #Ver las imágenes corriendo:
2. docker-compose ps
3. #Dentener una imagen especifica:
4. docker stop nombredelaimagen

Ejecutando esto verificamos las imágenes activas, obteniendo el resultado siguiente:

Texto

Descripción generada automáticamente

Ilustración 24 - Imágenes de docker activas

Si ingresamos a la URL el balanceador de carga aleatoriamente o según el algoritmo que tenga definido nos regresara una de las dos instancias. En el primer ingreso nos devuelve la página de wordpress:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ilustración 25 - Resultado del balanceador de carga

Desactivaremos la instancia de wordpress, para que nos redirija a la aplicación de Ghost. Obteniendo los siguientes resultados:

Texto

Descripción generada automáticamente

Ilustración 26 - Deshabilitando una instancia de docker

Observamos que la instancia de Docker aparece con un estatus Exit. Nuevamente ingresamos a la URL y esta vez nos redirige hacia la aplicación de Ghost:

Interfaz de usuario gráfica, Texto, Sitio web

Descripción generada automáticamente

Ilustración 27 - Redirección del balanceador de carga

Como resultado podemos verificar que el balanceador de carga Nginx funciona correctamente con la redirección del tráfico HTTPS.

### Certificados de Let’s Encrypt

Parte de la validación final de esta guía, está en asegurar a completitud que las aplicaciones posean certificados de confianza válidos. Para obtener estos resultados se implementó de manera eficaz utilizando el proceso automatizado descrito en la configuración de archivo docker-compose.yml el cual se utiliza una imagen de Docker de Let’s Encrypt que funciona en conjunto con Nginx para generar los certificados digitales de las aplicaciones de Wordpress y Ghost.

Let’s Ecrypt automáticamente gener certificados TLS/SSL gratuitos, destacando que para poder obtener estos certificados el dominio tiene que ser legítimo y estar registrado y validado, lo cual se logro por el dominio generado por la plataforma de Microsoft Azure.

Procedemos a realizar la validación que tanto el navegador y como el propio Intertet reconozca nuestra dirección de dominio URL como una dirección de confianza la cual podemos verificar en la ilustración 28. También podemos validar el certificado generado posee claves RSA privada y pública de 4096 bits en la ilustración 29 y 30.

Interfaz de usuario gráfica

Descripción generada automáticamente

Ilustración 28 - Reconocimiento como sitio seguro

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ilustración 29 - CA de confianza

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ilustración 30 - Claves RSA de 4096 bits

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

# CONCLUSIONES

En conclusión, podemos destacar los siguientes puntos:

* Para asegurar la disponibilidad de los servicios de una aplicación con una alta demanda, la aplicación de un balanceador de carga es una solución viable porque nos asegura una escalabilidad horizontal mejorando el rendimiento de respuesta cuando un servidor falle otro responderá en su lugar.
* La automatización de servicios que componen un entorno de Docker es una solución que nos permite definer, en un fichero YAML, las imágenes que se necesitan, puertos de conexión, acceso a los volúmenes de la aplicación, así como también una red interna de contenedores. Cuando se requiera realizar un cambio solo habrá que editar el fichero para que este ejecute todos los cambios en el contendor.
* Wordpress es uno de los gestores de contenido mas utilizados a nivel mundial, y a su vez, una de las cuales es mas atacadas por cibercriminales. Por lo tanto, es muy importante el aseguramiento de la información de esta plataforma, siguiendo guias bien establecidas desde protección básica como contraseñas seguras, hasta seguridad avanzada empleando reforzamiento en los códigos de confiracion de la aplicación, el forzamiento a usar formas de encriptación, uso de certificados digitales de confianza, entre otros como los dispuestos en esta investigación.
* Let’s Encrypt es unas soluciones mas factibles ya que nos permite obtener certificados de confianza gratuitos, disponibles por 90 dias y con renovación automática. Permite generar confianza para los usuarios de la aplicación, así como también bloquear el contenido en caso de que un tercero pretenda realizar una suplantación de dominio el cual será reconocido como un sitio inseguro y no valido.

# INFOGRAFÍA

|  |  |
| --- | --- |
| [1] | DigitalOcean, «5 Configuraciones Comunes para tus Aplicaciones Web,» DigitalOcean, [En línea]. Available: https://www.digitalocean.com/community/tutorials/5-configuraciones-comunes-para-tus-aplicaciones-web-es. [Último acceso: 30 06 2022]. |
| [2] | Azure, «Microsoft Azure,» [En línea]. Available: https://azure.microsoft.com/es-es/overview/what-is-azure/. [Último acceso: 30 06 2022]. |
| [3] | AWS, «¿Qué es Docker?,» AWS, [En línea]. Available: https://aws.amazon.com/es/docker/. [Último acceso: 30 06 2022]. |
| [4] | dev.to, «Docker,» [En línea]. Available: https://dev.to/ebarrioscode/que-demonios-es-docker-docker-compose-y-como-dockerizar-dotnet-core-webapi-y-sql-server-en-un-ambiente-de-desarrollo-ideal-95a. [Último acceso: 30 06 2022]. |
| [5] | kinsta, «Como configurar un proxy inverso,» [En línea]. Available: https://kinsta.com/es/blog/proxy-inverso/#:~:text=Un%20proxy%20inverso%20se%20ubica,el%20usuario%20o%20el%20cliente.. [Último acceso: 30 06 2022]. |
| [6] | Let´s Encrypt, «Acerca de Let´s Encrypt,» [En línea]. Available: https://letsencrypt.org/es/about/. [Último acceso: 30 06 2022]. |
| [7] | Wikipedia, «Ghost Plataforma de blogs,» [En línea]. Available: https://en.wikipedia.org/wiki/Ghost\_(blogging\_platform). [Último acceso: 30 06 2022]. |
| [8] | AWS, «Qué es Redis,» [En línea]. Available: https://aws.amazon.com/es/redis/. [Último acceso: 30 06 2022]. |
| [9] | DigitalOcean, «Cómo instalar y usar Docker en Ubuntu 20.04,» [En línea]. Available: https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-20-04-es. [Último acceso: 30 06 2022]. |
| [10] | Backtrack Academy, «Multi-Site con Lets Encrypt,» [En línea]. Available: https://backtrackacademy.com/articulo/docker-multi-site-con-let-s-encrypt. [Último acceso: 06 2022]. |