



UNIVERSIDAD TECNOLÓGICA DE PANAMÁ
FACULTAD DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

Parcial #1

Asignación:

Parcial 1 Usando dockers

Curso:

Introducción a la Ciberseguridad

Estudiante:

Eduardo Samaniego

Profesor:

José Moreno

Grupo:

IS3201

Indicaciones

UNIVERSIDAD TECNOLÓGICA DE PANAMA

FACULTAD DE INGENIERÍA DE SISTEMAS COMPUTACIONALES

SEGURIDAD EN LOS SISTEMAS DE INFORMACION

Profesor: Mgtr. José Moreno

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 continua y desarrollo continuo, alta disponibilidad, infraestructura inmutable, Devops.

El proyecto final incluye los siguientes:

- Presentar en formato Microsoft Word
- Página de Presentación
- Marco teórico
- Bibliografías, Conclusiones
- Debe captar por tema toda la configuración del proyecto y explicarla cada una.
- Se entrega en formato digital.
- La sustentación es en grupo.

Configuración Web Server

- Implementar un balanceador de carga para alta disponibilidad
- Instanciar dos servidores web como mínimo con dockers, debe responder por HTTPS y no por HTTP.
- Instalar CMS Wordpress con todos los plugins de seguridad.
- Wordpress debe usar variables de entorno para los datos de configuración.
- La infraestructura de wordpress debe contar con REDIS.
- Crear Certificado auto firmado de 4096 o 2048 bits TLS 1.2

Nota: Sustentación final del parcial1. Día indicado por el profesor, entregar documentación completa para nota.

Marco teórico

A grandes rasgos vamos a implementar varias paginas web de WordPress utilizando nginx-proxy, redis, plugins de seguridad, la misma imagen de Docker, el plugin Docker-compose entre otras funcionalidades.

Vamos a empezar con lo primero, la máquina virtual.

Máquina Virtual

El proceso de instalación es completamente fácil y se los explicare a continuación mediante el software que yo utilizo para virtualizar el cual es VMWare Workstation.

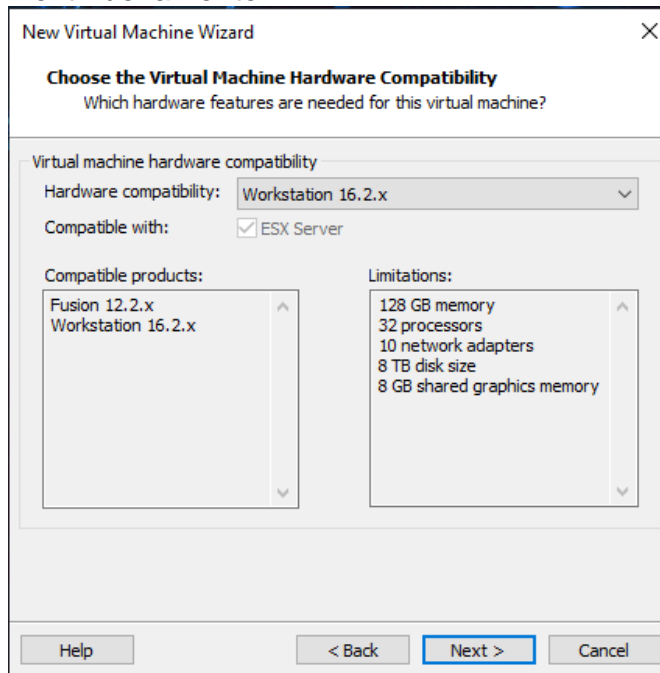
1. Le damos click derecho en la parte en blanco y le damos a New Virtual Machine



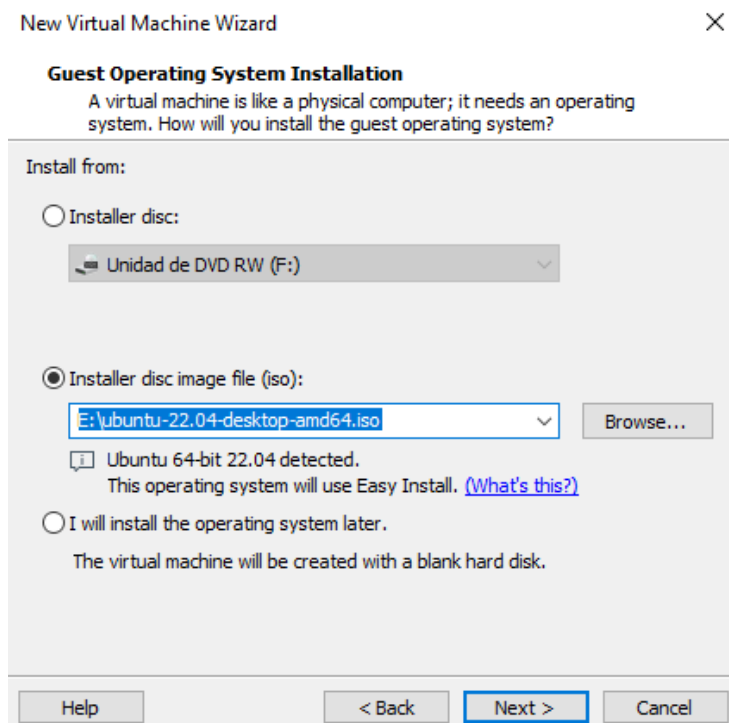
2. Le damos a next



3. Next nuevamente



4. Elegimos el iso y le damos a next



5. Elegimos un nombre de usuario, contraseña y le damos a next

New Virtual Machine Wizard ×

Easy Install Information
This is used to install Ubuntu 64-bit.

Personalize Linux

Full name:

User name:

Password:

Confirm:

6. Elegimos la localización y le damos a next

New Virtual Machine Wizard ×

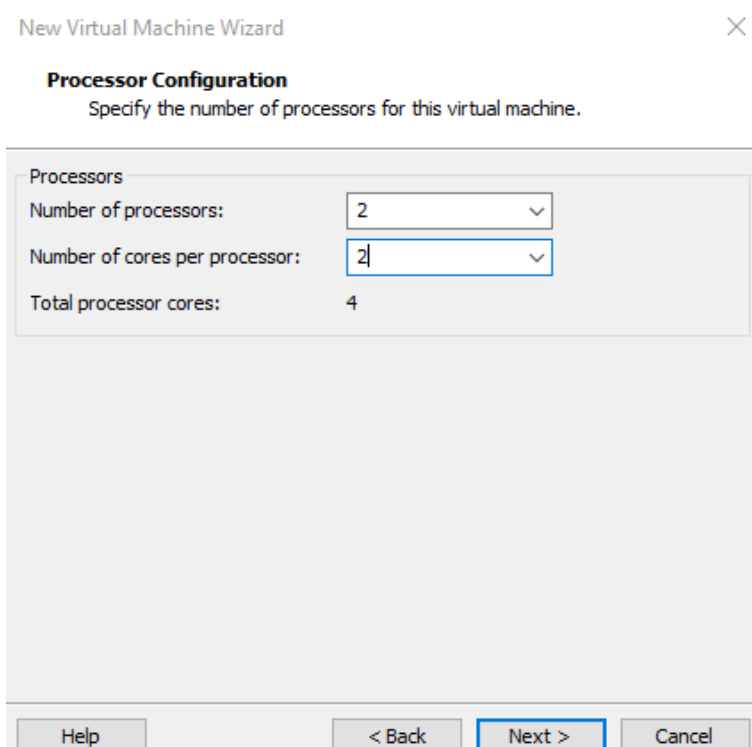
Name the Virtual Machine
What name would you like to use for this virtual machine?

Virtual machine name:

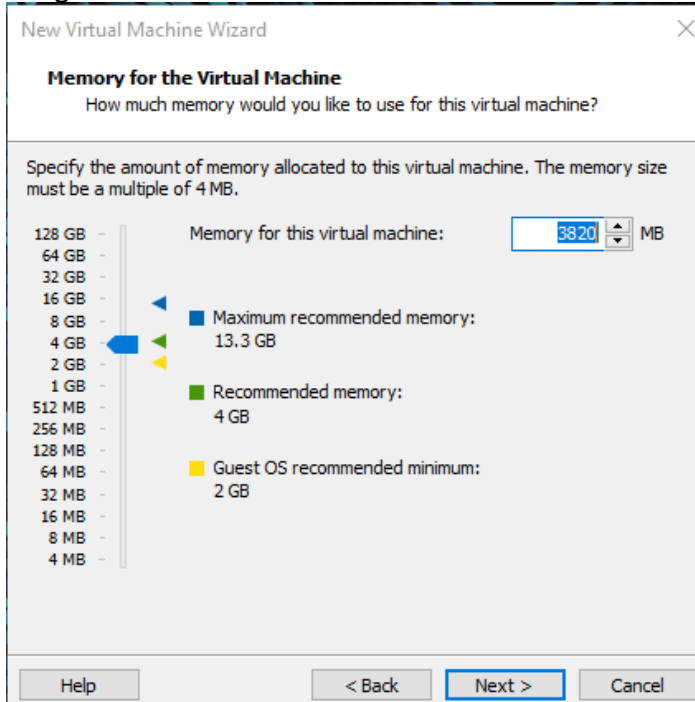
Location:

The default location can be changed at Edit > Preferences.

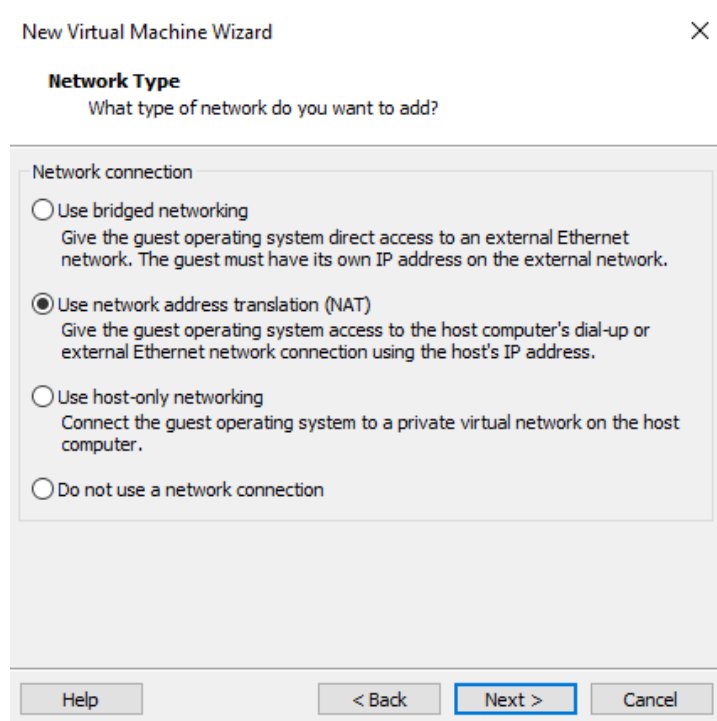
7. Elegimos el numero de procesadores y el core por procesador para objeto de pruebas se puede utilizar un 2 de 1 y tener 2 cores para reducir el poder de la máquina, en mi caso usare la maquina en posteriores trabajos así que voy a hacer una maquina un tanto generosa



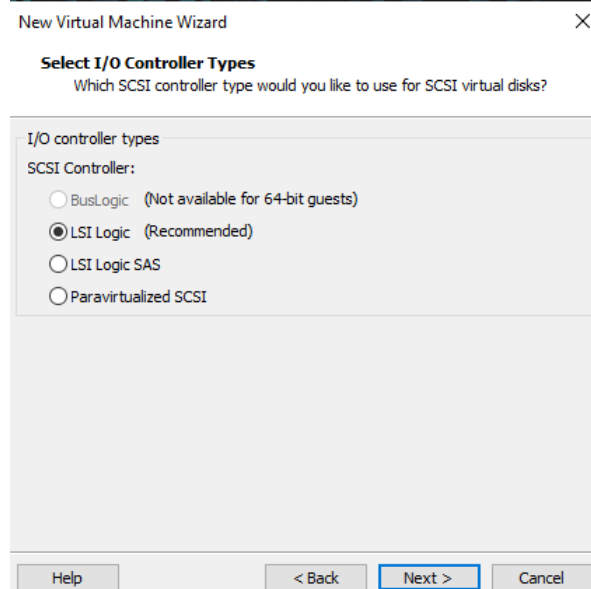
8. Elegimos el tamaño de la memoria



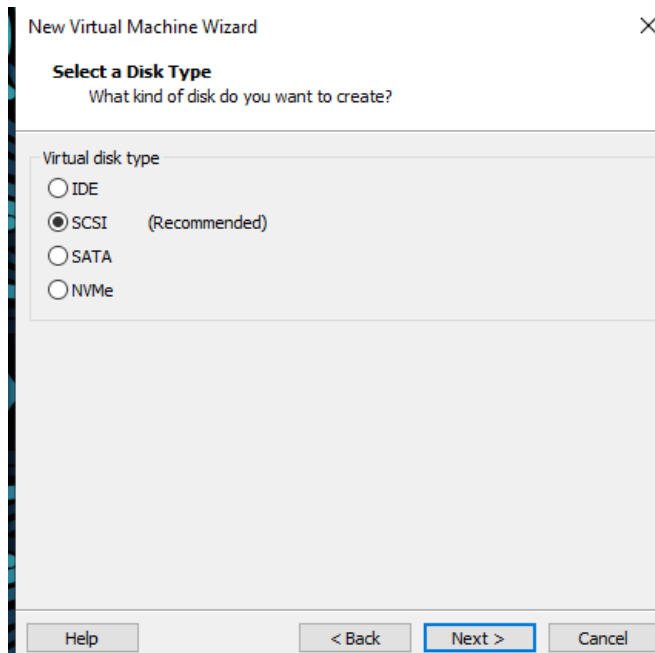
9. Elegimos el tipo de tarjeta que tendrá, le pondré una tarjeta en bridge y le doy a next



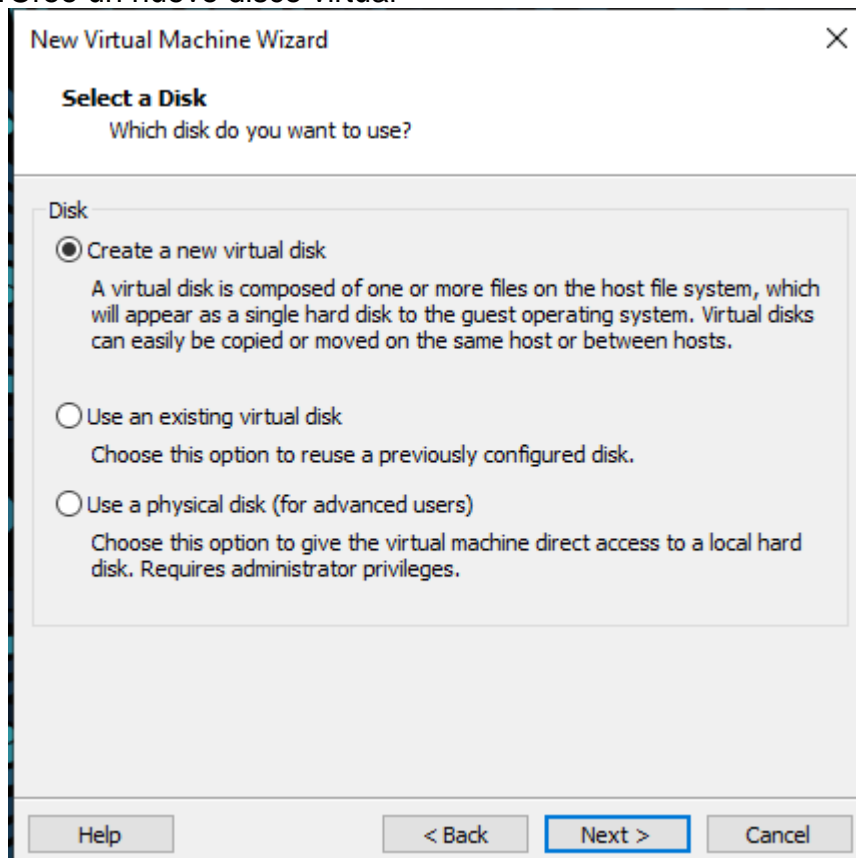
10. Controlador tipo LSI Logic que es el recomendado



11. SCSI recomendado y next



12. Creo un nuevo disco virtual



13. Elijo el tamaño del disco duro en mi caso elegiré 50 lo cual es mucho para una maquina virtual pero nuevamente repito posteriormente la utilizare entonces no me viene mal tenerla así.

New Virtual Machine Wizard ×

Specify Disk Capacity
How large do you want this disk to be?

Maximum disk size (GB):

Recommended size for Ubuntu 64-bit: 20 GB

☐ Allocate all disk space now.
Allocating the full capacity can enhance performance but requires all of the physical disk space to be available right now. If you do not allocate all the space now, the virtual disk starts small and grows as you add data to it.

☐ Store virtual disk as a single file

☒ Split virtual disk into multiple files
Splitting the disk makes it easier to move the virtual machine to another computer but may reduce performance with very large disks.

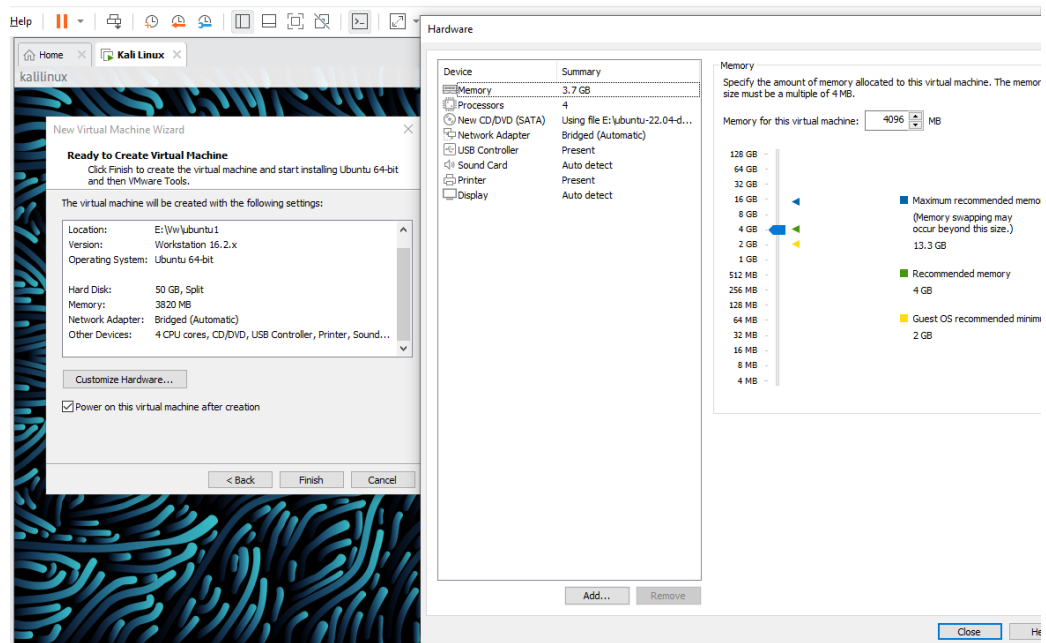
14. Elijo el nombre del archivo vmdk en mi caso lo dejare como Ubuntu

New Virtual Machine Wizard ×

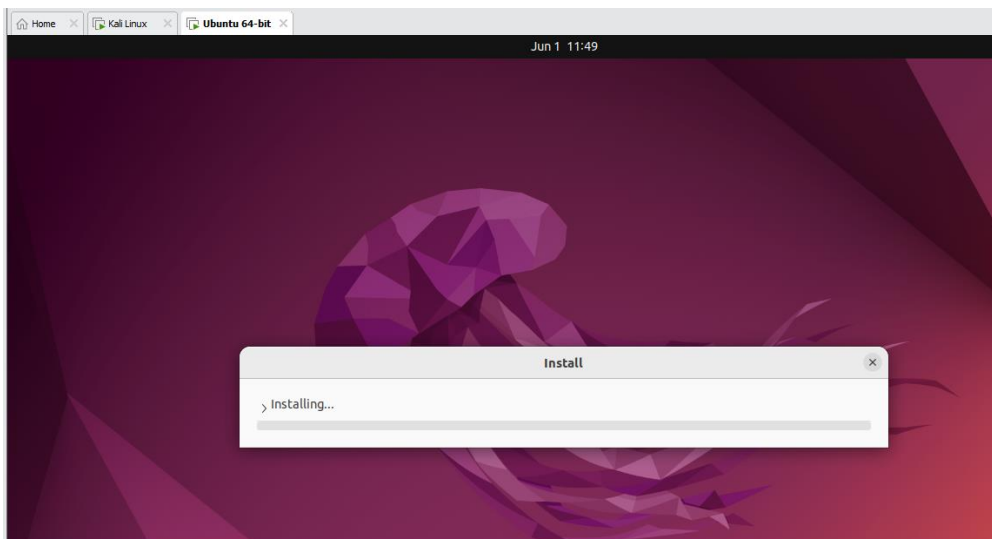
Specify Disk File
Where would you like to store the disk file?

Disk file
A 50 GB virtual disk be created using multiple disk files. The disk files will be automatically named based on this file name.

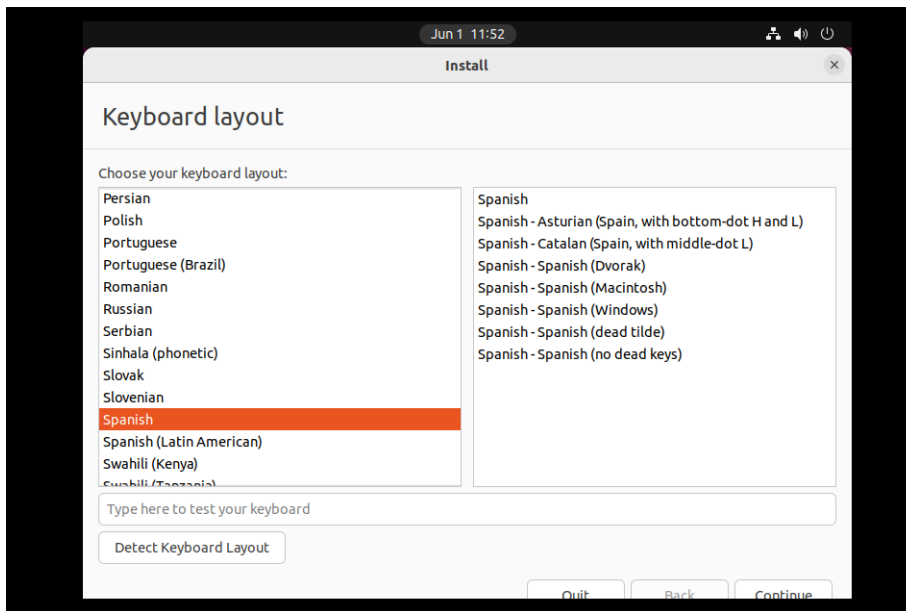
15. Finalmente nos sale un resumen de lo que pusimos en la maquina virtual de su composición, esto siempre se puede arreglar después (excepto el tamaño del disco duro) y le damos a finish



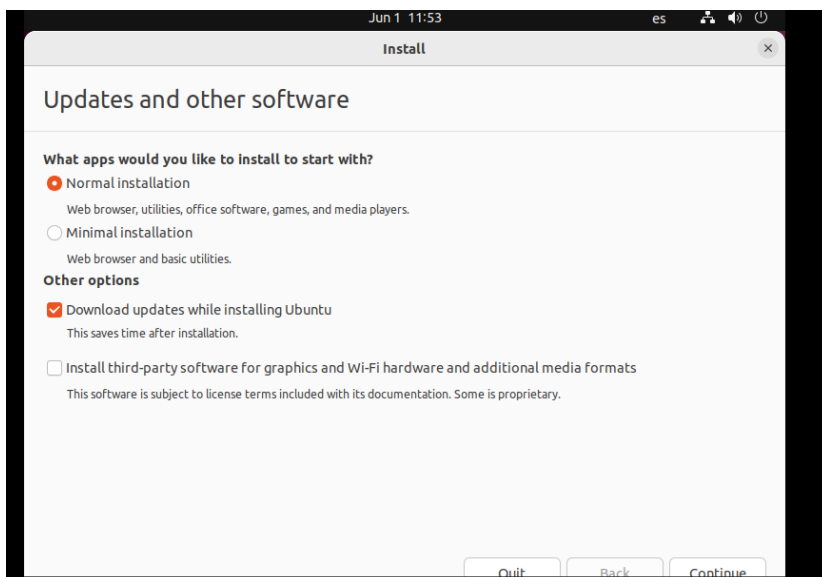
Instalación del Ubuntu



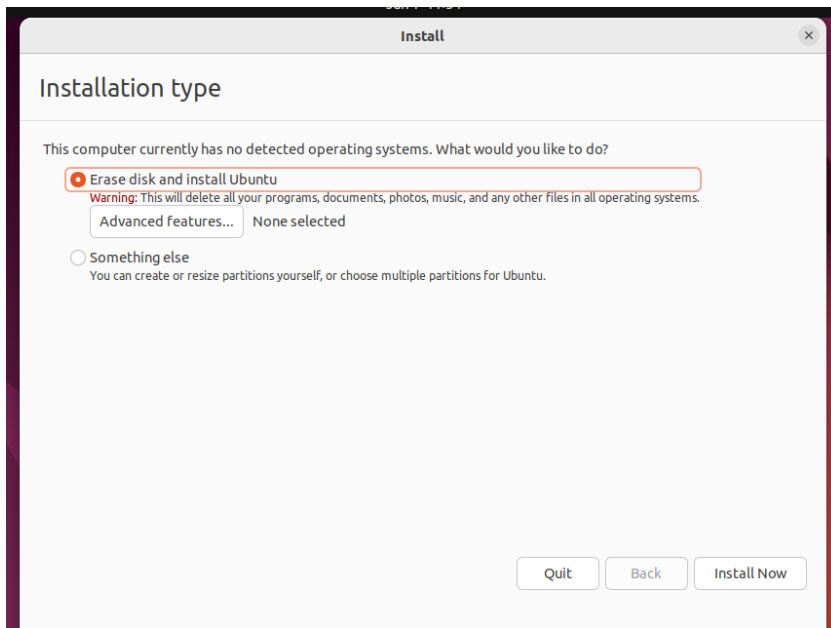
Elegimos el tipo keyboard layout



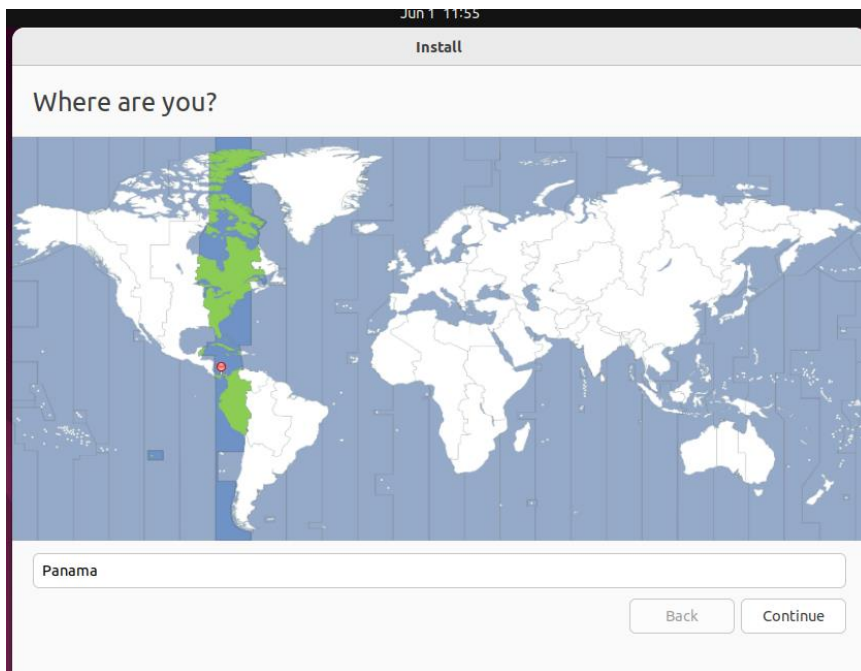
Le damos a continuar esto tiene que ver con las updates



Le damos a install, nos va a decir que borrarla todo en el disco, en nuestro caso es una maquina nueva entonces no tenemos que preocuparnos por tener algo dentro del disco.



Elegimos el pais



Install

Who are you?

Your name: Eduardo Samaniego

Your computer's name: yuru-virtual-machine

The name it uses when it talks to other computers.

Pick a username: yuru

Choose a password: Fair password

Confirm your password:

☒ Log in automatically

☐ Require my password to log in

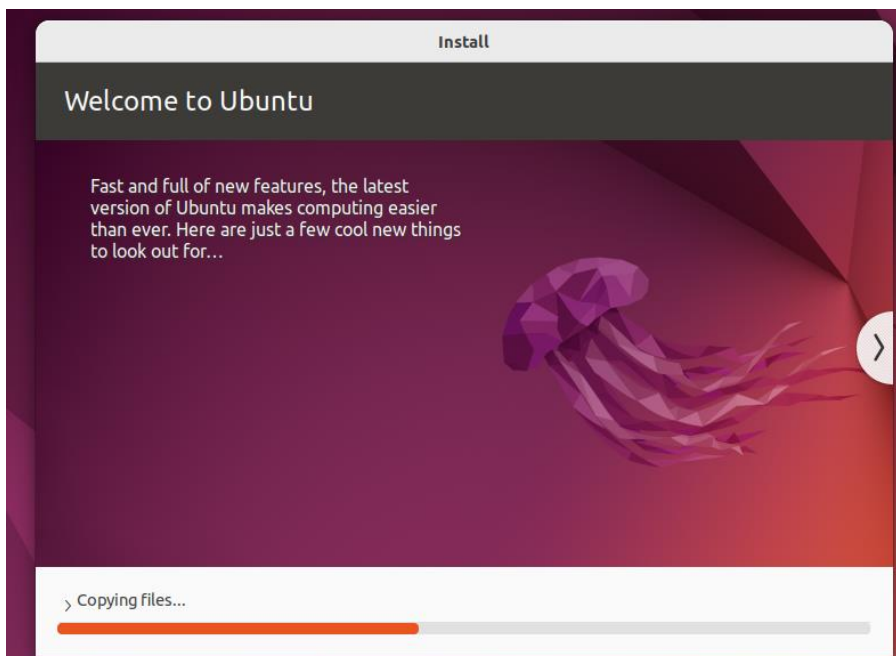
☐ Use Active Directory

You'll enter domain and other details in the next step.

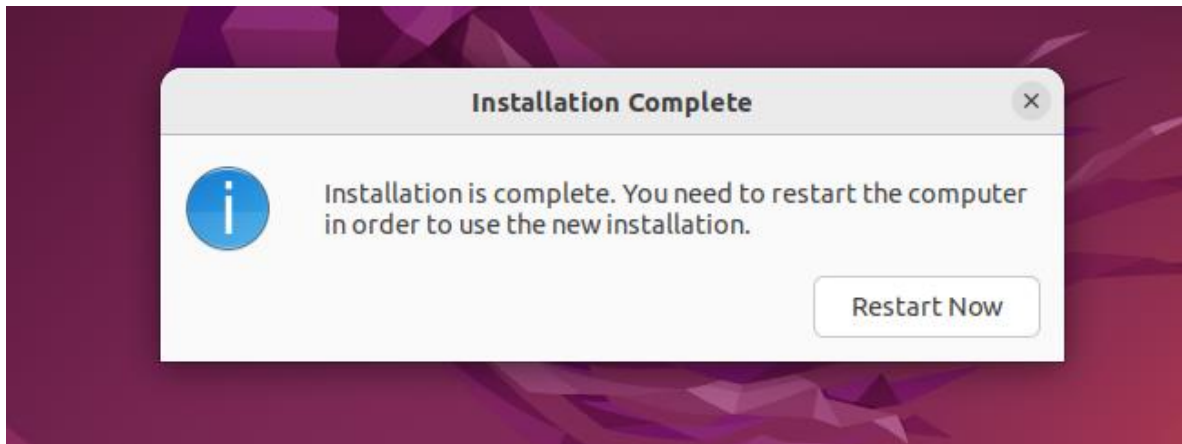
Back

Continue

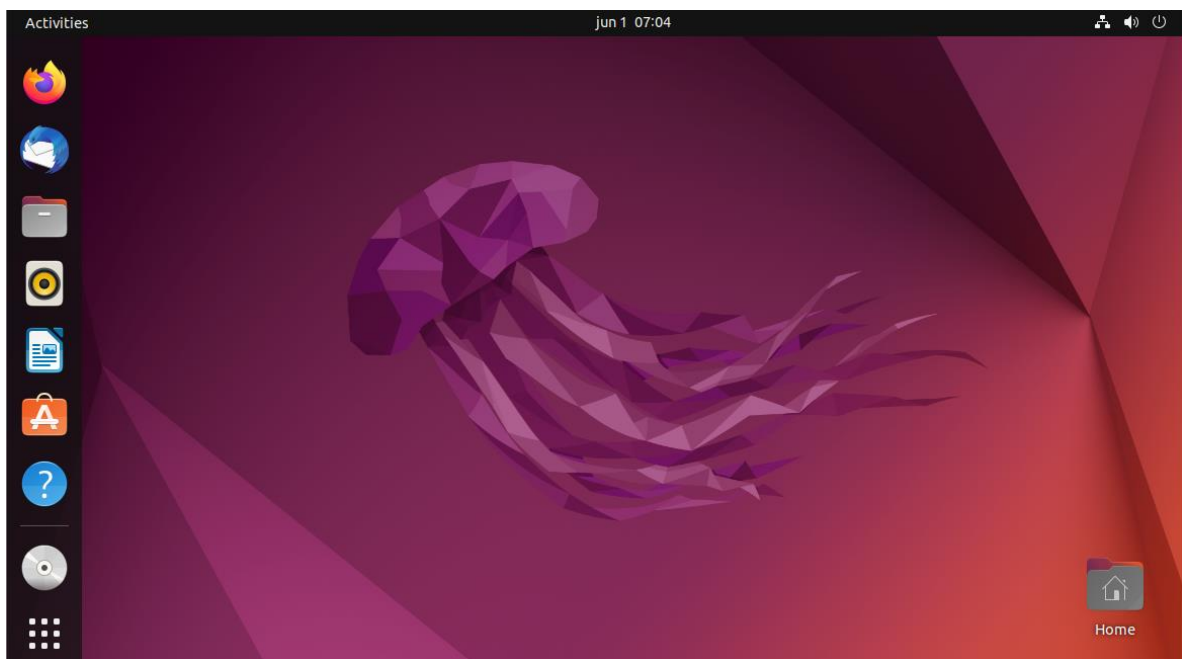
Por fin se está instalando el ubuntu



Completado la instalación nos pedirá reiniciar



Y al iniciar tendremos el Ubuntu instalado.



Instalación de Docker y Docker-Compose

Ahora que ya tenemos nuestro Ubuntu por fin podemos iniciar con el proceso de instalación de lo que utilizaremos para el desarrollo de este proyecto.

Primero que todo voy a loguearme como super usuario con sudo su y después voy a hacer un apt update

```
root@yuru-virtual-machine: /home/yuru
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

yuru@yuru-virtual-machine:~$ sudo su
[sudo] password for yuru:
root@yuru-virtual-machine:~# apt update
Hit:1 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:2 http://pa.archive.ubuntu.com/ubuntu jammy InRelease
Hit:3 http://pa.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:4 http://pa.archive.ubuntu.com/ubuntu jammy-backports InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
150 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

Instalare el Docker con el comando “apt install docker.io -y”

```
root@yuru-virtual-machine:~# apt install docker.io -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd git git-man liberror-perl pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools btrfs-progs cgroupfs-mount | cgroup-lite debootstrap
  docker-doc rinse zfs-fuse | zfsutils git-daemon-run | git-daemon-sysvinit
  git-doc git-email git-gui gitk gitweb git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  bridge-utils containerd docker.io git git-man liberror-perl pigz runc
  ubuntu-fan
```

Luego instalare el plugin de Docker compose con el comando “apt install docker-compose -y”

```
Processing triggers for Man-db (2.10.2-1) ...
root@yuru-virtual-machine:~# apt install docker-compose -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  python3-attr python3-distutils python3-docker python3-dockerpty
  python3-dockerpty python3-dotenv python3-jsonschema python3-pyrsistent
  python3-setuptools python3-texttable python3-websocket
Suggested packages:
  python-attr-doc python-jsonschema-doc python-setuptools-doc
The following NEW packages will be installed:
  docker-compose python3-attr python3-distutils python3-docker
  python3-dockerpty python3-dotenv python3-jsonschema
  python3-pyrsistent python3-setuptools python3-texttable python3-websocket
0 upgraded, 12 newly installed, 0 to remove and 150 not upgraded.
Need to get 909 kB of archives.
```

Proceso de creación del directorio y del archivo docker-compose.yml

Primero que todo vamos a ver en que carpeta estamos con un pwd

```
root@yuru-virtual-machine:/home/yuru# pwd
/home/yuru
```

Crearemos un directorio llamado prueba1 con “mkdir prueba1” y entraremos al mismo con “cd prueba1”

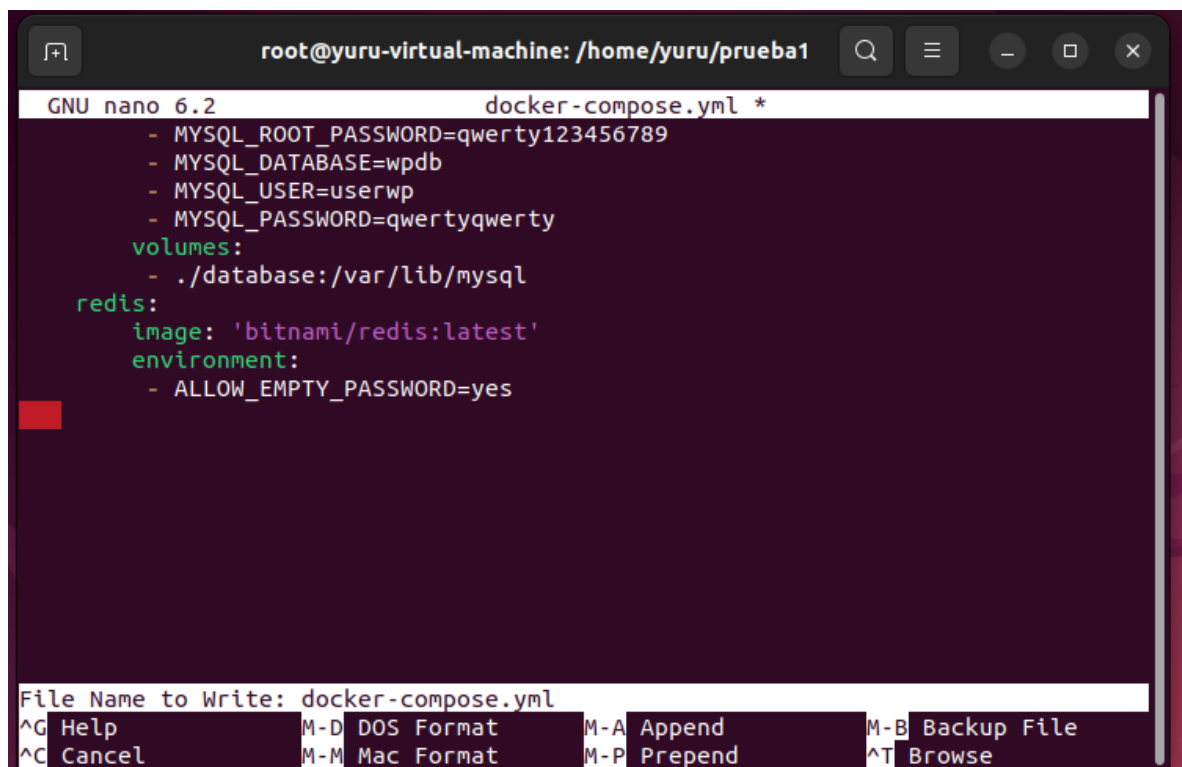
```
root@yuru-virtual-machine:/home/yuru# mkdir prueba1
root@yuru-virtual-machine:/home/yuru# cd prueba1
```

Luego crearemos el documento docker-compose.yml con el comando “nano docker-compose.yml”

```
root@yuru-virtual-machine:/home/yuru# cd prueba1
root@yuru-virtual-machine:/home/yuru/prueba1# nano docker-compose.yml
```

Dentro del documento agregaremos los siguientes parámetros

Le damos a ctrl + x y luego Y para guardar el documento posteriormente le damos a iniciar



```
root@yuru-virtual-machine: /home/yuru/prueba1
GNU nano 6.2 docker-compose.yml *
- MYSQL_ROOT_PASSWORD=qwerty123456789
- MYSQL_DATABASE=wpdb
- MYSQL_USER=userwp
- MYSQL_PASSWORD=qwertyqwerty
volumes:
- ./database:/var/lib/mysql
redis:
  image: 'bitnami/redis:latest'
  environment:
    - ALLOW_EMPTY_PASSWORD=yes

File Name to Write: docker-compose.yml
^G Help      M-D DOS Format  M-A Append     M-B Backup File
^C Cancel    M-M Mac Format  M-P Prepend    ^T Browse
```

Ahora vamos a explicar paso a paso cada parte del documento docker-compose.yml

version: '2'

services:

nginx-proxy:

image: jwilder/nginx-proxy

ports:

- "80:80"

- "443:443"

volumes:

- /var/run/docker.sock:/tmp/docker.sock:ro

- ./certs:/etc/nginx/certs:ro

- ./vhostd:/etc/nginx/vhost.d

- ./html:/usr/share/nginx/html

labels:

- com.github.jrcs.letsencrypt_nginx_proxy_companion.nginx_proxy

letsencrypt:

image: jrcs/letsencrypt-nginx-proxy-companion

image: jrcs/letsencrypt-nginx-proxy-companion:v1.13

restart: always

environment:

- NGINX_PROXY_CONTAINER=nginx-proxy

- DEFAULT_EMAIL=example@gmail.com

volumes:

- ./certs:/etc/nginx/certs:rw

- ./vhostd:/etc/nginx/vhost.d

- ./html:/usr/share/nginx/html

- /var/run/docker.sock:/var/run/docker.sock:ro

wordpress:

image: wordpress

container_name: wordpress_1

links:

- mariadb:mysql
- redis

expose:

- 80

environment:

- WORDPRESS_DB_PASSWORD=qwertyqwerty
- WORDPRESS_DB_USER=userwp
- WORDPRESS_DB_NAME=wpdb
- WP_REDIS_HOST="redis"
- "VIRTUAL_HOST=prueba.local"
- "LESENCRYPT_HOST=prueba.local"
- "LESENCRYPT_EMAIL=example@gmail.com"

volumes:

- ./html:/var/www/html

wordpress2:

image: wordpress

container_name: wordpress_2

links:

- mariadb:mysql
- redis

expose:

- 80

environment:

- WORDPRESS_DB_PASSWORD=qwertyqwerty
- WORDPRESS_DB_USER=userwp
- WORDPRESS_DB_NAME=wpdb
- WP_REDIS_HOST="redis"

- "VIRTUAL_HOST=prueba.local"
- "LETSENCRYPT_HOST=prueba.local"
- "LETSENCRYPT_EMAIL=example@gmail.com"

volumes:

- ./html:/var/www/html

wordpress3:

image: wordpress

container_name: wordpress_3

links:

- mariadb:mysql
- redis

expose:

- 80

environment:

- WORDPRESS_DB_PASSWORD=qwertyqwerty
- WORDPRESS_DB_USER=userwp
- WORDPRESS_DB_NAME=wpdb
- WP_REDIS_HOST="redis"
- "VIRTUAL_HOST=prueba.local"
- "LETSENCRYPT_HOST=prueba.local"
- "LETSENCRYPT_EMAIL=example@gmail.com"

volumes:

- ./html:/var/www/html

wordpress4:

image: wordpress

container_name: wordpress_4

links:

- mariadb:mysql
- redis

expose:

- 80

environment:

- WORDPRESS_DB_PASSWORD=qwertyqwerty
- WORDPRESS_DB_USER=userwp
- WORDPRESS_DB_NAME=wpdb
- WP_REDIS_HOST="redis"
- "VIRTUAL_HOST=prueba.local"
- "LESENCRYPT_HOST=prueba.local"
- "LESENCRYPT_EMAIL=example@gmail.com"

volumes:

- ./html:/var/www/html

mariadb:

image: mariadb

environment:

- MYSQL_ROOT_PASSWORD=qwerty123456789
- MYSQL_DATABASE=wpdb
- MYSQL_USER=userwp
- MYSQL_PASSWORD=qwertyqwerty

volumes:

- ./database:/var/lib/mysql

redis:

image: 'bitnami/redis:latest'

environment:

- ALLOW_EMPTY_PASSWORD=yes

La implementación del balanceador de carga estará a cargo del nginx-proxy y la parte de que responda por https y no por http está a cargo del letsencrypt

```

1. version: '2'
2.
3. services:
4.     nginx-proxy:
5.         image: jwilder/nginx-proxy
6.
7.         ports:
8.             - "80:80"
9.             - "443:443"
10.        volumes:
11.            - /var/run/docker.sock:/tmp/docker.sock:ro
12.            - ./certs:/etc/nginx/certs:ro
13.            - ./vhostd:/etc/nginx/vhost.d
14.            - ./html:/usr/share/nginx/html
15.        labels:
16.            -
17.        com.github.jrcs.letsencrypt_nginx_proxy_companion.nginx_proxy
18.        letsencrypt:
19.            #         image: jrcs/letsencrypt-nginx-proxy-companion
20.            image: jrcs/letsencrypt-nginx-proxy-
21.            companion:v1.13
22.            restart: always
23.            environment:
24.                - NGINX_PROXY_CONTAINER=nginx-proxy
25.            #         - DEFAULT_EMAIL=example@gmail.com
26.            volumes:
27.                - ./certs:/etc/nginx/certs:rw
28.                - ./vhostd:/etc/nginx/vhost.d
29.                - ./html:/usr/share/nginx/html
30.                - /var/run/docker.sock:/var/run/docker.sock:ro

```

Luego viene lo que es el wordpress como tal, en mi caso utilizare 4 WordPress para implementar el letsencrypt debemos ponerle los parámetros virtual host, letsencrypt_host y el letsencrypt_email en mi caso será prueba.local . para conectar posteriormente el redis utilizare el wp_redis_host y lo conectare en los links, también los wordpress estarán conectados a una base de datos de mariadb

```

29.        wordpress:
30.            image: wordpress
31.            container_name: wordpress_1
32.            links:
33.                - mariadb:mysql
34.                - redis
35.            expose:
36.                - 80
37.            environment:
38.                - WORDPRESS_DB_PASSWORD=qwertyqwerty
39.                - WORDPRESS_DB_USER=userwp
40.                - WORDPRESS_DB_NAME=wpdb

```

```
41.         - WP_REDIS_HOST="redis"
42.         - "VIRTUAL_HOST=prueba.local"
43.         - "LETSENCRYPT_HOST=prueba.local"
44.         - "LETSENCRYPT_EMAIL= example@gmail.com "
45.     volumes:
46.         - ./html:/var/www/html
47. wordpress2:
48.     image: wordpress
49.     container_name: wordpress_2
50.     links:
51.         - mariadb:mysql
52.         - redis
53.     expose:
54.         - 80
55.     environment:
56.         - WORDPRESS_DB_PASSWORD=qwertyqwerty
57.         - WORDPRESS_DB_USER=userwp
58.         - WORDPRESS_DB_NAME=wpdb
59.         - WP_REDIS_HOST="redis"
60.         - "VIRTUAL_HOST=prueba.local"
61.         - "LETSENCRYPT_HOST=prueba.local"
62.         - "LETSENCRYPT_EMAIL= example@gmail.com "
63.     volumes:
64.         - ./html:/var/www/html
65. wordpress3:
66.     image: wordpress
67.     container_name: wordpress_3
68.     links:
69.         - mariadb:mysql
70.         - redis
71.     expose:
72.         - 80
73.     environment:
74.         - WORDPRESS_DB_PASSWORD=qwertyqwerty
75.         - WORDPRESS_DB_USER=userwp
76.         - WORDPRESS_DB_NAME=wpdb
77.         - WP_REDIS_HOST="redis"
78.         - "VIRTUAL_HOST=prueba.local"
79.         - "LETSENCRYPT_HOST=prueba.local"
80.         - "LETSENCRYPT_EMAIL= example@gmail.com"
81.     volumes:
82.         - ./html:/var/www/html
83. wordpress4:
84.     image: wordpress
85.     container_name: wordpress_4
86.     links:
87.         - mariadb:mysql
88.         - redis
89.     expose:
90.         - 80
91.     environment:
92.         - WORDPRESS_DB_PASSWORD=qwertyqwerty
```

```

93.         - WORDPRESS_DB_USER=userwp
94.         - WORDPRESS_DB_NAME=wpdb
95.         - WP_REDIS_HOST="redis"
96.         - "VIRTUAL_HOST=prueba.local"
97.         - "LETSENCRYPT_HOST=prueba.local"
98.         - "LETSENCRYPT_EMAIL= example@gmail.com"
99.     volumes:
100.         - ./html:/var/www/html
101. mariadb:
102.     image: mariadb
103.
104.     environment:
105.         - MYSQL_ROOT_PASSWORD=qwerty123456789
106.         - MYSQL_DATABASE=wpdb
107.         - MYSQL_USER=userwp
108.         - MYSQL_PASSWORD=qwertyqwerty
109.     volumes:
110.         - ./database:/var/lib/mysql

```

La ultima parte es la imagen del redis

```

1.     redis:
2.         image: 'bitnami/redis:latest'
3.         environment:
4.             - ALLOW_EMPTY_PASSWORD=yes
5.

```

Ahora realmente si le damos a iniciar a nuestro servidor no vamos a poder entrar en el mismo, ya que dentro del documento docker-compose.yml estamos diciendo que responda por https y no hemos configurado un host en mi caso prueba.local vamos a configurar este paso.

Configurando prueba.local para responder por HTTPS y no por HTTP

Ya que es una maquina virtual nueva de Ubuntu necesitaremos instalar algunas dependencias entonces instalaremos el net-tools para poder utilizar el comando ifconfig

```
root@yuru-virtual-machine:/home/yuru/prueba1# apt install net-tools
```

Una vez instalado vemos la ip

```

root@yuru-virtual-machine:/home/yuru/prueba1# ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255
    ether 02:42:f9:ac:9a:80 txqueuelen 0 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0

```

Vemos las ips con ifconfig

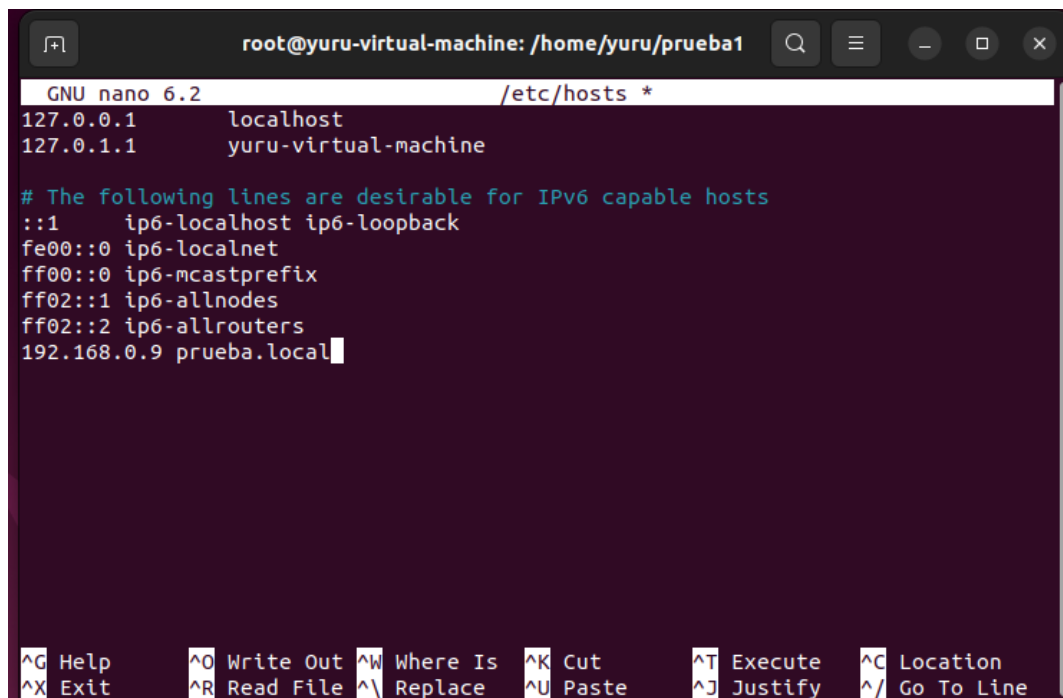
```
root@yuru-virtual-machine:/home/yuru/prueba1# ifconfig
```

Vemos que tenemos dos ip una que crea docker automáticamente al instalarse y otra creada en modo bidge, apuntamos la que está en modo bridge. “192.168.0.9” en mi caso.

```
root@yuru-virtual-machine:/home/yuru/prueba1# ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255
    ether 02:42:f9:ac:9a:80 txqueuelen 0 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.9 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 fe80::273e:3b44:4eb2:a003 prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:95:c9:97 txqueuelen 1000 (Ethernet)
    RX packets 50152 bytes 74969779 (74.9 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 11073 bytes 765282 (765.2 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Luego vamos a modificar el documento hosts con el comando “nano /etc/hosts” y agregaremos 192.168.0.9 prueba.local luego daremos a ctrl+x y Y , posteriormente le damos a enter para salir



```
root@yuru-virtual-machine: /home/yuru/prueba1
GNU nano 6.2 /etc/hosts *
127.0.0.1 localhost
127.0.1.1 yuru-virtual-machine

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
192.168.0.9 prueba.local

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute  ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify  ^_ Go To Line
```

Con esto nos conectaremos mediante https

Ahora vamos a iniciar el docker-compose.yml con el comando “docker-compose up -d”

```
root@yuru-virtual-machine:/home/yuru/prueba1# nano /etc/hosts
root@yuru-virtual-machine:/home/yuru/prueba1# docker-compose up -d
```

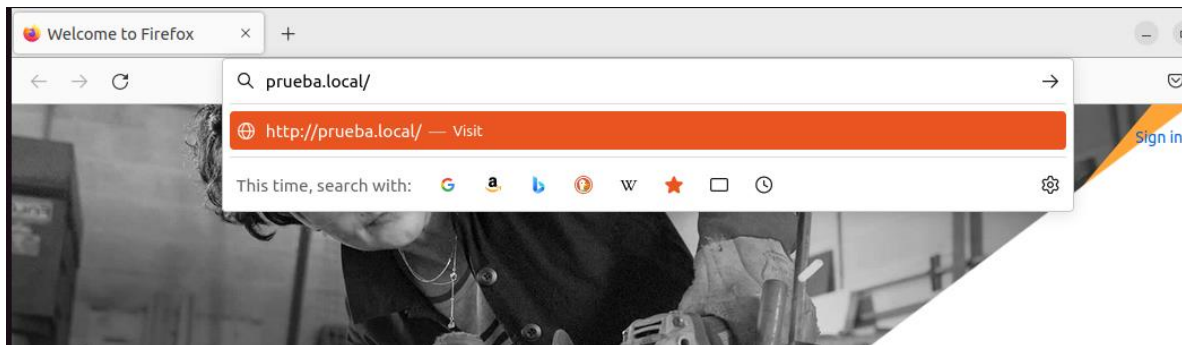

Se van a bajar todas las imágenes necesarias

```
latest: Pulling from jwilder/nginx-proxy
214ca5fb9032: Downloading [==>
66eec13bb714: Downloading [=====>
 3.664MB/25.35MBwnload complete
 602B/602B Waiting
c4547ad15a20: Waiting
d31373136b98: Waiting
c6152356a9b4: Waiting
18751395f202: Waiting
a6e84eb2ea8d: Waiting
0dcbb9c3079: Waiting
c3095acc487a: Waiting
197d7b412f2a: Waiting
4f4fb700ef54: Waiting
```

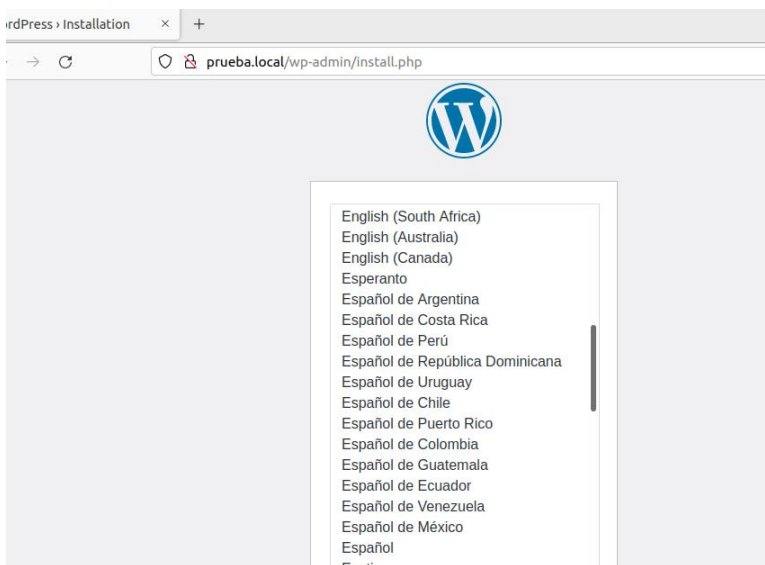
Cuando se termine la subida del docker-compose veremos un done en todos, aunque aun no hemos terminado

```
Creating prueba1_nginx-proxy_1 ... done
Creating prueba1_mariadb_1     ... done
Creating prueba1_redis_1       ... done
Creating prueba1_letsencrypt_1 ... done
Creating wordpress_2           ... done
Creating wordpress_3           ... done
Creating wordpress_1           ... done
Creating wordpress_4           ... done
root@yuru-virtual-machine:/home/yuru/prueba1#
```

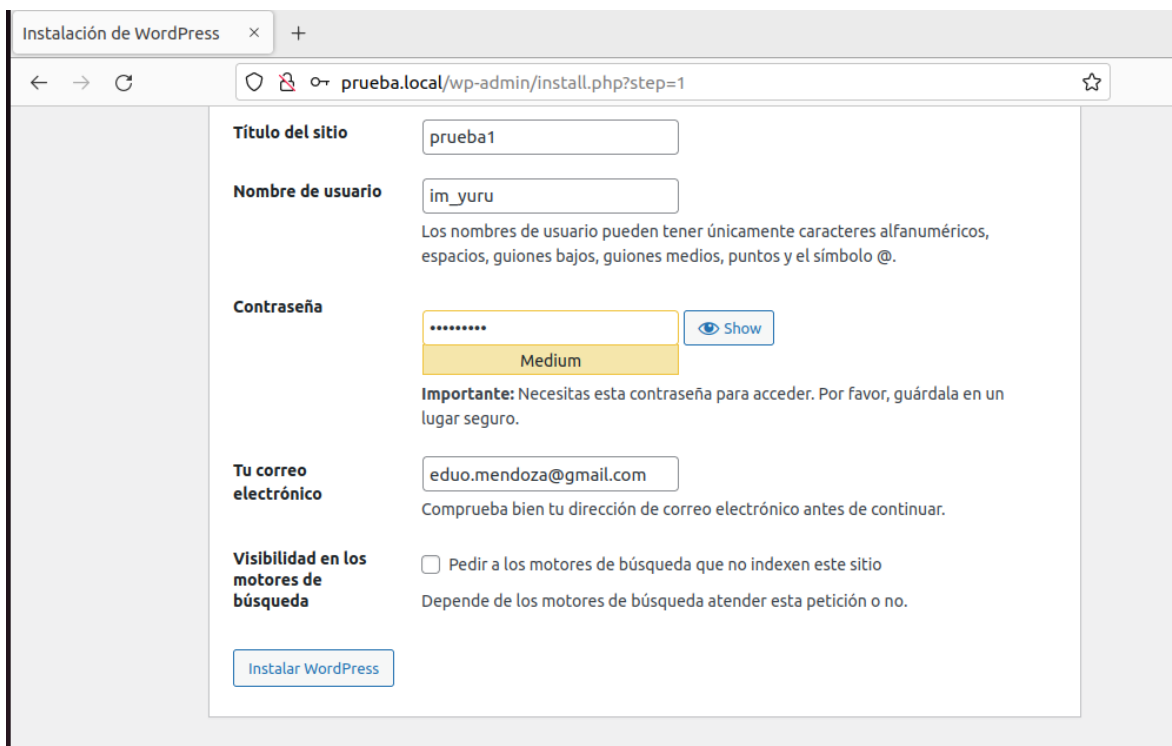
Vamos a comprobar que todo esté funcionando correctamente, nos vamos a nuestro navegador y ponemos prueba.local/



Y entramos, si hemos hecho todo bien debería aparecernos el cliente de instalación del WordPress, elegimos nuestro idioma y le damos a continuar



Agregamos un titulo para el sitio, usuario y la contraseña, también un correo



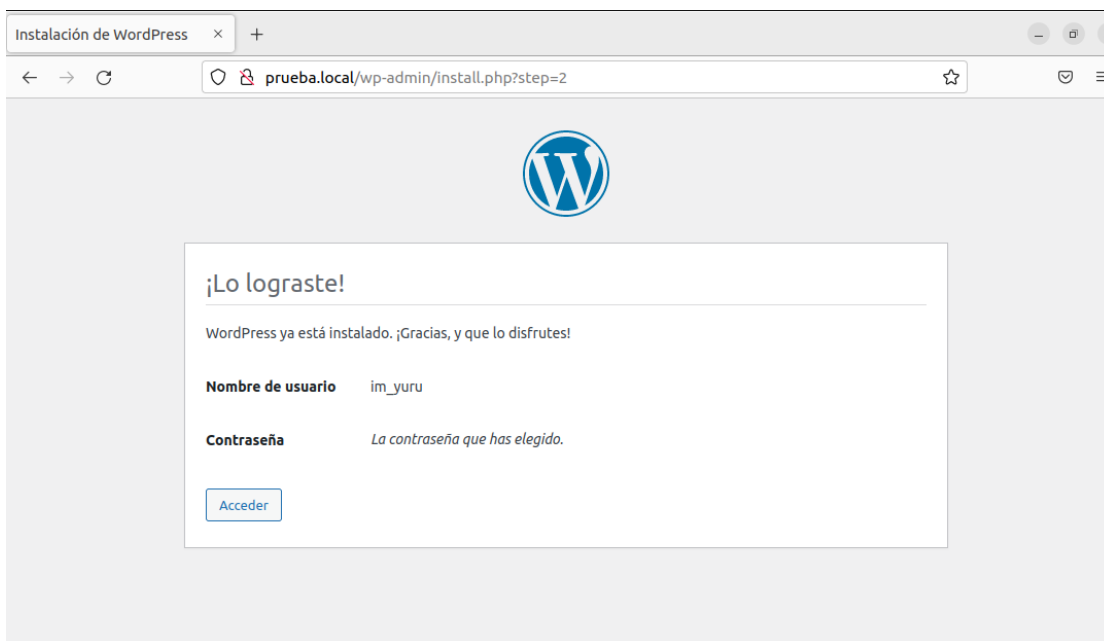
The screenshot shows the first step of the WordPress installation process in a web browser. The address bar indicates the URL is `prueba.local/wp-admin/install.php?step=1`. The form contains the following fields and options:

- Título del sitio:** A text input field containing the value "prueba1".
- Nombre de usuario:** A text input field containing the value "im_yuru". Below this field is a note: "Los nombres de usuario pueden tener únicamente caracteres alfanuméricos, espacios, guiones bajos, guiones medios, puntos y el símbolo @."
- Contraseña:** A password input field with a strength indicator showing "Medium". A "Show" button is located to the right of the field. Below this field is an important note: "Importante: Necesitas esta contraseña para acceder. Por favor, guárdala en un lugar seguro."
- Tu correo electrónico:** A text input field containing the value "eduo.mendoza@gmail.com". Below this field is a note: "Comprueba bien tu dirección de correo electrónico antes de continuar."
- Visibilidad en los motores de búsqueda:** A checkbox labeled "Pedir a los motores de búsqueda que no indexen este sitio". Below this is a note: "Depende de los motores de búsqueda atender esta petición o no."

At the bottom of the form is a blue button labeled "Instalar WordPress".

Posteriormente le damos a instalar wordpress

Si lo hicimos todo bien entonces nos debe salir que lo logramos. Le damos a acceder



The screenshot shows the second step of the WordPress installation process. The address bar indicates the URL is `prueba.local/wp-admin/install.php?step=2`. The page features the WordPress logo at the top center. Below the logo is a white box with the following content:

- ¡Lo lograste!** (Congratulations!)
- A message: "WordPress ya está instalado. ¡Gracias, y que lo disfrutes!" (WordPress is now installed. Thank you, and enjoy it!).
- Nombre de usuario:** The value "im_yuru" is displayed.
- Contraseña:** The text "La contraseña que has elegido." (The password you have chosen.) is displayed.

At the bottom of the white box is a blue button labeled "Acceder" (Log in).

Nos logueamos

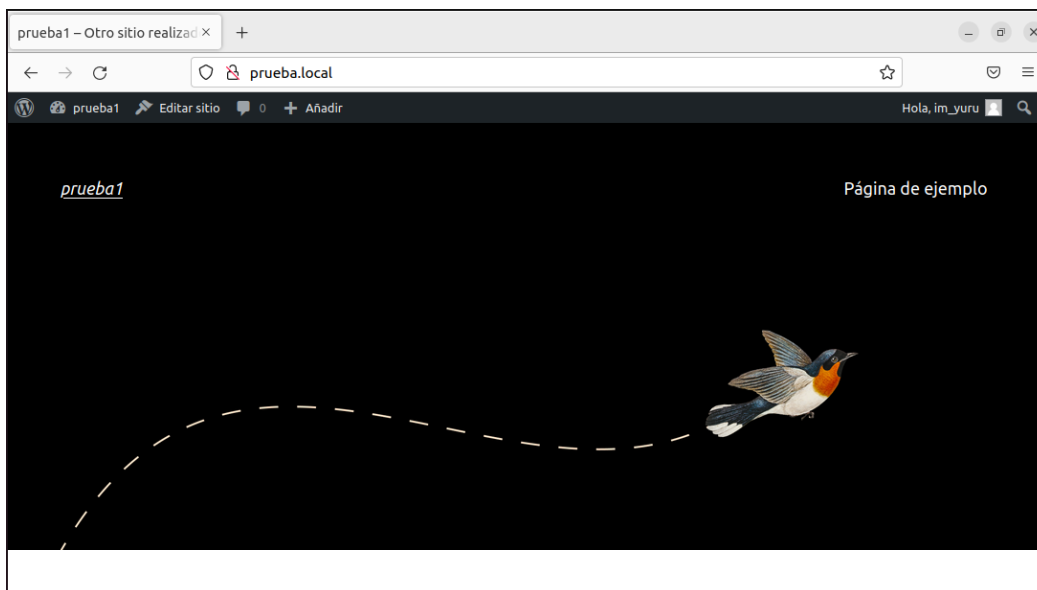


The image shows the WordPress login interface. At the top is the WordPress logo. Below it is a login box with the following elements:

- A text input field labeled "Nombre de usuario o correo electrónico" containing the text "im_yuru".
- A password input field labeled "Contraseña" with masked characters (dots) and an eye icon to toggle visibility.
- A checkbox labeled "Recuérdame".
- A blue button labeled "Acceder".

Below the login box, there is a link that says "¿Has olvidado tu contraseña?" and another link below it that says "← Ir a prueba1".

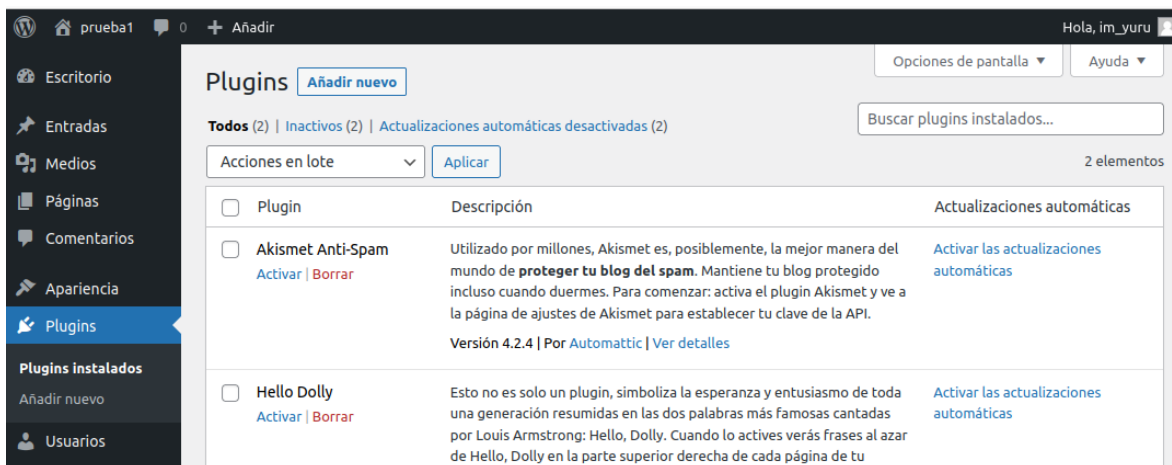
Al iniciar veremos que tenemos el wordpress activado



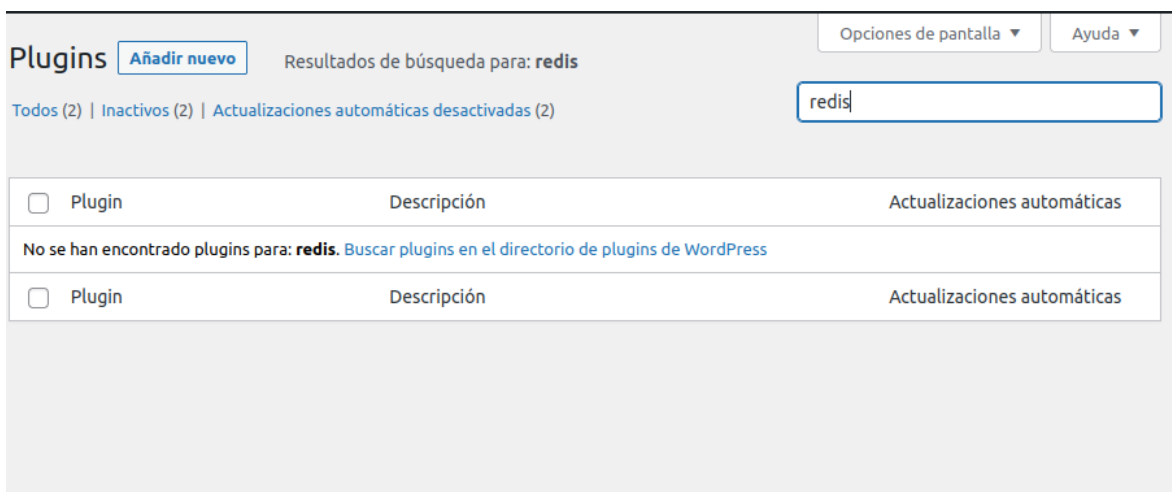
Instalación de plugins y Redis Object Cache

Ahora vamos configurar Redis Object Cache, si bien es cierto ya lo configuramos dentro del docker-compose.yml hay ciertos puntos que tenemos que configurar de manera distinta.

Dentro del wordpress nos vamos al apartado de plugins



Ponemos redis en el buscador y luego a buscar plugins en el directorio de plugins de wordpress



Encontraremos el plugin de redis object cache, le damos a instalar ahora



Redis Object Cache

[Instalar ahora](#)[Más detalles](#)

A persistent object cache backend powered by Redis. Supports Predis, PhpRedis, Credis, Relay, HHVM, replication,...

Por Till Krüss

★★★★★ (105)

100.000+ instalaciones activas

Última actualización: hace 7 días

✓ **Compatible** con tu versión de WordPress

Posteriormente nos vamos al plugin y le damos a activar cache de objetos

Redis Object Cache

[Resumen](#)[Métricas](#)[Diagnóstico](#)

Resumen

Estado	✗ El dependiente no está instalado
Dependiente:	✗ No está instalada
Sistema de archivos:	✓ Con permisos de escritura

Activar la caché de objetos

Recursos

Al darle a activar la cache de objetos vemos que no se conecta, entonces que vamos a hacer

Redis Object Cache

Resumen
Métricas
Diagnóstico

Resumen

Estado ✗ No conectado

Dependiente: ✓ Válido

Sistema de archivos: ✓ Con permisos de escritura

[Desactivar la caché de objetos](#)

Recursos

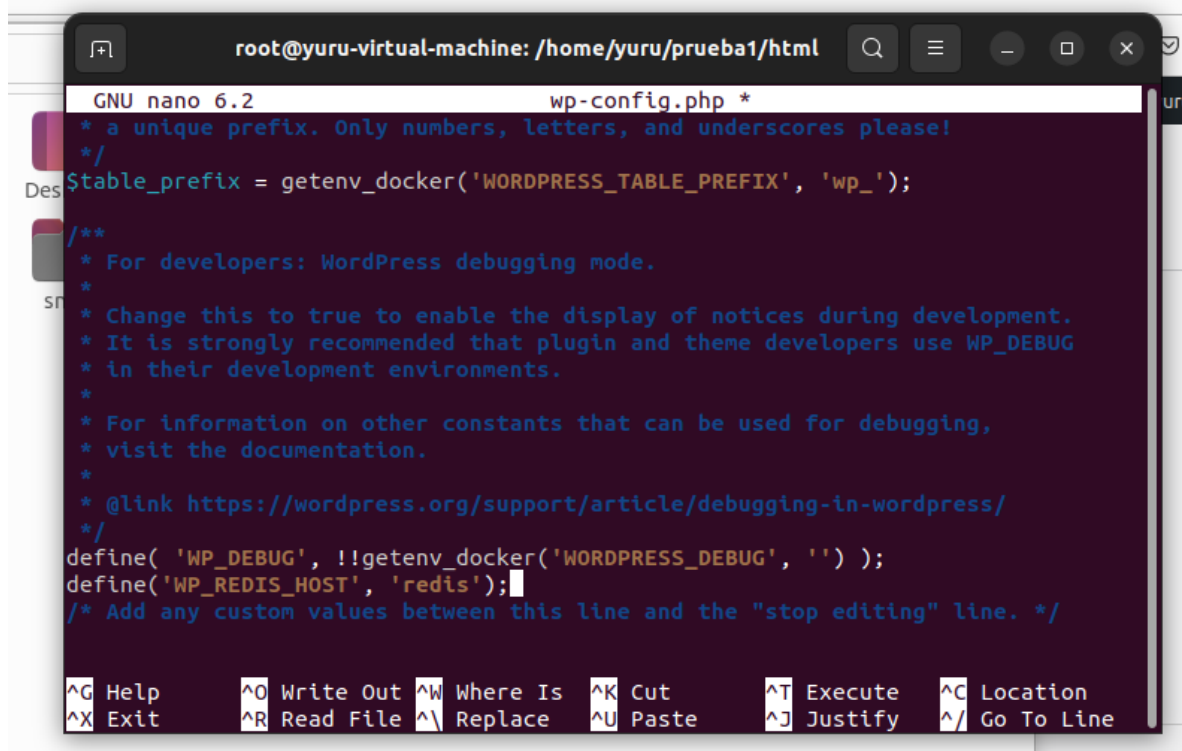
Nos vamos a ir a nuestra consola y vamos a escribir “cd /html” y luego vamos a hacer un “ls -la” para ver todos los archivos.

```
root@yuru-virtual-machine:/home/yuru/prueba1# cd html
root@yuru-virtual-machine:/home/yuru/prueba1/html# ls -la
total 248
drwxr-xr-x  5 www-data www-data 4096 jun  1 08:09 .
drwxr-xr-x  6 root      root    4096 jun  1 07:44 ..
-rw-r--r--  1 www-data www-data  553 jun  1 07:52 .htaccess
-rw-r--r--  1 www-data www-data  405 feb  6  2020 index.php
-rw-r--r--  1 www-data www-data 19915 dic 31 19:15 license.txt
-rw-r--r--  1 www-data www-data  7401 mar 22 16:11 readme.html
-rw-r--r--  1 www-data www-data  7165 ene 20  2021 wp-activate.php
drwxr-xr-x  9 www-data www-data 4096 may 24 14:02 wp-admin
-rw-r--r--  1 www-data www-data   351 feb  6  2020 wp-blog-header.php
-rw-r--r--  1 www-data www-data 2338 nov  9  2021 wp-comments-post.php
-rw-r--r--  1 root      root         3 jun  1 08:06 wp-config
-rw-rw-r--  1 www-data www-data 5480 may 28 23:10 wp-config-docker.php
-rw-r--r--  1 www-data www-data 5584 jun  1 07:44 wp-config.php
-rw-r--r--  1 www-data www-data 3001 dic 14 03:44 wp-config-sample.php
drwxr-xr-x  7 www-data www-data 4096 jun  1 08:02 wp-content
-rw-r--r--  1 www-data www-data  3943 abr 28 04:49 wp-cron.php
drwxr-xr-x 26 www-data www-data 16384 may 24 14:02 wp-includes
-rw-r--r--  1 www-data www-data  2494 mar 19 15:31 wp-links-opml.php
-rw-r--r--  1 www-data www-data  3973 abr 11 20:47 wp-load.php
-rw-r--r--  1 www-data www-data 48498 abr 29 09:36 wp-login.php
```

Vamos a modificar el archivo wp-config.php

```
root@yuru-virtual-machine:/home/yuru/prueba1/html# nano wp-config.php
```

Nos iremos hasta debajo de wp_debug y pondremos define('WP_REDIS_HOST', 'redis'); y lo guardaremos con ctrl + x y Y + enter

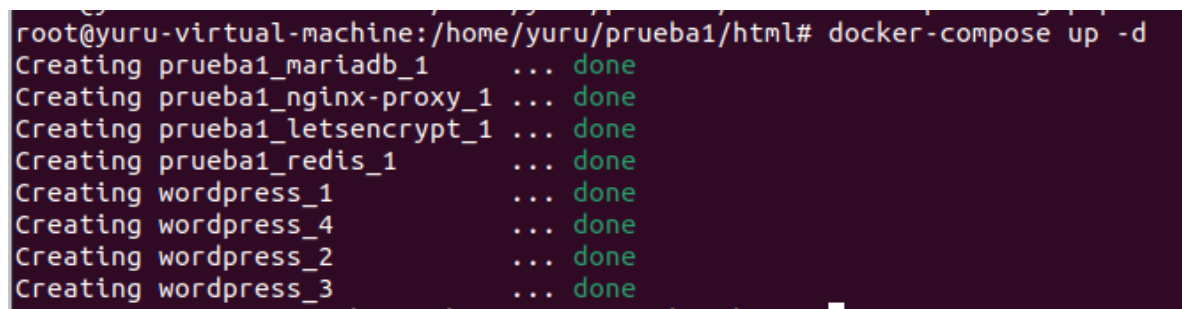


```
root@yuru-virtual-machine: /home/yuru/prueba1/html
GNU nano 6.2 wp-config.php *
/*
 * a unique prefix. Only numbers, letters, and underscores please!
 */
$table_prefix = getenv_docker('WORDPRESS_TABLE_PREFIX', 'wp_');

/**
 * For developers: WordPress debugging mode.
 *
 * Change this to true to enable the display of notices during development.
 * It is strongly recommended that plugin and theme developers use WP_DEBUG
 * in their development environments.
 *
 * For information on other constants that can be used for debugging,
 * visit the documentation.
 *
 * @link https://wordpress.org/support/article/debugging-in-wordpress/
 */
define( 'WP_DEBUG', !getenv_docker('WORDPRESS_DEBUG', '') );
define('WP_REDIS_HOST', 'redis');
/* Add any custom values between this line and the "stop editing" line. */

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

Luego iniciaremos nuevamente los servicios de docker-compose con docker-compose up -d



```
root@yuru-virtual-machine:/home/yuru/prueba1/html# docker-compose up -d
Creating prueba1_mariadb_1      ... done
Creating prueba1_nginx-proxy_1 ... done
Creating prueba1_letsencrypt_1 ... done
Creating prueba1_redis_1       ... done
Creating wordpress_1           ... done
Creating wordpress_4           ... done
Creating wordpress_2           ... done
Creating wordpress_3           ... done
```

Si nos vamos nuevamente al plugin veremos que ya estaría instalado correctamente

Redis Object Cache

Resumen
Métricas
Diagnóstico

Resumen

Estado ✓ Conectado
Dependiente: ✓ Válido
Sistema de archivos: ✓ Con permisos de escritura

Conexión

Cliente: Predis (v1.1.10)
Host: redis
Puerto: 6379
Base de datos: 0
Tiempo de inactividad de conexión: 1s

Ahora vamos a instalar dos plugins mas , uno para control de file integrity y limitaremos la cantidad de veces fallidas de login

Primero la cantidad de logins, utilizaremos limit login attempts reloaded le daremos a buscar e instalaremos el plugin.

Todos (3) | Activo (1) | Inactivos (2) | Dependiente (1) | Actualizaciones automáticas desactivadas (3)

☐ Plugin
Descripción
Actualizaciones automáticas

No se han encontrado plugins para: **limit login attempts reloaded** . [Buscar plugins en el directorio de plugins de WordPress](#)

☐ Plugin
Descripción
Actualizaciones automáticas



Limit Login Attempts Reloaded

Instalar ahora
Más detalles

Block excessive login attempts and protect your site against brute force attacks. Simple, yet powerful...

Por Limit Login Attempts Reloaded

★★★★★ (914)
2+ millones instalaciones

Última actualización: hace 7 días
✓ Compatible con tu versión de

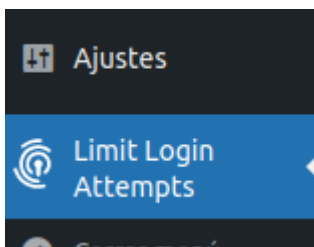


Bienvenido a Limit Login Attempts Reloaded

Gracias por elegir Limit Login Attempts Reloaded - Un plugin sencillo, pero potente, de bloqueo de bots que mantiene segura tu página de acceso.

Con este plugin, puedes...

- ★ Ver quién está intentando acceder a tu web
- ★ Protégete de futuros ataques permitiendo o rechazando IP



Una vez instalado podremos entrar desde la barra de la izquierda al plugin

Limit Login Attempts Reloaded

[Escritorio](#)[Ajustes](#)[Registros](#)[Depuración](#)

Escritorio de Limit Login Attempts Reloaded

Intentos de acceso fallidos

0

¡Hurra! Ningún intento de acceso fallido hoy

1

Intentos de acceso fallidos

Protección Premium desactivada

Como usuario gratuito, tu servidor local está absorbiendo el tráfico provocado por los ataques de fuerza bruta, ralentizando potencialmente tu web. Actualiza hoy a la versión Premium para externalizar estos ataques a través de nuestra aplicación en la nube y ralentizar los futuros ataques con la regulación avanzada.

[ACTUALIZAR A PREMIUM](#)

Si nos vamos a ajuste podremos ajustar el tiempo de minutos por bloqueo yo lo pondré en 5 minutos

Ajustes de la aplicación

La aplicación absorbe la carga principal provocada por los ataques de fuerza bruta, analizan los intentos de acceso y bloquean a los visitantes no deseados. También podría realizar otras funciones de servicio.

▼ Aplicación local

Bloqueo

4

reintentos permitidos

5

minutos por bloqueo

4

bloqueos incrementan el tiempo de bloqueo a

24

 horas

24

 horas hasta reiniciar los reintentos

Orígenes IP de confianza

REMOTE_ADDR

Especifica los orígenes en los que confías por orden de prioridad y separados por comas.
Te recomendamos encarecidamente que **no utilices** otra cosa que no sea
'REMOTE_ADDR' ya que otros orígenes se pueden falsificar fácilmente. Ejemplos:

Ahora vamos a ver un plugin de file integrity instalaremos el Wordpress file monitor



WordPress File Monitor

[Instalar ahora](#) [Más detalles](#)

Monitor your website for added, changed and deleted files! Track changes in all website directories and get email alerts! Stay secure for free!

Por nicolly

★★★★★ (12)

3.000+ instalaciones activas

Última actualización: hace 3 años

No probado con tu versión de WordPress

<input type="checkbox"/>	WordPress File Monitor Manual Scan Settings Desactivar	Monitor your website for added, changed and deleted files! Track changes in all website directories and get email alerts! Stay secure for free! Versión 1.0.9 Por nicolly Ver detalles	Activar las actualizaciones automáticas
--------------------------	--	---	---

Y vamos a instalar un plugin para cambiar el url de login y que no se vea wp-admin

Instalaremos el wps hide login



WPS Hide Login

Activo [Más detalles](#)

Cambia wp-login.php a lo que quieras.

Por *WPServeur, Nicolaskulka, wpformation*

★★★★★ (2.043)

1+ millón instalaciones activas

Última actualización: hace 1 semana

✓ **Compatible** con tu versión de WordPress

Antes lo veríamos así

← → ↻

prueba.local/wp-login.php?loggedout=true&wp_lang=es_ES



Ahora estás desconectado.

Nombre de usuario o correo electrónico

im_yuru

Contraseña

● ● ● ● ● ● ● ● 

☐ Recuérdame Acceder

[¿Has olvidado tu contraseña?](#)

Una vez instalado el plugin entramos a ajustes-> wps hide login y elegimos el que queremos

WPS Hide Login

¿Necesitas ayuda? Inténtalo en el [foro de soporte](#). Este plugin te lo ofrece amablemente [WPSeurveur](#) (Alojamiento especializado para WordPress)
Descubre nuestros otros plugins: el plugin [WPS Bidouille](#), el plugin [WPS Cleaner](#) y [WPS Limit Login](#)

URL de acceso

`http://prueba.local/`

Protege tu web cambiando la URL de acceso y evitando el acceso a la página «wp-login.php» y al directorio «wp-admin» a quien no esté conectado.

URL de redirección

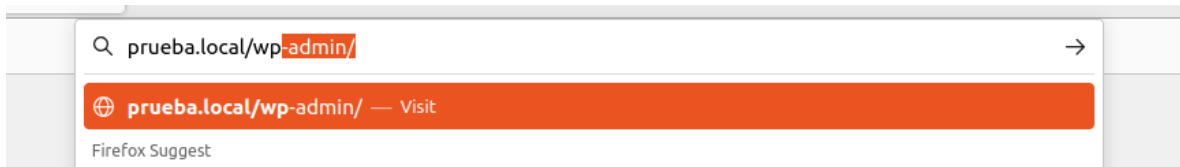
`http://prueba.local/`

Redirige la URL cuando alguien intenta acceder a la página «wp-login.php» y al directorio «wp-admin» sin haber accedido.

Cuando la guardemos podremos entrar desde prueba.local/login



Ahora si intentamos ingresar a wp-admin



Nos redirecciona al error 404

prueba1

Página de ejemplo

404

Conclusiones

De este parcial pude sacar grandes conclusiones las cuales podre implementar en futuros proyectos, lo primero y mas importante es que docker es una herramienta poderosa junto a docker-compose, puede facilitar gran cantidad de procedimientos con unos simples pasos, la implementación del nginx y letsencrypt se torno algo complicada sin la orientación del docente y el redis en ciertos puntos también confuso, pero poco a poco se pudo lograr el objetivo del proyecto el cual era aprender a utilizar estas herramientas en armonía para crear un producto final.

Bibliografía

Overview of Docker Compose. (2022, 1 junio). Docker Documentation.
<https://docs.docker.com/compose/>

Docker Hub. (n.d.). Hub.docker.com. Retrieved June 1, 2022, from
<https://hub.docker.com/r/bitnami/redis/>

Docker Hub. (2019). Docker.com. https://hub.docker.com/_/wordpress

Docker Hub. (n.d.). Hub.docker.com. https://hub.docker.com/_/mariadb

Docker Hub. (n.d.). Hub.docker.com. Retrieved June 1, 2022, from
<https://hub.docker.com/r/linuxserver/letsencrypt>