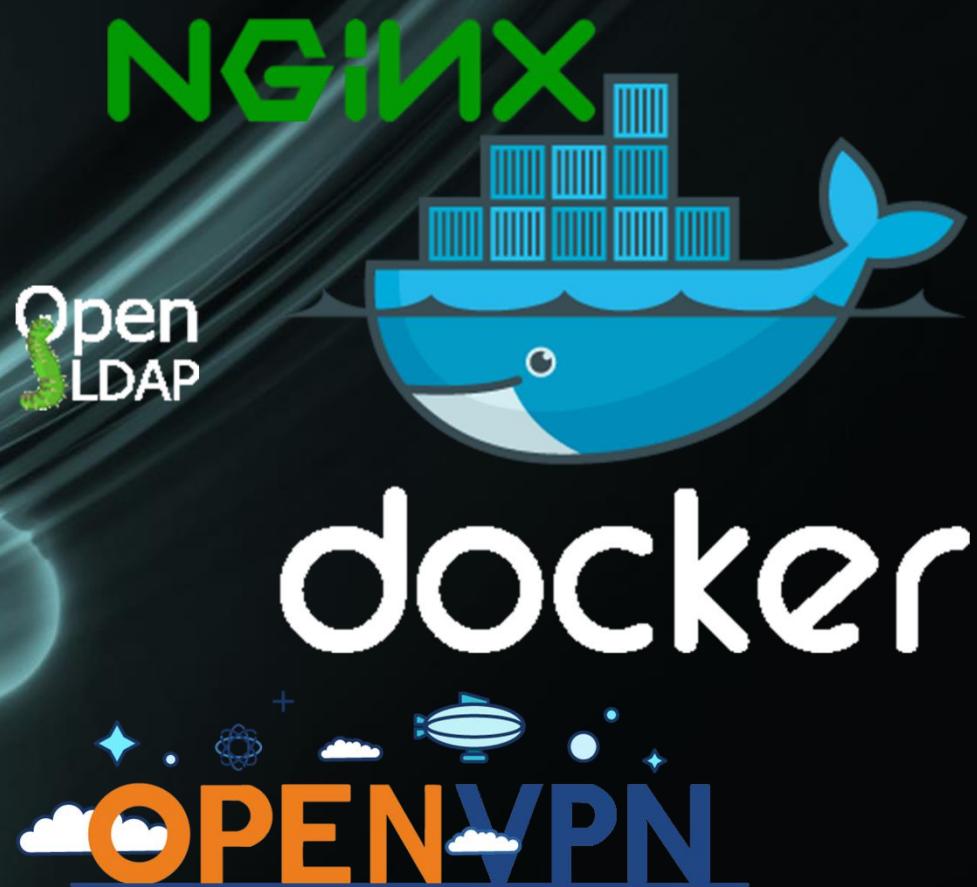


Virtualización de servicios en Docker



Coordinador

Antonio Sepúlveda Ruiz

Grupo de trabajo

Luis Valderas Torres

Daniel De Tomas Plaza

Daniel Palencia Origüel



IES GASPAR MELCHOR DE
Jovellanos
COLEGIO

Contenido

¿Por qué Docker? Y objetivos del proyecto.....	4
Abstract.....	5
Proyecto Docker	5
Instalación De Docker	7
Instalación Docker-Compose	10
Instalación De Openldap en docker.....	11
Instalación de Interfaz Web Openldap en docker	12
Creación y configuración de Usuario, Grupos y Unidades Organizativas en LDAP en Docker	13
Para justificar el usar Docker, hemos replicado el mismo proceso en el propio sistema Debian, sin usar Docker, para ver como Docker nos ayuda hacer despliegues mucho más rápido.....	18
Instalación OpenLdap Debian puro	18
Creación y configuración de Usuario, Grupos y Unidades Organizativas en LDAP en Debian puro	22
Unidades organizativas.....	24
Grupos	24
Usuarios	25
Instalación De Interfaz Web Openldap en Debian puro	27
Instalación OPENVPN DOCKER	33
Configuración Máquina Virtual	33
Docker-Compose Instalación	34
Servicio WEB: NginX	40

¿Que es NginX?	40
NginX vs apache	41
Incluir NginX con docker:	41
Bootstrap:	43
Nnigx con base de datos:	46
Instalación de portainer (Gui para contenedores)	46
Conclusión	51
Web grafía	52

¿Por qué Docker? Y objetivos del proyecto.

Hemos decidido usar estas herramientas, las cuales son desconocidas para nosotros, para expandir nuestras competencias de cara al mundo laboral. Docker es una herramienta de virtualización muy completa y se está ampliando su uso en el mundo laboral.

Se va a simular que es una empresa nueva con pocos recursos, por ende recurre al uso de docker para optimizar los recursos y gastos de la empresa. Debido la situación actual en el mundo laboral se ha incluido la idea de añadir un servidor VPN para el Teletrabajo.

Los objetivos principales son:

Implementación del servidor OpenVPN.

Implementación de servidor web Nginx.

Implementación de OpenLDAP.

Creación de un dominio.

Creación de usuarios de un dominio.

Configuración de la conexión vía VPN entre cliente y debían.

Objetivos secundarios:

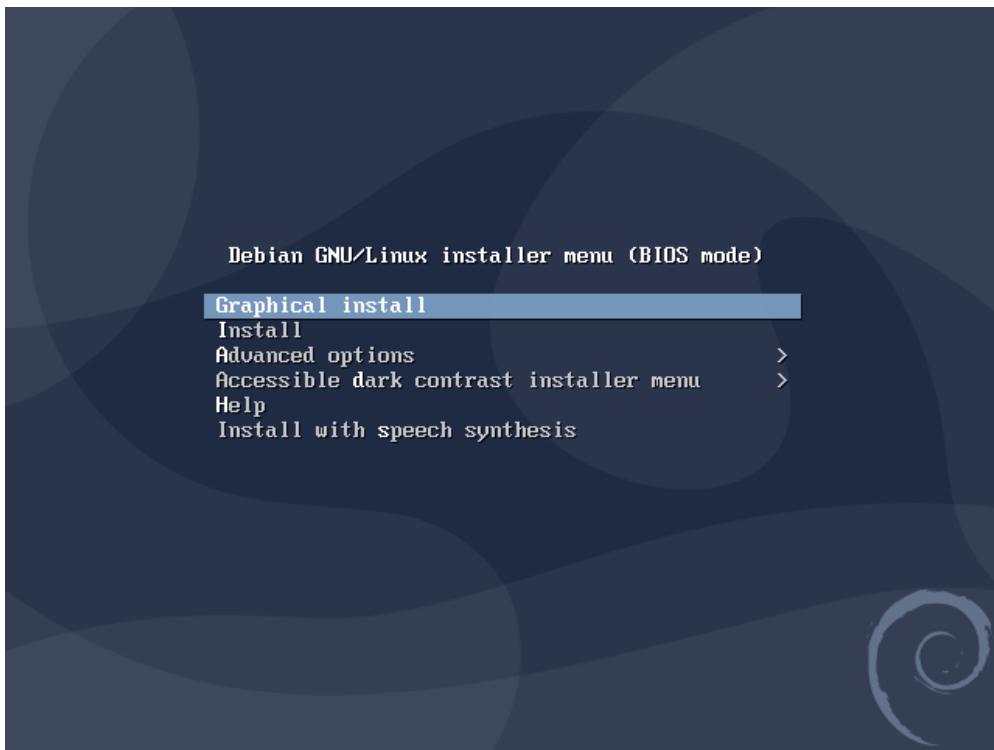
Configuración de portainer para la administración grafica de los contenedores.

Abstract

In this Project we going to work with Docker, Docker is an open source tool that simplifies managing Linux containers. A container is a sandbox environment that runs a collection of processes. Containers are light-weight Virtualmachines that share the same kernel as the host operating system. This kind of software who makes our work as administrators easy, cause we can fast deploy new services to the users and fix the errors quickly, Docker work with containers, Containers let us move our software and programs between machines, no matter the hardware we just need Docker installed in the destiny computer.

Proyecto Docker

Para empezar el proyecto, lo primero que tenemos que hacer es simular la máquina de la empresa que va a contener Docker y sus servicios, para ello vamos a descargar una ISO de Debian en concreto la versión Debian 10.3.0 de su página web (www.debian.org).



Una vez descargada usando VMware vamos a crear una máquina virtual con Debian que simulara ser el equipo de la empresa, vamos a configurar el nombre del equipo “Debian-Docker” y como usuario “Proyecto”.

Una vez instalado procedemos a añadir los repositorios de debian para así poder instalar/actualizar paquetes utilizando el siguiente comando.

```
sudo nano /etc/apt/sources.list
```

```
GNU nano 3.2                                         /etc/apt/sources.list

#
# deb cdrom:[Debian GNU/Linux 10.3.0 _Buster_ - Official amd64 DVD Binary-1 20200208-12:08]/ buster contrib main
#deb cdrom:[Debian GNU/Linux 10.3.0 _Buster_ - Official amd64 DVD Binary-1 20200208-12:08]/ buster contrib main
deb http://security.debian.org/debian-security buster/updates main contrib
deb-src http://security.debian.org/debian-security buster/updates main contrib
deb http://deb.debian.org/debian buster main
deb-src http://deb.debian.org/debian buster main

deb http://deb.debian.org/debian-security/ buster/updates main
deb-src http://deb.debian.org/debian-security/ buster/updates main

deb http://deb.debian.org/debian buster-updates main
deb-src http://deb.debian.org/debian buster-updates main
..
```

En la web de debian podremos obtener la lista con los repositorios correspondientes.

Una vez añadidos los repositorios actualizaremos la lista de paquetes disponibles y actualizaremos los paquetes que no estén actualizados utilizaremos los siguientes comandos.

```
sudo apt update & sudo apt upgrade
```

```
proyecto@debian-docker:~$ sudo apt update & sudo apt upgrade
[2] 2829
[sudo] password for proyecto:
Obj:1 http://deb.debian.org/debian buster InRelease
Des:2 http://security.debian.org/debian-security buster/updates InRelease [65,4 kB]
Des:3 http://deb.debian.org/debian-security buster/updates InRelease [65,4 kB]
Obj:4 http://deb.debian.org/debian buster-updates InRelease
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Calculando la actualización... Hecho
0% [Trabajando]0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
Des:5 http://security.debian.org/debian-security buster/updates/main Sources [107 kB]
Des:6 http://security.debian.org/debian-security buster/updates/main amd64 Packages [186 kB]
Des:7 http://security.debian.org/debian-security buster/updates/main Translation-en [99,1 kB]
Des:8 http://deb.debian.org/debian-security buster/updates/main Sources [107 kB]
Des:9 http://deb.debian.org/debian-security buster/updates/main amd64 Packages [186 kB]
Des:10 http://deb.debian.org/debian-security buster/updates/main Translation-en [99,1 kB]
Descargados 914 kB en 1s (787 kB/s)
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Todos los paquetes están actualizados.
```

Instalación De Docker

Con el siguiente comando vamos a permitir a apt usar paquetes a través de HTTPS.

```
sudo apt install apt-transport-https ca-certificates curl gnupg2 software-properties-common apt update & sudo apt upgrade
```

```
proyecto@debian-docker:~$ sudo apt install apt-transport-https ca-certificates curl gnupg2 software-properties-common
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
ca-certificates ya está en su versión más reciente (20190110).
fijado ca-certificates como instalado manualmente.
software-properties-common ya está en su versión más reciente (0.96.20.2-2).
fijado software-properties-common como instalado manualmente.
Se instalarán los siguientes paquetes NUEVOS:
  apt-transport-https curl gnupg2
0 actualizados, 3 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
Se necesita descargar 806 kB de archivos.
Se utilizarán 976 kB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] s
```

Procedemos a añadir la clave GPG del repositorio oficial de Docker para poder descargar el paquete.

```
sudo curl -fsSL https://download.docker.com/linux/debian/gpg | sudo apt-key add -
```

```
proyecto@debian-docker:~$ curl -fsSL https://download.docker.com/linux/debian/gpg | sudo apt-key add -
OK
```

Ahora añadimos el repositorio de Docker alas fuentes del APT.

```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/debian $(lsb_release -cs) stable"
```

```
proyecto@debian-docker:~$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/debian $(lsb_release -cs) stable"
```

Volvemos a actualizar la base de datos de paquetes para que se añadan los datos de Docker.

```
sudo apt update
```

```
proyecto@debian-docker:~$ sudo apt update
Obj:1 http://deb.debian.org/debian buster InRelease
Obj:2 http://security.debian.org/debian-security buster/updates InRelease
Obj:3 http://deb.debian.org/debian-security buster/updates InRelease
Obj:4 http://deb.debian.org/debian buster-updates InRelease
Des:5 https://download.docker.com/linux/debian buster InRelease [44,4 kB]
Des:6 https://download.docker.com/linux/debian buster/stable amd64 Packages [11,5 kB]
Descargados 55,9 kB en 1s (106 kB/s)
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Todos los paquetes están actualizados.
```

Verificamos que la instalación de Docker se va a realizar desde el repositorio de Docker en lugar del repositorio predeterminado de Debian.

```
sudo apt-cache policy docker-ce
```

```
proyecto@debian-docker:~$ apt-cache policy docker-ce
docker-ce:
  Instalados: (ninguno)
  Candidato: 5:19.03.8~3-0~debian-buster
  Tabla de versión:
    5:19.03.8~3-0~debian-buster 500
      500 https://download.docker.com/linux/debian buster/stable amd64 Packages
    5:19.03.7~3-0~debian-buster 500
      500 https://download.docker.com/linux/debian buster/stable amd64 Packages
    5:19.03.6~3-0~debian-buster 500
      500 https://download.docker.com/linux/debian buster/stable amd64 Packages
    5:19.03.5~3-0~debian-buster 500
      500 https://download.docker.com/linux/debian buster/stable amd64 Packages
    5:19.03.4~3-0~debian-buster 500
      500 https://download.docker.com/linux/debian buster/stable amd64 Packages
    5:19.03.3~3-0~debian-buster 500
      500 https://download.docker.com/linux/debian buster/stable amd64 Packages
    5:19.03.2~3-0~debian-buster 500
      500 https://download.docker.com/linux/debian buster/stable amd64 Packages
    5:19.03.1~3-0~debian-buster 500
      500 https://download.docker.com/linux/debian buster/stable amd64 Packages
    5:19.03.0~3-0~debian-buster 500
      500 https://download.docker.com/linux/debian buster/stable amd64 Packages
    5:18.09.9~3-0~debian-buster 500
      500 https://download.docker.com/linux/debian buster/stable amd64 Packages
    5:18.09.8~3-0~debian-buster 500
      500 https://download.docker.com/linux/debian buster/stable amd64 Packages
    5:18.09.7~3-0~debian-buster 500
      500 https://download.docker.com/linux/debian buster/stable amd64 Packages
    5:18.09.6~3-0~debian-buster 500
      500 https://download.docker.com/linux/debian buster/stable amd64 Packages
    5:18.09.5~3-0~debian-buster 500
      500 https://download.docker.com/linux/debian buster/stable amd64 Packages
    5:18.09.4~3-0~debian-buster 500
      500 https://download.docker.com/linux/debian buster/stable amd64 Packages
    5:18.09.3~3-0~debian-buster 500
      500 https://download.docker.com/linux/debian buster/stable amd64 Packages
    5:18.09.2~3-0~debian-buster 500
      500 https://download.docker.com/linux/debian buster/stable amd64 Packages
    5:18.09.1~3-0~debian-buster 500
      500 https://download.docker.com/linux/debian buster/stable amd64 Packages
```

Procedemos a instalar Docker.

```
sudo apt install docker-ce
```

```
proyecto@debian-docker:~$ sudo apt install docker-ce
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  aufs-dkms aufs-tools binutils binutils-common binutils-x86-64-linux-gnu build-essential
  libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan5 libbb
  libfile-fcntllock-perl libgcc-8-dev libitm1 liblsan0 libmpx2 libstdc++-8-dev libtsan0
  linux-headers-amd64 linux-kbuild-4.19 linux-libc-dev make manpages-dev patch pigz
Paquetes sugeridos:
  aufs-dev binutils-doc python3-apport menu debian-keyring g++-multilib g++-8-multilib g
  gcc-8-multilib gcc-8-locales libgcc1-dbg libgomp1-dbg libitm1-dbg libatomic1-dbg libas
  l dit-daemon-svsvinit dit-doc dit-el dit-email dit-oui ditk ditweb dit-cvs dit-mediawi
```

Verificamos que Docker se ha instalado correctamente.

```
sudo systemctl status Docker
```

```
proyecto@debian-docker:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
  Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
  Active: active (running) since Tue 2020-03-31 21:30:37 CEST; 53s ago
    Docs: https://docs.docker.com
Main PID: 4922 (dockerd)
  Tasks: 8
    Memory: 47.1M
   CGroup: /system.slice/docker.service
           └─4922 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

```
proyecto@debian-docker:~$ docker
Usage: docker [OPTIONS] COMMAND
A self-sufficient runtime for containers
Options:
      --config string      Location of client config
```

Instalación Docker-Compose

Vamos a instalar Docker-Compose, que es una herramienta que permite realizar scripts para facilitar la creación de containers y servicios. Para instalar Docker-Compose procedemos a usar el siguiente comando.

```
sudo curl -L "https://github.com/docker/compose/releases/download/1.25.5/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

```
proyecto@debian-docker:~/openvpn$ sudo curl -L "https://github.com/docker/compose/releases/download/1.25.5/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
% Total    % Received % Xferd  Average Speed   Time     Time      Current
          Dload  Upload Total Spent   Left Speed
100  638  100  638    0     0  3288      0 --:--:-- --:--:-- 3271
100 16.7M  100 16.7M   0     0  7984K     0 0:00:02 0:00:02 --:--:-- 18.0M
proyecto@debian-docker:~/openvpn$
```

Ahora le tenemos que dar permisos de ejecución para ello ejecutaremos el siguiente comando.

```
sudo chmod +x /usr/local/bin/docker-compose
```

```
proyecto@debian-docker:~/openvpn$ sudo chmod +x /usr/local/bin/docker-compose
```

Por último, para comprobar que Docker-compose se ha instalado sin problemas lo verificaremos con el siguiente comando.

```
Docker-compose --version
```

```
proyecto@debian-docker:~/openvpn$ docker-compose --version
docker-compose version 1.25.5, build 8alc60f6
proyecto@debian-docker:~/openvpn$
```

Instalación De Openldap en docker

Para empezar, descargaremos la imagen de Docker Openldap

```
Docker run -name my-openldap-container -detach osixia/openldap:1.3.0
```

Al no tenerlo en local, se conectará a buscar la imagen

```
proyecto@debian-docker:~$ sudo docker run --name my-openldap-container --detach osixia/openldap:1.3.0
Unable to find image 'osixia/openldap:1.3.0' locally
1.3.0: Pulling from osixia/openldap
1ab2bdfe9778: Pull complete
0abcaf321aa9: Pull complete
6d688c3d4e02: Pull complete
e3560bcbb6e7: Pull complete
ff7bed76ee88: Pull complete
c32f0fe69c97: Pull complete
ae5e7b954ab8: Pull complete
0386ed728a79: Pull complete
63ad335042c4: Pull complete
Digest: sha256:cb3f5fea3c3203acddc3e6b8a70642a0f994d89be3ec5f0e50621b2a9ea17a83
Status: Downloaded newer image for osixia/openldap:1.3.0
0f56fac9ae87a4790b19d53289c93d8f61641f9c39bb44b9357f74cea135a48a
proyecto@debian-docker:~$
```

A continuación, vamos a añadir nuestro dominio y la configuración principal con este comando

```
docker run -p 389:389 --name ldap-service --hostname ldap-service --env
LDAP_ORGANISATION="Jovellanos" --env LDAP_DOMAIN="proyecto.docker" \
--env LDAP_ADMIN_PASSWORD="1234" --env LDAP_BASE_DN="dc=proyecto,dc=dock
er" --volume /data/slapd/database:/var/lib/ldap \
--volume /data/slapd/config:/etc/ldap/slapd.d --detach osixia/openldap:1
.3.0
```

Instalación de Interfaz Web Openldap en docker

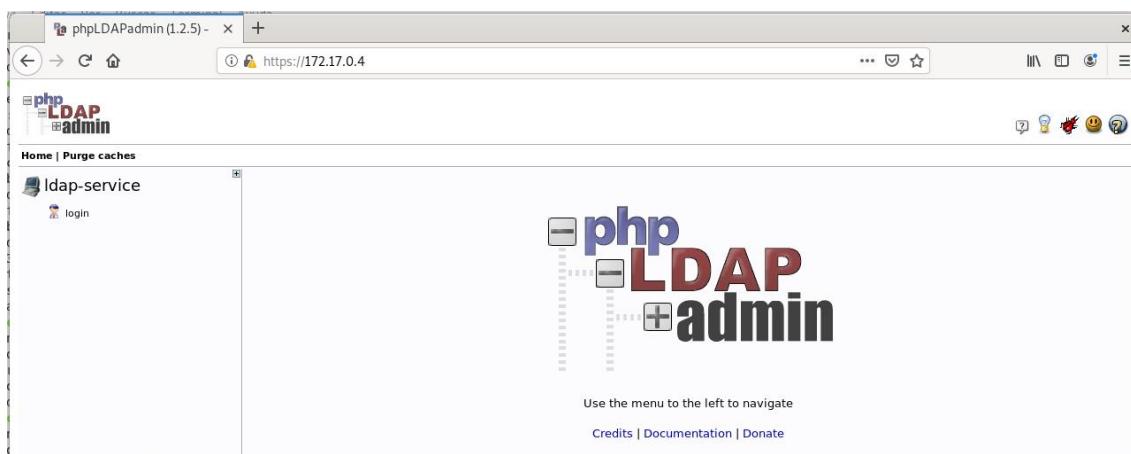
También para mayor comodidad vamos a añadir una interfaz web para nuestro openldap

```
docker run --name phpldapadmin-service --hostname phpldapadmin-service -  
-link ldap-service:ldap-host --env PHPLDAPADMIN_LDAP_HOSTS=ldap-service  
--detach osixia/phpldapadmin:0.9.0
```

Después con este comando averiguaremos cual es la ip asociada al proceso de ldapadmin

```
docker inspect -f "{{ .NetworkSettings.IPAddress }}" phpldapadmin-service
```

A continuación, si nos conectamos a esa ip accederemos a la interfaz web



Le daremos a la ventana de login poniendo los datos de nuestro dominio anteriormente creado

Home | Purge caches

ldap-service

login

Authenticate to server ldap-service

Login DN:
 cn=admin,dc=proyecto,dc=docker

Password:
 ••••

Anonymous

1 2 3

Y como comprobamos podemos acceder perfectamente

Home | Purge caches

ldap-service

schema search refresh info import export logout

Logged in as: cn=admin

dc=proyecto,dc=docker (1)

- cn=admin
- Create new entry here

Authenticate to server

Successfully logged into server.

Use the menu to the left to navigate

[Credits](#) | [Documentation](#) | [Donate](#)

Creación y configuración de Usuario, Grupos y Unidades Organizativas en LDAP en Docker

A continuación, creamos un par de entradas para comprobar su funcionalidad

Create Object

Server: ldap-service Container: dc=proyecto,dc=docker

Select a template for the creation process

Templates:

- Courier Mail: Account
- Courier Mail: Alias
- Generic: Address Book Entry
- Generic: DNS Entry
- Generic: LDAP Alias
- Generic: Organisational Role
- Generic: Organisational Unit
- Generic: Posix Group
- Generic: Simple Security Object
- Generic: User Account
- Kolab: User Entry
- Samba: Account
- Samba: Domain
- Samba: Group Mapping
- Samba: Machine
- Sendmail: Alias
- Sendmail: Cluster
- Sendmail: Domain
- Sendmail: Relays
- Sendmail: Virtual Domain
- Sendmail: Virtual Users
- Thunderbird: Address Book Entry
- Default

Para empezar, cogeremos crear una UO

Create Object

Server: ldap-service Container: dc=proyecto,dc=docker
Template: Generic: Organisational Unit (ou)

New Organisational Unit (Step 1 of 1)

Organisational Unit	alias, required, rdn, hint
Jovellanos	*

Comprobamos los datos metidos y le damos a commit

Create LDAP Entry

Server: ldap-service Container: dc=proyecto,dc=docker

Do you want to create this entry?

Attribute	New Value	Skip
ou=jovellanos,dc=proyecto,dc=docker		
Organisational Unit	Jovellanos	<input type="checkbox"/>
objectClass	organizationalUnit	<input type="checkbox"/>

Como vemos, ya aparece a la izquierda

The screenshot shows the 'Create Entry' page in the OpenLDAP web interface. The entry 'ou=Jovellanos,dc=proyecto,dc=docker' has been successfully created. The 'objectClass' field contains 'organizationalUnit' and 'top'. The 'ou' field is set to 'Jovellanos'. A 'Create Object' button is visible at the bottom.

Creamos ahora un grupo llamado usuarios dentro de la UO Jovellanos

The screenshot shows the 'Create Object' page for a 'New Posix Group (Step 1 of 1)'. The group name is 'Usuarios'. The 'GID Number' is set to 500. There are no users listed under 'Users'. A 'Create Object' button is at the bottom.

Comprobamos la entrada

The screenshot shows the 'Create LDAP Entry' confirmation dialog. It lists the attributes and their values: 'cn=Usuarios,ou=Jovellanos,dc=proyecto,dc=docker', 'Group' (value: Usuarios), 'GID Number' (value: 500), and 'objectClass' (value: posixGroup). 'Commit' and 'Cancel' buttons are at the bottom.

Y ya lo tenemos creado

Home | Purge caches

ldap-service

schema search refresh info import export logout

Logged in as: cn=admin

dc=proyecto,dc=docker (2)

- cn=admin
- ou=jovellanos (1+)
- cn=Usuarios

Create new entry here

cn=Usuarios

Server: ldap-service Distinguished Name: cn=Usuarios,ou=Jovellanos,dc=proyecto,dc=docker Template: Default

Refresh Switch Template Copy or move this entry Rename Create a child entry

Show internal attributes Export Delete this entry Compare with another entry Add new attribute

Hint: To delete an attribute, empty the text field and click save.

Hint: To view the schema for an attribute, click the attribute name.

cn required, rdn

Usuarios
(add value)
(rename)

gidNumber required

500

objectClass required

posixGroup (structural)
top
(add value)

A continuación, creamos un usuario dentro del grupo usuario

Create Object

Server: ldap-service Container: cn=Usuarios,ou=Jovellanos,dc=proyecto,dc=docker Template: Generic: User Account (posixAccount)

New User Account (Step 1 of 1)

First name alias

Alumno

Last name alias, required

Jovellanos *

Common Name alias, required, rdn

Alumno Jovellanos *

User ID alias, required

ajovellanos *

Password alias, hint

•••• md5 (confirm)

Check password...

UID Number alias, required, hint, ro

1000

GID Number	alias, required, hint
<input type="text" value="Usuarios"/>	*
Home directory	alias, required
<input type="text" value="/home/users/ajovellanos"/>	*
Login shell	alias
<input type="text" value="Bash"/>	
Create Object	
1.2.5	

Como vemos podemos configurar múltiples parámetros como password, directorio home, la Shell que usara etc...

Create LDAP Entry

Server: ldap-service Container: cn=Usuarios,ou=Jovellanos,dc=proyecto,dc=docker

Do you want to create this entry?

Attribute	New Value	Skip
cn=Alumno Jovellanos,cn=Usuarios,ou=Jovellanos,dc=proyecto,dc=docker		
First name	Alumno	<input type="checkbox"/>
Last name	Jovellanos	<input type="checkbox"/>
Common Name	Alumno Jovellanos	<input type="checkbox"/>
User ID	ajovellanos	<input type="checkbox"/>
Password	*****	<input type="checkbox"/>
UID Number	1000	<input type="checkbox"/>
GID Number	500	<input type="checkbox"/>
Home directory	/home/users/ajovellanos	<input type="checkbox"/>
Login shell	/bin/bash	<input type="checkbox"/>
objectClass	inetOrgPerson posixAccount	<input type="checkbox"/>

[Commit](#) [Cancel](#)

Y ya lo tendríamos creado

ldap-service

Logged in as: cn=admin

- dc=proyecto,dc=docker (2)
 - cn=admin
 - ou=Jovellanos (1)
 - cn=Usuarios (1+)
 - cn=Alumno Jovellanos
- ★ Create new entry here

cn=Alumno Jovellanos

Server: ldap-service Distinguished Name: cn=Alumno Jovellanos,cn=Usuarios,ou=Jovellanos,dc=proyecto,dc=docker Template: Default

Refresh Show internal attributes
 Switch Template Export
 Copy or move this entry Delete this entry
 Rename Compare with another entry
 ★ Create a child entry Add new attribute
 Hint: To delete an attribute, empty the text field and click save.
 Hint: To view the schema for an attribute, click the attribute name.

cn	required, rdn
Alumno Jovellanos	*
(add value)	
(rename)	
gidNumber	required
500	
Usuarios ()	
givenName	
Alumno	
(add value)	
homeDirectory	required
/home/users/ajovellanos	
loginShell	

Para justificar el usar Docker, hemos replicado el mismo proceso en el propio sistema Debian, sin usar Docker, para ver como Docker nos ayuda hacer despliegues mucho más rápido

Instalación OpenLdap Debian puro

Para empezar, realizaremos un

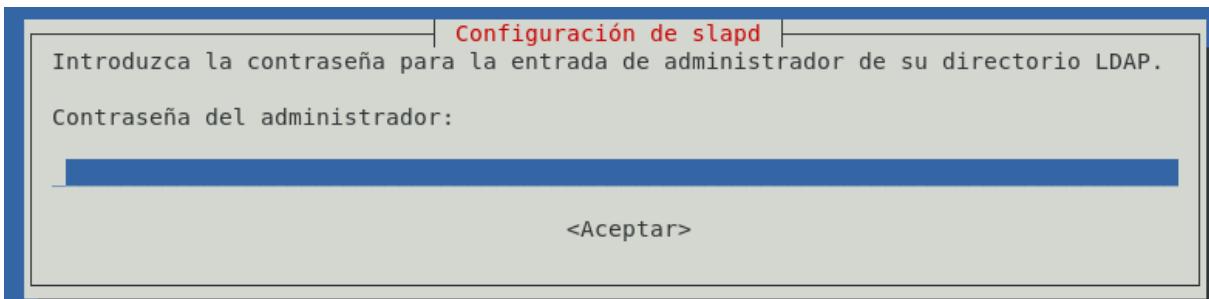
```
apt-get update
```

```
proyecto@debian-docker:~$ sudo apt-get update
Obj:1 http://security.debian.org/debian-security buster/updates InRelease
Obj:2 https://download.docker.com/linux/debian buster InRelease
Obj:3 http://deb.debian.org/debian buster InRelease
Obj:4 http://deb.debian.org/debian-security buster/updates InRelease
Obj:5 http://deb.debian.org/debian buster-updates InRelease
Leyendo lista de paquetes... Hecho
proyecto@debian-docker:~$ █
```

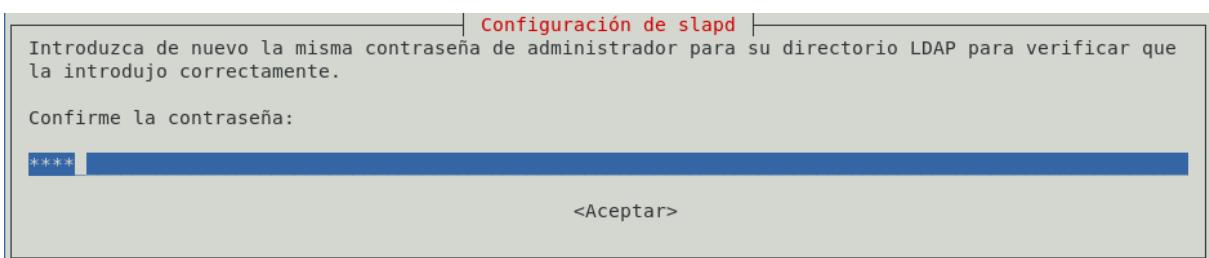
Una vez actualizado los repositorios nos descargaremos el paquete deseado

```
proyecto@debian-docker:~$ sudo apt-get install slapd ldap-utils
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  libodbc1
Paquetes sugeridos:
  libsasl2-modules-gssapi-mit | libsasl2-modules-gssapi-heimdal libmyodbc odbc-postgresql tdsodbc
  unixodbc-bin
Se instalarán los siguientes paquetes NUEVOS:
  ldap-utils libodbc1 slapd
0 actualizados, 3 nuevos se instalarán, 0 para eliminar y 6 no actualizados.
Se necesita descargar 1.857 kB de archivos.
Se utilizarán 17,4 MB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] s█
```

Una vez realizada la instalación del paquete nos pedirá la contraseña de administrador del directorio LDAP



Nos pide de nuevo una confirmación de contraseña

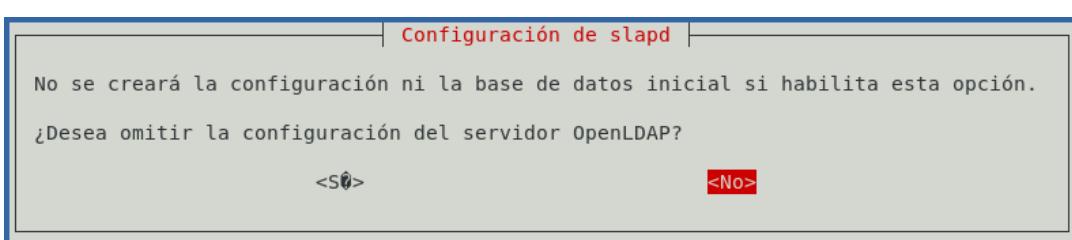


Tras esto, comprobamos que ya ha realizado por si solo una configuración inicial

```
Configurando slapd (2.4.47+dfsg-3+deb10u1) ...
  Creating new user openldap... done.
  Creating initial configuration... done.
  Creating LDAP directory... done.
Procesando disparadores para systemd (241-7~deb10u3) ...
Procesando disparadores para man-db (2.8.5-2) ...
Procesando disparadores para libc-bin (2.28-10) ...
proyecto@debian-docker:~$
```

Como nosotros queremos una configuración propia ejecutaremos

```
dpkg-reconfigure -plow slapd
```



Así que aquí le daremos a la opción No

A continuación, nos pide que introduzcamos un nombre de dominio dns en nuestro caso será proyecto.local

Configuración de slapd

El nombre de dominio DNS se utiliza para construir el DN base del directorio LDAP. Por ejemplo, si introduce «foo.example.org» el directorio se creará con un DN base de «dc=foo, dc=example, dc=org».

Introduzca el nombre de dominio DNS:

proyecto.local

<Aceptar>

Nos pide un nombre de organización para utilizar en LDAP en nuestro caso será proyecto

Configuración de slapd

Introduzca el nombre de la organización a utilizar en el DN base del directorio LDAP.

Nombre de la organización:

proyecto

<Aceptar>

A continuación, nos pide la contraseña del administrador del directorio LDAP

Configuración de slapd

Introduzca la contraseña para la entrada de administrador de su directorio LDAP.

Contraseña del administrador:

<Aceptar>

Como antes pide que la confirmemos introduciéndola de nuevo

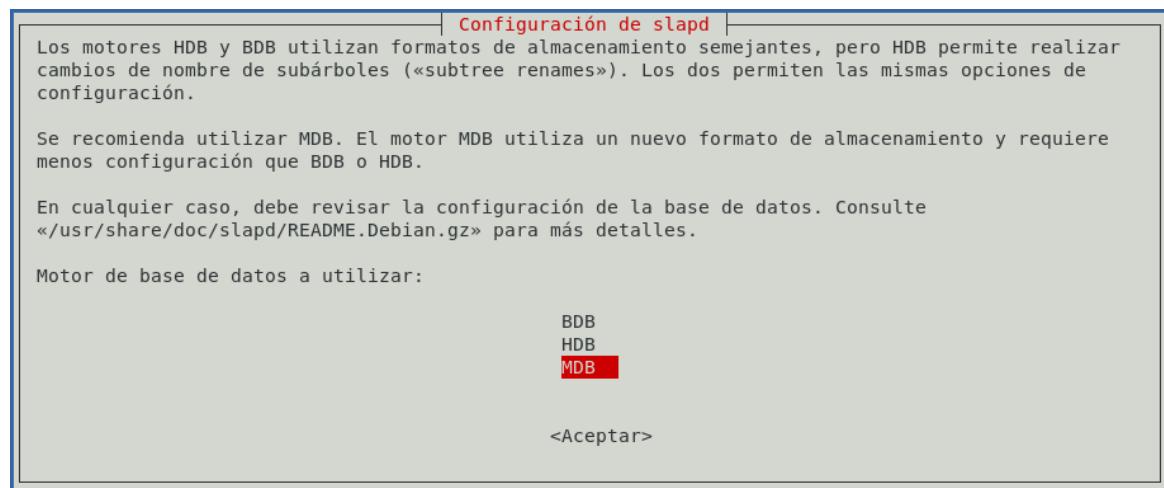
Configuración de slapd

Introduzca de nuevo la misma contraseña de administrador para su directorio LDAP para verificar que la introdujo correctamente.

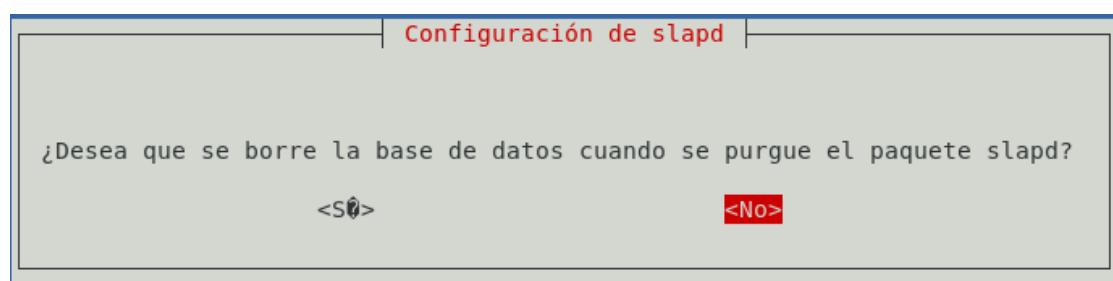
Confirme la contraseña:

<Aceptar>

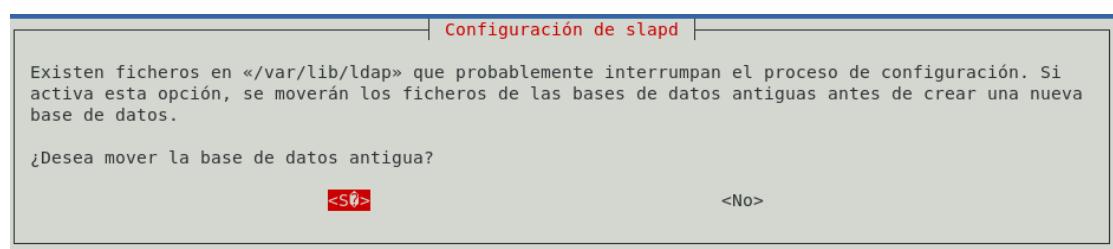
En el siguiente paso nos pide configurar el motor de base de datos que queremos usar, por defecto dejaremos el marcado MDB



A continuación, nos pregunta que si deseamos que se elimine el directorio slapd de la base de datos cuando slapd sea eliminado, en su defecto dejamos No



En la siguiente opción nos da a elegir si deseamos que se mueva el antiguo directorio, por defecto dejamos la opción Si



Una vez realizado el proceso nos ha vuelto a realizar la configuración, pero esta vez con nuestros parámetros

```
proyecto@debian-docker:~$ sudo dpkg-reconfigure -plow slapd
Backing up /etc/ldap/slapd.d in /var/backups/slapd-2.4.47+dfsg-3+deb10u1... done.
Moving old database directory to /var/backups:
- directory unknown... done.
Creating initial configuration... done.
Creating LDAP directory... done.
proyecto@debian-docker:~$
```

Para comprobar que todos los parámetros son correctos usaremos el comando slapcat que nos muestra el siguiente resultado

```
 proyecto@debian-docker:~$ sudo slapcat
dn: dc=projeto,dc=local
objectClass: top
objectClass: dcObject
objectClass: organization
o: projeto
dc: projeto
structuralObjectClass: organization
entryUUID: 084cc528-126c-103a-8865-b3f0b7026e8b
creatorsName: cn=admin,dc=projeto,dc=local
createTimestamp: 20200414071932Z
entryCSN: 20200414071932.923457Z#000000#000#000000
modifiersName: cn=admin,dc=projeto,dc=local
modifyTimestamp: 20200414071932Z

dn: cn=admin,dc=projeto,dc=local
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: admin
description: LDAP administrator
userPassword:: e1NTSEF9aVVqSGtMMETnR2ozNEFYmw2NnRha1NtR0gzdTh0TDI=
structuralObjectClass: organizationalRole
entryUUID: 084cdef0-126c-103a-8866-b3f0b7026e8b
creatorsName: cn=admin,dc=projeto,dc=local
createTimestamp: 20200414071932Z
entryCSN: 20200414071932.924149Z#000000#000#000000
modifiersName: cn=admin,dc=projeto,dc=local
modifyTimestamp: 20200414071932Z

proyecto@debian-docker:~$ █
```

Con esto hemos finalizado la configuración inicial de LDAP, a continuación, pasaremos a crear los grupos, unidades organizativas y usuarios que deseemos

Creación y configuración de Usuario, Grupos y Unidades Organizativas en LDAP en Debian puro

Para tener una organización fácil y optima vamos a manejar ficheros. Idif que a posterior se cargan en LDAPP a través de un comando, así a su vez podemos tener un fichero. Idif para poder separar usuarios, grupos y UO según nos venga en gana, A continuación, vamos a configurar los ficheros.

Primero crearemos la unidad organizativa Jovellanos

```
Archivo Editar Ver Buscar Terminal Ayuda
GNU nano 3.2                                         unidades.ldif

dn: ou=Jovellanos,dc=proyecto,dc=local
ou: Jovellanos
objectClass: top
objectClass: organizationalUnit
```

Ahora crearemos el grupo administradores

```
Archivo Editar Ver Buscar Terminal Ayuda
GNU nano 3.2                                         grupos.ldif

dn: cn=Administradores,ou=Jovellanos, dc=proyecto,dc=local
objectClass: top
objectClass: posixGroup
gidNumber: 2020
cn: Administradores
```

A continuación, hemos creado dos usuarios dentro de administradores, Tanto el cn como el sn que se requiere para este tipo de cuentas lo hemos realizado en base 64 con el comando

Echo dani | base64 y el resultado es el cn, repetimos el paso con el sn y en este caso ponemos el apellido, así hemos obtenido los valores requeridos

```

Archivo Editar Ver Buscar Terminal Ayuda
GNU nano 3.2                               usuarios.ldif

dn: uid=admin1, ou=Jovellanos,dc=proyecto,dc=local
objectClass: top
objectClass: posixAccount
objectClass: inetOrgperson
objectClass: person
cn:: ZGFuaQo=
uid: dani
uidNumber: 2001
gidNumber: 2020
homeDirectory: /home/nfs/dani
loginShell: /bin/bash
userPassword: 1234
sn:: dG9tYXMK
mail: dani@administradores.local
givenName: Dani

dn: uid=admin2, ou=Jovellanos,dc=proyecto,dc=local
objectClass: top
objectClass: posixAccount
objectClass: inetOrgperson
objectClass: person
cn :: THVpcwo=
uid: Luis
uidNumber: 2002
gidNumber: 2020
homeDirectory: /home/nfs/Luis
loginShell: /bin/bash
userPassword: 1234
sn:: dmFsZGVyYXMK
mail: Luis@administradores.local
givenName: Luis

```

Ahora procederemos aadir a ldap los datos de cada fichero. ldif creado

ldapadd -x -D dc=Jovellanos,dc=local -W -f unidades.ldif

Unidades organizativas

```

proyecto@debian-docker:~$ sudo ldapadd -x -D cn=admin,dc=Proyecto,dc=local -W -f unidades.ldif
Enter LDAP Password:
adding new entry "ou=Jovellanos,dc=proyecto,dc=local"

```

Grupos

```

proyecto@debian-docker:~$ sudo ldapadd -x -D cn=admin,dc=Proyecto,dc=local -W -f grupos.ldif
Enter LDAP Password:
adding new entry "cn=Administradores,ou=Jovellanos,dc=proyecto,dc=local"

```

Usuarios

```
proyecto@debian-docker:~$ sudo ldapadd -x -D cn=admin,dc=Proyecto,dc=local -W -f usuarios.ldif
Enter LDAP Password:
adding new entry "uid=admin1, ou=Jovellanos,dc=proyecto,dc=local"
adding new entry "uid=admin2, ou=Jovellanos,dc=proyecto,dc=local"
```

Tras estos procesos podemos usar el comando slapcat para comprobar la información

```
proyecto@debian-docker:~$ sudo slapcat
dn: dc=proyecto,dc=local
objectClass: top
objectClass: dcObject
objectClass: organization
o: proyecto
dc: proyecto
structuralObjectClass: organization
entryUUID: 084cc528-126c-103a-8865-b3f0b7026e8b
creatorsName: cn=admin,dc=proyecto,dc=local
createTimestamp: 20200414071932Z
entryCSN: 20200414071932.923457Z#000000#000#000000
modifiersName: cn=admin,dc=proyecto,dc=local
modifyTimestamp: 20200414071932Z

dn: cn=admin,dc=proyecto,dc=local
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: admin
description: LDAP administrator
userPassword:: e1NTSEF9aVVqSGtMMEtR2ozNEFHYmw2NnRha1NtR0gzdTh0TDI=
structuralObjectClass: organizationalRole
entryUUID: 084cdef0-126c-103a-8866-b3f0b7026e8b
creatorsName: cn=admin,dc=proyecto,dc=local
createTimestamp: 20200414071932Z
entryCSN: 20200414071932.924149Z#000000#000#000000
modifiersName: cn=admin,dc=proyecto,dc=local
modifyTimestamp: 20200414071932Z

dn: ou=Jovellanos,dc=proyecto,dc=local
ou: Jovellanos
objectClass: top
objectClass: organizationalUnit
structuralObjectClass: organizationalUnit
entryUUID: 99291734-12cd-103a-825d-abccf60050e5
creatorsName: cn=admin,dc=Proyecto,dc=local
createTimestamp: 20200414185757Z
entryCSN: 20200414185757.141992Z#000000#000#000000
modifiersName: cn=admin,dc=Proyecto,dc=local
modifyTimestamp: 20200414185757Z
```

```
dn: ou=Jovellanos,dc=proyecto,dc=local
ou: Jovellanos
objectClass: top
objectClass: organizationalUnit
structuralObjectClass: organizationalUnit
entryUUID: 99291734-12cd-103a-825d-abccf60050e5
creatorsName: cn=admin,dc=Proyecto,dc=local
createTimestamp: 20200414185757Z
entryCSN: 20200414185757.141992Z#000000#000#000000
modifiersName: cn=admin,dc=Proyecto,dc=local
modifyTimestamp: 20200414185757Z

dn: cn=Administradores,ou=Jovellanos,dc=proyecto,dc=local
objectClass: top
objectClass: posixGroup
gidNumber: 2020
cn: Administradores
structuralObjectClass: posixGroup
entryUUID: e2df682e-12cd-103a-825e-abccf60050e5
creatorsName: cn=admin,dc=Proyecto,dc=local
createTimestamp: 20200414190000Z
entryCSN: 20200414190000.810567Z#000000#000#000000
modifiersName: cn=admin,dc=Proyecto,dc=local
modifyTimestamp: 20200414190000Z

dn: uid=admin1,ou=Jovellanos,dc=proyecto,dc=local
objectClass: top
objectClass: posixAccount
objectClass: inetOrgPerson
objectClass: person
cn:: ZGFuaQo=
uid: dani
uid: admin1
uidNumber: 2001
gidNumber: 2020
homeDirectory: /home/nfs/dani
loginShell: /bin/bash
userPassword:: MTIzNA==
sn:: dG9tYXMK
mail: dani@administradores.local
givenName: Dani
structuralObjectClass: inetOrgPerson
entryUUID: 2da41f26-12d3-103a-825f-abccf60050e5
creatorsName: cn=admin,dc=Proyecto,dc=local
createTimestamp: 20200414193753Z
entryCSN: 20200414193753.734804Z#000000#000#000000
modifiersName: cn=admin,dc=Proyecto,dc=local
modifyTimestamp: 20200414193753Z
```

```
dn: uid=admin2,ou=Jovellanos,dc=proyecto,dc=local
objectClass: top
objectClass: posixAccount
objectClass: inetOrgPerson
objectClass: person
cn:: THVpcwo=
uid: Luis
uid: admin2
uidNumber: 2002
gidNumber: 2020
homeDirectory: /home/nfs/Luis
loginShell: /bin/bash
userPassword:: MTIzNA==
sn:: dmFsZGVyYXMK
mail: Luis@administradores.local
givenName: Luis
structuralObjectClass: inetOrgPerson
entryUUID: 2da44de8-12d3-103a-8260-abccf60050e5
creatorsName: cn=admin,dc=Proyecto,dc=local
createTimestamp: 20200414193753Z
entryCSN: 20200414193753.736002Z#000000#000#000000
modifiersName: cn=admin,dc=Proyecto,dc=local
modifyTimestamp: 20200414193753Z
```

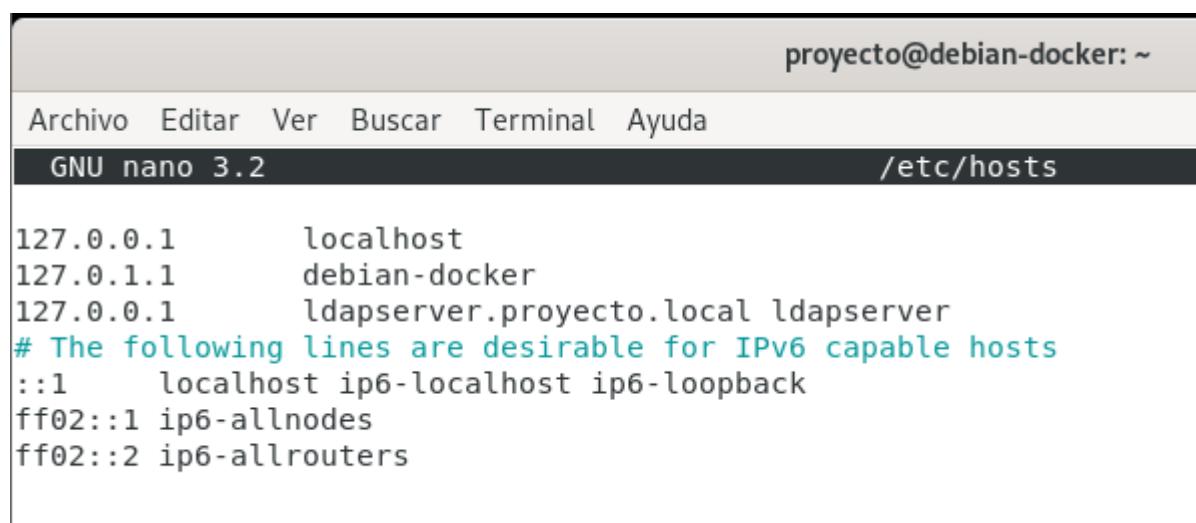
```
proyecto@debian-docker:~$ █
```

Con esto hemos finalizado la creación de UO, grupos y usuarios deseados para esta demo

Instalación De Interfaz Web Openldap en Debian puro

Para un manejo más fácil de Openldap instalaremos una interfaz web

Primero modificaremos el fichero /etc/hosts y añadimos la entrada del server



The screenshot shows a terminal window with the following content:

```
proyecto@debian-docker: ~
Archivo Editar Ver Buscar Terminal Ayuda
GNU nano 3.2                               /etc/hosts

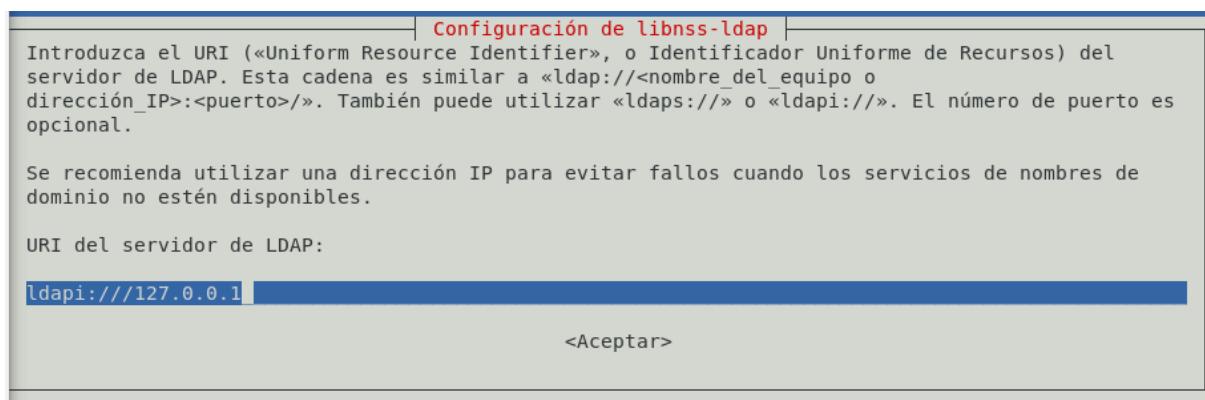
127.0.0.1      localhost
127.0.1.1      debian-docker
127.0.0.1      ldapserver.proyecto.local ldapserver
# The following lines are desirable for IPv6 capable hosts
::1      localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

A continuación, procederemos a instalar el paquete libnss-ldap

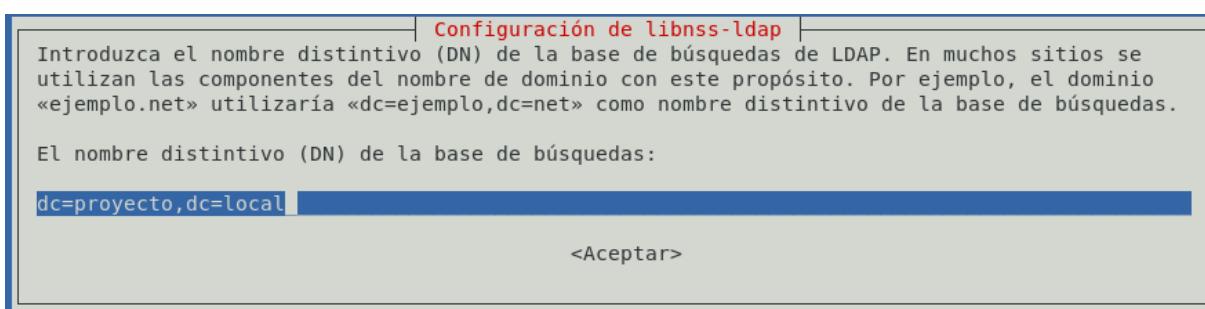
```
proyecto@debian-docker:~$ sudo apt install libnss-ldap
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  libpam-ldap nscd
Se instalarán los siguientes paquetes NUEVOS:
  libnss-ldap libpam-ldap nscd
0 actualizados, 3 nuevos se instalarán, 0 para eliminar y 12 no actualizados.
Se necesita descargar 483 kB de archivos.
Se utilizarán 940 kB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n]
```

Proseguiremos con la configuración correspondiente

Aquí nos pide la ip que usaremos



Aquí nos pide rellenar el nombre distintivo



Aquí cogeremos el valor mas alto como indica

Configuración de libnss-ldap
Introduzca la versión del protocolo de LDAP que debería usar ldapns. Se recomienda utilizar el número de versión más alto que esté disponible.

Versión de LDAP a utilizar:

3
2

<Aceptar>

Configurando la cuenta admin para LDAP

Configuración de libnss-ldap
Escoja que cuenta se utilizará para las consultas nss con privilegios de root.

Nota: Para que funcione esta opción la cuenta necesita permisos para poder acceder a los atributos LDAP que están asociados con las entradas «shadow» de los usuarios así como a las contraseñas de los usuarios y grupos.

Cuenta LDAP para root:

`cn=admin,dc=proyecto,dc=local`

<Aceptar>

Contraseña admin

Configuración de libnss-ldap
Introduzca la contraseña se utilizará cuando libnss-ldap intente autenticarse al directorio LDAP con la cuenta LDAP de root.

La contraseña se guardará en un fichero independiente («/etc/libnss-ldap.secret») al que sólo podrá acceder root.

Si introduce una contraseña vacía se reutilizará la antigua contraseña.

Contraseña para la cuenta LDAP de root:

<Aceptar>

Aviso del fichero nsswitch.conf

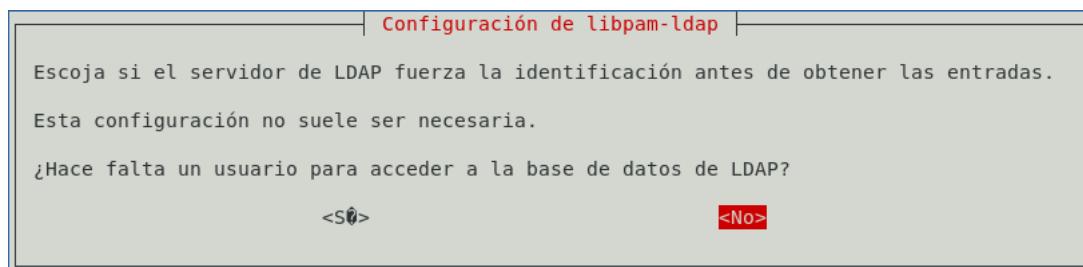
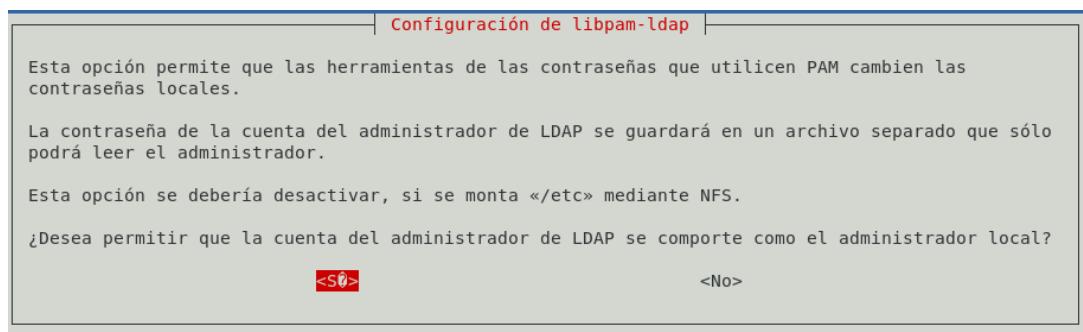
Configuración de libnss-ldap
nsswitch.conf no se gestiona automáticamente

Debe modificar su fichero «/etc/nsswitch.conf» para que utilice una fuente de datos LDAP si quiere que funcione el paquete libnss-ldap. Puede utilizar como ejemplo de la configuración de nsswitch el fichero de ejemplo en «/usr/share/doc/libnss-ldap/examples/nsswitch.ldap» o puede copiarlo sobre su configuración actual.

Tenga en cuenta que antes de eliminar este paquete puede ser conveniente eliminar las entradas «ldap» del fichero nsswitch.conf para que los servicios básicos sigan funcionando.

<Aceptar>

Aquí marcamos la opción por defecto



Vamos a configurar los clientes

```
proyecto@debian-docker:~$ sudo nano /etc/ldap/ldap.conf
```

```
proyecto@debian-docker:~$ nano /etc/ldap/ldap.conf

# 
# LDAP Defaults
#
# See ldap.conf(5) for details
# This file should be world readable but not world writable.

#BASE    dc=projeto,dc=local
#URI     ldap://127.0.0.1
```

Ejecutaremos el siguiente comando para comprobar que funciona el servidor

```
proyecto@debian-docker:~$ ldapsearch -x
```

Nos devuelve un 0 así que es correcto

```
# search result
search: 2
result: 0 Success

# numResponses: 7
# numEntries: 6
proyecto@debian-docker:~$
```

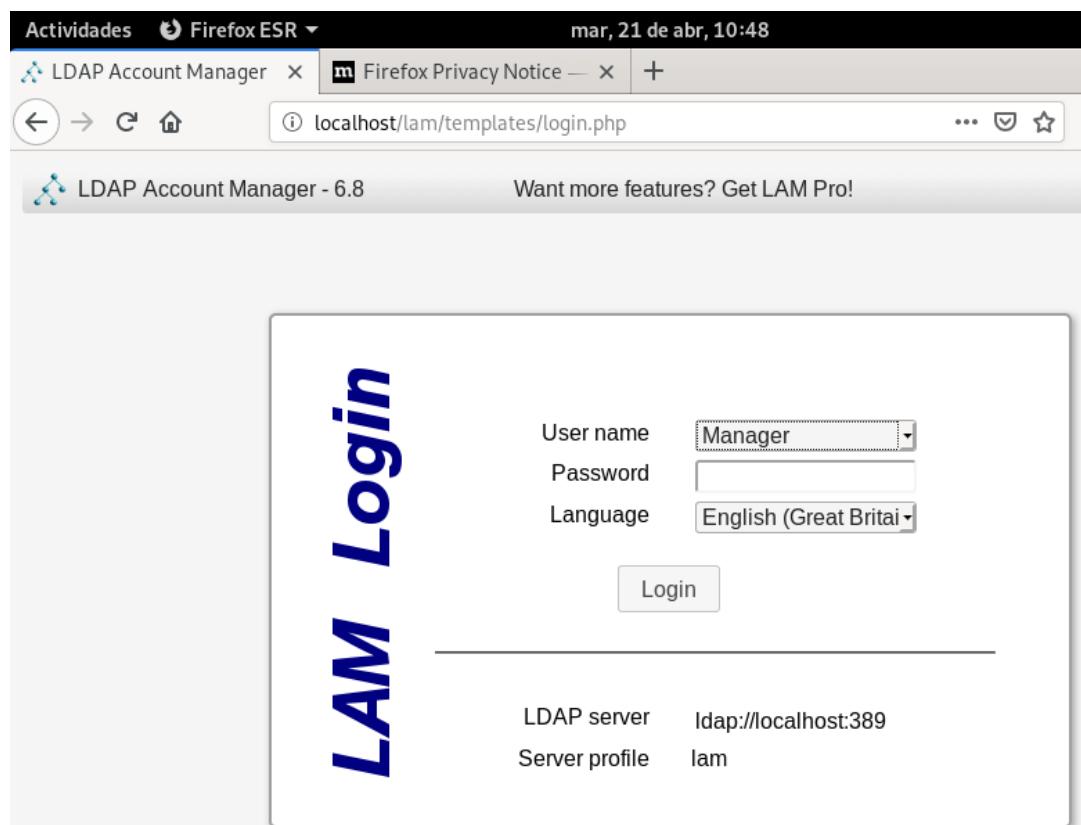
Terminamos bajando la interfaz web deseada

```
proyecto@debian-docker:~$ wget http://prdownloads.sourceforge.net/lam/ldap-account-manager_6.8-1_all.deb
--2020-04-21 10:44:59--  http://prdownloads.sourceforge.net/lam/ldap-account-manager_6.8-1_all.deb
```

```
proyecto@debian-docker:~$ sudo dpkg -i ldap-account-manager_6.8-1_all.deb
(Leyendo la base de datos ... 166850 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar ldap-account-manager_6.8-1_all.deb ...
Desempaquetando ldap-account-manager (6.8-1) sobre (6.8-1) ...
Configurando ldap-account-manager (6.8-1) ...
proyecto@debian-docker:~$
```

Comprobamos que tras esto podemos acceder

A la interfaz web



Vamos a proceder a configurarlo

General settings

Security settings

Session timeout: 30
Allowed hosts:

Encrypt session:

SSL certificates:
Browse... No file selected.
use system certificates

Idaps://

Password policy

Minimum password length: 0
Minimum lowercase characters: 0
Minimum uppercase characters: 0
Minimum numeric characters: 0
Minimum symbolic characters: 0
Minimum character classes: 0

Number of rules that must match: all
 Password must not contain user name
 Password must not contain part of user/first/last name

 General settings  Account types  Modules  Module settings

Server settings

Server address *: ldap://localhost:389
Activate TLS: no
Tree suffix: dc=proyecto,dc=local
LDAP search limit: -

Security settings

Login method: Fixed list
List of valid users *: cn=admin,dc=proyecto,dc=local

Como vemos después de configurar los parámetros, nos reconoce los ficheros ldif creados

The screenshot shows the LDAP Account Manager interface version 6.8. The top navigation bar includes links for Tree view, Tools, Help, and Logout. Below the header, there are tabs for Users and Groups, with 'Users' selected. A toolbar contains buttons for New user, Delete selected users, and File upload. The main area displays a table of users with columns: Actions, User name, First name, Last name, UID number, and GID number. The table shows two entries: admin1; dani and Luis; admin2. The 'Actions' column for each entry includes icons for edit, delete, and export.

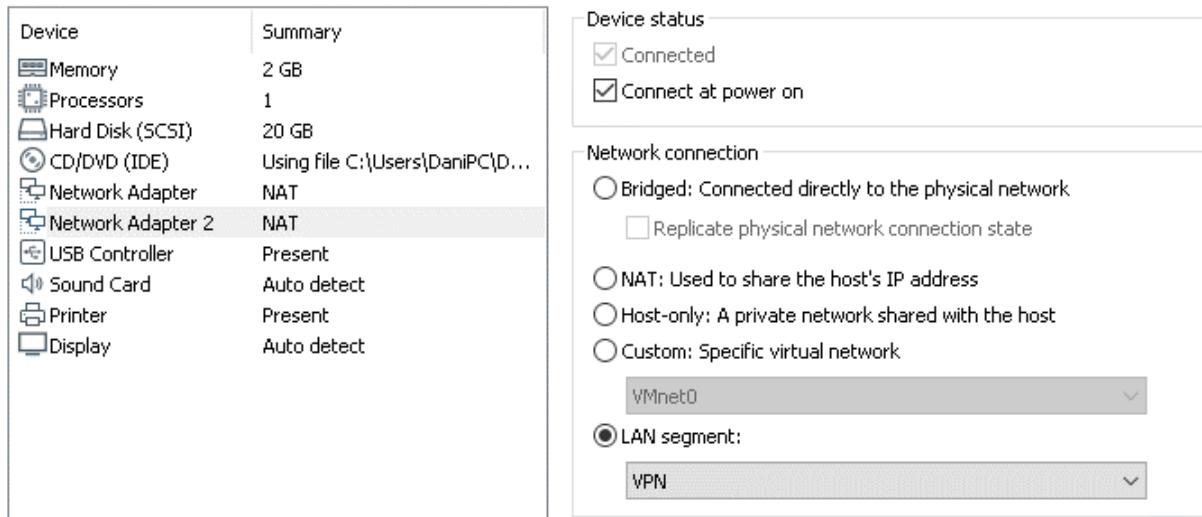
Actions	User name	First name	Last name	UID number	GID number
<input type="checkbox"/> Sort sequence <input type="checkbox"/> Filter	admin1; dani	Dani	tomas	2001	2020
<input type="checkbox"/>	Luis; admin2	Luis	valderas	2002	2020

Y hasta aquí como se haría si no usásemos Docker, como vemos es un proceso más tedioso, pero con el mismo resultado (hemos usado otras interfaces web para comprobar e investigar la funcionalidad de cada una)

Instalación OPENVPN DOCKER

Configuración Máquina Virtual

El primer paso que vamos a realizar va a ser instalar una segunda tarjeta de red que vamos a poner en un segmento llamado VPN para poder realizar posteriormente las pruebas de conexión mediante la VPN.



Dentro de la maquina vamos a asignarle la siguiente dirección de red.

```
auto ens37
iface ens37 inet static
    address 10.0.0.10
    netmask 255.255.255.0
```

Docker-Compose Instalación

Para poder instalar OPENVPN vamos a acudir a buscar su imagen, para ello acudimos a <https://hub.docker.com/>, donde dentro del mismo podremos encontrar todas las imágenes publicadas para Docker de distintos servicios, aplicaciones.

Una vez dentro podremos encontrar haciendo una búsqueda la imagen oficial de OPENVPN que corresponde al repositorio [kylemanna/openvpn](#).

En el repositorio podemos encontrar una guía de cómo realizar la instalación de OPENVPN en Docker.

En nuestro caso vamos a utilizar la opción de Docker-Compose para realizar la instalación.

Para ello vamos a generar un fichero .yml con el nombre de Docker-compose.

```
|proyecto@debian-docker:~$ nano docker-compose.yml|
```

Vamos a rellenar el fichero con los siguientes parámetros para la instalación.

```

version: '2'
services:
  openvpn:
    cap_add:
      - NET_ADMIN
    image: kylemanna/openvpn
    container_name: proyectodockervpn
    ports:
      - "1194:1194/udp"
    restart: always
    volumes:
      - / proyecto/openvpn-data/conf:/etc/openvpn

```

- Version : Corresponde con la versión de Docker-Compose que vamos a utilizar en este caso la 2.
- Services: Le indicamos que servicio vamos a utilizar en este caso OPENVPN.
- Cap_add: Nos permite añadirle al contenedor privilegios, en este caso con NET_ADMIN nos permite darle la capacidad de poder manipular cosas como, cambiar la configuración de la Interfaz, administrar el Firewall, modificar la tabla de rutas etc.
- Podemos buscar todas las opciones que nos da el kernel de Linux buscando en google man 7 capabilities, de esta forma podremos ver una lista completa de todas las opciones.
- Image: Le indicamos que imagen queremos usar, en este caso vamos a usar la oficial.
- Container_name: Indicamos a Docker el nombre que tendrá el contenedor.
- Ports: Le indicamos los puertos que queremos utilizar.
- Restart: Le indicamos que siempre queremos que el contenedor se reinicie en caso de que se detenga.
- Volúmenes: Le indicamos donde queremos que se cree el volumen que va a contener el contenedor, esto es algo muy importante ya que, en caso de no indicarle un punto de montaje, Docker generaría un volumen que en caso de actualizar el contenedor se borraría, en este caso vamos a mapear el volumen a la carpeta /proyecto/openvpn-data/conf. Esto mismo se podría realizar con la opción -v en caso de no hacer la instalación con Docker-compose.

Una vez generado el fichero, vamos a iniciar generando la configuración del VPN, le vamos a indicar el dominio a resolver para conectarse, también le podríamos poner directamente la IP.

*Con el modificar –rm conseguimos crear un contenedor temporal una vez acabe de ejecutarse se eliminará.

```
docker-compose run --rm proyectodockervpn ovpn_genconfig -u udp://vpn.roy  
ectodocker.com
```

```
proyecto@debian-docker:~$ sudo docker-compose run --rm openvpn ovpn_genconfig -u udp://vpn.proyectodocker.co
Creating network "proyecto_default" with the default driver
Pulling openvpn (kylemanna/openvpn)...
latest: Pulling from kylemanna/openvpn
c9b1b535fdd9: Pull complete
48ae345d34a0: Pull complete
9660e571431b: Pull complete
294766247fac: Pull complete
e0fcf9e9959e: Pull complete
Digest: sha256:f638aff0997e519ab356ea4770d33df08c69cbc8fde6071130e5459e92d266d7
Status: Downloaded newer image for kylemanna/openvpn:latest
Processing PUSH Config: 'block-outside-dns'
Processing Route Config: '192.168.254.0/24'
Processing PUSH Config: 'dhcp-option DNS 8.8.8.8'
Processing PUSH Config: 'dhcp-option DNS 8.8.4.4'
Processing PUSH Config: 'comp-lzo no'
Successfully generated config
Cleaning up before Exit ...
```

Para hacer funcionar el Servidor VPN tenemos que generar una llave publica, una privada y un certificado. Para realizar esta tarea vamos a utilizar el siguiente comando.

```
docker-compose run --rm openvpn ovpn_initpki
```

```
proyecto@debian-docker:~$ sudo docker-compose run --rm openvpn ovpn_initpki
init-pki complete; you may now create a CA or requests.
Your newly created PKI dir is: /etc/openvpn/pki
```

```
Using SSL: openssl OpenSSL 1.1.1d  10 Sep 2019
```

```
Enter New CA Key Passphrase:
Re-Enter New CA Key Passphrase:
Generating RSA private key, 2048 bit long modulus (2 primes)
.....+++++
Generating RSA private key, 2048 bit long modulus (2 primes)
.....+++++
.....+++++
e is 65537 (0x010001)
Can't load /etc/openvpn/pki/.rnd into RNG
139790256741704:error:2406F079:random number generator:RAND_load_file:Cannot open fi
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Common Name (eg: your user, host, or server name) [Easy-RSA CA]:ProyectoDocker
```

```
THE SUBJECT'S DISTINGUISHED NAME IS AS FOLLOWS  
commonName :ASN.1 12:'vpn.proyectodocker.com'  
Certificate is to be certified until Apr 8 11:53:33 2023 GMT (1080 days
```

```
Write out database with 1 new entries  
Data Base Updated
```

```
Using SSL: openssl OpenSSL 1.1.1d 10 Sep 2019  
Using configuration from /etc/openvpn/pki/safessl-easyrsa.cnf  
Enter pass phrase for /etc/openvpn/pki/private/ca.key:
```

```
An updated CRL has been created.  
CRL file: /etc/openvpn/pki/crl.pem
```

Una vez acabada habrá generado las claves publica/privada y el PKI. Una vez generados ya podemos iniciar el contenedor.

```
Sudo docker-compose up -d openvpn
```

```
| proyecto@debian-docker:~$ sudo docker-compose up -d openvpn  
Creating projectodockervpn ... done  
proyecto@debian-docker:~$
```

Ahora vamos a generar una variable donde almacenaremos el nombre del usuario del vpn, esto no es necesario si le indicáramos directamente el nombre funcionaria de la misma forma.

```
| proyecto@debian-docker:~$ export CLIENTNAME="dockerprojeto"
```

Vamos a generar al usuario dentro de la base de datos del servidor VPN.

```
projeto@debian-docker:~$ sudo docker-compose run --rm openvpn easyrsa build-client-full $CLIENTNAME

Using SSL: openssl OpenSSL 1.1.1d  10 Sep 2019
Generating a RSA private key
.....+++++
.....+++++
writing new private key to '/etc/openvpn/pki/private/dockerprojeto.key.XXXXGFhDLg'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
Using configuration from /etc/openvpn/pki/safessl-easyrsa.cnf
Enter pass phrase for /etc/openvpn/pki/private/ca.key:
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName :ASN.1 12:'dockerprojeto'
Certificate is to be certified until Apr 8 11:54:51 2023 GMT (1080 days)

Write out database with 1 new entries
Data Base Updated
```

Una vez creado lo único que nos queda por hacer es generar el fichero de configuración que tendrá que usar el usuario para conectarse.

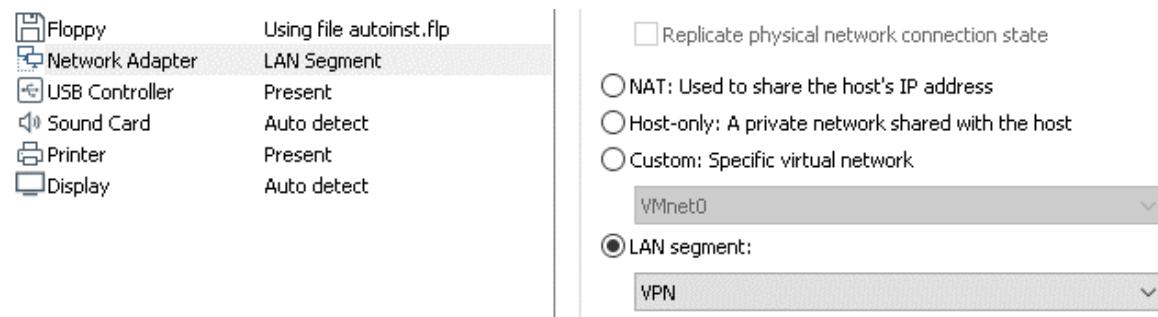
```
Sudo docker-composer run -rm openvpn ovpn_getclient $CLIENTNAME > $CLIENTNAME.ovpn
```

```
projeto@debian-docker:~$ sudo docker-compose run --rm openvpn ovpn_getclient $CLIENTNAME > $CLIENTNAME.ovpn
```

```
projeto@debian-docker:~$ ls
Descargas docker-compose.yml dockerprojeto.ovpn Documentos Escritorio Imágenes Música openvpn-data Plantillas Público Videos
```

Comprobación Funcionamiento VPN

Para comprobar el funcionamiento de la VPN vamos a crear una maquina cliente, en este caso vamos a usar una distribución de Ubuntu, la tarjeta de red la vamos a configurar en el segmento de red de VPN.



Una vez configurado la interfaz con su correspondiente IP, vamos a realizar varias comprobaciones:

- Revisar la configuración de la interfaz.
- Tener conectividad con el servidor VPN.
- No tener salida a internet hasta pasar por el VPN.
- Configurar el dns para resolver vpn.proyectodocker.com

```
uservpn@ubuntu:~/Desktop$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host
            valid_lft forever preferred_lft forever
2: ens37: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:ec:7f:b3 brd ff:ff:ff:ff:ff:ff
        inet 10.0.0.13/24 brd 10.0.0.255 scope global ens37
            valid_lft forever preferred_lft forever
        inet6 fe80::20c:29ff:feec:7fb3/64 scope link
            valid_lft forever preferred_lft forever
```

Para comprobar la conectividad con el servidor VPN vamos a realizar ping a la IP 10.0.0.10.

```
v4lto_t4t forever preferred_t4t forever
uservpn@ubuntu:~/Desktop$ ping 10.0.0.10
PING 10.0.0.10 (10.0.0.10) 56(84) bytes of data.
64 bytes from 10.0.0.10: icmp_seq=1 ttl=64 time=0.270 ms
64 bytes from 10.0.0.10: icmp_seq=2 ttl=64 time=0.213 ms
64 bytes from 10.0.0.10: icmp_seq=3 ttl=64 time=0.206 ms
64 bytes from 10.0.0.10: icmp_seq=4 ttl=64 time=0.194 ms
64 bytes from 10.0.0.10: icmp_seq=5 ttl=64 time=0.207 ms
```

Como podemos comprobar podemos llegar a esa IP.

Ahora vamos a configurar el fichero /etc/hosts para que resuelva el dominio y nos llevo a la dirección IP 10.0.0.10

```
uservpn@ubuntu:~/Desktop$ cat /etc/hosts
127.0.0.1      localhost
127.0.1.1      ubuntu

10.0.0.10  vpn.proyectodocker.com
```

Ya sabemos que podemos llegar a resolver la dirección del vpn, por lo tanto, vamos a ejecutar el perfil del vpn para poder conectarnos.

Servicio WEB: NginX

Hemos decidido instalar una alternativa a apache que es el que hemos estado estudiando en el curso, a la hora de elegir uno diferente hemos optado por NginX por ser muy popular al igual que apache.

¿Que es NginX?

NGINX, pronunciado en inglés como «engine-ex», es un famoso software de servidor web de código abierto. En su versión inicial, funcionaba en servidores web HTTP. Sin embargo, hoy en día también sirve como proxy inverso, balanceador de carga HTTP y proxy de correo electrónico para IMAP, POP3 y SMTP.

NGINX fue lanzado oficialmente en octubre del 2004. El creador del software, Igor Sysoev, comenzó su proyecto en el 2002 como un intento de solucionar el problema C10k. C10k es el reto de gestionar diez mil conexiones al mismo tiempo. Hoy en día, los servidores web tienen que manejar un número aún más grande de conexiones. Por esa razón, NGINX ofrece una arquitectura asíncrona y controlada por eventos, característica que hace de NGINX uno de los servidores más confiables para la velocidad y la escalabilidad.

Debido a su excelente capacidad para manejar muchas conexiones y a su velocidad, muchos sitios web de alto tráfico usan el servicio de NGINX. Algunos de estos gigantes del internet son Google, Netflix, Adobe, Cloudflare, WordPress.com y muchos más.



NginX vs apache

Nginx usa dramáticamente menos memoria que Apache, y puede manejar aproximadamente cuatro veces más solicitudes por segundo. Este aumento de rendimiento viene con un costo de disminuida flexibilidad, como por ejemplo la capacidad de anular las configuraciones de acceso del sistema por archivo (Apache logra esto con un archivo [.htaccess](#), mientras que Nginx no tiene desarrollada tal funcionalidad).

Incluir NginX con docker:

Ahora vamos a introducir nginx en nuestra maquina que presta servicios con contenedores, la pagina que usaremos para mostrar estará hecha con bootstrap 4.5, lo que quiere decir que queremos una instalación limpia de nginx con la carpeta necesaria para subir el "index.html".

Siendo administradores introduciremos el comando para instalar el contendor con las características que buscamos.

```
proyecto@debian-docker:~$ sudo docker run --restart always --name proyectonginx -d -p 80:80 -v $HOME/docker/nginx:/usr/share/nginx/html:ro nginx  
3eac568eb263520042c229285276c20f4ee0974272c9358e4dcef30bb9cf8f1c  
proyecto@debian-docker:~$ █
```

```
Sudo docker run --restar always --name proyectonginx -d -p 80:80 -v  
$HOME/docker/nginx:/usr/share/nginx/html:ro nginx
```

¿Cómo sacaríamos el fichero de configuración de nginx?

```
/host/path/nginx.conf:/etc/nginx/nginx.conf:ro
```

Entre las propiedades del comando con “-restar always” hacemos que el docker arranque siempre cuando la maquina se encienda, -p elegimos los puertos que usara en este caso el “80” y con –V seleccionamos donde estará la carpeta que hará referencia a la carpeta donde se encuentra por lo general el fichero index en nginx.

Con el comando `docker ps -a |grep nginx`

Podremos observar su estado en este caso esta “UP”

```
proyecto@debian-docker:~$ sudo docker ps -a |grep nginx
3eac560eb263        nginx           "nginx -g 'daemon off;"   11 minutes ago      Up 11 minutes   0.0.0.0:80->80/tcp    proyectonginx
proyecto@debian-docker:~$
```

Se puede para con el comando stop:



Y volverlo a lanzar con el comando start:



Bootstrap:

Nos dirigimos al a carpeta que creamos en el comando de instalación del contenedor de nginx:

```
$HOME/docker/nginx/
```

```
proyecto@debian-docker:~/docker/nginx$ tree
.
├── css
│   └── style.css
├── docker.html
└── img
    ├── 1.jpeg
    ├── 2.jpg
    ├── 3.jpg
    ├── docker.png
    ├── glyphicons-halflings.png
    ├── glyphicons-halflings-white.png
    ├── logo.jpeg
    ├── net.jpg
    ├── nginx.png
    ├── openldap.png
    └── openvpn.jpg
├── index.html
└── js
    ├── nginx.html
    ├── openldap.html
    └── openvpn.html

3 directories, 17 files
```

Introducimos el fichero index y la carpeta de css e imágenes.

A diferencia de cómo se usó en clase, no descargaremos los ficheros de bootstrap si no que los cargaremos directamente desde internet con lo cual no es necesario descargarlos y cargarlos en la carpeta.

```

<!DOCTYPE html>
<html lang="en">

<head>
<meta charset="utf-8">


<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css">
<link rel="stylesheet" href="./css/style.css">


<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>


<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"></script>


<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"></script>
<link href="https://fonts.googleapis.com/css?family=Bangers&display=swap" rel="stylesheet">
</head>

```

Cargamos las librerías de ficheros css y JavaScript desde internet con estas líneas en la cabecera del fichero index.

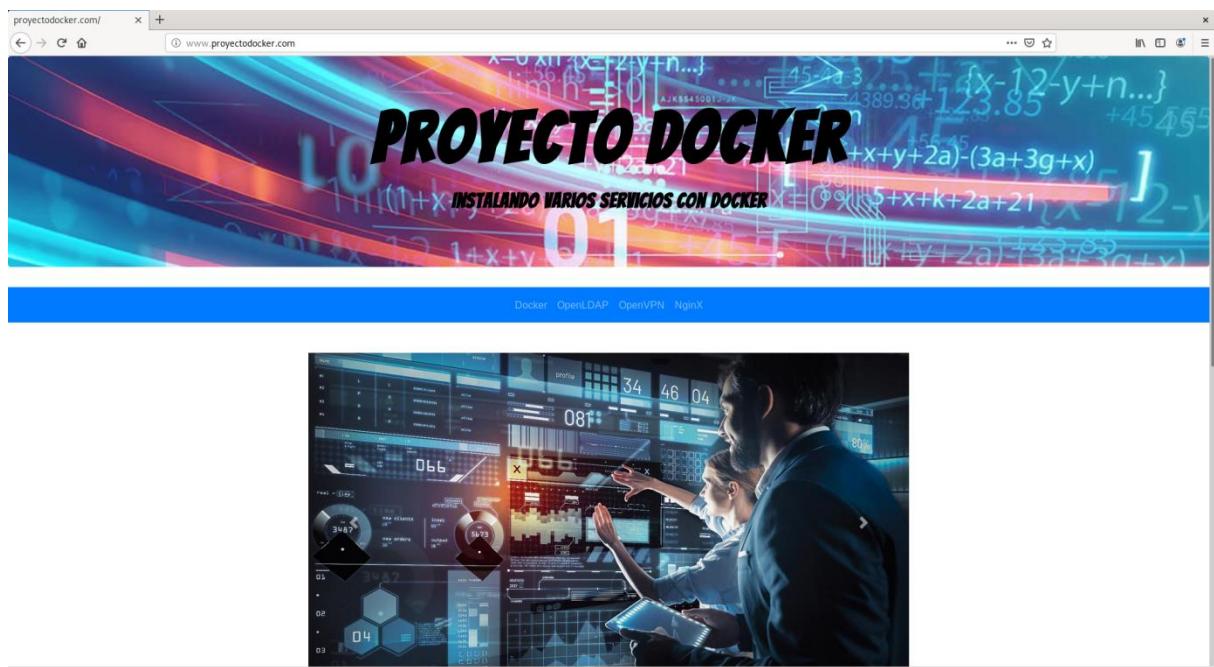
Una vez hecho esto nos guiarímos con las guías de bootstrap para crear una página acorde a nuestro gusto desde 0 sin necesidad de tener que almacenar ficheros extra y con las propiedades de las clases de bootstrap no te tienes que preocupar del tamaño de la página ya que se ajustara automáticamente según la resolución del navegador (se irá auto-ajustando como pueda).

Ejemplo de la barra de navegación:

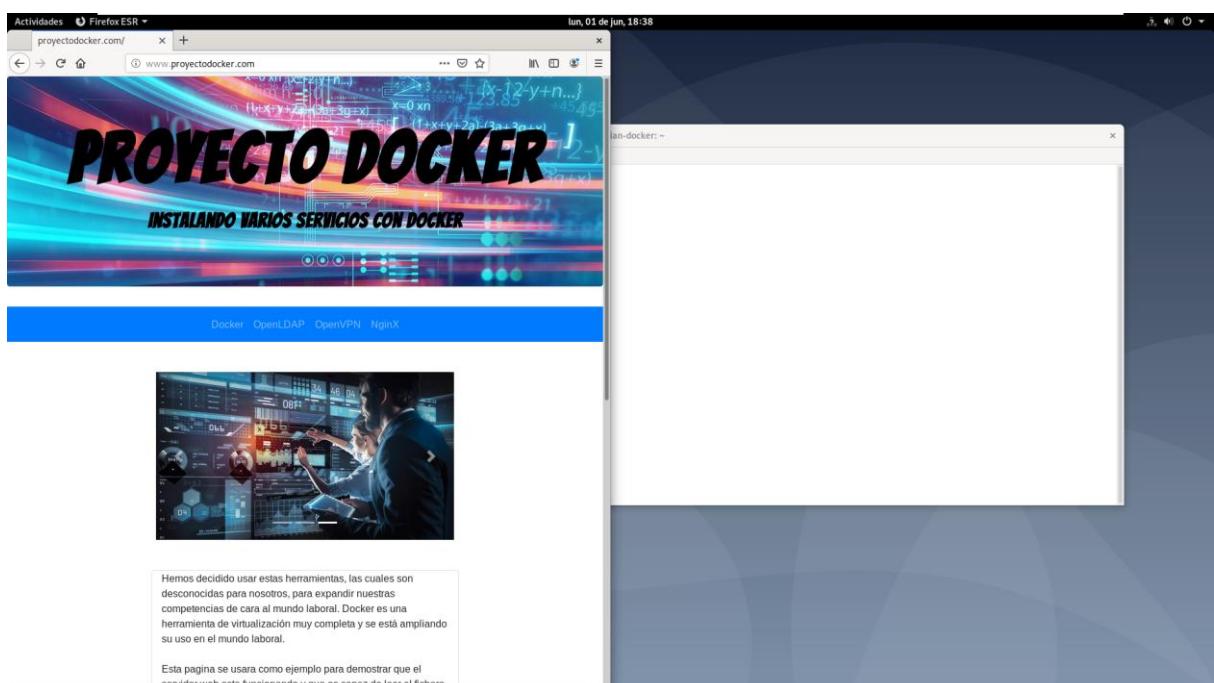
```

<body>
<a href=".index.html">
<div class="jumbotron text-center" style="background-image: url('./img/net.jpg'); background-size: cover;">
<h1 class="titulo"><b> Proyecto Docker</b> </h1>
<p class="subtitulo"><b>Instalando varios servicios con Docker</b></p>
</div></a>
<nav class="navbar navbar-expand-sm bg-primary navbar-dark">
<!-- Links -->
<ul class="navbar-nav barra">
<li class="nav-item">
<a class="nav-link" href=".docker.html">Docker</a>
</li>
<li class="nav-item">
<a class="nav-link" href=".openldap.html">OpenLDAP</a>
</li>
<li class="nav-item">
<a class="nav-link" href=".openvpn.html">OpenVPN</a>
</li>
<li class="nav-item">
<a class="nav-link" href=".nginx.html">NginX</a>
</li>
</ul>
</nav>
<br><br>

```



Comparativa de la página en distintos tamaños.



Nnigx con base de datos:

Docker nos permite mucha flexibilidad y personalización, como por ejemplo juntar una base de datos y nginx en un contenedor de esta forma podríamos instalar joomla o wordpress de una forma muy rápida sin tener que pasar por instalar y gestionar una base de datos.

Dentro del fichero de docker-compose para crear scripts de contenedores podremos incluir ambos como en este ejemplo:

```
#Nginx Service
webserver:
  image: nginx:alpine
  container_name: webserver
  restart: unless-stopped
  tty: true
  ports:
    - "80:80"
    - "443:443"
  networks:
    - app-network

#MySQL Service
db:
  image: mysql:5.7.22
  container_name: db
  restart: unless-stopped
  tty: true
  ports:
    - "3306:3306"
  environment:
    MYSQL_DATABASE: laravel
    MYSQL_ROOT_PASSWORD: your_mysql_root_password
    SERVICE_TAGS: dev
    SERVICE_NAME: mysql
  networks:
    - app-network
```

Instalación de portainer (Gui para contenedores)

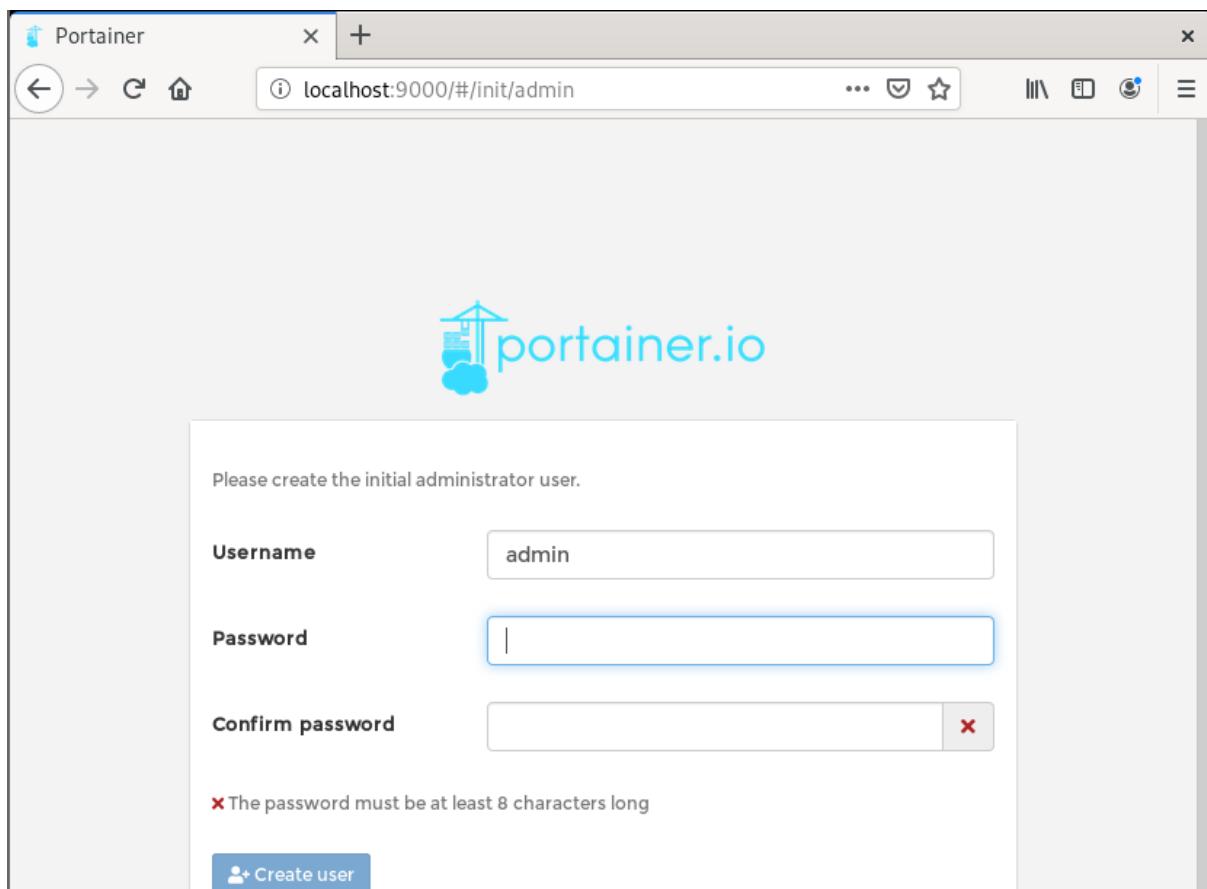
Para obtener portainer pondremos el siguiente comando

```
docker run -d \
--name portainer \
--restart=always \
-p 9000:9000 \
-v /var/run/docker.sock:/var/run/docker.sock \
-v portainer_data:/data \
```

portainer/portainer

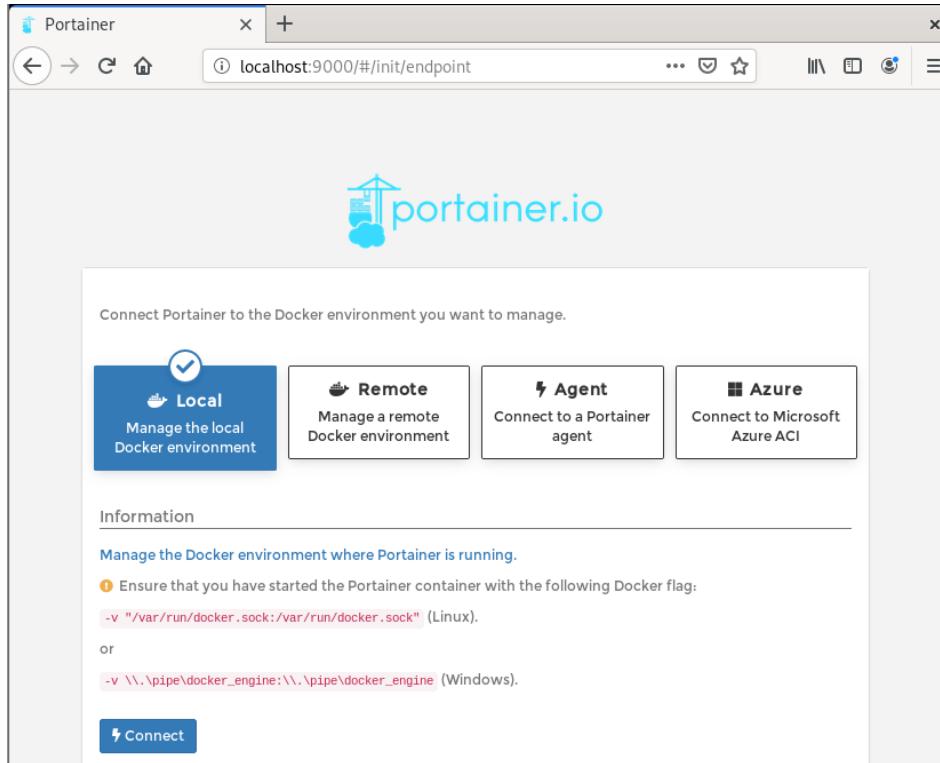
```
projeto@debian-docker:~$ sudo docker run -d \
> --name portainer \
> --restart=always \
> -p 9000:9000 \
> -v /var/run/docker.sock:/var/run/docker.sock \
> -v portainer_data:/data \
> portainer/portainer
[sudo] password for projeto:
Unable to find image 'portainer/portainer:latest' locally
latest: Pulling from portainer/portainer
d1e017099d17: Pull complete
b8084bf83dcf: Pull complete
Digest: sha256:55c7614b1ad61eabc27214299c42d41bb49e5ef78238c0e5783031f041499284
Status: Downloaded newer image for portainer/portainer:latest
4404894c65d4ae8615e63587dd2dc6269821a4709b8184c9908e67c5b74214a3
projeto@debian-docker:~$
```

Para conectarnos a portainer usaremos localhost:9000

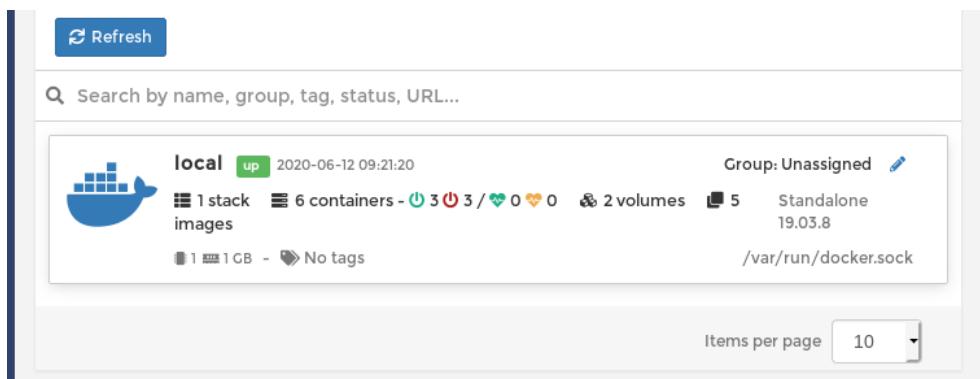


Al ser la primera vez que conectamos nos pedirá crear una contraseña para el admin

A continuación, nos pide una configuración de como vamos a administrar nuestros contenedores, en este caso va a ser Local



Después accederemos al menú donde se muestra nuestros contenedores



Y aquí ya veremos nuestros contenedores

Containers								<input type="checkbox"/> Columns	<input type="checkbox"/> Settings
	<input type="button"/> Start	<input type="button"/> Stop	<input type="button"/> Kill	<input type="button"/> Restart	<input type="button"/> Pause	<input type="button"/> Resume	<input type="button"/> Remove	<input type="button"/> + Add container	
<input type="text"/> Search...									
<input type="checkbox"/> Name	State	Quick actions	Stack	Image	Created	Published Ports	Ownership		
Filter									
<input type="checkbox"/> portainer					-	portainer/portainer	2020-06-12 09:16:50		
<input type="checkbox"/> projectonginx					-	nginx	2020-06-01 18:19:56		
<input type="checkbox"/> projectodockervpn					proyecto	kyfemanna/openvpn	2020-04-23 14:21:53		
<input type="checkbox"/> phpldapadmin-service					-	osixia/phpldapadmin:0.9.0	2020-04-22 11:03:23	-	
<input type="checkbox"/> ldap-service					-	osixia/openldap:1.3.0	2020-04-22 11:00:23	-	
<input type="checkbox"/> my-openldap-container					-	osixia/openldap:1.3.0	2020-04-22 10:59:15	-	

Como podemos observar nos muestra los contenedores y las opciones que podemos usar con ellos. Para esta demo, dejamos uno de los contenedores sin uno de los parámetros para que inicie directamente con la maquina y así ver como podemos iniciarla a posteriori.

<input checked="" type="checkbox"/> phpldapadmin-service				-	osixia/phpldapadmin:0.9.0	2020-04-22 11:03:23	-	
<input checked="" type="checkbox"/> ldap-service				-	osixia/openldap:1.3.0	2020-04-22 11:00:23	-	
<input checked="" type="checkbox"/> my-openldap-container				-	osixia/openldap:1.3.0	2020-04-22 10:59:15	-	

Seleccionamos los 3 servicios a arrancar Y pinchamos en start

Containers							
	<input type="button"/> Start	<input type="button"/> Stop	<input type="button"/> Kill	<input type="button"/> Restart	<input type="button"/> Pause	<input type="button"/> Resume	
<input type="text"/> Search...							
<input type="checkbox"/> Name	State	Quick a	Filter				
<input type="checkbox"/> portainer							
<input type="checkbox"/> projectonginx							
<input type="checkbox"/> projectodockervpn							
<input checked="" type="checkbox"/> phpldapadmin-service							
<input checked="" type="checkbox"/> ldap-service							
<input checked="" type="checkbox"/> my-openldap-container							

Después de esperar 1 minuto, portainer ha arrancado los servicios

<input type="checkbox"/>	phpldapadmin-service	running	
<input type="checkbox"/>	ldap-service	running	
<input type="checkbox"/>	my-openldap-container	running	

Una vez arrancado el servicio, desde portainer nos podemos meter a saber cual es la ip asignada para acceder a el

The screenshot shows the Portainer interface with the 'Container status' section for the 'phpldapadmin-service'. The container ID is b04fb3097cff15885655667fd9645e77bce20b13e116ec5f1b758ab7e304bcdd. The IP address listed is 172.17.0.4. Other details include the name, status (Running for 3 minutes), creation time (2020-04-22 11:03:23), and start time (2020-06-12 09:35:18). Below the table are navigation links: Logs, Inspect, Stats, Console, and Attach.

ID	b04fb3097cff15885655667fd9645e77bce20b13e116ec5f1b758ab7e304bcdd
Name	phpldapadmin-service
IP address	172.17.0.4
Status	Running for 3 minutes
Created	2020-04-22 11:03:23
Start time	2020-06-12 09:35:18

Logs Inspect Stats Console Attach

Al introducir la ip en la web, vemos como accedemos al servicio sin problema

The screenshot shows a browser window with the URL https://172.17.0.4. The page displays the 'phpLDAPadmin' logo and a sidebar with a 'ldap-service' entry and a 'login' link. The main content area is currently empty.

Aparte de esto portainer nos permite hacer despliegues rápido de aplicaciones ya que incluye un “repositorio” de donde podemos descargar diversos contenedores

 Plone	A free and open-source CMS built on top of Zope	CMS
 Magento 2	container Open-source e-commerce platform	<input checked="" type="checkbox"/> Update <input type="button" value="Delete"/>
 Sematext Docker Agent	container Collect logs, metrics and docker events	<input checked="" type="checkbox"/> Update <input type="button" value="Delete"/>
 Datadog agent	container Collect events and metrics	<input checked="" type="checkbox"/> Update <input type="button" value="Delete"/>
 Mautic	container Open-source marketing automation platform	<input checked="" type="checkbox"/> Update <input type="button" value="Delete"/>
 Wowza	container Streaming media server	<input checked="" type="checkbox"/> Update <input type="button" value="Delete"/>
 Jenkins	container Open-source continuous integration tool	<input checked="" type="checkbox"/> Update <input type="button" value="Delete"/>
 Redmine	container Open-source project management tool	<input checked="" type="checkbox"/> Update <input type="button" value="Delete"/>
 Odoo	container Open-source business apps	<input checked="" type="checkbox"/> Update <input type="button" value="Delete"/>
 Urbackup	container	<input checked="" type="checkbox"/> Update <input type="button" value="Delete"/>

Copias De Seguridad

Hay que tener en cuenta que con Docker tenemos dos cosas a tener en cuenta a la hora de hacer Backups, por un lado la imagen del servicio y por otra los datos (volúmenes).

Para hacer un backup de la imagen tendremos que hacer un Commit del contenedor

```
Docker commit -p [ContainerID] nombre_del_backup
```

Una vez acabado el commit podremos crear un tar de la imagen con

```
Docker save -o backup_name.tar nombre_del_backup
```

Para restaurar el backup, solo tendremos que usar un

```
Docker load -i /ruta_del_tar
```

Para los datos solo tendremos que ir al volumen que tenemos mapeado y hacer un tar

Conclusión

Trabajar con Docker ha sido muy rápido y sencillo, puedes hacer una gran cantidad de contenedores personalizados a tu gusto e instalar los servicios que quieras sin necesidad de tener muchos recursos. Hemos probado otros servicios que nunca habíamos instalado en la teoría y práctica de clase por lo que hemos ganado capacidades nuevas.

También está la posibilidad de instalar la herramienta portainer la cual permite tener de manera centralizada la administración de estos contenedores.

Los contenedores se quitan y ponen con mucha facilidad y apenas se nota que gaste recursos del equipo, estas prácticas realizadas con docker han sido interesantes de aprender y vemos la posibilidad de seguir aprendiendo y usando docker en el futuro.

Web gráfia

<https://www.w3schools.com/bootstrap4/default.asp>

<https://www.digitalocean.com/community/tutorials/como-configurar-laravel-nginx-y-mysql-con-docker-compose-es>

<https://ugeek.github.io/blog/post/2019-03-25-monta-un-servidor-web-nginx-con-una-sola-linea-de-terminal-docker.html>

https://hub.docker.com/_/nginx

<https://www.hostinger.es/tutoriales/que-es-nginx/>

<https://es.wikipedia.org/wiki/Nginx>

<https://www.digitalocean.com/community/tutorials/como-instalar-y-usar-docker-en-debian-9-es>

<https://dockertips.com/utilizando-docker-compose>

<https://github.com/kylemanna/docker-openvpn>