

S.I.V.E.

Sistemas Operativos III

AFM Tech System

Rol	Apellido	Nombre	C.I	Email	Tel/Cel.
Integrante 2	Aguilera	Estela	4621249-5	estela231092@gmail.com	092856282
Sub-Coordinadora	Aguirre	Ashelem	5254868-6	ashe_713@hotmail.com	095890552
Integrante 1	Fernández	Matías	4550079-0	mfalfasio@gmail.com	091080985
Coordinador	Martínez	Federico	4591407-6	martinez.fl@gmail.com	094540813
Integrante 3	Tomassini	Dino	4235739-8	dinotomassini@gmail.com	099746158

Docente: Pajares, Juan

Fecha de entrega

10/09/2021

SEGUNDA ENTREGA

I.S.B.O.

3 IF



Índice

Índice	1
Configuración en el servicio SSH	2
Servidor	2
Cliente	3
Configuración de red del servidor	5
Configuración de red en Docker	7
Creación de scripts para la creación de usuarios	8
SSOO	8
BD	8
Perfiles de usuario - Casos de uso PGP (ssh-keygen)	9
Diagrama de implementación	10
Desarrollo	10
Testing	11
Pre-producción	12
Webgrafía	13
HOJA TESTIGO	14



Configuración en el servicio SSH

Para poder acceder de forma remota al servidor se utiliza el protocolo ssh para realizar una conexión segura a la shell del servidor. Para esto se necesita instalar y configurar openssh, en nuestro caso este paquete venía instalado por defecto en CentOS 7.

Servidor

Instalar el paquete “openssh-server” y configurar el archivo “sshd_config” que se encuentra en “/etc/ssh/” y agregar la línea “PermitRootLogin no” para que no se pueda acceder remotamente con el usuario root. También se puede cambiar el puerto por defecto. Luego de esto se pasa a reiniciar el servicio de ssh y abrir el puerto en el firewall del sistema.

```
$ sudo yum install openssh-server -y
```

Agregar la línea “PermitRootLogin no”:

```
$ sudo vi /etc/ssh/sshd_config
```

Esto deja el servicio para iniciar automáticamente con el sistema:

```
$ sudo systemctl reload sshd
```

```
$ sudo systemctl enable sshd
```



Así abrimos el puerto para la conexión ssh:

```
$ sudo firewall-cmd --permanent --add-port=22/tcp
```

Cliente

En nuestro caso el cliente es un sistema GNU/Linux por lo que el paquete necesario para esta conexión viene instalado por defecto. Solo es necesario ejecutar el comando “ssh usuario@ip.del.servidor”. También generamos un par de claves público privadas para la conexión al servidor.

Conexión con el servidor:

```
$ ssh dino@192.168.1.220
```

Crear la clave:

```
$ ssh-keygen -t rsa -b 4096 -C "Servidor Sive CentOS 7"
```

Agregar la clave pública al servidor:

```
$ ssh-copy-id -i .ssh/id_rsa_server-sive.pub dino@192.168.1.220
```



Conectarse nuevamente con la llave:

```
$ ssh -i .ssh/id_rsa_server-sive dino@192.168.1.220
```

```
~ 12:07:25 dino
> tail -n 20 .bash_history
#1630853953
ssh-keygen -t rsa -b 4096 -C "Servidor Sive CentOS 7"
#1630854236
mv id_rsa_server-* .ssh/
#1630854247
ls .ssh/
#1630854281
ssh-copy-id -i .ssh/id_rsa_server-sive.pub dino@192.168.1.220
#1630854331
ssh dino@192.168.1.220
#1630854358
ssh -i .ssh/id_rsa_server-sive.pub dino@192.168.1.220
#1630854392
ll .ssh/id_rsa_server-sive
#1630854396
ll .ssh/id_rsa_server-sive*
#1630854417
ssh -i .ssh/id_rsa_server-sive dino@192.168.1.220
#1630854444
tail .bash_history
~ 12:07:38 dino
```

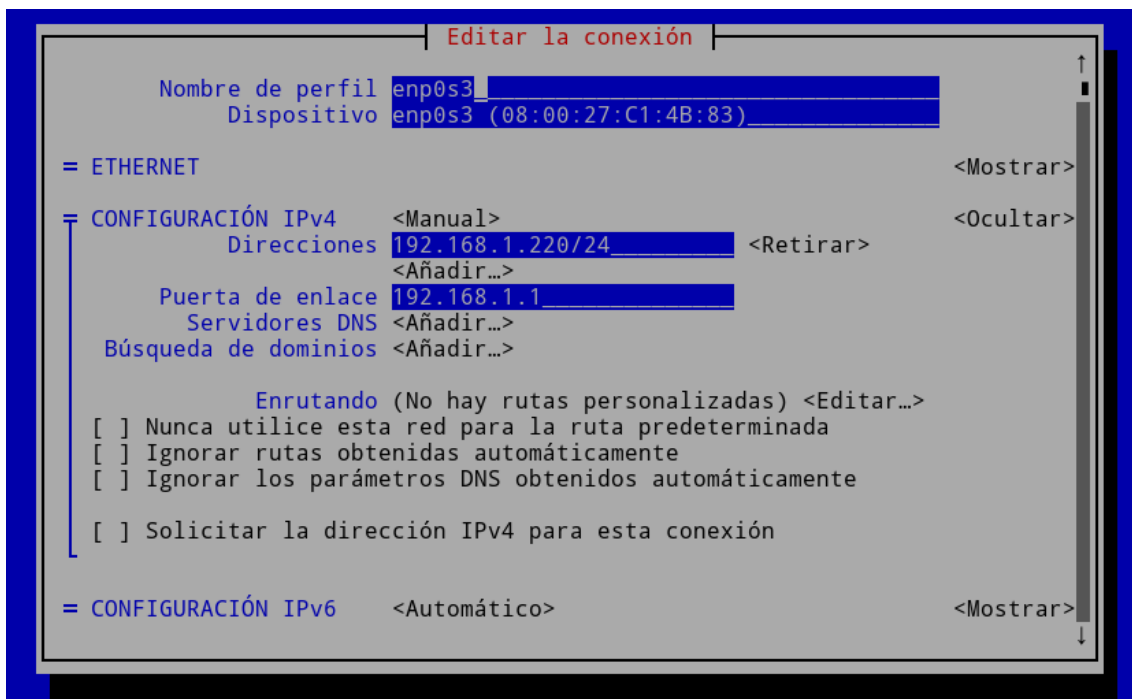


Configuración de red del servidor

Para la configuración de red se utilizó la herramienta “nmtui” la cual es gráfica y muy intuitiva. Se configuró para el ejemplo los siguientes datos:

- IP: 192.168.1.220/24
- GW: 192.168.1.1
- Hostname: servidor-sive

La herramienta nmtui





Verificando la ip del servidor

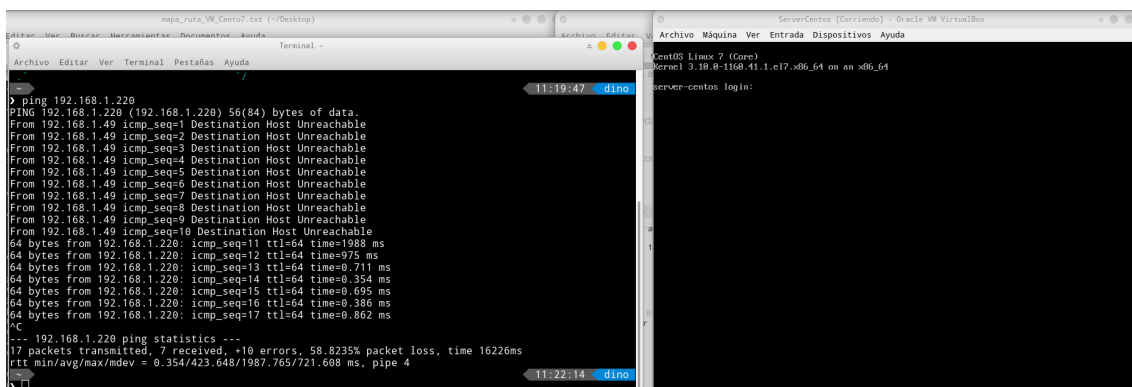
```
[dino@server-centos ~]$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:c1:4b:83 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.220/24 brd 192.168.1.255 scope global noprefixroute enp0s3
        valid_lft forever preferred_lft forever
    inet6 fe80::f346:8083:7b4a:f829/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:5f:2a:68:11 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
        valid_lft forever preferred_lft forever
```

```
[dino@server-centos ~]$ ip r
default via 192.168.1.1 dev enp0s3 proto static metric 100
172.17.0.0/16 dev docker0 proto kernel scope link src 172.17.0.1
192.168.1.0/24 dev enp0s3 proto kernel scope link src 192.168.1.220 metric 100
```

Verificando conexión

```
[dino@server-centos ~]$ ping google.com
PING google.com (142.250.79.78) 56(84) bytes of data:
64 bytes from eze06s11-in-f14.1e100.net (142.250.79.78): icmp_seq=1 ttl=116 time=15.3 ms
64 bytes from eze06s11-in-f14.1e100.net (142.250.79.78): icmp_seq=2 ttl=116 time=16.2 ms
^C
--- google.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1004ms
rtt min/avg/max/mdev = 15.323/15.803/16.284/0.496 ms
```

Verificando desde fuera



```
mpa_ruta_VB Centos7.txt (~/.Desktop)
Archivo Editar Ver Terminal Pestañas Ayuda
11:19:47 dino
$ ping 192.168.1.220
PING 192.168.1.220 (192.168.1.220) 56(84) bytes of data:
From 192.168.1.49 icmp_seq=1 Destination Host Unreachable
From 192.168.1.49 icmp_seq=2 Destination Host Unreachable
From 192.168.1.49 icmp_seq=3 Destination Host Unreachable
From 192.168.1.49 icmp_seq=4 Destination Host Unreachable
From 192.168.1.49 icmp_seq=5 Destination Host Unreachable
From 192.168.1.49 icmp_seq=6 Destination Host Unreachable
From 192.168.1.49 icmp_seq=7 Destination Host Unreachable
From 192.168.1.49 icmp_seq=8 Destination Host Unreachable
From 192.168.1.49 icmp_seq=9 Destination Host Unreachable
From 192.168.1.49 icmp_seq=10 Destination Host Unreachable
64 bytes from 192.168.1.220: icmp_seq=11 ttl=64 time=1988 ms
64 bytes from 192.168.1.220: icmp_seq=12 ttl=64 time=975 ms
64 bytes from 192.168.1.220: icmp_seq=13 ttl=64 time=0.711 ms
64 bytes from 192.168.1.220: icmp_seq=14 ttl=64 time=0.354 ms
64 bytes from 192.168.1.220: icmp_seq=15 ttl=64 time=0.695 ms
64 bytes from 192.168.1.220: icmp_seq=16 ttl=64 time=0.386 ms
64 bytes from 192.168.1.220: icmp_seq=17 ttl=64 time=0.862 ms
^C
--- 192.168.1.220 ping statistics ---
17 packets transmitted, 7 received, +10 errors, 58.8235% packet loss, time 16226ms
rtt min/avg/max/mdev = 0.354/423.648/1987.765/721.688 ms, pipe 4
11:22:14 dino
```



Configuración de red en Docker

Docker por defecto utiliza el modo Bridge lo cual hace un puente entre el contenedor y el sistema en donde se está ejecutando utilizando la interfaz de red del demonio de docker la cual se denomina “docker0”, para poder acceder desde fuera del contenedor es necesario exponer los puertos a utilizar, esto nos permite utilizar la ip del servidor en vez de la ip del contenedor. Todo esto se realiza con algunas opciones al ejecutar el docker:

```
$ docker run -p 80:80 php:7.4.16-apache
```

Pero si queremos asignar ip estática a los contenedores podemos hacerlos con algunas opciones más.

Crear una red:

```
$ docker network create --subnet=172.19.0.0/16 red-sive
```

Ejecutar el docker asignando esa red, una ip y un hostname:

```
$ docker run --net red-sive --ip 172.19.0.20 --hostname sive-web  
php:7.4.16-apache
```




Creación de scripts para la creación de usuarios

Estos script tienen como propósito generar los usuarios necesarios tanto en el sistema operativo como en el motor de base de datos. Estos usuarios son: app, respaldo, admin, dba.

Todos los código utilizados estan en <https://github.com/dinotomassini/sive-ssoo>

SSOO

Para crear los usuarios utilizamos el comando “useradd”:

```
$ sudo useradd -m -p respaldo-sive.21 respaldo
```

```
$ sudo useradd -m -p dba-sive.21 dba
```

```
$ sudo useradd -m -p app-sive.21 app
```

```
$ sudo useradd -m -p admin-sive.21 admin
```

BD

Debemos ingresar a la consola de mysql para poder crear los usuarios y darles sus permisos:

```
$ create user 'dba'@'localhost' identified by 'dba-sive.21';
```

```
$ create user 'app'@% identified by 'app-sive.21';
```

```
$ create user 'respaldo'@'localhost' identified by 'respaldo-sive.21';
```

```
$ grant all privileges on *.* to 'dba'@'localhost';
```

```
$ grant select,insert,update,delete on *.* to 'app'@%;
```



```
$ grant all privileges on *.* to 'respaldo'@'localhost';
```

```
$ flush privileges;
```

Perfiles de usuario - Casos de uso PGP (ssh-keygen)

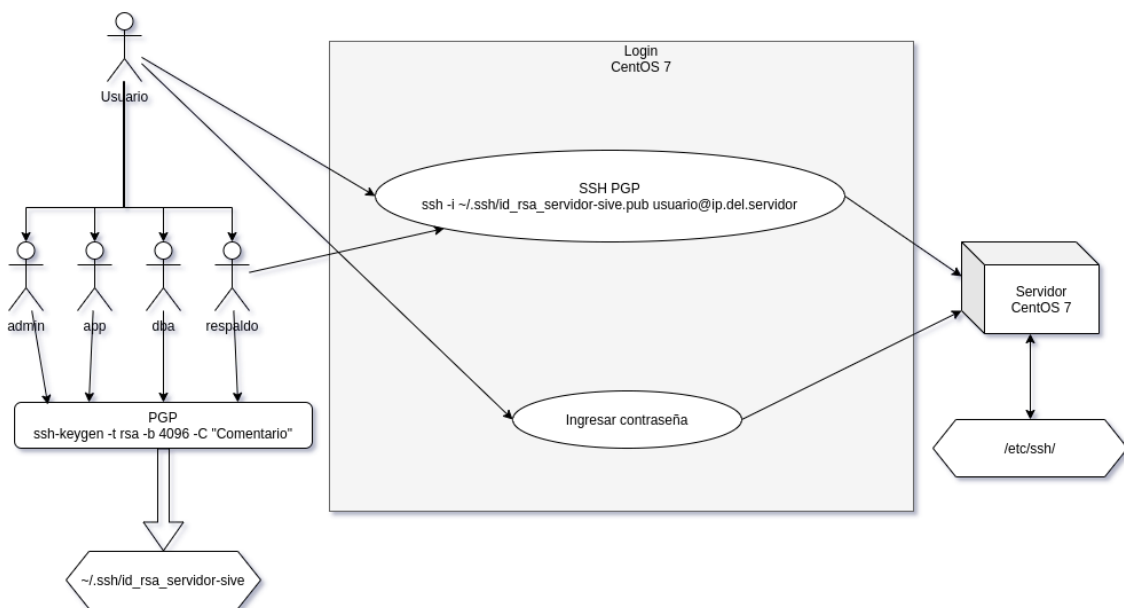
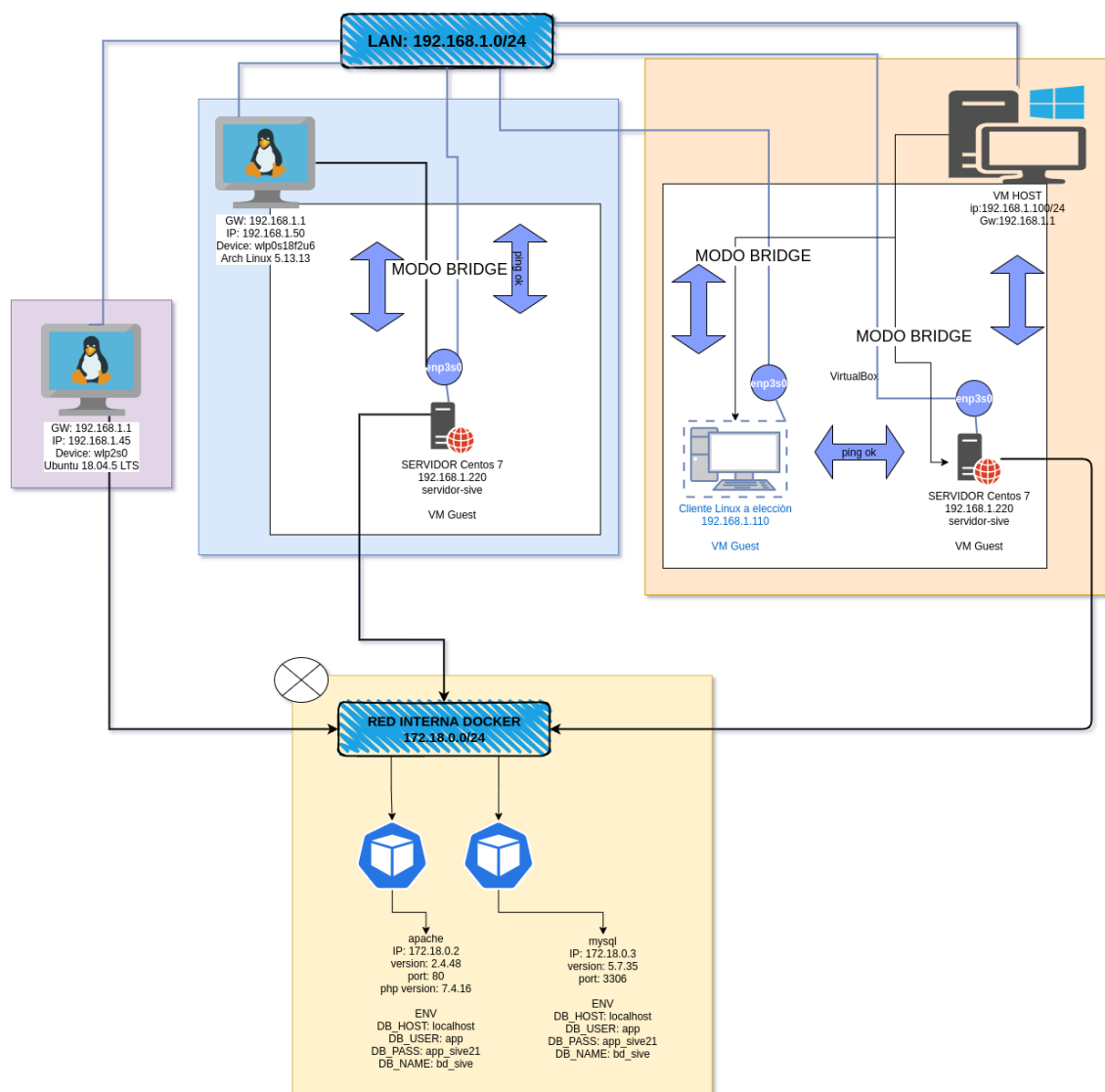


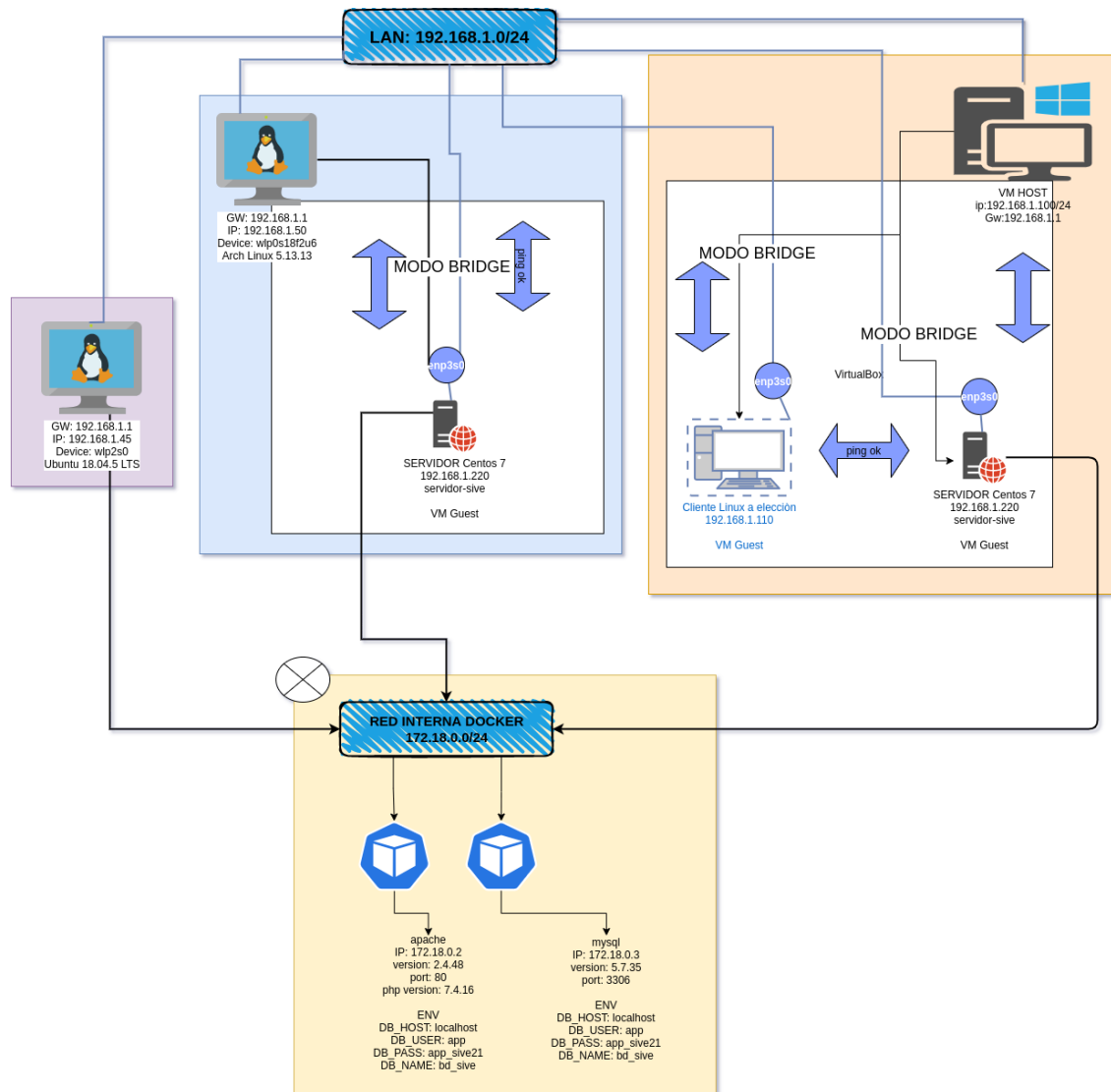


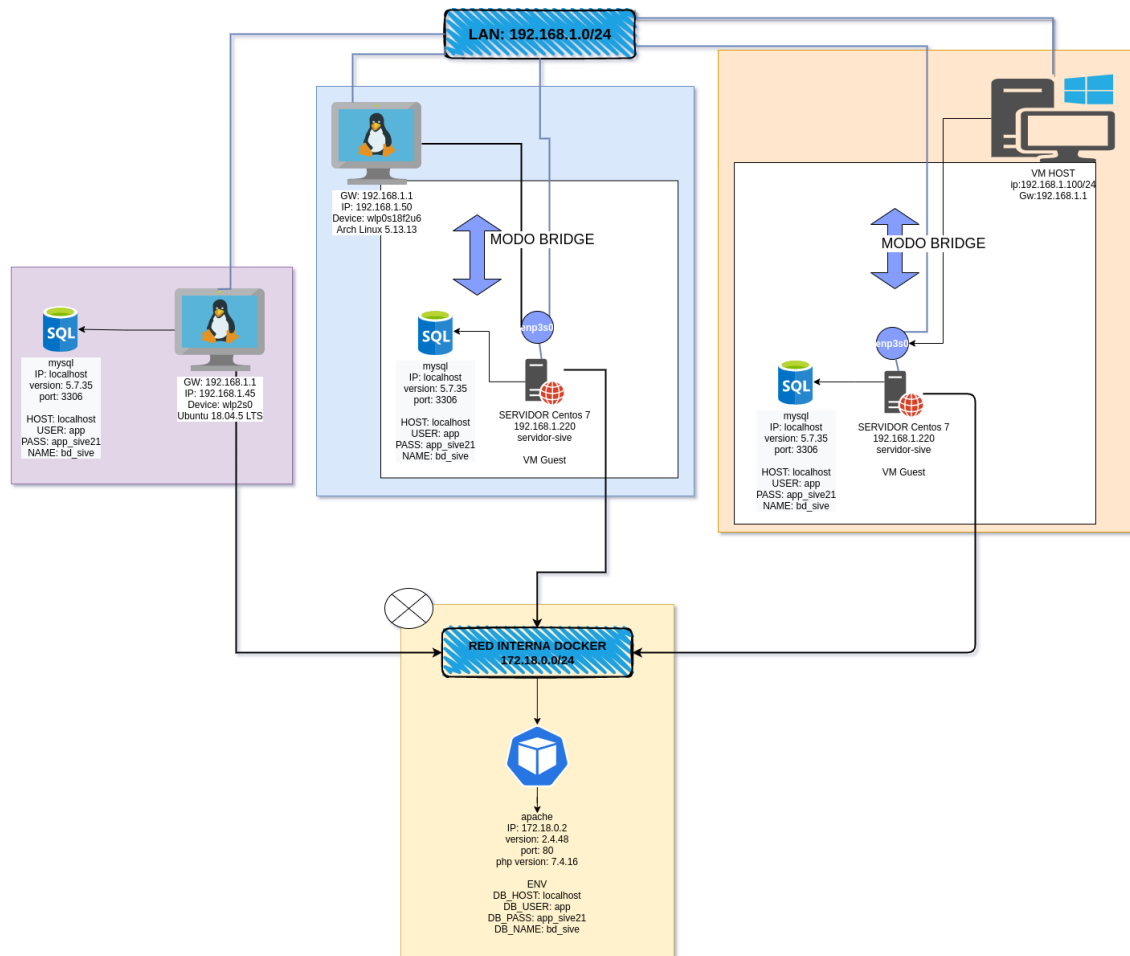
Diagrama de implementación

Se muestran los diagramas de implementación para cada ambiente, de desarrollo, de testing y de pre-producción.

Desarrollo









Webgrafía

Red Hat Enterprise Linux 4: Manual de referencia

<https://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-rg-es-4/ch-ssh.html>

Página del manual

<https://man.openbsd.org/ssh>

<http://www.openssh.com/manual.html>

Guía básica sobre SSH

<https://raiolanetworks.es/blog/ssh/>

Configuración de red de CentOS 7

<https://rm-rf.es/configurar-red-centos-7-rhel-7/>

Configuración de red (Networking) en Dockers

<https://www.maquinasvirtuales.eu/configuracion-red-networking-dockers/>

Conceptos Básicos De Redes Docker

<https://ricardogeek.com/conceptos-basicos-de-redes-docker/>

Creación de red Docker

https://docs.docker.com/engine/reference/commandline/network_connect/

https://docs.docker.com/engine/reference/commandline/network_create/

<https://qastack.mx/programming/27937185/assign-static-ip-to-docker-container>



HOJA TESTIGO