CentOS, nginx, Flask y gunicorn

1. Verficar el nombre del servidor

 # cat /etc/os-release (Verificar el nombre de servidor)

2.Ver en que host estamos trabajando

 # hostname

3.Crear y cambiar a otro host perzonalizado(aqui donde se desplegara la app web)

 # hostnamectl set-hostname redes-curso

4. Agrear el host que creamos a /etc/hosts

 # nano /etc/hosts

/etc/hosts

| 127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
|
| redes-curso 192.168.0.32 (esta varía)
|
|
| ::1 localhost localhost.localdomain localhost6 localhost6.localdomain6

Terminado esto guardan, ctr + o, enter y ctr + x

5. Creamos un usuario designado para que pueda hacer la configuración de los servidores

 # adduser redes

 # passwd redes (en este paso tendran que poner la contraseña del usuario)

6. Luego asignarle al grupo de usuario "wheel" o admnistradores, ser miembro de Wheel para poder convertirse en root. Se utiliza como una capa adicional de rotección para el sistema. No conozco ningún significado especial ahora, es solo un legado como los grupos uucp y dialout.

 # usermod -aG wheel redes

7. Cerrar sesión del root, y luego probar, sí,podemos acceder desde nuestro usuario creado

 # exit

8. Una vez estemos en sesión con el usuario "redes", crear un directorio ".ssh", esto para poder alojar nuestra llave publica

$ mkdir /home/redes/.ssh

9. Cerrar sesión del usuario "redes"

$ exit

10. Para eso tenemos que agregar más seguridad a nuestro servidor ya que tenemos hasta el momento a un al usuario raíz "root" y el usuario "redes".

En este punto tendremos que genera una llave para accesso a servidores remotos

*Requisito: tener instalado el plugin de openSSH de windows o instalado un terminal unix en nuestro computador (puede el git bash)

abrimos la terminal de comandos (o el bash que biene instalado con git)

> ssh-keygen -b 4096 (full enter nomas)

11.Aqui tendremos que copiar nuestra llave creada a nuestro servidor de trabajo, esto porque solo nosotros podremos acceder a nuestro servidor remoto

a) Si estan en la terminal de windows (cmd)

> scp C:\Users\ASUS\.ssh\id_rsa.pub redes@192.168.0.32: ~/.ssh/authorized_keys

b) Si estan en el git bash

$ scp ~/.ssh/id_rsa.pub redes@192.168.0.32: ~/.ssh/authorized_keys

Si es la primera vez que se conecta a un nuevo servidor les pedira una confirmación solo escriben yes o si, en este paso les pedira la contraseña que crearon en el servidor CentOS, solo ingresar y copiara nuetra llave al directorio .ssh de nuestro usuario redes.

12. Se conectan desde su terminal o el git bash con el protocolo ssh

a) Si estan en la terminal de windows (cmd)

> ssh redes@192.168.0.32

b) Si estan en el git bash

$ scp redes@192.168.0.32

Deberan autenticarse con la contraseña que crearon

13. Una vez iniciado sesión con el usuario "redes", cambiar los permisos de nuestro directorio .ssh, esto para mayor seguridad dentro de nuestro que más adelante nos autenticaremos por medio de esta llave para esto ejecutar los siguiente comandos.

Esto solo para que el grupo de los propietarios puedan leer, cambiar y ejecutar

$ chmod 700 ~/.ssh/

Esto solo para que el grupo de los propietarios puedan leer, cambiar.

$ chmod 600 ~/.ssh/*

14.Modificar la configuración de acceso remoto a nuestro servidor con el ssh

$ sudo nano /etc/ssh/sshd_config #les pedira una contraseña, solo ponen la contraseña del usuario "redes"

Dentro del archivo tendran que cambiar la opción de yes a no

|

|

| PermitRootLogin no

|

| PasswordAuthentication no

|

|

PermitRootLogin -> Se deshabilita el inicio de sesión con el usaurio raíz

PasswordAuthentication -> Se deshabilita la autenticación por contraseña

Terminado esto guardan, ctr + o, enter y ctr + x

15. Reiniciar el servicio de ssh (su servicio es sshd)

$ sudo systemctl restart sshd

En este punto deberan probar solo conectandose desde su equipo local al servidor remoto, aqui no les pedira contraseña, ya que nosotrs cambiamos la autenticación por llave secreta.

$ ssh redes@192.168.0.32

16. En este paso debemos de limpiar los paquetes que tenemos y actualizar todos los paquetes de nuestro servidor,

$ sudo dnf clean all

$ sudo dnf update & dnf upgrade

17. Instalación de nginx

$ sudo dnf install nginx -y

18. Agregar al grupo de nginx y dar permisos en directorio del usaurio redes de lectura, escritura y ejecución, y el en grupo "wheel" el permiso de ejecutar

$ sudo usermod -aG redes nginx

$ chmod 710 /home/redes

19. Instalar el software de supervisor

$ sudo dnf install epel-release -y

$ sudo dnf install supervisor -y

epel-release: repositorio de otros paquetes de linux

supervisor: software para poder ....

20. Para nuestro ejemplo usaremos una aplicación de Flask microframework de python y tenemos que instalar python3 y git.

***$ sudo dnf install git -y**

***$ sudo dnf install python3 -y (esto de no tener python instalado en nuestro servidor)**

a) Creamos un directorio nuevo "project-redes", para mantener aqui nuestra aplicación

$ mkdir /home/redes/project-redes

b) Aqui clonamos la aplicación https://github.com/josetanta/my_blog.git

$ cd /home/redes/project-redes/

$ git clone https://github.com/josetanta/my_blog.git

- Paso extras no pertenecientes a la configuración del servidor, sino del proyecto clonado

Creamos un entorno virtual para nuestro proyecto de Flask

$ sudo pip3 install virutalenv

$ virtualenv env

$ source env/bin/activate

- Dirigirse al directorio de "my_blog"

$ cd my_blog/

$ pip3 install -r requirements.txt

Esperamos a que termine la instalación

1. Configurar nuestro nginx y crear un nuevo server

$ sudo touch /etc/nginx/conf.d/webredes.conf (el nombre puede ser cualquiera)

$ sudo nano /etc/nginx/conf.d/webredes.conf

```
|
| server {
|
|     listen 80; # A la escucha en el puerto :80
|     server_name 192.168.0.32; #Aqui puede ir el nombre en DNS y/o el IP(el de tu servidor)
|
|     location /static { #Archivos estaticos (html, css, js)
|           alias /home/redes/project-redes/my_blog/app/static;
```

```
|     }
|
|     location / { #path inicial de nuestra aplicación (más conocido como el home, index o inicio)
|           include /etc/nginx/fastcgi_params;
|           proxy_pass http://localhost:8000/; # puerto por default de gunicorn
|           proxy_redirect off;
|     }
|
| }
```

2. Tenemos que cambiar el firewall a tipo "drop" por defecto, solo dispnible para tráfico de salidas, pero deshabilitado traficos entrantes.

$ sudo firewall-cmd --set-default-zone=drop

3. Agregar los servicios a nuestro firewall-cmd drop

$ sudo firewall-cmd --zone=drop --add-service=http --permanent

$ sudo firewall-cmd --zone=drop --add-service=ssh --permanent

$ sudo firewall-cmd --zone=drop --add-service=dhcpv6-client --permanent

$ sudo firewall-cmd --zone=drop --add-service=cockpit --permanent

Por ultimo recargar nuestro firewall-cmd

$ sudo firewall-cmd --reload

$ sudo firewall-cmd --list-all (con esto verificamos si se agregaron nuestros servicio)

4. Configuraciones extras

a) Hacemos copia de .env.example

$ cp .env.example .env

$ nano .env

|

| APP_NAME="Redes de Computadoras"

| FLASK_APP=manage:app

| FLASK_ENV=production

| SECRET_KEY="miclavesecreta"

|

5. Activamos nuestro servidor nginx

$ sudo systemctl start nginx

$ sudo systemctl enable nginx

$ sudo systemctl status nginx (verificamos si está activo)

6. Ahora nuestra primera prueba

en /home/redes/project-redes/my_blog/, para esto debemos de tener activado nuestro entorno virtual

$ export FLASK_APP=manage:app

$ flask db migrate

$ flask db upgrade

$ gunicorn -w 3 manage:app

Probar en el navegador con nuestra ip -> 192.168.0.32

7. Si obtenemos el error 502, es porque SELinux es por que Centos no soporta a nuestra aplicación, para realizar esto desactivamos temporalmente el SElinux

$ sudo setenforce 0

$ sudo systemctl restart nginx

¡¡¡Con esto ya tenemos desplegado nuestra aplicación!!!

8. Ahora configuraremos para que nuestro supervisor haga las tareas de ejecutar nuestro proyecto ya desplegado

```
$ sudo nano /etc/supervisord.d/webredes.ini
```

```
|
| [program:webredes]
| directory=/home/redes/project-redes/my_blog
| command=/home/redes/project-redes/env/bin/gunicorn -w 3 manage:app
| user=redes
| autostart=true
| autorestart=true
| stopasgroup=true
| killasgroup=true
| stderr_logfile=/var/log/webredes/webredes.err.log
| stdout_logfile=/var/log/webredes/webredes.out.log
|
|
```

Guardar el archivo

9. Creamos los logs de errores y salidas

```
$ sudo mkdir -p /var/log/webredes
$ sudo touch /var/log/webredes/webredes.err.log
$ sudo touch /var/log/webredes/webredes.out.log
```

10. Y por último!!!

Reiniciamos el supervisord

```
$ sudo systemctl restart supervisord
$ sudo supervisorctl reload
$ sudo supervisorctl restart webredes
```

Y ya tienes el servidor con una aplicación