Laboratorio N° 2

“Configuración de NGINX Web Server”

LABORATORIO

Objetivos:

- ✓ Configurar el servidor NGINX como Web Server.
- ✓ Comprender el archivo de configuración.
- ✓ Configurar hosts virtuales.
- ✓ Crear y asignar páginas de error a cada host virtual
- ✓ Trabajar con el parámetro try_files.

PROCEDIMIENTO

Pasos previos.

Configurar el archivo /etc/nginx/nginx.conf

1. **Modificar el parámetro workers_processes, conforme a nuestro servidor.**

```
# grep ^processor /proc/cpuinfo | wc -l
```

2. **Las limitaciones en el Core del sistema también se pueden visualizar con el siguiente comando, esto servirá para trabajar con el parámetro worker_connections dentro del archivo de configuración.**

```
# ulimit -n
```

3. **El comando anterior dara como resultado un número, dependiendo de este valor será el total de conexiones que tendrá nuestro servidor. Por defecto este número es 1024, a lo que nosotros lo modificaremos. Para ello agregamos el final del archivo las siguientes líneas.**

```
# vim /etc/security/limits.conf
...
*  soft nofile  60000
*  hard nofile  60000
```

4. **Procedemos a reiniciar el servidor luego de este cambio.**

5. **Habiendo modificado el archivo limits.conf y conociendo el número de CPU's en nuestro sistema, colocamos estos valores en el archivo de configuración. En los parámetros worker_processes y worker_connections en el archivo nginx.conf.**

```
# vim /etc/nginx/nginx.conf
...
user nginx;
worker_processes 2;

error_log /var/log/nginx/error.log warn;
pid /var/run/nginx.pid;

events {
    worker_connections 60000;
    use epoll;
    multi_accept on;
}
...
```

Los siguientes parámetros dan a conocer:

- ***use epoll:*** Un mecanismo escalable de notificación de eventos de I/O, asegurándose de que I/O es utilizando en su mejor capacidad.
- ***multi_accept on:*** Con el fin de que un worker acepte todas las nuevas conexiones al mismo tiempo.

6. **Con el servidor nginx configurado, modificamos el hostname del servidor para trabajar.**

```
# hostnamectl -set-hostname nginxpc[X].ose.pe
# logout
```

7. **Ingresamos mediante una consola SSH al servidor. Nos dirigimos al directorio de backup y realizamos un backup del archivo de configuración.**

```
# cd /et/nginx/conf.d/
# mv default.conf default.conf.backup
```

8. **Crearemos un archivo de texto que mostrara información de esta website.**

```
# mkdir /opt/web1
# cd /opt/web1
# vim index.html
<html>
<body><h1>WEB SITE WEB1 USER X NGINX OSE</h1></body>
</html>
```

9. Creamos un nuevo host virtual para especificar la configuración del server1.

```
# vim /etc/nginx/conf.d/web1.conf

server {
    listen 80;
    server_name web1.u[x]-server1.nginx.ose;

    location / {
        root /opt/web1;
        index index.html;
    }
}
```

Nota: Donde [X], es el número de PC al que corresponde.

10. Validamos que la configuración del archivo sea correcta.

```
# nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

11. Si todo esta Ok, reiniciamos nginx.

```
# nginx -s reload
```

12. Para validar que esté funcionando correctamente, ingresamos a un navegador y colocamos al nombre que se le asigno en el parámetro server_name.

```
# nginx -s reload
```

13. Creamos un nuevo virtual host, este tendrá más especificaciones. Además, de modificar el primero para que todas las solicitudes realizadas hacia este, se redirijan al segundo virtual host.
14. Creamos un nuevo archivo para el virtual host 2.

```
# vim /etc/nginx/conf.d/web2.conf
server {
    listen 80;
    server_name web2.u[x]-server1.nginx.ose;

    location / {
        root /opt/web2;
```

```
        index index.html;
    }
}
```

15. Creamos el directorio para este nuevo web site.

```
# mkdir /opt/web2
# cd /opt/web2
# vim index.html
<html>
<body><h1>WEB SITE WEB2 USER X NGINX OSE</h1></body>
</html>
```

16. Ahora agregaremos el contexto de error_pages para el segundo host virtual. Creamos el archivo del contenido 404 error file.

```
# vim /opt/web2/custom404.html
<html>
<body>
<h1>404 ERROR PAGE, TRY AGAIN OR OTHER CONTEXT</h1>
</body>
</html>
```

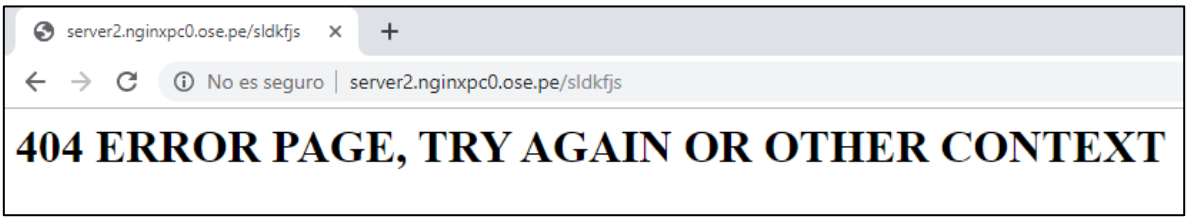
17. Modificamos el archivo de configuración, para especificar la página de error.

```
# vim /etc/nginx/conf.d/web2.conf
...
error_page 404 /custom404.html;
    location = /custom404.html {
        root /opt/web2;
        internal;
}
```

18. Teniendo todo esto, procedemos a validar que la configuración de nginx este bien, y reiniciamos el servicio.

```
# nginx -t
# nginx -s reload
```

19. Para validar esta configuración. Probamos con un contexto que no existe.



20. **Ahora si en algún caso queremos implementar más error_pages para otros números de errores. Agregamos las siguientes líneas al archivo de configuración.**

```
# vim /etc/nginx/conf.d/web2.conf
...
    error_page 500 502 503 504 /custom50x.html;
    location = /custom50x.html{
          root /opt/web2;
          internal;
}
```

21. **Creamos el html para los errores 500.**

```
# vim /opt/web2/custom50x.html
<html>
<body>
<h1>Uy! Something is WRONG</h1>
</body>
</html>
```

22. **Sin embargo para que este tipo de errores se llegue a visualizar agregamos un bloque más al archivo de configuración web2.conf.**

```
# vim /etc/nginx/conf.d/web2.conf
...
    location /errortest {
        fastcgi_pass unix:/does/not/exist;
    }
```

23. **Teniendo todo esto, procedemos a validar que la configuración de nginx este bien, y reiniciamos el servicio.**

```
# nginx -t
# nginx -s reload
```

24. Validamos que el nuevo error_page para valores de 50X estén funcionando, ingresamos al contexto que configuramos con fastcgi.

25. Ahora en un caso nosotros no queramos que los usuarios no vean la página de error y sean redirigidos a la página principal. Para ello modificamos en el archivo de configuración.

```
# vim /etc/nginx/conf.d/web2.conf
...
error_page 404 /custom404.html;
    location = /custom404.html {
        root /opt/web2;
        internal;
        rewrite ^/(.*)$ / redirect;
}
```

26. Validamos la configuración de nginx y reiniciamos.

```
# nginx -t
# nginx -s reload
```

27. Ahora para que se pueda redirigir todo el tráfico del servidor1 hacia el servidor2. Modificamos el archivo de configuración del server1.

```
# vim /etc/nginx/conf.d/web1.conf
server {
    listen 80;
    server_name web1.u[x]-server1.nginx.ose;

    location / {
        root /opt/web1;
        index index.html;
        rewrite ^/(.*)$ http://web2.u[x]-server1.nginx.ose redirect;
    }
}
```

28. Validamos la configuración de nginx y reiniciamos el servicio.

```
# nginx –t
# nginx –s reload
```

29. Para validar esto ingresamos a la dirección del server1. Y nos llevara al web2.

30. Sin embargo, si se ingresa al servidor 1, agregando un contexto a lado del hostname, esto nos redirigirá a la página central del servidor 2. Para que esto no afecte a la redirección de la pagina agregamos el parámetro $uri al rewrite.

```
# vim /etc/nginx/conf.d/web1.conf
server {
    listen 80;
    server_name web1.u[x]-server1.nginx.ose;

    location / {
     root /opt/server1;
     index index.html;
     rewrite ^/(.*)$ http://web1.u[x]-server1.nginx.ose$uri redirect;
    }
}
```

31. Validamos y reiniciamos el servidor nginx.

```
# nginx –t
# nginx –s reload
```

32. Ahora probamos ingresando al servidor 1 y este nos redirigirá al servidor 2, si en un caso lleva un contexto, el valor también pasará al servidor 2.