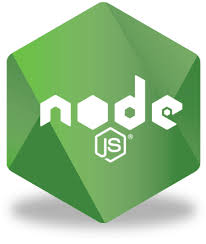
**Proxy Apache sirviendo una aplicación de node.**

**Grupo ED2.**

| **Revisor** | **Estado** | **Fecha** | **Notas** |
| --- | --- | --- | --- |
| [Hasna Zalmat](mailto:hasnaehasnae063@gmail.com) | **Aprobada** | **07/02(24** | Configuración de Node.js con Apache como proxy inverso. Implica dirigir el tráfico entrante del servidor web Apache hacia un servidor Node.js que se ejecuta en un puerto diferente. |
| [Hasna Zalmat](mailto:hasnaehasnae063@gmail.com) | **En revisión** | **11/02/24** | Actualización del título y otros detalles. |
| **Persona** | **No iniciada** |  |  |

**ÍNDICE**

[**Descripción.**](#_k2ghbwwmr3g6) **3**

[**Configuración.**](#_d915b5ilktc4) **4**

[**Conectividad.**](#_ojm2njswtqb4) **10**

[**Creación reglas pfsense.**](#_tkyz0s9vr5s8) **11**

[**Fuentes.**](#_dkc4jf6x4tgu) **11**

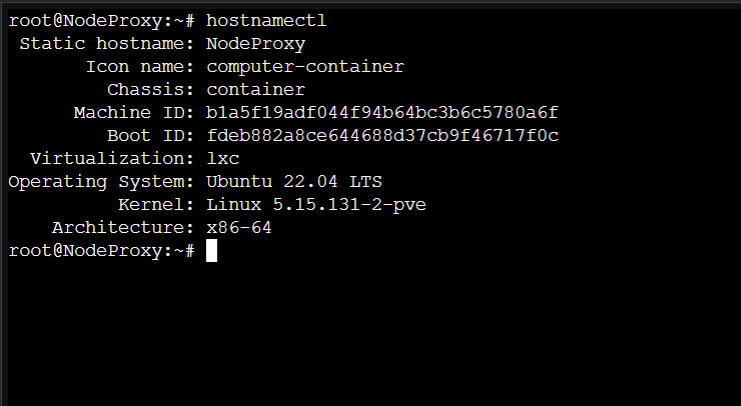
# **Descripción.**

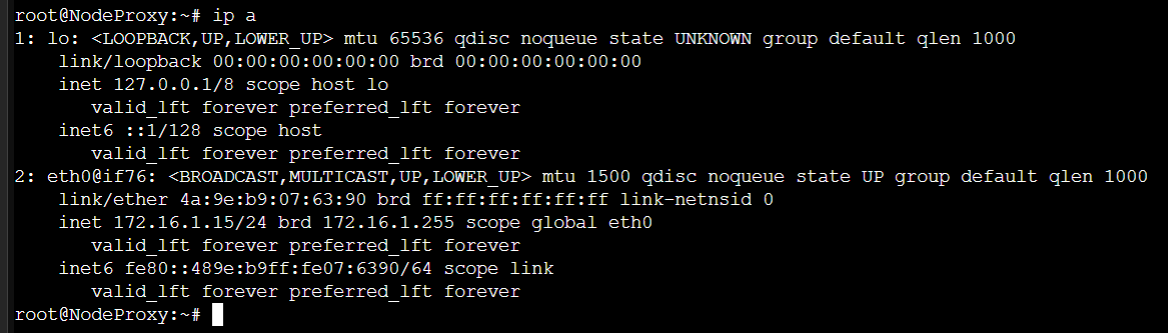
Node.js es un popular entorno de ejecución de JavaScript gratuito y de código abierto basado en el motor JavaScript V8 de Chrome. Se utiliza para sitios web tradicionales y servicios API back-end.

Apache es un servidor web de código abierto que también se puede utilizar Apache como un servidor proxy front-end para aplicaciones back-end incluyendo, Node.js.

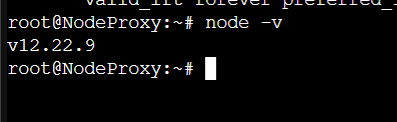
En esta actividad, se muestra cómo configurar Apache como un proxy inverso para la aplicación Node.js en Ubuntu.

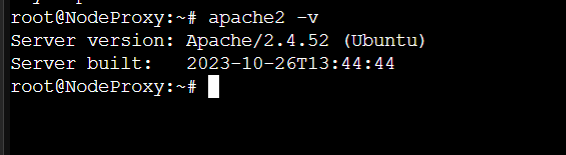
**Máquinas utilizadas.**

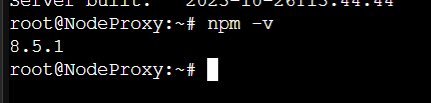




**Paquetes utilizados.**

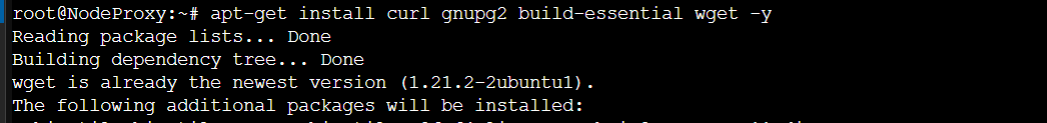




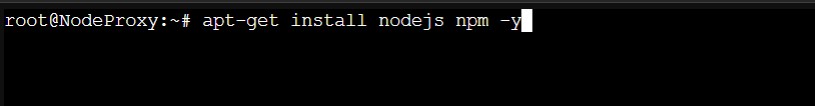


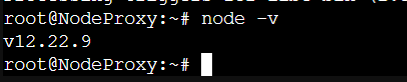
# **Configuración.**

En primer lugar, instalamos las dependencias necesarias.

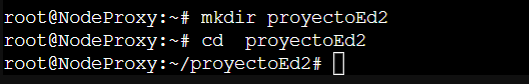


Luego, instalar Node.js con el siguiente comando.





Creamos un directorio para nuestra aplicación Node.js y un archivo nodeapp.js dentro de él:



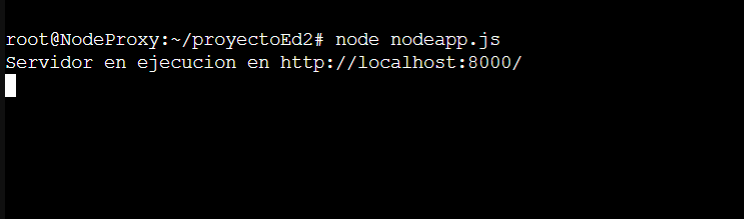


* Importación del módulo HTTP: Se importa el módulo http, que proporciona funcionalidades para crear servidores HTTP y manejar solicitudes y respuestas HTTP en Node.js.
* Definición de variables de host y puerto: Se definen las variables hostname y port para especificar en qué dirección IP y puerto el servidor escuchará las solicitudes entrantes

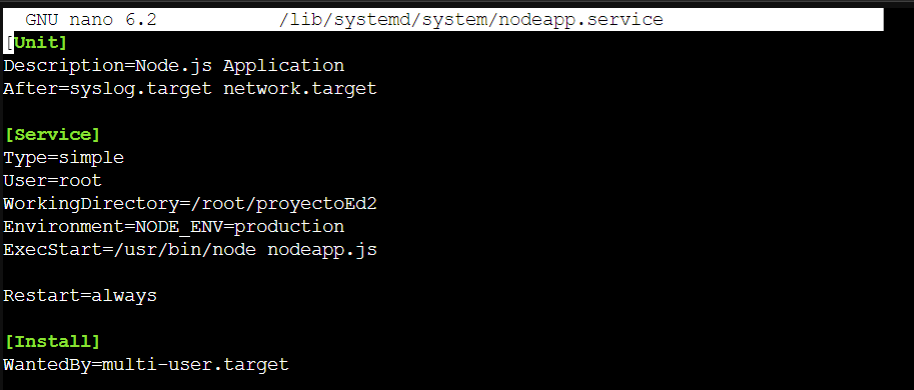
.

* Creación del servidor HTTP: Se utiliza el método createServer() del módulo http para crear un servidor HTTP. Este método toma como argumento una función de devolución de llamada que se ejecuta cada vez que el servidor recibe una solicitud HTTP.
* Manejo de solicitudes y respuestas: Dentro de la función de devolución de llamada, se configura la respuesta que el servidor enviará al cliente. Se establece el código de estado de la respuesta en 200 para indicar que la solicitud se ha procesado correctamente. Además, se establece el tipo de contenido de la respuesta como texto plano ('text/plain'), y se envía un mensaje de bienvenida al cliente.
* Inicio del servidor: Se llama al método listen() en el objeto del servidor para hacer que el servidor comience a escuchar las solicitudes entrantes en el puerto y la dirección IP especificados.

Guardamos el archivo y ejecutamos la aplicación.

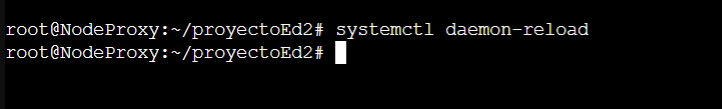


Creamos un archivo de servicio systemd para administrar la aplicación Node.js

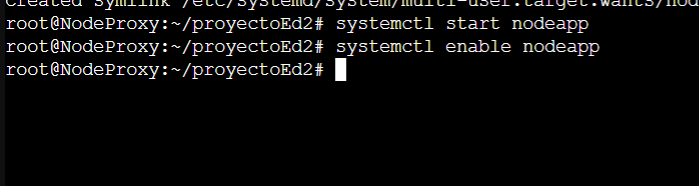


* [Unit]: Esta sección proporciona una descripción del servicio y establece las dependencias del servicio. Especifica que el servicio debe iniciarse después de que se hayan iniciado los servicios syslog.target y network.target.
* [Service]: Esta sección define cómo se ejecuta el servicio. Aquí se especifica el tipo de servicio (simple), el usuario bajo el cual se ejecutará (root), el directorio de trabajo para el servicio (/root/proyectoEd2), y las variables de entorno (NODE\_ENV=production). También se establece el comando de inicio (ExecStart) que ejecuta el script de la aplicación Node.js. Además, se indica que el servicio debe reiniciarse automáticamente en caso de fallo (Restart=always).
* [Install]: Esta sección especifica cómo se instalará el servicio. En este caso, indica que el servicio será deseado por el multi-user.target, lo que significa que se iniciará automáticamente durante el inicio del sistema en entornos multiusuario.

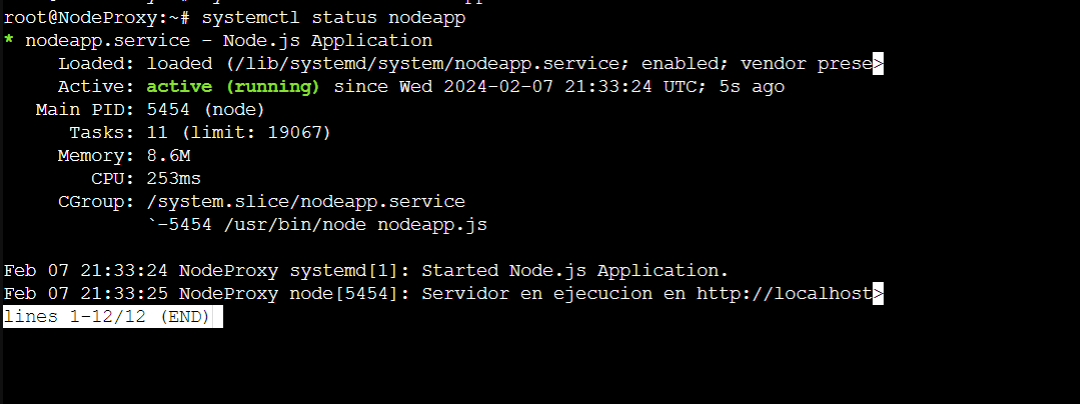
Guardamos y recargamos el daemon de systemd.



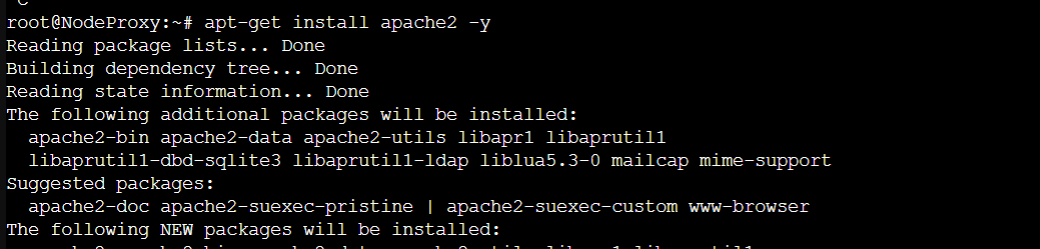
Iniciamos el servicio y habilitamos el arranque.

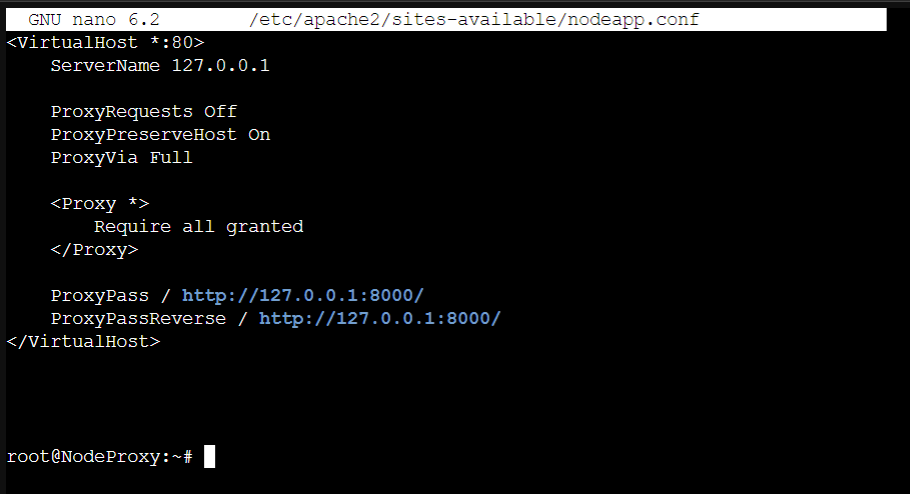


Verificamos el status de nuestro node app, y todo está correcto.



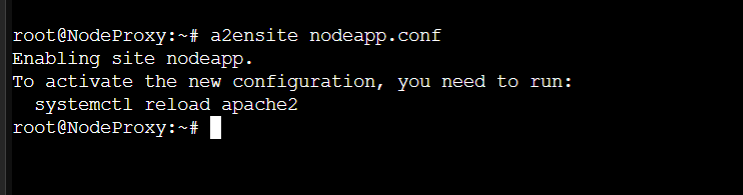
Ahora toca configurar Apache como un proxy inverso.

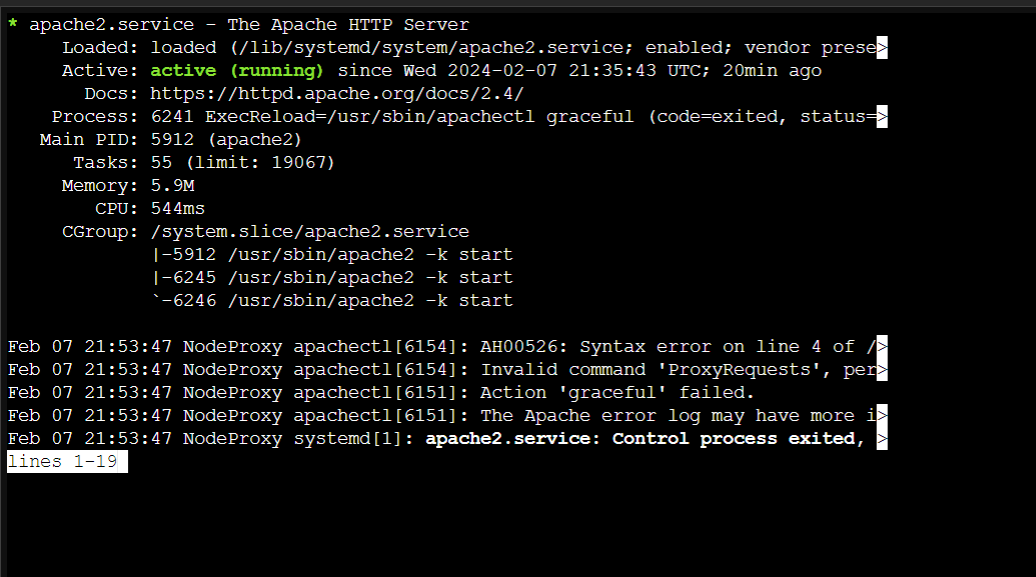




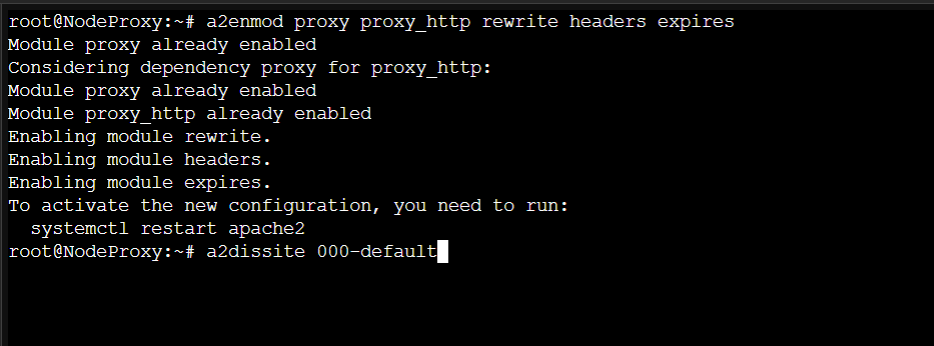
* <VirtualHost \*:80>: Define un virtual host que escucha en todas las interfaces en el puerto 80.
* ServerName your-server-ip: Especifica el nombre del servidor al que se le están dirigiendo las solicitudes.
* ProxyRequests Off: Deshabilita las solicitudes de proxy. Esto asegura que el servidor Apache no actúe como un proxy abierto.
* ProxyPreserveHost On: Preserva la información del host en las solicitudes al servidor Node.js. Esto permite que el servidor Node.js pueda determinar correctamente el host al que se dirige la solicitud.
* ProxyVia Full: Especifica que el encabezado Via en las solicitudes y respuestas debe estar completo, lo que significa que debe contener tanto el nombre del servidor proxy como la versión del servidor proxy.
* <Proxy \*> ... </Proxy>: Define un proxy que permitirá todas las solicitudes. En entornos de producción, debes ajustar esto para limitar las solicitudes solo a las que son necesarias.
* ProxyPass / http://127.0.0.1:8000/: Especifica que todas las solicitudes que lleguen a este virtual host en el puerto 80 deben ser dirigidas al servidor Node.js que se ejecuta en localhost en el puerto 8000.
* ProxyPassReverse / http://127.0.0.1:8000/: Ajusta los encabezados de las respuestas del servidor Node.js para que parezca que están viniendo del servidor Apache, lo que ayuda a evitar problemas con las URLs generadas por la aplicación Node.js.

Habilitamos la configuración del host virtual.



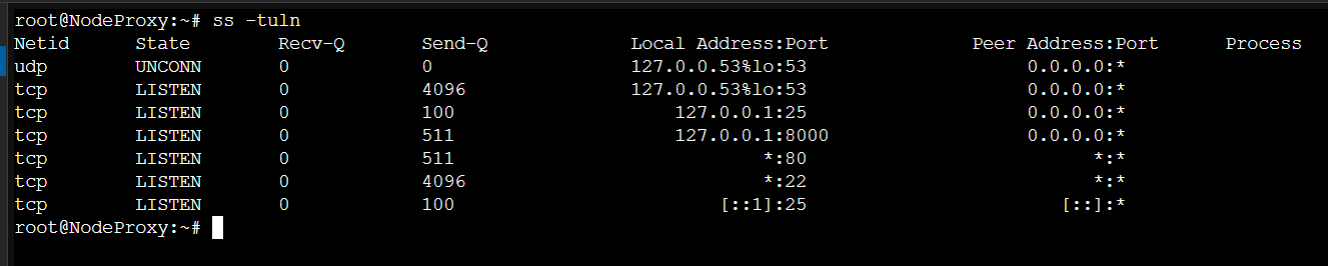


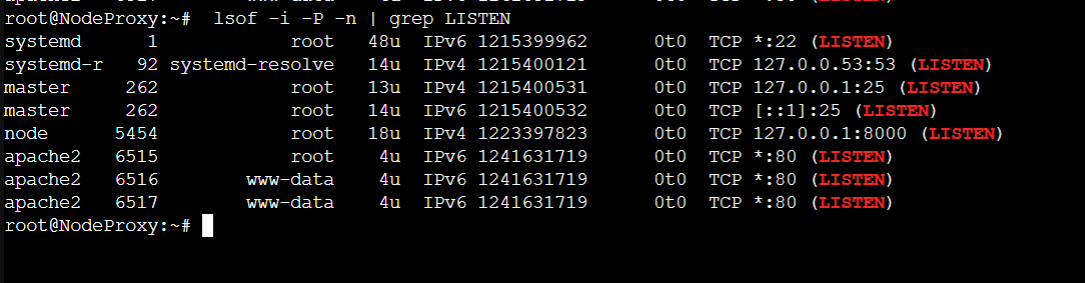
Deshabilitamos la configuración del host virtual predeterminado y habilitamos los módulos de proxy necesarios.



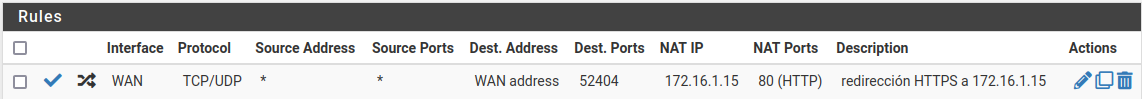
# **Conectividad.**







# **Creación reglas pfsense.**



# 

# **Fuentes.**

<https://www.atlantic.net/dedicated-server-hosting/how-to-configure-reverse-proxy-for-node-js-application-using-apache-on-ubuntu/>

<https://www.digitalocean.com/community/tutorials/how-to-set-up-a-node-js-application-for-production-on-ubuntu-20-04-es>