Balanceo de carga y Failover con servidor Nginx en el borde y una granja de servidores web IPv6 Only

Autor: Alejandro Acosta
Coordinación/revisión: Guillermo Cicileo, Carlos MArtínez
Edición: Área de Comunicaciones
Área: Área de Tecnología

Agosto 2023

lacnic
www.lacnic.net

Introducción

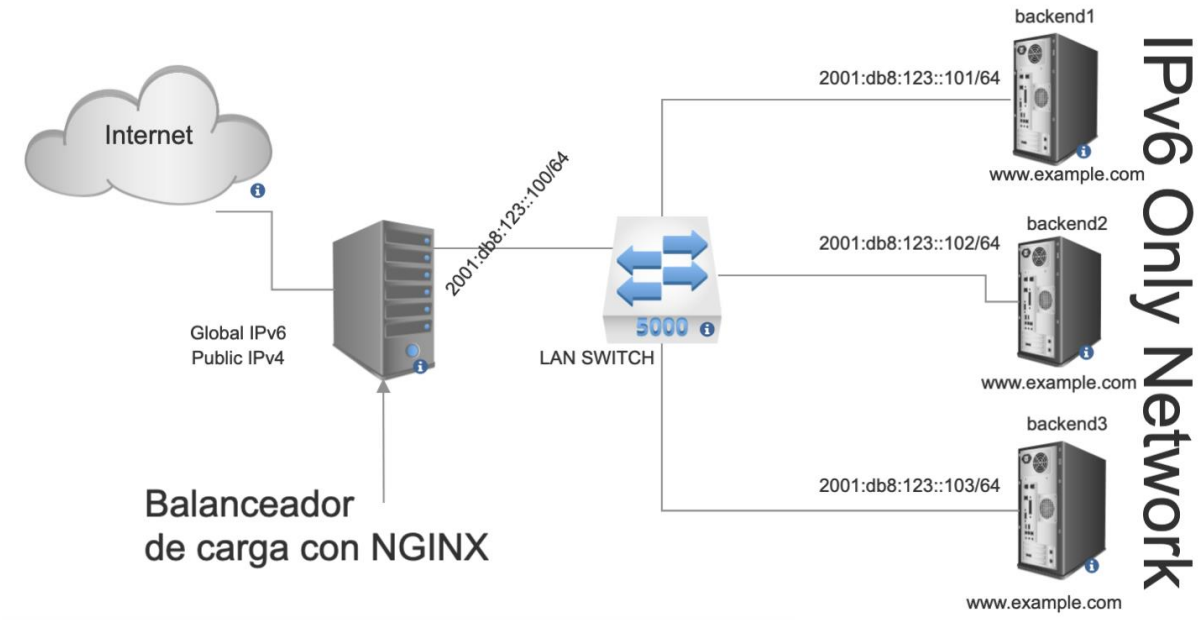
Este trabajo es la continuación del documento NGINX Reverse Proxy y Granja de Servidores IPv6 Only: Conectividad Web Eficiente. En él estuvimos configurando un Proxy Reverso Nginx donde con un Servidor pudimos dar acceso Web Dual Stack (IPv4 e IPv6) desde una granja de servidores IPv6 Only. Un camino muy interesante para ahorrar direcciones IPv4 y obtener otra gran cantidad de beneficios.

En este artículo, exploraremos cómo puedes implementar funcionalidades de balanceo de carga utilizando un servidor Nginx en el borde y una granja de servidores web que operan exclusivamente en IPv6. Descubriremos los beneficios de esta configuración y los pasos necesarios para lograr una arquitectura robusta y confiable, así como los diferentes métodos de implementación.

¿Por qué utilizar Nginx en el borde?

El servidor Nginx es conocido por su rendimiento, escalabilidad y capacidades avanzadas de balanceo de carga. Colocar Nginx en el borde de la red te permite tener un punto de entrada único para tus servicios web, donde puedes administrar y distribuir el tráfico de manera eficiente hacia tu granja de servidores IPv6 Only.

Topología



Métodos de NGINX para balanceo de carga

Nginx maneja varios métodos para el balanceo de carga, te explicamos cada uno:

- IP-hash: este método utiliza un algoritmo que toma la dirección IP de origen y destino del cliente y el servidor para generar una clave hash única. Lo anterior permite la persistencia de la sesión.
- Round-robin (por defecto): es el método predeterminado para el equilibrio de carga. Indica al equilibrador de carga que vuelva a la parte superior de la lista y se repita de nuevo.
- "El Menos conectado" (least_conn): este método utiliza un algoritmo de equilibrio de carga dinámico. Redistribuye las conexiones al miembro del grupo que administran la menor cantidad de conexiones abiertas en el momento en que se recibe la nueva solicitud de conexión.

Requisitos Configuración de Nginx para el balanceo de carga (Servidor Proxy -borde-)

- Instalar Nginx en tu servidor en el borde y asegúrate de que esté configurado correctamente para trabajar con IPv4 e IPv6. Recordemos que este servidor puede escuchar de la calle IPv4 e IPv6, hará proxy de las solicitudes y las reenviará internamente a la granja de servidores solo por IPv6
- Crear un archivo de configuración de Nginx y definir el upstream block con las direcciones IPv6 de tus servidores web.
- Configurar los algoritmos de balanceo de carga, como round-robin, least_conn o ip_hash, para distribuir las solicitudes entre los servidores web de la granja.
- Servidor Linux en el borde con Nginx instalado y el mismo tendrá una dirección IPv4 y una dirección IPv6
- Es necesario que cada uno de tus servidores web en la granja tenga configurada una dirección IPv6 diferente y que la misma sea alcanzable desde el servidor Proxy.

Configuraciones

Configuración en el lado del balanceador

```
#Archivo: /etc/nginx/sites-enabled/example.com
upstream backend {  #El upstream de la granja de servidores se llama upstream
    server [2001:db8:123::101]; #server backend1
    server [2001:db8:123::102]; #server backend2
    server [2001:db8:123::103]; #server backend3
}

server {  #esta ya es una directiva conocida de ningx
    listen 80; #puerto en el que escucha el Web server
    server_name example.com www.example.com;  #nombre del dominio

    location / {
        proxy_pass http://backend;    #nótese que backend es el nombre del
upstream
    }
}
```

Configuración del lado de los servidores de la granja

Todos los servidores backend en la granja tienen la misma configuración

```
#/etc/nginx/sites-available/default
server {
      listen [::]:80 default_server;
      root /var/www/html;
      index index.html index.htm index.nginx-debian.html;
      server_name _;
      location / {
            try_files $uri $uri/ =404;
      }
}
```

Pruebas y monitoreo

Luego de realizar las configuraciones procedemos a probar. Aquí te dejo una lista de posibles pruebas que se pueden realizar:

a) Crear un archivo diferente en cada servidor de la granja y cargar www.example.com desde Internet. Cada vez que cargamos y recarguemos la página debería mostrar una página por cada servidor en la directiva upstream
b) Revisar los logs en nginx en cada servidor backend, por ejemplo se podría realizar: *tail -f /var/log/nginx/*.log* y revisar los accesos y/o errores
c) Revisar los logs en balanceador, igualmente se podría utilizar: *tail -f /var/log/nginx/*.log*
d) *Puedes correr este script muy sencillo para apreciar el round robin:*
for ((i=1;i<=10;i++)); do curl -v "http://www.example.com"; sleep 1; done

Configuración de Nginx para el failover y otras opciones

El failover en Nginx se maneja pasando parámetros a los servidores de backend especificados en el upstream.

Los diferentes parámetros son (ver ejemplo más abajo)

weight: Esta opción permite especificar el peso relativo de cada servidor en el grupo upstream. Como ya has utilizado en tu ejemplo, el peso determina la proporción de solicitudes que cada servidor manejará en comparación con los demás servidores.

max_fails: Esta opción permite especificar el número máximo de intentos fallidos de conexión con un servidor antes de considerarlo temporalmente no disponible. Por defecto, el valor es 1. Por ejemplo, max_fails=3; indica que un servidor se marcará como no disponible después de tres intentos fallidos de conexión consecutivos.

fail_timeout: Esta opción establece el tiempo durante el cual un servidor se considerará no disponible después de alcanzar el número máximo de intentos fallidos especificado por max_fails. Por defecto, el valor es 10 segundos. Por ejemplo, fail_timeout=30s; establece un tiempo de espera de 30 segundos para un servidor marcado como no disponible.

backup: Esta opción indica que un servidor debe utilizarse como reserva o backup. El servidor marcado como backup solo se utilizará si todos los demás servidores están marcados como no disponibles.

down: Esta opción marca un servidor como no disponible de forma permanente. Nginx no enviará solicitudes a un servidor marcado como "down", incluso si todos los demás servidores están marcados como no disponibles. Por ejemplo, down; marca un servidor como no disponible.

```
upstream backend {
    server [2001:db8:123::101] weight=3;
    server [2001:db8:123::102] max_fails=2 fail_timeout=10s;
    server [2001:db8:123::103] backup;
    server [2001:db8:123::104] down;
}
```

Chequear configuración y reiniciar el server para que tome los cambios

```
# nginx -t
# systemctl restart nginx
```

Conclusión

Implementar un balanceo de carga y failover utilizando un servidor Nginx en el borde y una granja de servidores web IPv6 Only te brinda una arquitectura escalable y robusta. Puedes distribuir eficientemente el tráfico entrante entre tus servidores web y garantizar la alta disponibilidad de tus servicios.

Github con los archivos de configuración utilizados

https://github.com/LACNIC/BlogPostHelpFiles/tree/main/2023_07_Balanceo_Carga_y_Failover_NGINX_IPv6

Referencias

https://help.clouding.io/hc/es/articles/360019908839-C%C3%B3mo-configurar-un-servidor-de-balanceo-de-carga-Nginx-en-Ubuntu-20-04
https://cloud.google.com/load-balancing/docs/https
https://stackoverflow.com/questions/69285690/nginx-load-balancer-configuration-not-working