¿QUÉ ES NGINX?

Es un servidor web de código abierto que, desde su éxito inicial como servidor web, ahora también es usado como proxy inverso, cache de HTTP, y balanceador de carga.

Además de otras tareas, los servidores web son los encargados de la entrega de aplicaciones web, respondiendo a peticiones HTTPS realizadas por usuarios, normalmente desde un navegador web.

¿CÓMO FUNCIONA NGINX?

Nginx está diseñado para ofrecer un bajo uso de memoria y alta concurrencia. En lugar de crear nuevos procesos para cada solicitud web, Nginx usa un enfoque asincrónico basado en eventos donde las solicitudes se manejan en un solo hilo.

Con Nginx, un proceso maestro puede controlar múltiples procesos de trabajo. El proceso maestro mantiene los procesos de trabajo, y son estos lo que hacen el procesamiento real.

Algunas características comunes que se ven en Nginx incluyen:

Proxy inverso con caché

IPv6

Balanceo de carga

Soporte FastCGI con almacenamiento en caché

Websockets
Manejo de archivos estáticos, archivos de índice y auto indexación

TLS / SSL con SIN

Ventajas del servidor Nginx

° **Rendimiento superior**
° **Consumo de recursos bajos**
° **Escalabilidad fácil**
° **Balanceó de carga y alta disponibilidad**
° **Proxy inverso**
° **Servir contenido estático eficientemente**
° **Configuración flexible**
° **Compatibilidad con muchos módulos y extensiones** ° **Gran comunidad de soporte**

Desventajas del servidor Nginx

° **Menor cantidad de herramientas de gestión**
° **Curva de aprendizaje pronunciada**
° **Menos soporte para tecnologías específicas**
° **Limitaciones en el manejo de archivos grandes**
° **Menor adopción en comparación con otros servidores web**
° **Configuración de SSL más complicada**

Ejemplo 1:

Configuración de front-end

Para la configuración de front-end de NGINX, el usuario debe configurar los servidores, que incluyen las reglas de puerto y distribución.

- `listen`

  Equivalente a HAProxy `bind`, el puerto externo con el que se comunicará la aplicación.

- location

  Reglas de distribución, basadas en `path`, igual que en HAProxy.

  - Todo el tráfico va a la ubicación / por defecto y pasa al grupo `thingworx` ascendente.
  - La coincidencia de ubicaciones de Connection Server utiliza un regex en lugar de mostrar cada ruta. Estas se transferirán al grupo `cxserver` ascendente.

Por ejemplo:

```
server {
 listen 80;
 # connection server paths
 location ~ ^/Thingworx/(WS|WSTunnelClient|WSTunnelServer|TWS)\.* {
   proxy_set_header X-Real-IP $remote_addr;
   proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
   proxy_set_header Host $http_host;
   proxy_set_header Upgrade $http_upgrade;
   proxy_set_header Connection "Upgrade";
   proxy_pass http://cxserver;
 }
 # everything else to thingworx
 location / {
   proxy_set_header X-Real-IP $remote_addr;
   proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
   proxy_set_header Host $http_host;
   proxy_pass http://thingworx;
 }
}
```

Ejemplo 2:

Configuración de back-end de plataforma

En esta sección se proporcionan la lista de servidores que pueden gestionar solicitudes y el algoritmo de equilibrio de la carga.

- `ip_hash`

  Permite hacer que la solicitud se pueda recordar en un servidor en función de la dirección IP. Las sesiones que se pueden recordar son una función de NGINX Plus.
- `server`

  Hay entradas para cada servidor de la lista.
- No se han configurado verificaciones de estado, pero si falla un servidor, el tráfico se redirecciona. Las comprobaciones de estado son una función de NGINX Plus.

Por ejemplo:

```
upstream thingworx {
 ip_hash;
 server platform1:80;
 server platform2:80;
}
```

Configuración de back-end de Connection Server

- `ip_hash`

  Permite hacer que la solicitud se pueda recordar en un servidor en función de la dirección IP. Esto es equivalente a `source` en HAProxy.
- servidor

  Hay entradas para cada servidor de la lista.
- No se han configurado verificaciones de estado, pero si falla un servidor, el tráfico se redirecciona. Las comprobaciones de estado son una función de NGINX Plus.

Por ejemplo:

```
upstream cxserver {
 ip_hash;
 server cxserver1:80;
 server cxserver2:80;
}
```

Ejemplo 3: