How to load balance your servers using Nginx Proxy Manager and Cloudflare

Link: https://silicon.blog/2023/05/17/how-to-load-balance-your-servers-using-nginx-proxy-manager-and-cloudflare/

In the previous posts, you have learned how to self-hosted a WordPress Docker container, reverse proxy it with Nginx Proxy Manager and configure a Cloudflare Tunnel to access it.

This article will modify the docker-compose.yml to host 2 more WordPress Docker containers first. After that, you will learn how to load balance it with Nginx Proxy Manager.

If you have followed my previous tutorials. Your docker-compose.yml should look like this:

```
version: '3'
services:
  app:
    image: 'jc21/nginx-proxy-manager:2.9.18'
    hostname: npm
    container_name: npm
    restart: unless-stopped
    ports:
      - '81:81'
      - '443:443'
    volumes:
      - ./data:/data
    networks:
      - npm



  mysql:
    image: mysql:8.0
```

```
    hostname: mysql
    container_name: mysql
    env_file: .env
    environment:
      MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
      MYSQL_DATABASE: ${MYSQL_DATABASE}
      MYSQL_USER: ${MYSQL_USER}
      MYSQL_PASSWORD: ${MYSQL_PASSWORD}
    volumes:
       - mysql:/var/lib/mysql
    networks:
      - npm


  wordpress:
    image: wordpress:6.2-php8.0-apache
    hostname: wordpress-1
    container_name: wordpress-1
    ports:
      - 8080:80
    environment:
      WORDPRESS_DB_HOST: mysql
      WORDPRESS_DB_USER: ${MYSQL_USER}
      WORDPRESS_DB_PASSWORD: ${MYSQL_PASSWORD}
      WORDPRESS_DB_NAME: ${MYSQL_DATABASE}
    networks:
      - npm


  tunnel:
    image: cloudflare/cloudflared
    hostname: cloudflared
    container_name: cloudflared
    restart: unless-stopped
    command: tunnel run
    environment:
      TUNNEL_TOKEN: ${CLOUDFLARE_TOKEN}
    networks:
      - npm
```

```
volumes:
  mysql:



networks:
  npm:
    name: npm_network
```

The tunnel container is optional. You can comment on it if you decide not to use Cloudflare Tunnel. Your .env should look like this:

```
MYSQL_ROOT_PASSWORD="your_mysql_root_password"
MYSQL_USER="your_mysql_user"
MYSQL_PASSWORD="your_mysql_user_password"
MYSQL_DATABASE="your_wordpress_db1"
CLOUDFLARE_TOKEN="xxx"
```

The CLOUDFLARE_TOKEN is optional. You can remove that line if you decide not to use Cloudflare Tunnel.

Step 1: Start your docker containers if they are not running.

```
sudo docker compose up -d
```

Step 2: Copy the required files to your project directory.

```
sudo docker cp npm:app/templates templates
sudo docker cp npm:etc/nginx/conf.d conf.d
```

Step 3: Update your docker-compose.yml using the sample below. You are going to create 2 more WordPress containers. Comment on the tunnel container if you will not use Cloudflare Tunnel.

```
version: '3'
services:
  app:
    image: 'jc21/nginx-proxy-manager:2.9.18'
    hostname: npm
    container_name: npm
    restart: unless-stopped
    ports:
      - '81:81'
```

```
    - '443:443'
  volumes:
    - ./templates:/app/templates
    - ./conf.d:/etc/nginx/conf.d
    - ./data:/data
    - ./letsencrypt:/etc/letsencrypt #optional
  networks:
    - npm


mysql:
  image: mysql:8.0
  hostname: mysql
  container_name: mysql
  env_file: .env
  environment:
    MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
    MYSQL_DATABASE: ${MYSQL_DATABASE}
    MYSQL_USER: ${MYSQL_USER}
    MYSQL_PASSWORD: ${MYSQL_PASSWORD}
  volumes:
     - mysql:/var/lib/mysql
  networks:
    - npm


wordpress:
  image: wordpress:6.2-php8.0-apache
  hostname: wordpress-1
  container_name: wordpress-1
  ports:
    - 8080:80
  environment:
    WORDPRESS_DB_HOST: mysql
    WORDPRESS_DB_USER: ${MYSQL_USER}
    WORDPRESS_DB_PASSWORD: ${MYSQL_PASSWORD}
    WORDPRESS_DB_NAME: ${MYSQL_DATABASE}
  networks:
    - npm
```

```
mysql2:
  image: mysql:8.0
  hostname: mysql2
  container_name: mysql2
  env_file: .env
  environment:
    MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
    MYSQL_DATABASE: ${MYSQL_DATABASE2}
    MYSQL_USER: ${MYSQL_USER2}
    MYSQL_PASSWORD: ${MYSQL_PASSWORD2}
  volumes:
     - mysql2:/var/lib/mysql
  networks:
    - npm


wordpress2:
  image: wordpress:6.2-php8.0-apache
  hostname: wordpress-2
  container_name: wordpress-2
  ports:
    - 8081:80
  environment:
    WORDPRESS_DB_HOST: mysql2
    WORDPRESS_DB_USER: ${MYSQL_USER2}
    WORDPRESS_DB_PASSWORD: ${MYSQL_PASSWORD2}
    WORDPRESS_DB_NAME: ${MYSQL_DATABASE2}
  networks:
    - npm


mysql3:
  image: mysql:8.0
  hostname: mysql3
  container_name: mysql3
  env_file: .env
  environment:
    MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
    MYSQL_DATABASE: ${MYSQL_DATABASE3}
```

```
      MYSQL_USER: ${MYSQL_USER3}
      MYSQL_PASSWORD: ${MYSQL_PASSWORD3}
    volumes:
       - mysql3:/var/lib/mysql
    networks:
      - npm


  wordpress3:
    image: wordpress:6.2-php8.0-apache
    hostname: wordpress-3
    container_name: wordpress-3
    ports:
      - 8082:80
    environment:
      WORDPRESS_DB_HOST: mysql3
      WORDPRESS_DB_USER: ${MYSQL_USER3}
      WORDPRESS_DB_PASSWORD: ${MYSQL_PASSWORD3}
      WORDPRESS_DB_NAME: ${MYSQL_DATABASE3}
    networks:
      - npm


  tunnel:
    image: cloudflare/cloudflared
    hostname: cloudflared
    container_name: cloudflared
    restart: unless-stopped
    command: tunnel run
    environment:
      TUNNEL_TOKEN: ${CLOUDFLARE_TOKEN}
    networks:
      - npm

volumes:
  mysql:
  mysql2:
  mysql3:
```

```
networks:
  npm:
    name: npm_network
```

Step 4: Edit the .env file as you will host 2 more WordPress containers. Remove the CLOUDFLARE_TOKEN if you are not going to use Cloudflare Tunnel.

```
sudo nano .env
MYSQL_ROOT_PASSWORD="your_mysql_root_password"
MYSQL_USER="your_mysql_user"
MYSQL_PASSWORD="your_mysql_user_password"
MYSQL_DATABASE="your_wordpress_db1"
MYSQL_USER2="your_mysql_user2"
MYSQL_PASSWORD2="your_mysql_user_password2"
MYSQL_DATABASE2="your_wordpress_db2"
MYSQL_USER3="your_mysql_user3"
MYSQL_PASSWORD3="your_mysql_user_password3"
MYSQL_DATABASE3="your_wordpress_db3"
CLOUDFLARE_TOKEN="xxx"
```

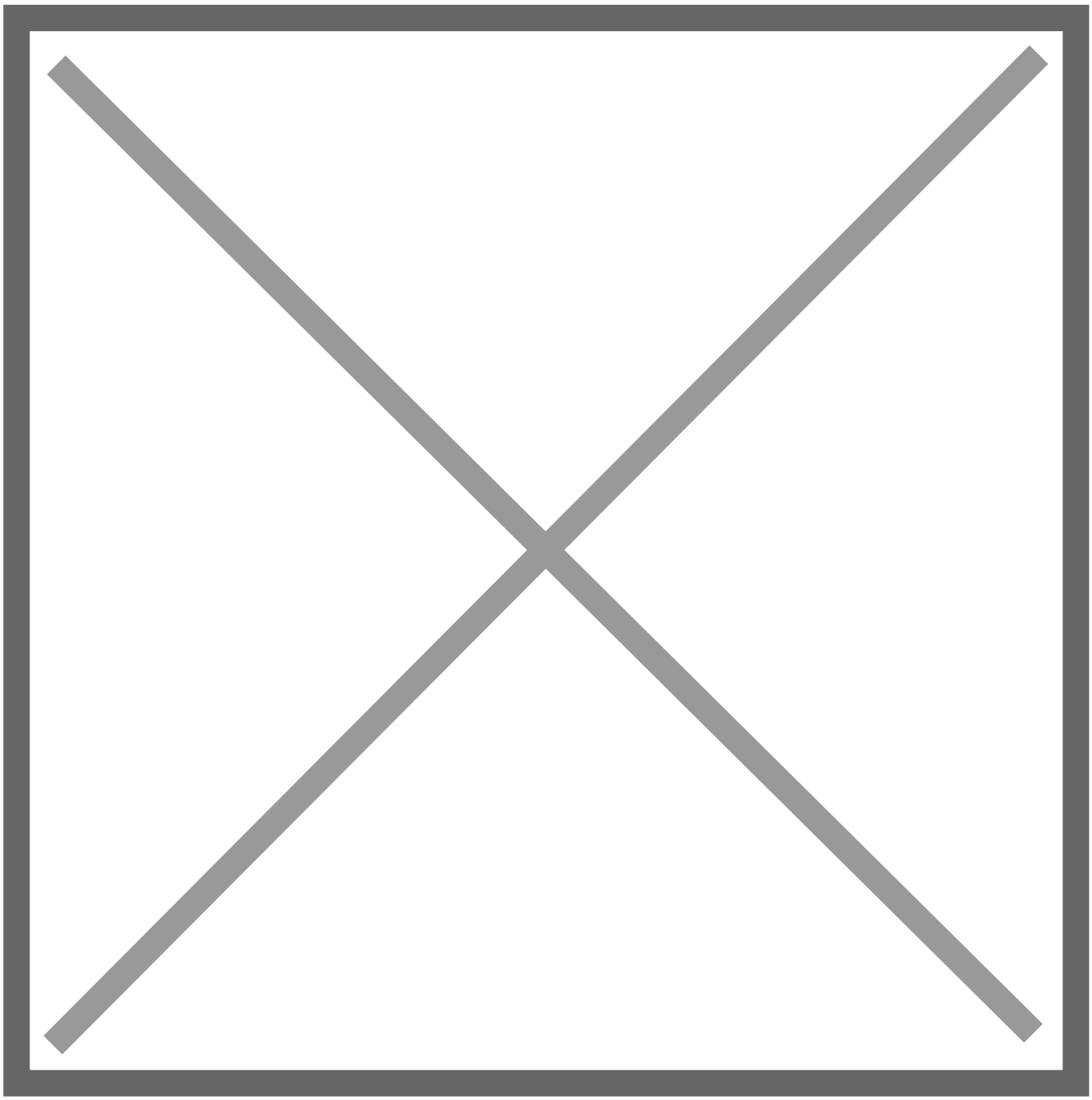
Ctrl + X to save the file.

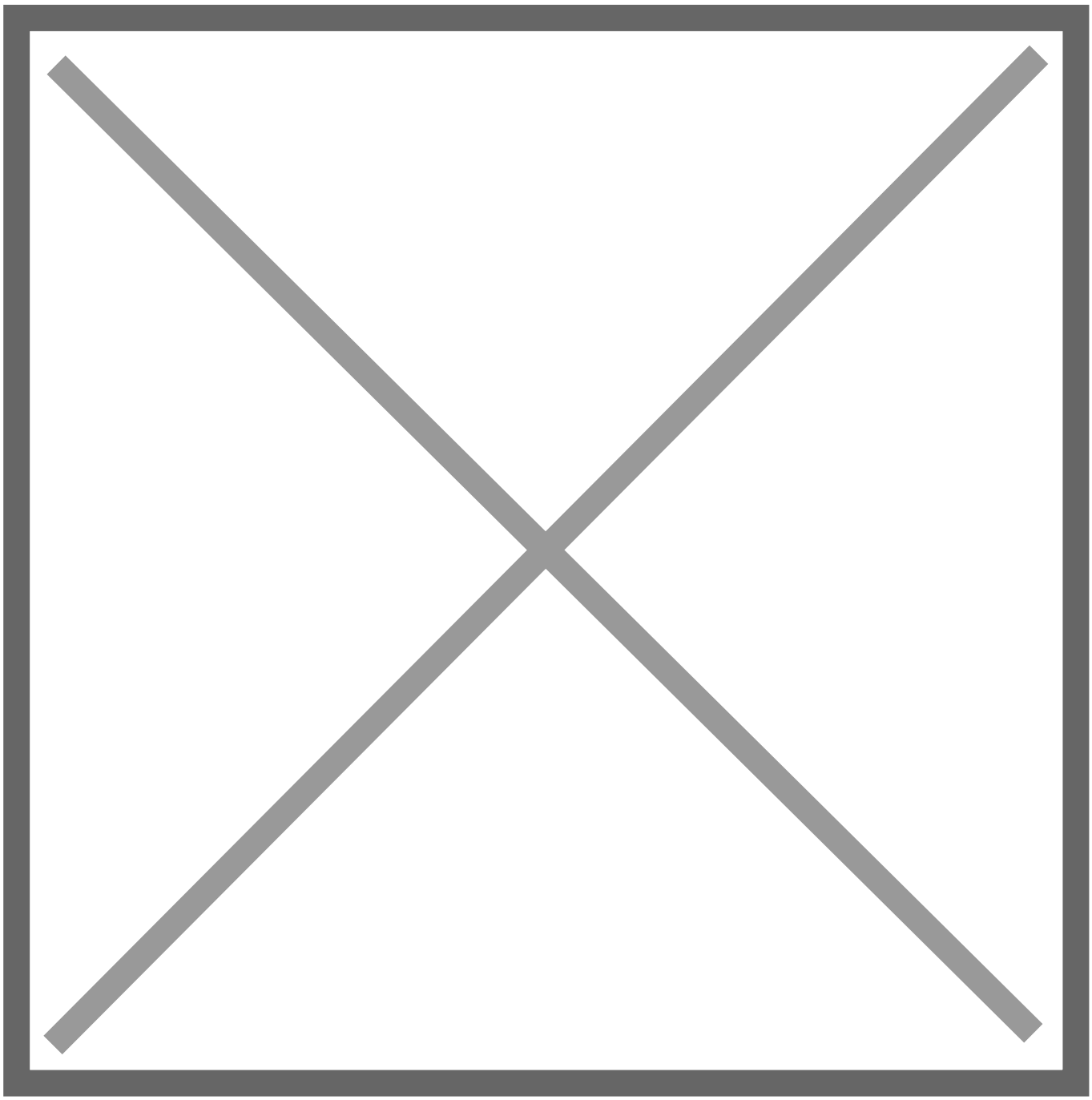
Step 5: Rebuild the docker containers by

```
sudo docker compose up -d
```

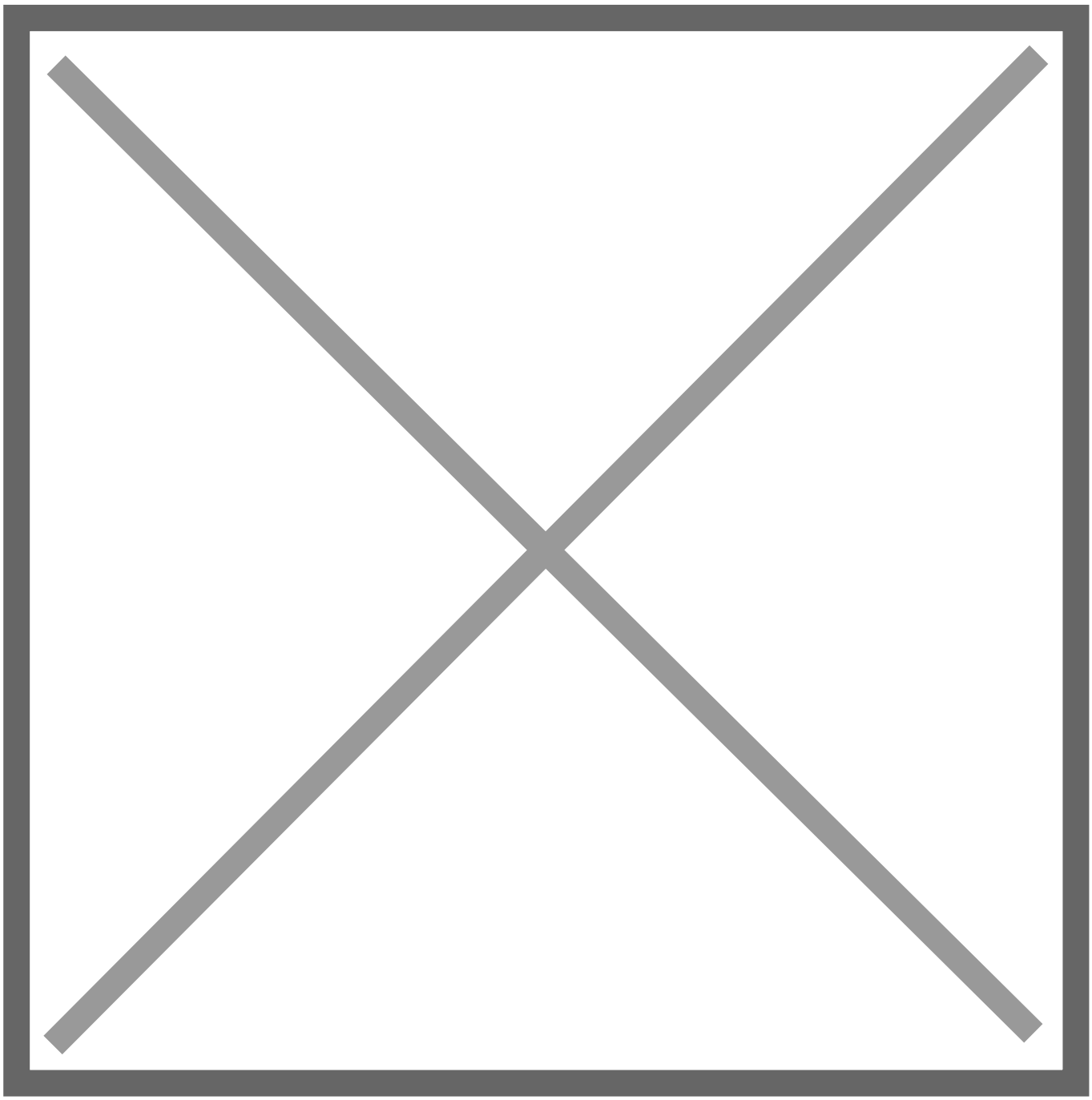
Step 6: Go to WordPress 2 container and WordPress 3 container to install WordPress. Use test2 and test3 as the Site Title.

```
http://your_ip:8081/
http://your_ip:8082/
```





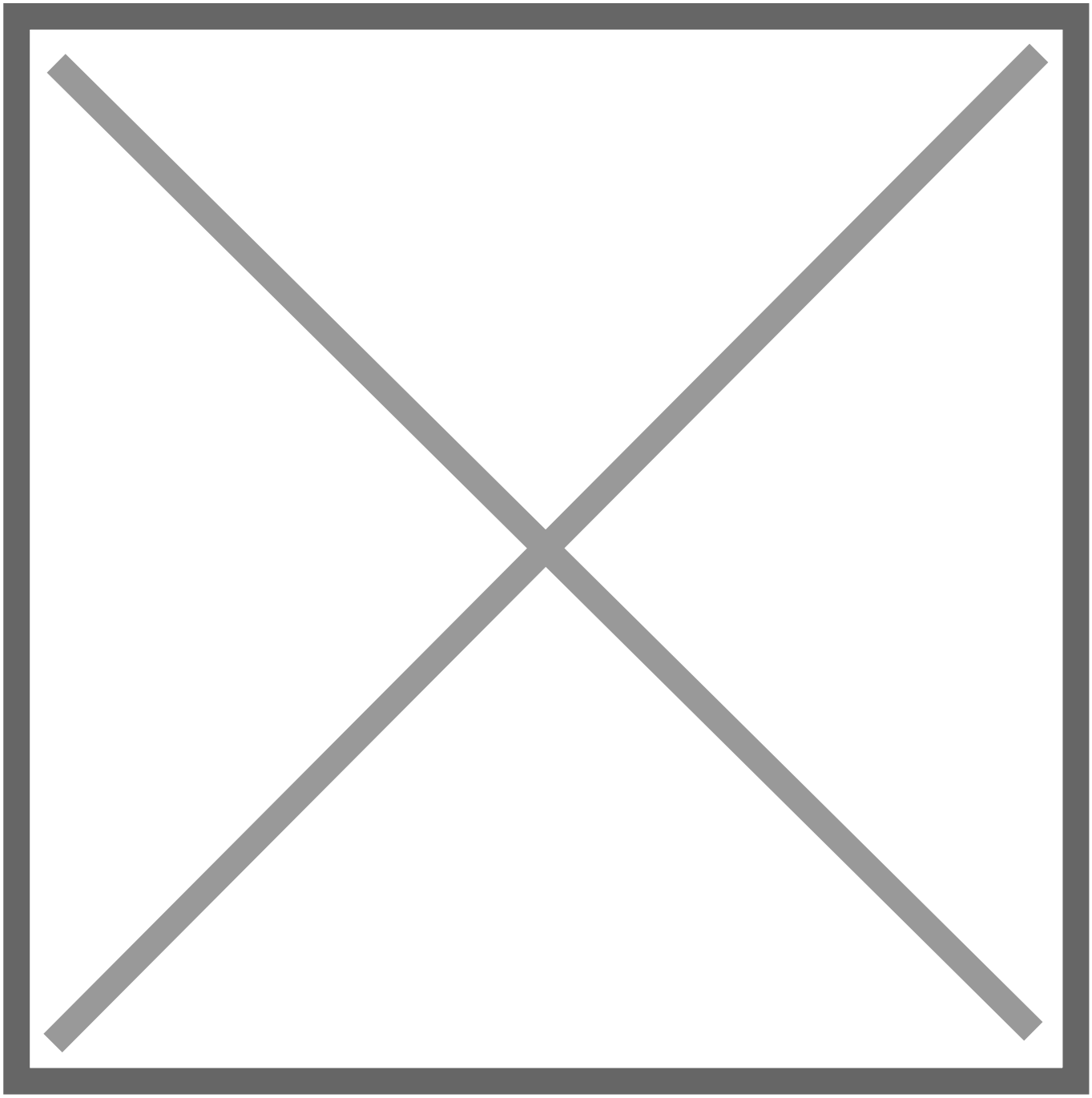
Step 7: After installation, go to the Settings page, and change the WordPress Address (URL) and Site Address (URL) with your WordPress domain name. In my case, it is https://test.silicon.blog.



It is normal if your browser returns ‘your ip sent an invalid response’ errors.

Everything will be fine after configuring the Nginx Proxy Manager.

Step 8: Modify the proxy_host.conf of the Nginx Proxy Manager docker container. sudo nano ./templates/proxy_host.conf At the top {% if enabled %}, add the following lines

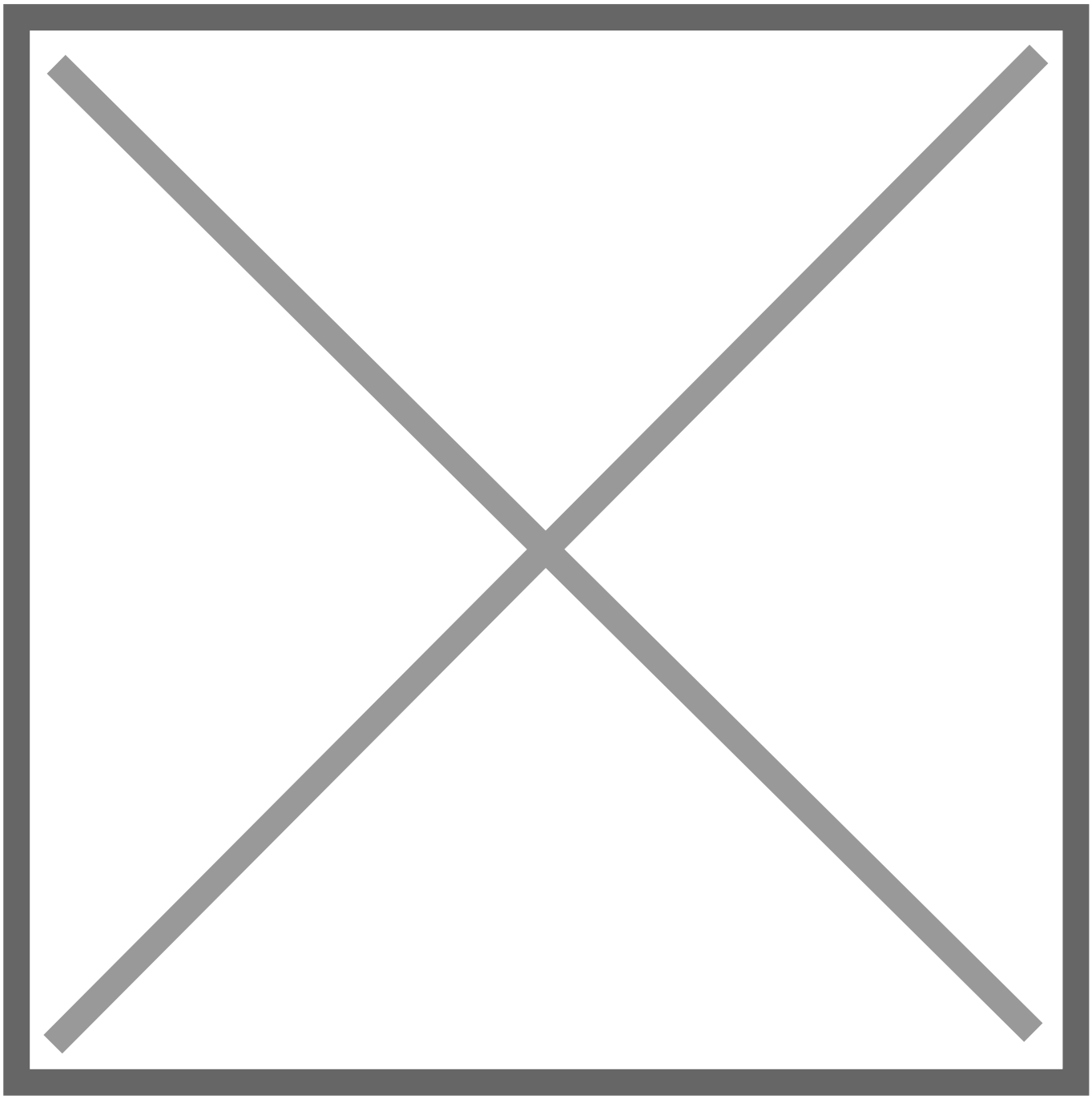


```
# Custom%
upstream lbtest{{ id }}{
   include /data/nginx/custom/load_balancer{{ id }}.conf;
   keepalive 200;
   keepalive_timeout 120s;
}
```

Ctrl + X to save the file.

Step 9: Edit proxy.conf and comment out everything. Those headers will be added manually later.

```
sudo nano ./conf.d/include/proxy.conf
```



Ctrl + X to save the file.

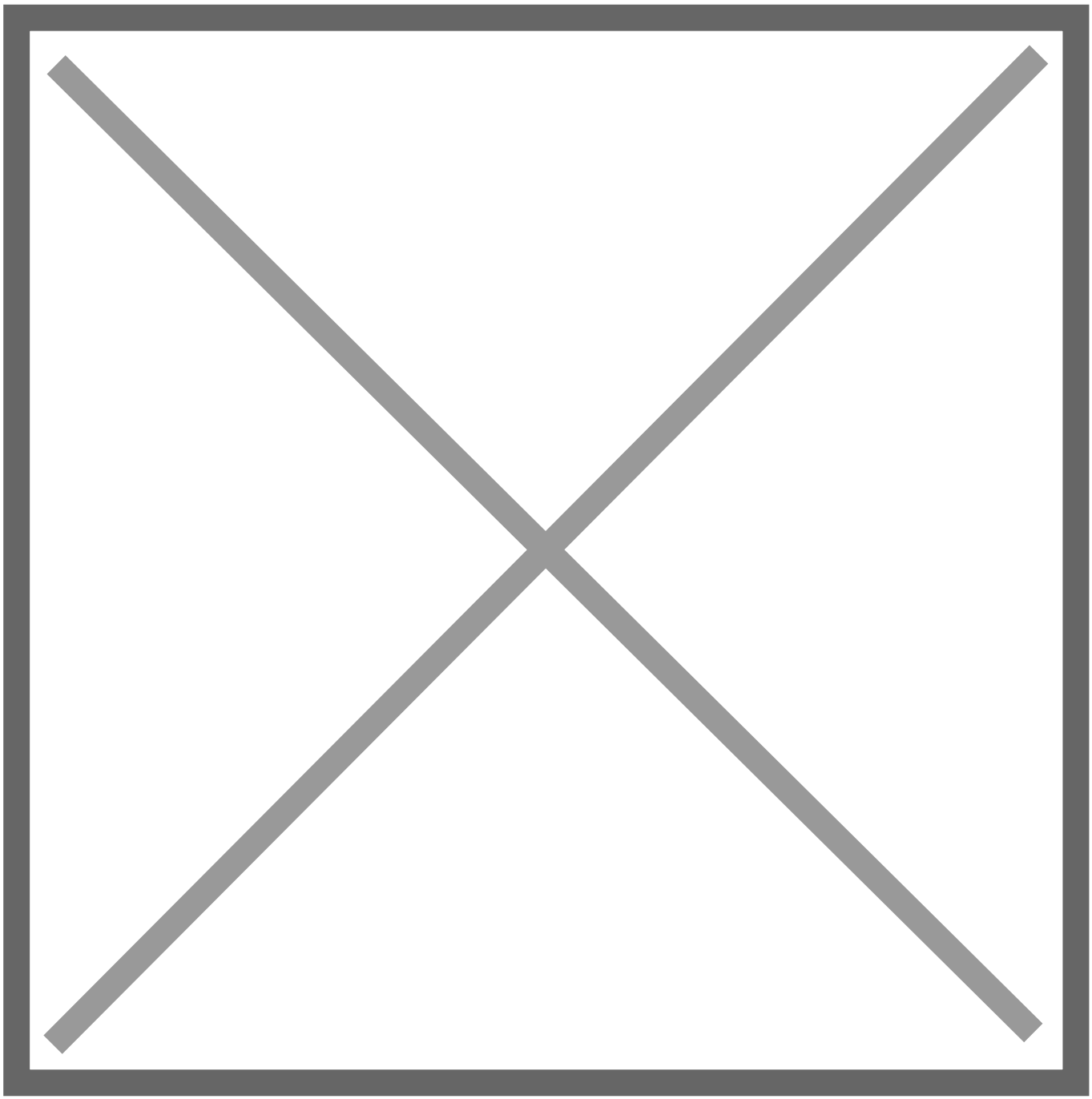
Step 10: Since you will enable load balancing in Nginx Proxy Manager in a hacky way, the load balancer configuration file will not be generated automatically when we click save.

We have to add and edit the load balancer configuration manually. It may crash if Nginx Proxy Manager cannot find the related load_balancerX.conf or the load_balancerX.conf is empty at the start. Therefore, you need to create 10 load balancer configuration files and fill contents to them for later use by

```
sudo touch ./data/nginx/custom/load_balancer{1..10}.conf
echo "server 127.0.0.1 weight=1;" | sudo tee ./data/nginx/custom/load_balancer{1..10}.conf
1>/dev/null
```

You can generate as many load balancer configuration files as you want, but the Nginx Proxy Manager will take a long time to load if you create more than 100 load balancer configuration files.

Step 11: On the Nginx Proxy Manager dashboard, find out the number of your Proxy host.



In my case, it is proxy host 3.

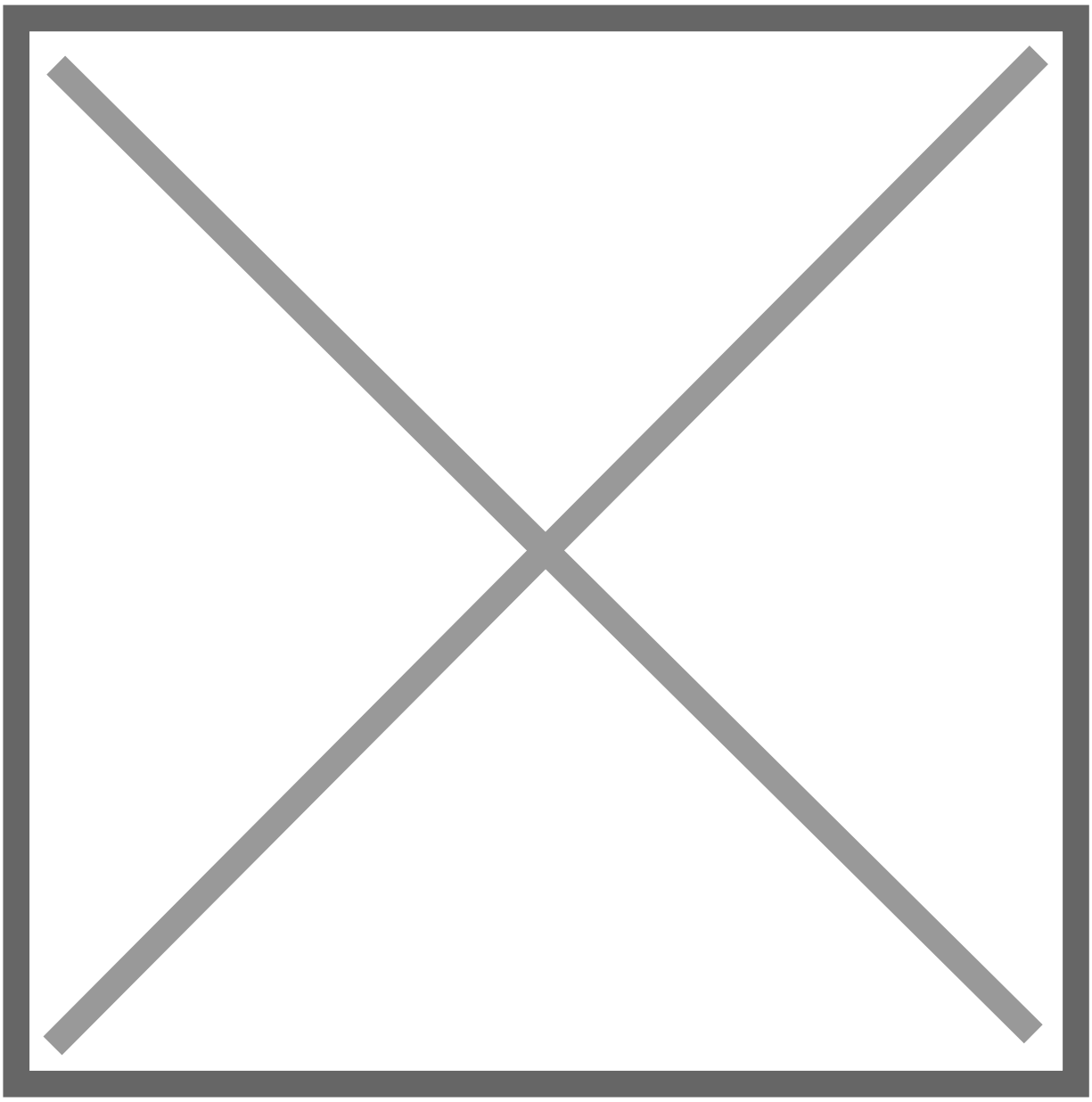
Step 12: Edit the load balancer configuration file matching the proxy host number. Replace X with your proxy host number. In my case, it is load_balancer3.conf.

```
sudo mkdir ./data/nginx/custom/
sudo nano ./data/nginx/custom/load_balancerX.conf
```

Add the WordPress server to the load balancer.

```
server your_wordpress_server1_ip:port weight=1;
server your_wordpress_server2_ip:port weight=1;
server your_wordpress_server3_ip:port weight=1;
```

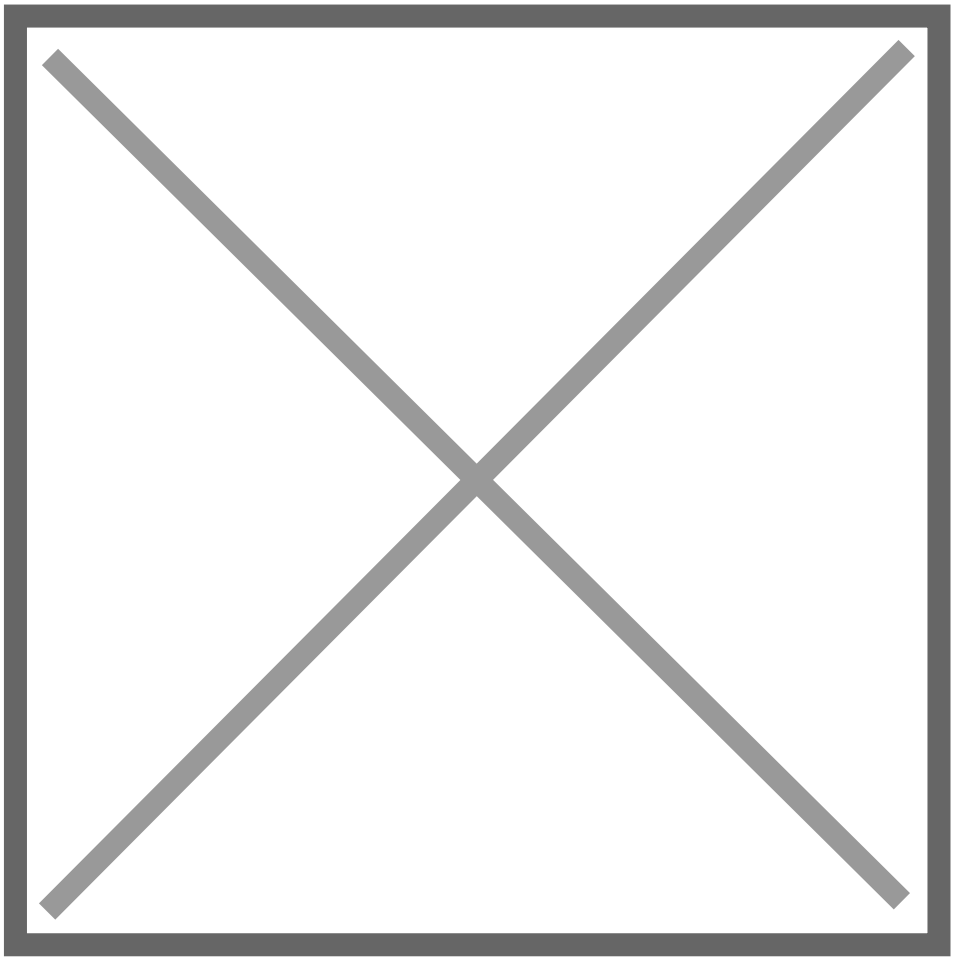
In my case, it is



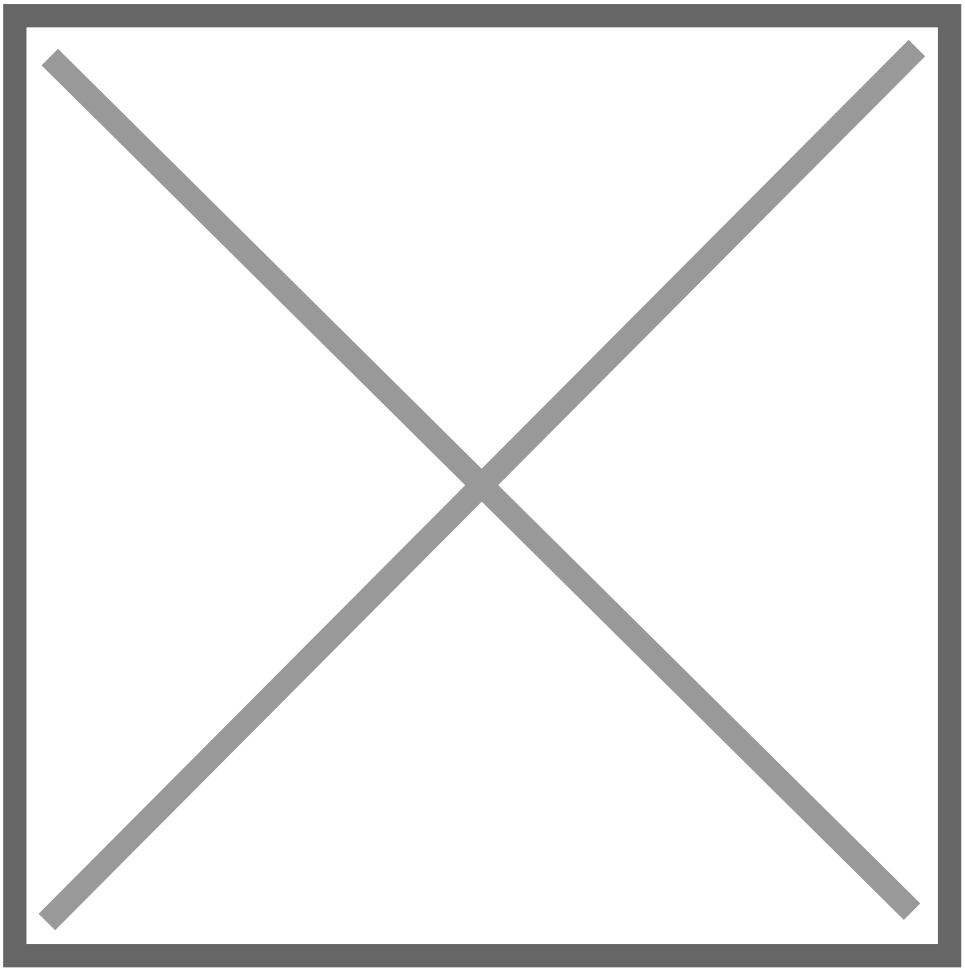
```
server wordpress-1 weight=1;
server wordpress-2 weight=1;
server wordpress-3 weight=1;
```

Ctrl + X to save the file.

Step 13: Edit your proxy host. Change the Forward Hostname / IP to loadbalancer. The Forward Port does not matter. The scheme will be HTTP in this article, and you can change it to HTTPS later.



Step 14: Remember to add an SSL certificate to your site in the SSL section. HTTP/2 Support is optional.

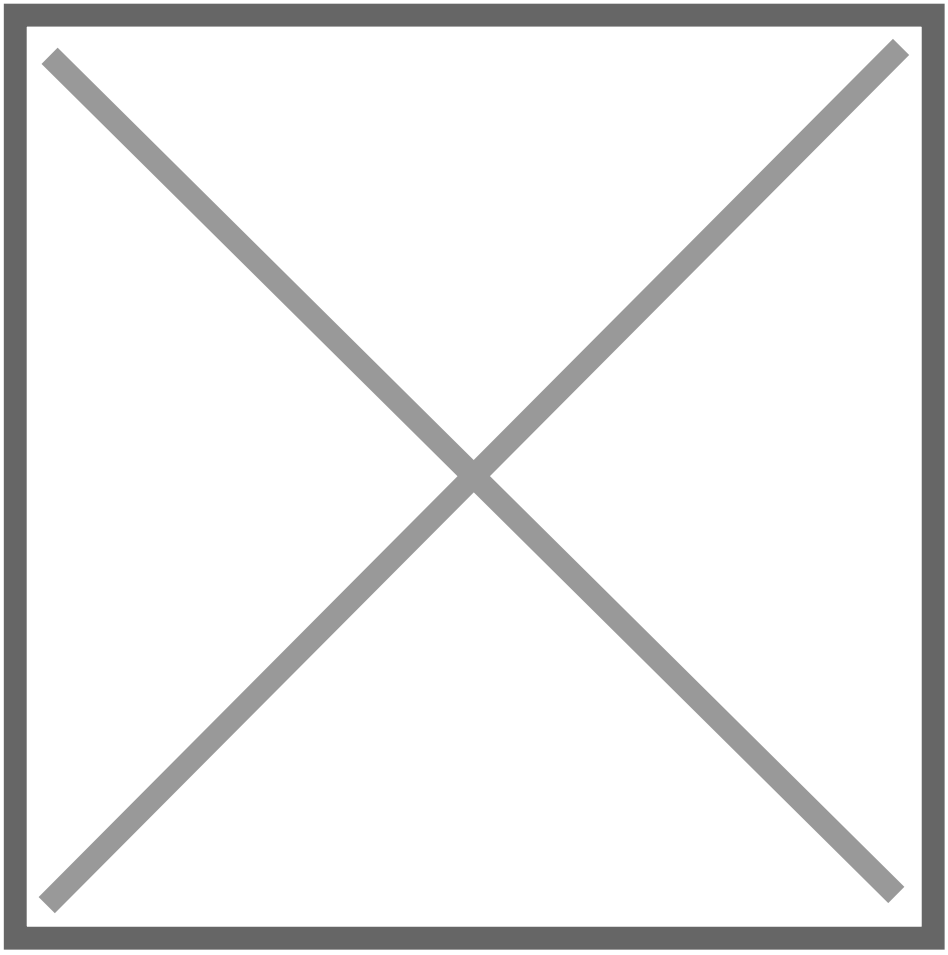


Step 15: Go to the Advanced page, and add the following script.

Replace lbtestX with your proxy host number. In my case, it is lbtest3.

If needed, you should return to step 10 and manually create more load balancer files.

Your Nginx Proxy Manager may crash at the start if it cannot find the corresponding load balancer file.



```
location / {
proxy_set_header Host $host;
proxy_set_header X-Forwarded-Host $server_name;
proxy_connect_timeout   15s;
proxy_read_timeout  15s;
proxy_next_upstream error timeout http_500 http_502 http_503 http_504 non_idempotent;
proxy_set_header    Connection  "";

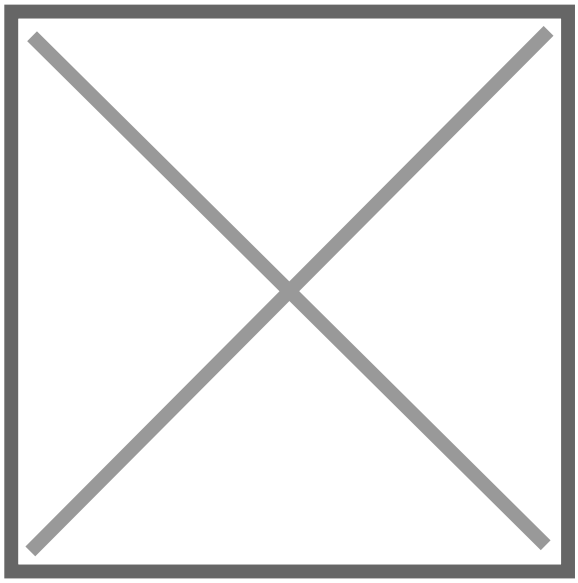
if ($server != "loadbalancer"){
  proxy_pass $forward_scheme://$server:$port$request_uri;
}
if ($server = "loadbalancer"){
  proxy_pass $forward_scheme://lbtestX$request_uri;
}
}
```

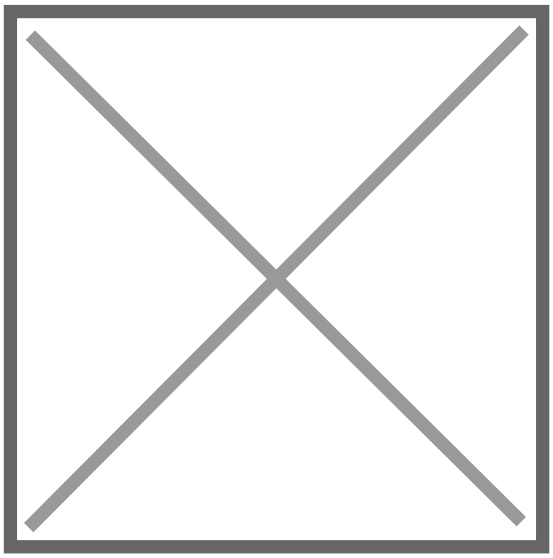
In my case, it is

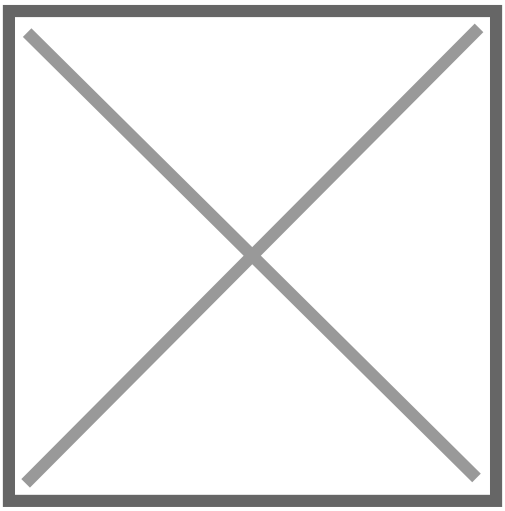
```
location / {
proxy_set_header Host $host;
proxy_set_header X-Forwarded-Host $server_name;
proxy_connect_timeout   15s;
proxy_read_timeout  15s;
proxy_next_upstream error timeout http_500 http_502 http_503 http_504 non_idempotent;
proxy_set_header    Connection  "";

if ($server != "loadbalancer"){
  proxy_pass $forward_scheme://$server:$port$request_uri;
}
if ($server = "loadbalancer"){
  proxy_pass $forward_scheme://lbtest3$request_uri;
}
}
```

Try to access your site a few times. The site title should change randomly if everything works properly.







Congratulation if everything runs smoothly on your side.

The following article will teach you how to create a failover site using Nginx Proxy Manager and cPanel (or other web hosting platforms).

Check out this article if you want to enable HTTPS (install SSL certificate on your Apache server) on your WordPress docker container.

I am not using it in a large-scale high-traffic production site. Take risks if you use this method to load balance your web servers.

Feel free to comment on any potential security issues with the proxy header. I am not familiar with it.

Revision #1
Created 2024-07-06 11:29:27 UTC by Administrador
Updated 2024-07-06 11:34:47 UTC by Administrador