Step by step procedure to demonstrate load balancing using Nginx

Abstract

The following document demonstrates how to perform load balancing using nginx for two containers deployed using docker.

The document explains the step by step procedure to install necessary software, creating docker images and containers, mapping these containers to host ports and configure nginx to perform load balancing.

Table of Contents

Installing Nginx

Step 1 : Update local package index

$ sudo apt-get update

Step 2 : Use the apt package management suite to complete the installation.

$ sudo apt-get install nginx

On Ubuntu 16.04, Nginx is configured to start running upon installation.

Step 3 : Allow firewall for nginx connections

$ sudo ufw allow 'Nginx HTTP'

Verify the change using

$ **Sudo ufw status**

Step 4 :

Go to your browser and enter your host URL, it should take you to Nginx's default loading page.

Installing Docker

Step 1 : Install packages to allow apt to use a repository over HTTPS

```
$ sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    software-properties-common
```

Step 2 : Add Docker's official GPG key:

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg |
sudo apt-key add –
```

Step 3 : Use the following command to set up the stable repository

```
$ sudo add-apt-repository \
   "deb [arch=amd64]https://download.docker.com/linux/ubuntu\
   $(lsb_release -cs) \
   stable"
```

Step 4 : Install the latest version of Docker CE, or install a specific version

```
$ sudo apt-get install docker-ce
        Or
$ sudo apt-get install docker-ce=<VERSION>
```

Step 5 : Verify that Docker CE is installed correctly by running the hello-world image.

$ sudo docker run hello-world

Creating Tomcat servers using Docker

Step 1 : Pull the docker image for tomcat 8.5 (You can choose a version of your choice) from Docker Hub

$ sudo docker pull tomcat:8.5

verify if the image has been downloaded

$ sudo docker images

Step 2 : Create two containers from the image downloaded previously

$ sudo docker run -d -p 8082:8080 --name tomcat1 tomcat

$ sudo docker run -d -p 8083:8080 --name tomcat2 tomcat

Notice the tag –p 8082:8080, this maps the containers 8080 port to the 8082 port of the host. Make sure the host ports (8082 and 8083) is free and if any service is running then stop it.

Verify if the containers are up and running

$ sudo docker ps

Step 3 : Configure firewall to allow the ports

$ sudo ufw allow 8082

$ sudo ufw allow 8083

Configure Nginx to perform Load Balancing

Step 1 : Go to the nginx directory

$ cd /etc/nginx/sites-enabled

Step 2 : Create a file with relevant name to configure nginx and add the following lines of code

```
upstream backend {
server <Server URL>:8082;
server <Server URL>:8083;
}

server {
listen 80;
server_name <Server URL>;
location / {
proxy_pass http://backend;
}
}
```

With the above lines of code we group the host ports under upstream backend. Every time the host receives the request from browser our nginx server redirects it to one of the containers.

You can verify this by going to your browser and typing

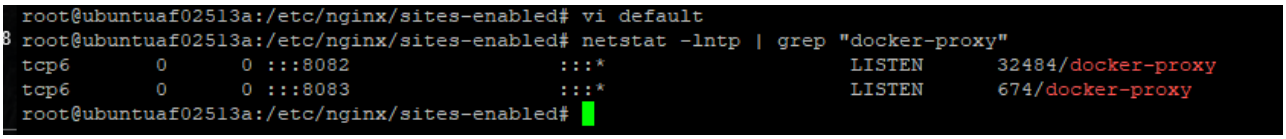
http://<Server URL>/

You will see the tomcat home page displayed

To verify whether nginx was able to map both the containers, execute the following command :

$ netstat -lntp | grep "docker-proxy"

You should be able to see something like this :



As we can see both the ports configured in the nginx file are up and running.

Test the Load Balancing

Step 1 : Stop one of the docker containers.

$ sudo docker ps

This will give all running docker containers. Copy the CONTAINER_ID of the server you want to stop

$ sudo docket stop < CONTAINER_ID >

Verify that the container is not running by running the command

$ sudo docker ps

Step 2 : Try to access tomcat

Go to your browser and typing

http://<Server URL>/

You will see the tomcat home page displayed. Nginx automatically redirects to the container which is up and running instead of the server which we stopped earlier.

You can also check the same by running the command

$ netstat -lntp | grep "docker-proxy"

You may notice that only one docker-proxy is running.