Loadbalancer with nginx

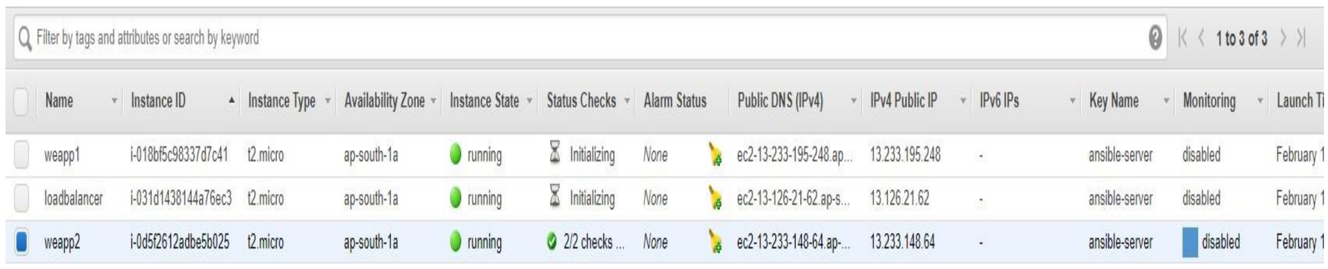
Prerequisites:

To achieve this we will take 3 ubuntu instances in aws.

In one machine we will install nginx for load balancer.

Another two machines we will deploy one static web page using nginx.

First we will install docker in 2 and 3 machines.



Filter by tags and attributes or search by keyword

1 to 3 of 3

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP	IPv6 IPs	Key Name	Monitoring	Launch Ti
weapp1	i-018bf5c98337d7c41	t2.micro	ap-south-1a	running	Initializing	None	ec2-13-233-195-248.ap...	13.233.195.248	-	ansible-server	disabled	February 1
loadbalancer	i-031d1438144a76ec3	t2.micro	ap-south-1a	running	Initializing	None	ec2-13-126-21-62.ap-s...	13.126.21.62	-	ansible-server	disabled	February 1
weapp2	i-0d5f2612adbe5b025	t2.micro	ap-south-1a	running	2/2 checks ...	None	ec2-13-233-148-64.ap-...	13.233.148.64	-	ansible-server	disabled	February 1

Here webapp1 and webapp2 is running our application instances.

Loadbalancer is running with nginx load balancer.

First we will install docker on webapp1 and awebapp2 instances (as of now I am installing manually. We will automate this step using ansible to install docker).

sudo apt-get update

sudo apt-get upgrade -y

sudo apt-get install docker.io –y

Now I will write a small static page using html

sudo mkdir -p /data/www

Create a index.html under /data/www path

cd /data/www

sudo vi index.html

<h1>Hello 1</h1>

And save it

Crate a html page in weapp2 as same a above and change the response content for testing purpose.

sudo mkdir -p /data/www

Create a index.html under /data/www path

cd /data/www

sudo vi index.html

<h1>Hello 2</h1>

And save it

Deploy this static page in nginx webserver

sudo docker run --name nginx -v /data/www:/usr/share/nginx/html:ro -d -p 80:80 nginx

for testing access with http://<hostip>

now let's configure loadbalncer

sudo apt-get update

sudo apt-get upgrade -y

sudo apt-get install nginx -y

Now open the /etc/nginx/nginx.conf file add your server details.

sudo vi /etc/nginx/nginx.conf

Back up the configuration file for Nginx, called nginx.conf. Then, delete the contents of the current file, and replace them with the code shown below:

sudo cp nginx.conf nginx_back.conf

remove the content in nginx.conf file (All line at a time delete :1,$d in vi editor)

```
ubuntu@ip-172-31-20-76:/etc/nginx$ cat nginx.conf

http {

        upstream backend {

                server ec2-13-233-195-248.ap-south-1.compute.amazonaws.com;

                server ec2-13-233-148-64.ap-south-1.compute.amazonaws.com;

        }

        server {

                listen 80;

                location / {

                        proxy_pass http://backend;

                }

        }

}
```

The upstream back end, which can have any name, sets up the group of servers that perform the load balancing. The next two lines of code define the servers, but it's also possible to use domain names, if desired.

The second group of commands specifies that the default port 80 will accept requests.

The next two lines essentially implement the proxy service. The location / forwards all traffic for that website or IP address to the Nginx load balancer. The proxy pass defines the group of servers to use, which are those configured in the previous group of commands.

Next, use the following command to restart the Nginx load balancer and make the service live:

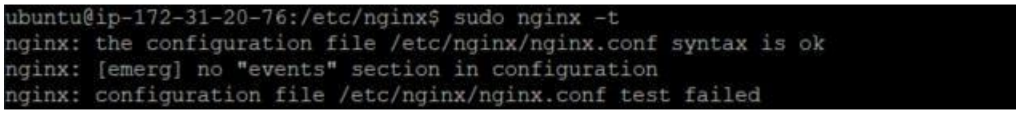
```
sudo systemctl reload nginx
```

if you see below error

Feb 18 13:50:14 ip-172-31-20-76 systemd[1]: Reload failed for A high performance

To check the syntax error use below command

sudo nginx –t



```
ubuntu@ip-172-31-20-76:/etc/nginx$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: [emerg] no "events" section in configuration
nginx: configuration file /etc/nginx/nginx.conf test failed
```

Added event section in nginx.conf

ubuntu@ip-172-31-20-76:/etc/nginx$ cat nginx.conf

```
events {
        worker_connections 768;
        # multi_accept on;
}


http {
        upstream backend {
                server ec2-13-233-195-248.ap-south-1.compute.amazonaws.com;
                server ec2-13-233-148-64.ap-south-1.compute.amazonaws.com;
        }
        server {
                listen 80;
                location / {
                        proxy_pass http://backend;
                }
        }
}
```

Lastly, open a web browser, and enter the IP address or URL of the front-facing load balancer. It should display Hello 1 or Hello 2, depending on which server the load balancer passes the request to.