Nginx Configuration

Nginx

Along with presenting the static files, nginx provides the feature of Load Balancing.

Load Balancing

Load Balancing distributes incoming requests among multiple application running instances such that not any single instance is overloaded thus preventing higher latency and eventual crashes.

Steps to implement Nginx Load Balancing

1. Create and open the *default.conf* file for nginx configuration
2. Define the upstream context
    - Inside upstream define server directive with EC2 instance IP address as the value.
    - You can define several server directives depending on how many servers you want to distribute the incoming requests.
    - For this example, we will define two server directives
        - first server directive will have contianer-name with container-port instead of IPv4 address as it is running in the very same EC2 instance as nginx.
        - second server directive will have public IPv4 address of another EC2 instance with `port 80`.
    - Optionally, you can provide weight value so the request distribution will be according to weightage.
3. Define the server block
    - Inside server block, define directive to `listen` on `port 80`.
    - Define the location context with `/` so all requests will be passed.
    - Inside location context, `proxy_pass` the incoming request to Upstream context (example:- *`proxy_pass http://upstream_name`*)
4. Validation of load balancing
    - Define the *log_format* to access the incoming request distribution logs.
    - Paste this in the main context

```
log_format upstreamlog '$server_name to: $upstream_addr [$request] '
    'upstream_response_time $upstream_response_time '
    'msec $msec request_time $request_time';
```

- Paste this in the server context

```
access_log /var/log/nginx/access.log upstreamlog;
```

Nginx configuration example

```
log_format upstreamlog '$server_name to: $upstream_addr [$request] '
    'upstream_response_time $upstream_response_time '
    'msec $msec request_time $request_time';

upstream app{
    server app-service:8000 weight=1;
    server 3.110.189.99:80 weight=2;
}

server{
    listen 80;

    access_log /var/log/nginx/access.log upstreamlog;

    location / {
        proxy_pass http://app;
    }
}
```

AWS Configuration

- Log in to your AWS account.
- Search for EC2 service.
- Click on Instances.
- Click on launch instance.
- Provide the instance name, create and save the `.pem` key, and create new security group allowing the `http` requests from `0.0.0.0/0`.
- Keep everything else as default.
- Launch the instance.
- Create another instance, similar to the first instance. This instance's public IPv4 address will be used in *default.conf* file to distribute the load.
- `ssh` into the firstly created instance using following commands and replacing public-IPv4 with first instance public IPv4 address.

```
chmod 400 '.pem'
ssh -i '.pem' ec2-user@public-IPv4
```

- Run the following commands.

```
sudo yum update
vim default.conf
```

- A new editor will be opened, copy the above-created *default.conf* file content and press `Esc`, type `:wq`.

> Don't forget to mention the second instance public IPv4 address in the upstream context in *default.conf* file.

- Run the following commands.

```
sudo yum install docker -y
sudo systemctl start docker
sudo systemctl enable docker
sudo usermod -a -G docker $(whoami)
newgrp docker
docker pull nginx
docker run -d --name nginx-service nginx
docker cp ./default.conf nginx-service:/etc/nginx/conf.d/default.conf
```

- Open the browser and request multiple times to the first instance IP address on which nginx is running.
- Go to the ssh shell and run `docker logs nginx-service`.
- Logs will show the incoming requests routed to different instances thus validating the load balancing.