***Nuage Networks Test App***

This phase relates to a test app development using nodejs. This app asks for querying a database (in mysql). The app written in nodejs and the database are maintained in different servers. AWS is used for this assignment.

***Choice of Technologies:***

1. **Nodejs:** The best possible choice of technologies would be to use nodejs or do the same in a rails framework, which is equally good. But if we observe rails framework, it is a blocking server. When our assignment goes further, there is a need to use Nginx loadbalancer. If we use a blocking server in case of many requests from the user, there may be the case like running X copies of rails server , after the never returning route hit X times, it would be very slow depending on the number of requests. If it is very slow, we will increase the processing time at the load balancer.

So, nodejs is better for this Assignment as it helps in maintaining a non blocking server. We used nodejs with html and javascript as the programming technologies.

1. **MySQL:** Out of the two best available databases open source (MySQL, MongoDB), MongoDB is the better one to use if we want to change the database schema dynamically. But our requirement will suffice if our database schema is fixed.

For our simple query and inserting into the database tables, MySQL would be sufficient and hence used that.

**Design and Architecture:**

Maintained two virtual machines from AWS each for the server and the database. App.js is run on the server machine and it is connected from the server remotely.

**Work Plan:**

**Database Implementation:**

Installed and created a sample database on the other. Now, to connect to this database from the server, we need to grant all privileges on the database for the user from the server ip address. This database is now accessible from the server machine.

**Server Side Implementation:**

Installed node and wrote the server code in app.js. The remote database is connected from the server as follows -

var connection = mysql.createConnection({

host : 'IP\_ADDRESS',

user : 'USER\_NAME',

password : 'PASSWORD’,

database : DATABASE\_NAME

});

This creates the connection and further queries and insertions can be done accordingly.

***Interact to API using python***

The API which is created can be interacted using python. There are four main features from which can be done - GET, POST, PUT, DELETE. We are importing ‘request’ from ‘urllib2’ library.

‘GET’ command is used to get the information about the url and act accordingly. It is used through the following command line and the response is stored in ‘Request’-

Response = request.get(“<http://example.com>”)

‘POST’ command is used to place the info into the url and act accordingly.

requests.post(“<http://example.com>”, json=body)

‘DELETE’ command is used to delete the particular data from the url.

requests.delete(“<http://example.com>”, data=body)

These App can be verified by printing the response of these commands with the response status codes or the headers. The info can be processed using JSON for readability.

***Load Balancer***

The available two popular open source load balancers are HAProxy and Nginx. When we want to compare between the two, HAProxy is a general TCP load balancer whereas Nginx is a HTTP load balancer. Secondly, consider the case when in a huge network, a link to the server goes down or taken out of the deployment. Here HAProxy will send requests to the server almost immediately to the server once it restarts. But in case of Nginx, it will wait for sometime after it restarts and thus facilitating a better performance. In our case, we are using Nginx as we are dealing with only HTTP traffic and many functions like X-Forwarded-For can be used to maintain the ip address info of the incoming packet.

**Design**:

The mechanisms used in Nginx are described below.

1. Round Robin: It requests the application server to call in round robin fashion. This is the default case.
2. Least - connected: Request is handled to the server with least number of active connections. ‘Least\_conn’ is the key word used.
3. Ip- hash: In this mechanism, a hash table is used to determine to get the next server to send the request. ‘Ip-hash’ is the key word used. In this mechanism, we can stick a server for an ip address using the appropriate hash function.

Weighted load balancing can also be done. We have employed round robin fashion in this project.

**Work Plan:**

Here, round robin method is used. Nginx can be installed on the virtual machine using the following commands-

apt-get update;

apt-get install nginx;

‘sudo service /etc/nginx start’ will start Nginx and start listening to the port 80.

In the file ‘default’, in /etc/nginx/sites-available/, we will connect to the servers using the following lines of code -

upstream web\_backend {

server 172.31.26.61:8888;

server 172.31.26.60:8888;

}

We will use ‘proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;’ in the server to get the information about the source in the web server.

***IP Tables***

Server can be secured by implementing an efficient firewall. Here, iptables has been used to do the same. The modified iptables rules can be saved using iptables-persistent. Here, we have initially allowed the SSH connections from all the sources so that if we mess up with some rules, there would be an option to set it back. Below command is used for the same purpose.

iptables -A INPUT -p tcp -s 172.26.31.62 --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT

iptables -A INPUT -p tcp -s 172.26.31.62 --dport 80 -m state --state NEW,ESTABLISHED -j ACCEPT  
iptables -A OUTPUT -p tcp --sport 22 -m state --state ESTABLISHED -j ACCEPT

Now, all other ip addresses will be blocked. As these rules are matched in order, only the load balancer can access these servers using html or through html ports and all the other connections are blocked.

iptables -P INPUT DROP

iptables -P OUTPUT DROP

After writing into the iptables, save it using ‘sudo iptables-persistent save’ and reload it using ‘sudo iptables-persistent reload’.