**Assignment**

**Contents:**

|  |  |  |
| --- | --- | --- |
| **S. NO** | **Content** | **Page Number** |
| **1** | **REST WEBAPI** | **2** |
| **2** | **REST API Operations using Python and CURL** | **4** |
| **3** | **Nginx** | **6** |
| **4** | **IP Tables** | **6** |
| **5** | **Ansible Configuration Managment** | **7** |

**Basic Environment:**

1. **Used 5 Xubuntu 14.04 images in VirtualBox with bridge mode adaptor.**
2. **Connected all of them through ssh**
3. **Passwordless SSH login between Ansible VM and WebServers**
4. **DB Server with MongoDB installed**
5. **WebServers running websites on its own Nodejs server running on localhost:3000**
6. **MongoDB server on Database Server**
7. **IPTables on Webservers and DB server and accepting traffic through NGINX load balancers only**
8. **REST Web API:**

Using Nodejs REST WEB API to communicate on webservers connected to Mongo DB server.

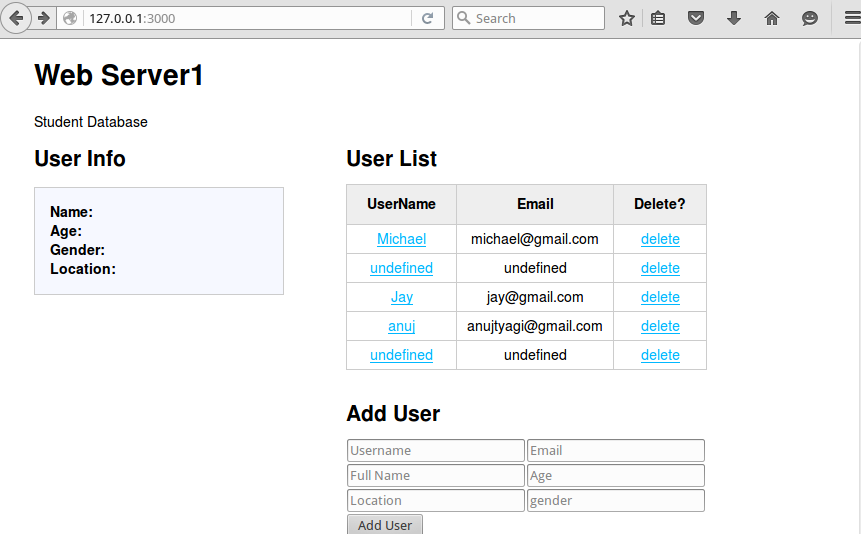


Fig. WEBAPI User Interface

1. WEBAPI providing addition, view and remove operations through GUI
2. Running on port 3000 Nodejs server
3. Connected with Mongo DB server 27017 port

Important code to check for the REST operations available in routes/users.js

**GET Operation:** Data of student information when adding saved in **users/userlist** json format. Below is the function for adding information to the database through GET.

router.get('/userlist', function(req, res) {

var db = req.db;

var collection = db.get('userlist');

collection.find({},{},function(e,docs){

res.json(docs);

});

});

**POST Operation:** To add the data, we need to use URL: localhost:3000/users/adduser.

router.post('/**adduser**', function(req, res) {

var db = req.db;

var collection = db.get('userlist');

collection.insert(req.body, function(err, result){

res.send(

(err === null) ? { msg: '' } : { msg: err }

);

});

});

**Delete Operation:** First, script extracting the data from db.get() and the sending the remove query by using id of user as key.

router.delete('/deleteuser/:id', function(req, res) {

var db = req.db;

var collection = db.get('userlist');

var userToDelete = req.params.id;

collection.remove({ '\_id' : userToDelete }, function(err) {

res.send((err === null) ? {msg: '' } : { msg:'error: ' + err });

});

});

module.exports = router;

1. **Interact With API (Python scripts to perform CRUD Operations)**

**GET: (Extracting the data from database)**

**First, we have assigned ip address with the path /users/userlist where user data is saved in JSON format. Its saved as array in list and each array is saved as dictionary.**

import urllib, json

url = <http://10.103.45.50:3000/users/userlist>

response = urllib.urlopen(url)

data = json.loads(response.read())

index = -1

for i in range(len(a)):

if a[i]['email']==email:

index = i

break

if index == -1:

print "Not found"

print "Student Details: ",data

**POST: (Adding/uploading data)**

First an empty dictionary is created so that we can add user info into this and then send it to database.

import requests

import json

payload = {}

#create an empty dictionary

url = "http://10.103.45.50:3000/users/adduser"

#Insert the data in dictionary

payload['username'] = str(raw\_input("Enter username : "))

payload['fullname'] = str(raw\_input("Enter fullname : "))

payload['email'] = str(raw\_input("Enter email : "))

payload['age'] = str(raw\_input("Enter username : "))

payload['location'] = str(raw\_input("Enter username : "))

payload['gender'] = str(raw\_input("Enter username : "))

r = requests.post(url, data=json.dumps(payload))

print(r.text)

**Delete: (Removing data)**

import urllib, json

url = "http://10.103.45.50:3000/users/userlist"

response = urllib.urlopen(url)

data = json.loads(response.read())

# removing the user information by taking it's email-id as input

def db\_search(a):

email = raw\_input("Enter the email id of user:\n")

index = -1

for i in range(len(a)):

if a[i]['email']==email:

index = i

break

if index == -1:

print "Not found"

a.remove(a[index])

print "Old database: ",data

db\_search(data)

print "New database: ",data

1. **Scale Up application with NGINX:** Nginx can be used as web server, proxy server, load balancer.

**Load Balancing: /etc/nginx/site-available**

1. Install NGINX: sudo apt­get install ­y nginx
2. Go to etc/nginx/sites-available/default
3. Open Default and add the below script.
4. Here, first web servers with their IP address are defined and informed to listen on port 80.

upstream web\_backend {

    server 10.0.0.15;

    server 10.0.0.16;

}

server {

    listen 80;

    location / {

        proxy\_set\_header X­Forwarded­For $proxy\_add\_x\_forwarded\_for;

        proxy\_pass http://web\_backend;

    }

}Here, we have added two servers to the NGINX and it’s balancing the traffic by using round robin algorithm.

1. **IP Tables:** IP table will drop all common ports which are unused. Also, accepting traffic from NGINX Servers and DB server only. All dropped packets by IP tables will be logged into file. Also, Web Servers can PING all servers but will not accept ICMP request from anyone.

sudo iptables -A INPUT -p tcp --match multiport --dports 23,100,25,20,21 -j DROP

sudo iptables -I INPUT -s 10.103.12.9 -j ACCEPT

sudo iptables -I INPUT -s 10.102.62.94 -j ACCEPT

sudo iptables -I INPUT -s 10.103.12.118 -j ACCEPT

sudo iptables -A OUTPUT -p icmp --icmp-type echo-request -j ACCEPT

sudo iptables -A INPUT -p icmp --icmp-type echo-reply -j ACCEPT

sudo iptables -A LOGGING -j DROP

sudo iptables -A INPUT -p ICMP --icmp-type 8 -j DROP

sudo iptables -A INPUT -j DROP

**Logs can be checked from**: /var/log/syslog

**To check iptables**: sudo iptables –L

**To Remove**: sudo iptables -F

1. **Automation with Ansible:**Ansible is a agentless configuration management tool and works with ssh connection only. To install Nginx, we need to provide ssh communication between devices without query of password. It can perform almost any operation to the remote device support ssh. There are 3 ways to configure device using Ansible: Adhoc Mode – Module, Playbook and Roles. I have used playbook to install NGINX.

In playlist, first we have defined action to perform on all hosts. It is then asking for sudo password from nodes. In task, ansible installing NGINX package and start it after installation.

**#Ansible playbook to install Nginx on web servers:**

**/etc/ansible/ansible-playbook.yml**

**And define users in host file at /etc/ansible/hosts**

---

- host: all

sudo: True

tasks:

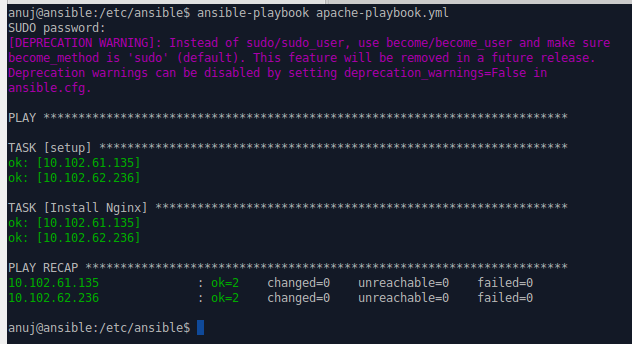
- name: Install Nginx

apt : pkg=nginx state=present

handlers:

- name: start nginx

service: name=nginx state=started



**NOTE: Before installing Ansible, we need to have password less login to nodes from Ansible server.**

First, install ssh package like Openssh then generate ssh keys by using $ssh-keygen.

Second, copy ssh keys from local system to remote system.

Also, we can create alias at home/<user>/.ssh/config by defining any name with respect to their ip addess. We can use those alias name to login to the nodes from server.