6. Vprofile Project Setup Manual, Automated

Vprofile project setup [LOCAL]

we will setup

- Multi tier web application stack
- Setup on laptop/desktop
- Baseline for upcoming projects
- Helps you setup any project locally

Scenario

- Working in a project
- varieties of services that powers yur project runtime.
- Runbook / Setup Documents

Problems

- Not comfortable in making changes in real servers.
- Local setup is complex
- Time consuming
- Not repeatable

Solutions

- Automated Local setup
- Repeatable local setup
- IAAC
- R&D in your own machine

Tools

- Hypervisor (Oracle VM virtualbox)
- Automation (Vagrant)
- CLI (GitBash)
- IDE (Sublime Text)

Objectives

- VM Automation locally

- Baseline for upcoming projects
- Real world project setup locally [For R&D]

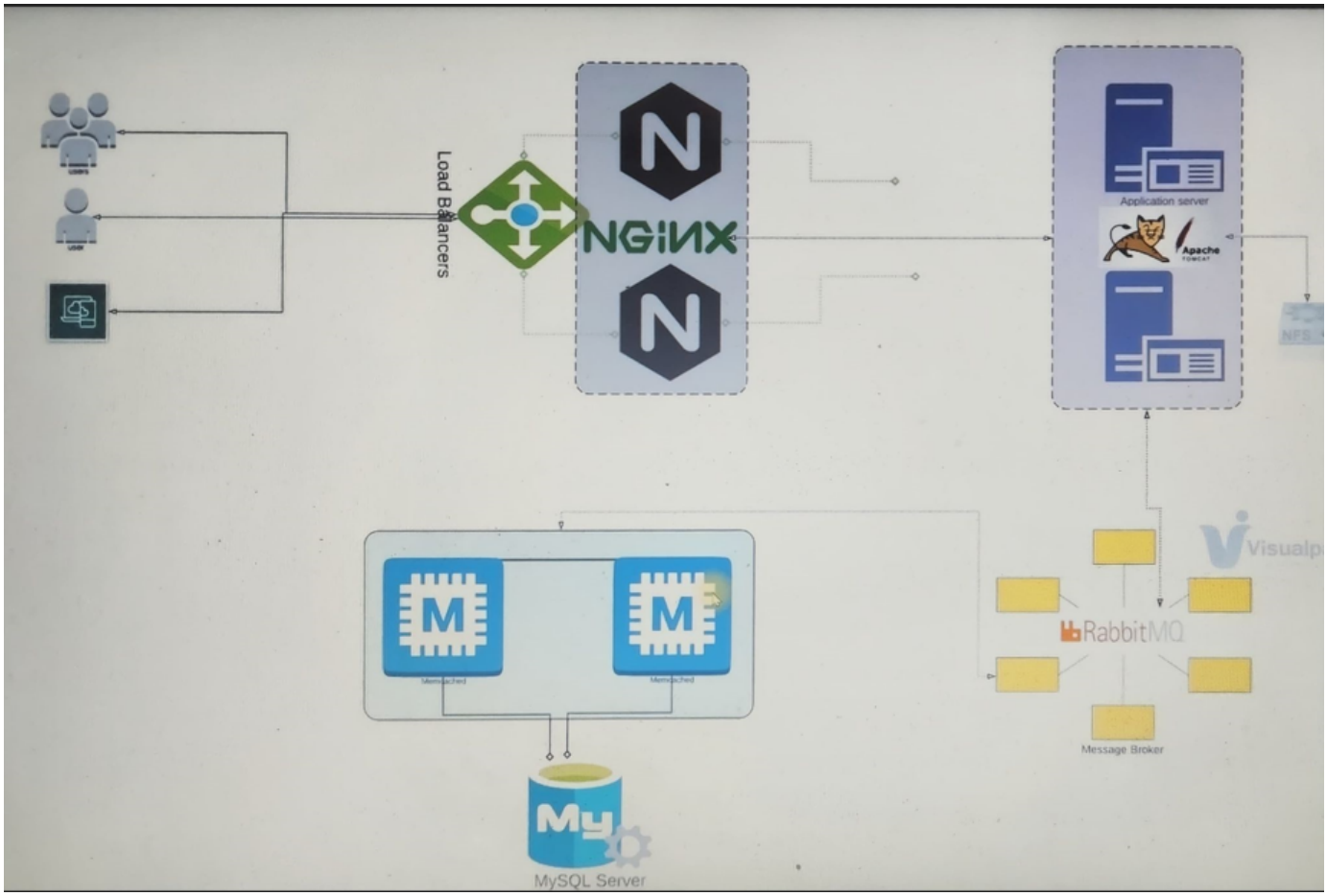
Vprofile Project
Architecture of project services

- NGINX
- TOMCAT
- RABBITMQ
- MEMCACHED
- MYSQL

Architecture of Automated Setup

- VAGRANT
- VIRTUALBOX
- GITBASH

ARCHITECTURAL DESIGN
after setting up our stack **user** can access all services from browser by entering ip addresses or endpoint then the user will be redirected to **load balancer** (**NGINX** service as a loadbalancer) then it will forward the request to **application server** (**Apache tomcat** will be our application server) where our java application will be running we can even have a **shared storage** using **NFS** Then application server will forward the request to **RabbitMQ** which will be our **message broker**. And that will send the request to Memcached for database caching. Memcached is gonna cache the sql queries which was executed for the MYSQL server



We will be using vagrant to automatically setup Vms. Vagrant will communicate to Oracle Vitualbox (Hypervisor) which will create vitual machines then we will be using bash scripts and bash commands to setup our services which are nginx, apache tomcat, memcached, rabbitmq, mysql.

Flow of execution

1. Setup tools mentioned in Prerequisite Video
2. Clone source code
3. cd into the vagrant dir
4. Bring up Vm's
5. Validate
6. Setup All the services
    1. Mysql
    2. memcached
    3. rabbit MQ
    4. tomcat
    5. nginx
    6. App Build and Deploy
7. Verify from Browser

VM setup

Manual Provisioning

we executed

`$ git clone https://github.com/devopshydclub/vprofile-project.git`

`$ git checkout local-setup` (This will switch to the branch local-setup)

`$ cd vagrant/`

`$ cd Manual_provisioning`

Then we open sublime text editor and select the vagrant folder which contains the vagrant file

Vagrantfile

```
Vagrant.configure("2") do |config|
  config.hostmanager.enabled = true
  config.hostmanager.manage_host = true

### Nginx VM ###
  config.vm.define "web01" do |web01|
    web01.vm.box = "ubuntu/xenial64"
    web01.vm.hostname = "web01"
  web01.vm.network "private_network", ip: "192.168.56.11"
  end

### tomcat vm ###
   config.vm.define "app01" do |app01|
    app01.vm.box = "geerlingguy/centos7"
    app01.vm.hostname = "app01"
    app01.vm.network "private_network", ip: "192.168.56.12"
  app01.vm.provider "virtualbox" do |vb|
     vb.memory = "1024"
   end
   end

### RabbitMQ vm  ####
  config.vm.define "rmq01" do |rmq01|
    rmq01.vm.box = "geerlingguy/centos7"
  rmq01.vm.hostname = "rmq01"
    rmq01.vm.network "private_network", ip: "192.168.56.16"
  end

### Memcache vm  ####
  config.vm.define "mc01" do |mc01|
    mc01.vm.box = "geerlingguy/centos7"
  mc01.vm.hostname = "mc01"
    mc01.vm.network "private_network", ip: "192.168.56.14"
  end

```

lets bring up the stack
before you bring up the VM you have to install the plugin necessarily ::

1. vagrant plugin install vagrant-hostmanager
2. vagrant plugin install vagrant-vbguest
   other wise you will get hostmanager unknown configuration error what this will do is automatically add host entries map it with their ip address of all the vms in every virtual machine
   the second plugin is optional

after running the command
vagrant plugin install vagrant-hostmanager

on git

we do
vagrant up

all 5 machines will be running

after that we check the host file by

cat /etc/hosts to see whether all the machines are in the file or not.

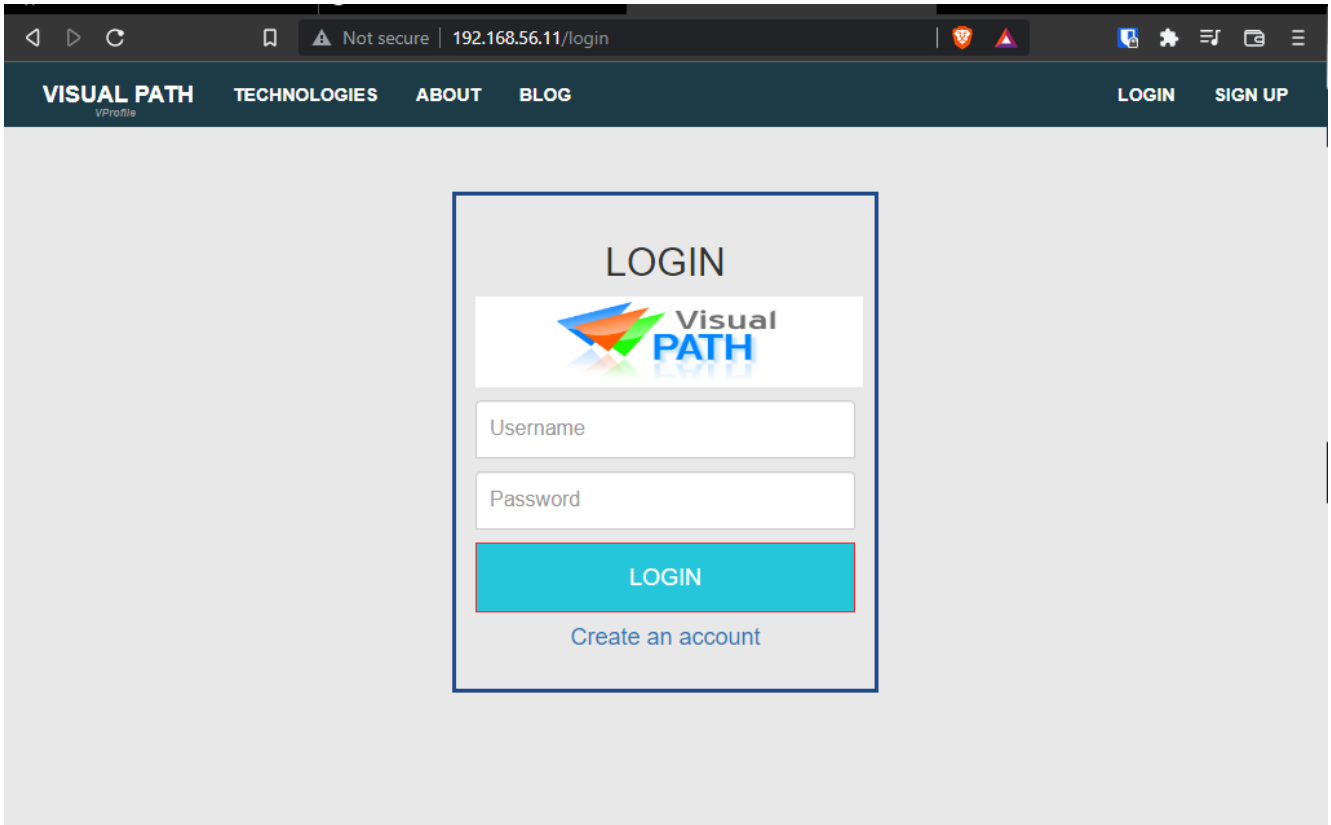
after that we check if they have connection or not, eg pinging.

for complete configuration follow the pdf guide which is :: linked
[VprofileProjectSetup.pdf]

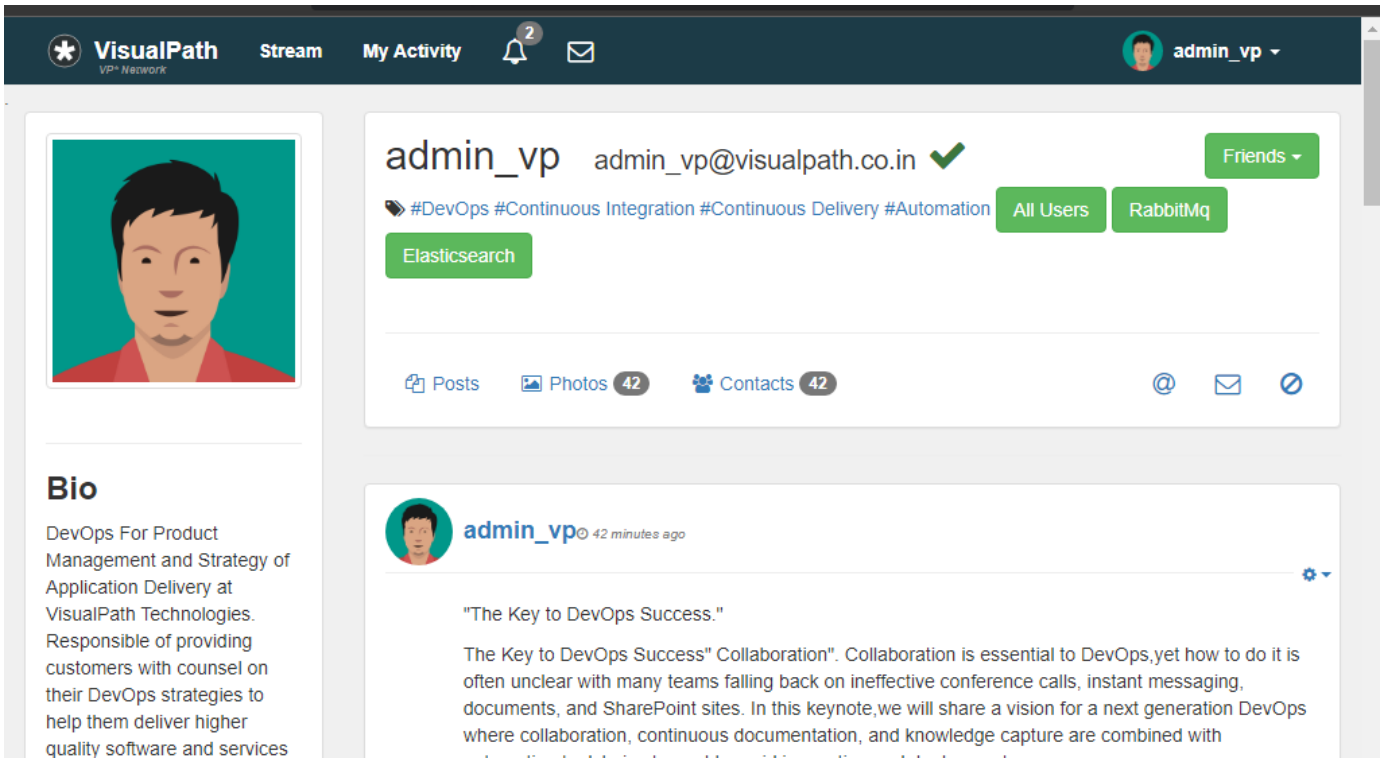
now finally lets see the **FLOW OF EXECUTION**

1. Setup tools mentioned in Prerequisite Vide
2. Clone source code
3. cd into the vagrant dir
4. Bring up VM's
5. validate
6. Setup all the services
    1. Mysql
    2. Memcached
    3. RabbitMQ
    4. tomcat
    5. Nginx
    6. App build and deploy

7. Verify from browser



we can see we set it up successfully



SO this is for the bringing up stack via manual provisioning.

so lets wrap up

By using vagrant we created virtual machines automatically on oracle virtualbox. Then we logged in to each and every machine and executed shell commands to settup various services. Once the stack was ready we verified as a user from the browser. we accessed nginx service. nginx service forwarded the service to tomcate service. tomcat forwared it to message broker rabbitmq and then to memcached and then to mysql server. So in the entire setup of this was all manual

Now we will do it from automatic provisioning

Automatic Provisioning

```
Vagrant.configure("2") do |config|
  config.hostmanager.enabled = true
  config.hostmanager.manage_host = true

### DB vm  ####
  config.vm.define "db01" do |db01|
    db01.vm.box = "geerlingguy/centos7"
    db01.vm.hostname = "db01"
    db01.vm.network "private_network", ip: "192.168.56.15"
    db01.vm.provision "shell", path: "mysql.sh"

  end

### Memcache vm  ####
  config.vm.define "mc01" do |mc01|
    mc01.vm.box = "geerlingguy/centos7"
    mc01.vm.hostname = "mc01"
    mc01.vm.network "private_network", ip: "192.168.56.14"
    mc01.vm.provision "shell", path: "memcache.sh"
  end

### RabbitMQ vm  ####
  config.vm.define "rmq01" do |rmq01|
    rmq01.vm.box = "geerlingguy/centos7"
  rmq01.vm.hostname = "rmq01"
    rmq01.vm.network "private_network", ip: "192.168.56.16"
    rmq01.vm.provision "shell", path: "rabbitmq.sh"
  end

### tomcat vm ###
   config.vm.define "app01" do |app01|
    app01.vm.box = "geerlingguy/centos7"
    app01.vm.hostname = "app01"
    app01.vm.network "private_network", ip: "192.168.56.12"
    app01.vm.provision "shell", path: "tomcat.sh"
```

in automatic provisioning the only difference in the vagrant file is that we have shell scripts executing for every vm

and the order is also different

first we are setting up db machine in this we are executing the shell script mysql.sh

```bash
#!/bin/bash
DATABASE_PASS='admin123'
sudo yum update -y
sudo yum install epel-release -y
sudo yum install git zip unzip -y
sudo yum install mariadb-server -y


# starting & enabling mariadb-server
sudo systemctl start mariadb
sudo systemctl enable mariadb
cd /tmp/
git clone -b local-setup https://github.com/devopshydclub/vprofile-project.git
#restore the dump file for the application
sudo mysqladmin -u root password "$DATABASE_PASS"
sudo mysql -u root -p"$DATABASE_PASS" -e "UPDATE mysql.user SET Password=PASSWORD('$DATABASE_PASS')
sudo mysql -u root -p"$DATABASE_PASS" -e "DELETE FROM mysql.user WHERE User='root' AND Host NOT IN (
sudo mysql -u root -p"$DATABASE_PASS" -e "DELETE FROM mysql.user WHERE User=''"
sudo mysql -u root -p"$DATABASE_PASS" -e "DELETE FROM mysql.db WHERE Db='test' OR Db='test\_%'"
sudo mysql -u root -p"$DATABASE_PASS" -e "FLUSH PRIVILEGES"
sudo mysql -u root -p"$DATABASE_PASS" -e "create database accounts"
sudo mysql -u root -p"$DATABASE_PASS" -e "grant all privileges on accounts.* TO 'admin'@'localhost'
sudo mysql -u root -p"$DATABASE_PASS" -e "grant all privileges on accounts.* TO 'admin'@'%' identifi
sudo mysql -u root -p"$DATABASE_PASS" accounts < /tmp/vprofile-project/src/main/resources/db_backup.
sudo mysql -u root -p"$DATABASE_PASS" -e "FLUSH PRIVILEGES"

# Restart mariadb-server
sudo systemctl restart mariadb


#starting the firewall and allowing the mariadb to access from port no. 3306
sudo systemctl start firewalld
sudo systemctl enable firewalld
sudo firewall-cmd --get-active-zones
sudo firewall-cmd --zone=public --add-port=3306/tcp --permanent
sudo firewall-cmd --reload
sudo systemctl restart mariadb
```

then we will setup memcache service in which the script memcache.sh will be executed.

```bash
#!/bin/bash
sudo yum install epel-release -y
sudo yum install memcached -y
sudo systemctl start memcached
sudo systemctl enable memcached
sudo systemctl status memcached
sudo memcached -p 11211 -U 11111 -u memcached -d
```

then it will setup rabbitmq service from the scipt rabbitmq.sh

Vagrantfile | rabbitmq.sh

```bash
#!/bin/bash
sudo yum install epel-release -y
sudo yum update -y
sudo yum install wget -y
cd /tmp/
wget http://packages.erlang-solutions.com/erlang-solutions-2.0-1.noarch.rpm
sudo rpm -Uvh erlang-solutions-2.0-1.noarch.rpm
sudo yum -y install erlang socat
curl -s https://packagecloud.io/install/repositories/rabbitmq/rabbitmq-server/script.rpm.sh | sudo bash
sudo yum install rabbitmq-server -y
sudo systemctl start rabbitmq-server
sudo systemctl enable rabbitmq-server
sudo systemctl status rabbitmq-server
sudo sh -c 'echo "[{rabbit, [{loopback_users, []}]}]." > /etc/rabbitmq/rabbitmq.config'
sudo rabbitmqctl add_user test test
sudo rabbitmqctl set_user_tags test administrator
sudo systemctl restart rabbitmq-server
```

next service is tomcat service which will execute the service tomcat.sh

```bash
TOMURL="https://archive.apache.org/dist/tomcat/tomcat-8/v8.5.37/bin/apache-tomcat-8.5.37.tar.gz"
yum install java-1.8.0-openjdk -y
yum install git maven wget -y
cd /tmp/
wget $TOMURL -O tomcatbin.tar.gz
EXTOUT=`tar xzvf tomcatbin.tar.gz`
TOMDIR=`echo $EXTOUT | cut -d '/' -f1`
useradd --shell /sbin/nologin tomcat
rsync -avzh /tmp/$TOMDIR/ /usr/local/tomcat8/
chown -R tomcat.tomcat /usr/local/tomcat8

rm -rf /etc/systemd/system/tomcat.service

cat <<EOT>> /etc/systemd/system/tomcat.service
[Unit]
Description=Tomcat
After=network.target

[Service]

User=tomcat
Group=tomcat

WorkingDirectory=/usr/local/tomcat8

#Environment=JRE_HOME=/usr/lib/jvm/jre
Environment=JAVA_HOME=/usr/lib/jvm/jre

Environment=CATALINA_PID=/var/tomcat/%i/run/tomcat.pid
Environment=CATALINA_HOME=/usr/local/tomcat8
Environment=CATALINE_BASE=/usr/local/tomcat8

ExecStart=/usr/local/tomcat8/bin/catalina.sh run
ExecStop=/usr/local/tomcat8/bin/shutdown.sh


RestartSec=10
Restart=always

[Install]
WantedBy=multi-user.target

EOT

systemctl daemon-reload
systemctl start tomcat
systemctl enable tomcat
```

```
Description=Tomcat
After=network.target

[Service]

User=tomcat
Group=tomcat

WorkingDirectory=/usr/local/tomcat8

#Environment=JRE_HOME=/usr/lib/jvm/jre
Environment=JAVA_HOME=/usr/lib/jvm/jre

Environment=CATALINA_PID=/var/tomcat/%i/run/tomcat.pid
Environment=CATALINA_HOME=/usr/local/tomcat8
Environment=CATALINE_BASE=/usr/local/tomcat8

ExecStart=/usr/local/tomcat8/bin/catalina.sh run
ExecStop=/usr/local/tomcat8/bin/shutdown.sh


RestartSec=10
Restart=always

[Install]
WantedBy=multi-user.target

EOT

systemctl daemon-reload
systemctl start tomcat
systemctl enable tomcat

git clone -b local-setup https://github.com/devopshydclub/vprofile-project.git
cd vprofile-project
mvn install
systemctl stop tomcat
sleep 60
rm -rf /usr/local/tomcat8/webapps/ROOT*
cp target/vprofile-v2.war /usr/local/tomcat8/webapps/ROOT.war
systemctl start tomcat
sleep 120
cp /vagrant/application.properties /usr/local/tomcat8/webapps/ROOT/WEB-INF/classes/application.properties
systemctl restart tomcat

```

before this stack comes up we have to make sure that application.properties file should be updated so from which the file contains our instructions. all the information in the applicaiton.properties file is based on the script that we are using like in mysql we settingup user admin and password admin123.

```
#JDBC Configutation for Database Connection
jdbc.driverClassName=com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://db01:3306/accounts?useUnicode=true&characterEncoding=UTF-8&zeroDateTimeBehavior=convertToNull
jdbc.username=admin
jdbc.password=admin123

#Memcached Configuration For Active and StandBy Host
#For Active Host
memcached.active.host=mc01
memcached.active.port=11211
#For StandBy Host
memcached.standBy.host=127.0.0.2
memcached.standBy.port=11211

#RabbitMq Configuration
rabbitmq.address=rmq01
rabbitmq.port=5672
rabbitmq.username=test
rabbitmq.password=test

#Elasticesearch Configuration
elasticsearch.host =192.168.1.85
elasticsearch.port =9300
elasticsearch.cluster=vprofile
elasticsearch.node=vprofilenode
```

and lastly it is going to be nginx which will execute the script nginx.sh

```
# adding repository and installing nginx
apt update
apt install nginx -y
cat <<EOT > vproapp
upstream vproapp {

 server app01:8080;

}

server {

  listen 80;

location / {

  proxy_pass http://vproapp;

}

}

EOT

mv vproapp /etc/nginx/sites-available/vproapp
rm -rf /etc/nginx/sites-enabled/default
ln -s /etc/nginx/sites-available/vproapp /etc/nginx/sites-enabled/vproapp

#starting nginx service and firewall
systemctl start nginx
systemctl enable nginx
systemctl restart nginx
```

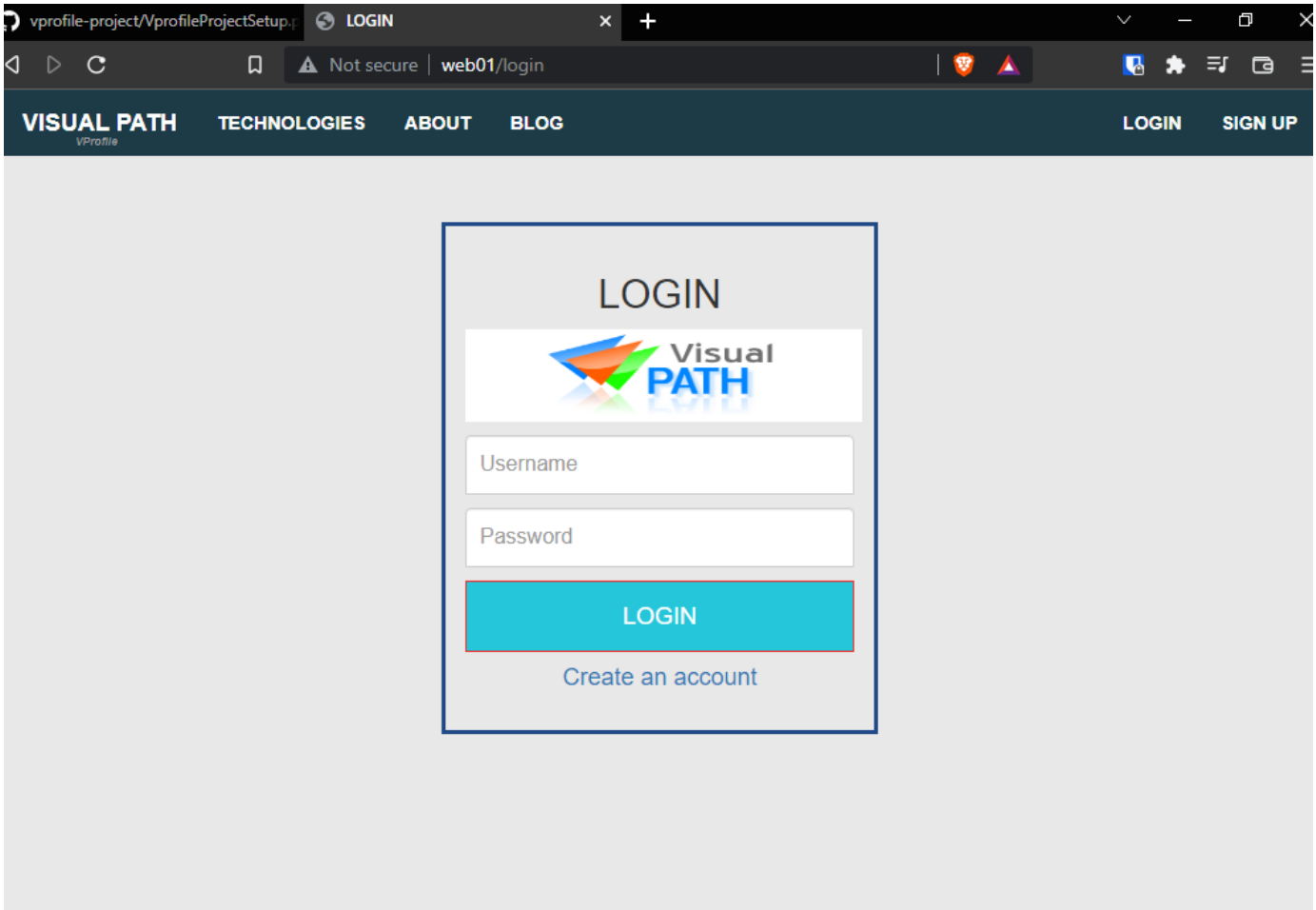
for this we only need to do

`vagrant up`

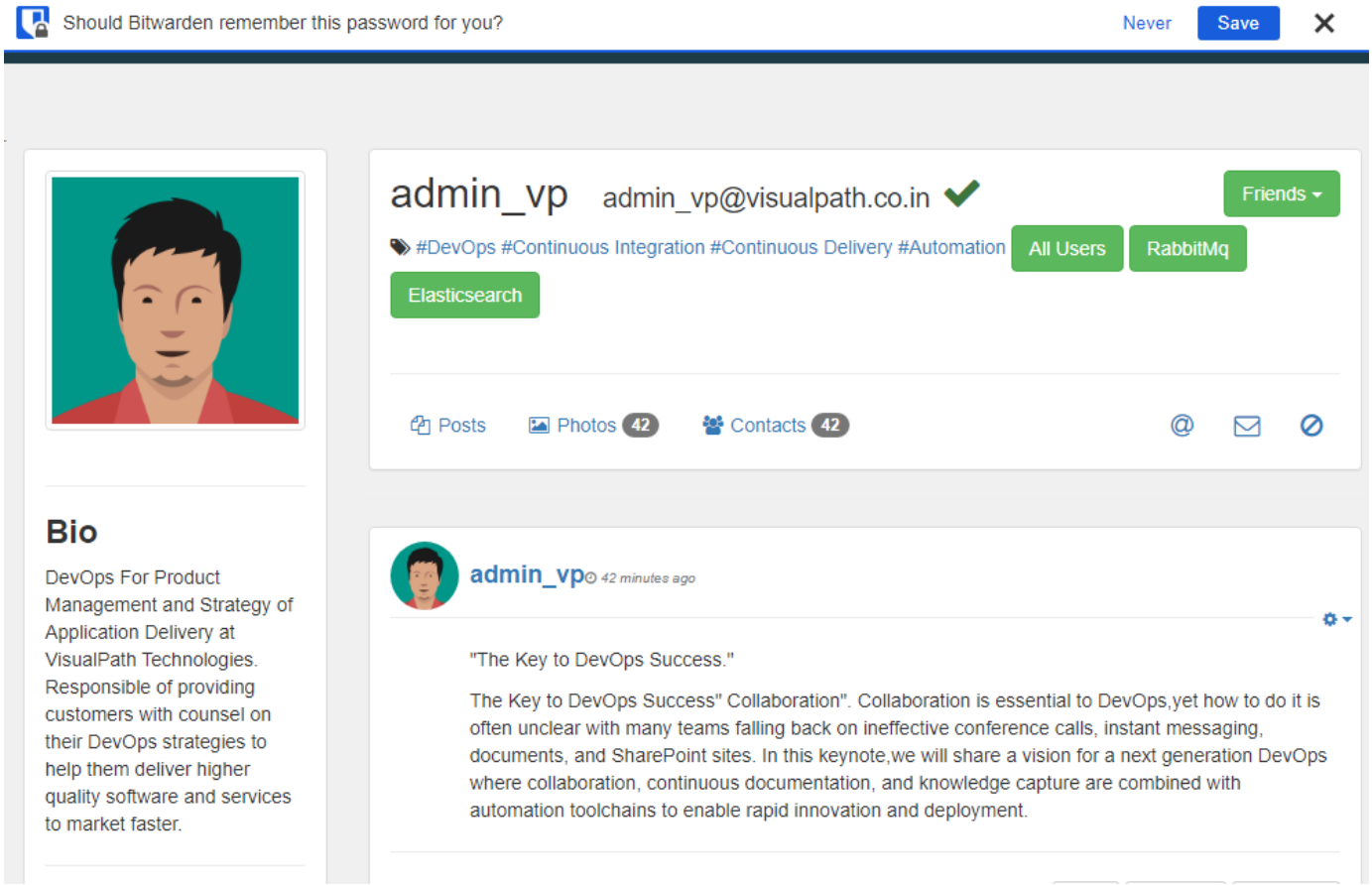
```
    web01: Unpacking libtiff5:amd64 (4.0.6-1ubuntu0.8) ...
    web01: Selecting previously unselected package libvpx3:amd64.
    web01: Preparing to unpack .../libvpx3_1.5.0-2ubuntu1.1_amd64.deb ...
    web01: Unpacking libvpx3:amd64 (1.5.0-2ubuntu1.1) ...
    web01: Selecting previously unselected package libxpm4:amd64.
    web01: Preparing to unpack .../libxpm4_1%3a3.5.11-1ubuntu0.16.04.1_amd64.deb ...
    web01: Unpacking libxpm4:amd64 (1:3.5.11-1ubuntu0.16.04.1) ...
    web01: Selecting previously unselected package libgd3:amd64.
    web01: Preparing to unpack .../libgd3_2.1.1-4ubuntu0.16.04.12_amd64.deb ...
    web01: Unpacking libgd3:amd64 (2.1.1-4ubuntu0.16.04.12) ...
    web01: Selecting previously unselected package nginx-common.
    web01: Preparing to unpack .../nginx-common_1.10.3-0ubuntu0.16.04.5_all.deb ...
    web01: Unpacking nginx-common (1.10.3-0ubuntu0.16.04.5) ...
    web01: Selecting previously unselected package nginx-core.
    web01: Preparing to unpack .../nginx-core_1.10.3-0ubuntu0.16.04.5_amd64.deb ...
    web01: Unpacking nginx-core (1.10.3-0ubuntu0.16.04.5) ...
    web01: Selecting previously unselected package nginx.
    web01: Preparing to unpack .../nginx_1.10.3-0ubuntu0.16.04.5_all.deb ...
    web01: Unpacking nginx (1.10.3-0ubuntu0.16.04.5) ...
    web01: Processing triggers for libc-bin (2.23-0ubuntu11.3) ...
    web01: Processing triggers for man-db (2.7.5-1) ...
    web01: Processing triggers for ureadahead (0.100.0-19.1) ...
    web01: Processing triggers for systemd (229-4ubuntu21.31) ...
    web01: Processing triggers for ufw (0.35-0ubuntu2) ...
    web01: Setting up libjpeg-turbo8:amd64 (1.4.2-0ubuntu3.4) ...
    web01: Setting up libjbig0:amd64 (2.1-3.1) ...
    web01: Setting up fonts-dejavu-core (2.35-1) ...
    web01: Setting up fontconfig-config (2.11.94-0ubuntu1.1) ...
    web01: Setting up libfontconfig1:amd64 (2.11.94-0ubuntu1.1) ...
    web01: Setting up libjpeg8:amd64 (8c-2ubuntu8) ...
    web01: Setting up libtiff5:amd64 (4.0.6-1ubuntu0.8) ...
    web01: Setting up libvpx3:amd64 (1.5.0-2ubuntu1.1) ...
    web01: Setting up libxpm4:amd64 (1:3.5.11-1ubuntu0.16.04.1) ...
    web01: Setting up libgd3:amd64 (2.1.1-4ubuntu0.16.04.12) ...
    web01: Setting up nginx-common (1.10.3-0ubuntu0.16.04.5) ...
    web01: Setting up nginx-core (1.10.3-0ubuntu0.16.04.5) ...
    web01: Setting up nginx (1.10.3-0ubuntu0.16.04.5) ...
    web01: Processing triggers for libc-bin (2.23-0ubuntu11.3) ...
    web01: Processing triggers for ureadahead (0.100.0-19.1) ...
    web01: Processing triggers for systemd (229-4ubuntu21.31) ...
    web01: Processing triggers for ufw (0.35-0ubuntu2) ...
    web01: Synchronizing state of nginx.service with SysV init with /lib/systemd/systemd-sysv-install...
    web01: Executing /lib/systemd/systemd-sysv-install enable nginx

ello world@DESKTOP-1BL5LTS MINGW64 /c/Devops/vprofile-project/vagrant/Automated_provisioning (local-setup)
```

so finally after some time the stack setup is complete.

we will access web01 from our browser by
http://web01



we can login from admin_vp and password admin_vp.



the login successful means that the db is validated

to check whether the rabbitmq is validated or not we click rabitmq

Should Bitwarden remember this password for you? Never Save

Rabbitmq initiated

Generated 2 Connections

6 Chanels 1 Exchage and 2 Que

we get this output which means it is validated as well.

to validate memcache

Users List

User Name	User Id
admin_vp	7
WahidKhan	8
Gayatri	9
WahidKhan2	10
KiranKumar	11
Saikumar	12
RamSai	13

we click all users then we click one. to see where it is being taken from

Should Bitwarden remember this password for you? Never Save

[Data is From DB and Data Inserted In Cache !!] Back

User Primary Details

Id	Name	Father's Name	Mother's Name	Email	Phone Number
11	KiranKumar	K K	RK	kiran@gmail.com	1010101010

User Extra Details

Date Of Birth	Gender	Marital Status	Permanent Address	Temporary Address	Primary Occupation	Secondary Occupation	Skills	Secondary PhoneNumber	Na
8/12/1993	male	unMarried	California	James Street	Software Engineer	Software Engineer	Java HTML CSS	1010101010	Inc

we can see that it is bring taken from database for now and it is inserted from cache

so now if we back and click it again it will be loaded directly from cache which will be much faster that loading it from database.

[Data is From Cache] Back

User Primary Details

Id	Name	Father's Name	Mother's Name	Email	Phone Number
11	KiranKumar	K K	RK	kiran@gmail.com	1010101010

User Extra Details

Date Of Birth	Gender	Marital Status	Permanent Address	Temporary Address	Primary Occupation	Secondary Occupation	Skills	Secondary PhoneNumber	Na
8/12/1993	male	unMarried	California	James Street	Software Engineer	Software Engineer	Java HTML CSS	1010101010	Inc

now we can see that it is loaded from cache
so from this we can validate the memcached

so from problem of provisioning manually we made a local setup which solves the problem of it.