Virtual profile project setup [Locally] Automated with Bashscript

[Multi tier web application setup (locally) Automatic provisioning]

About the project : Multi tier web application

Setup : Laptop / Desktop

Helps you setup any project Locally

Scenario :
Working in a project

Varieties of services that powers your project runtime like SQL services, application services

And also you have Runbook / Setup document to set up your project stack

Problem : Not comfortable making changes in real servers

Local setup is complex

Time consuming

Not repeatable

Automated with vagrant

So we avoid this setup

Solution:

- We can do local setup but it will be automated
- I would be repeatable because we are going to have Infrastructure as a code.
- So if we have code to set up the entire stack locally we can do it as many as time.
- So you can do as much as R&D you want on your local machine.

TOOLS

Hypervisor → Oracle VM virtual box

Automation → Vagrant

CLI → Git bash

IDE → VS code

OBJECTIVES

VM automation Locally

Real world project setup locally for R & D

Architecture of project services

NGINX → web service

TOMCAT → application server

RABBITMQ → Broker/Queuing agent

MEMCACHED → DB caching

MYSQL → Database service

Use cases :

1. NGINX → A high-performance web server and reverse proxy server for serving web content, load balancing, and handling HTTP, HTTPS, and mail protocols.

2. TOMCAT → An open-source Java servlet container and web server used to deploy and serve Java applications and dynamic web content.

3. RABBITMQ → A robust message broker that facilitates communication between distributed systems and applications through message queuing.

4. MEMCACHED → An in-memory key-value store used for caching data to accelerate web applications by reducing database load.

5. MYSQL → A widely-used open-source relational database management system for storing, managing, and retrieving structured data efficiently.

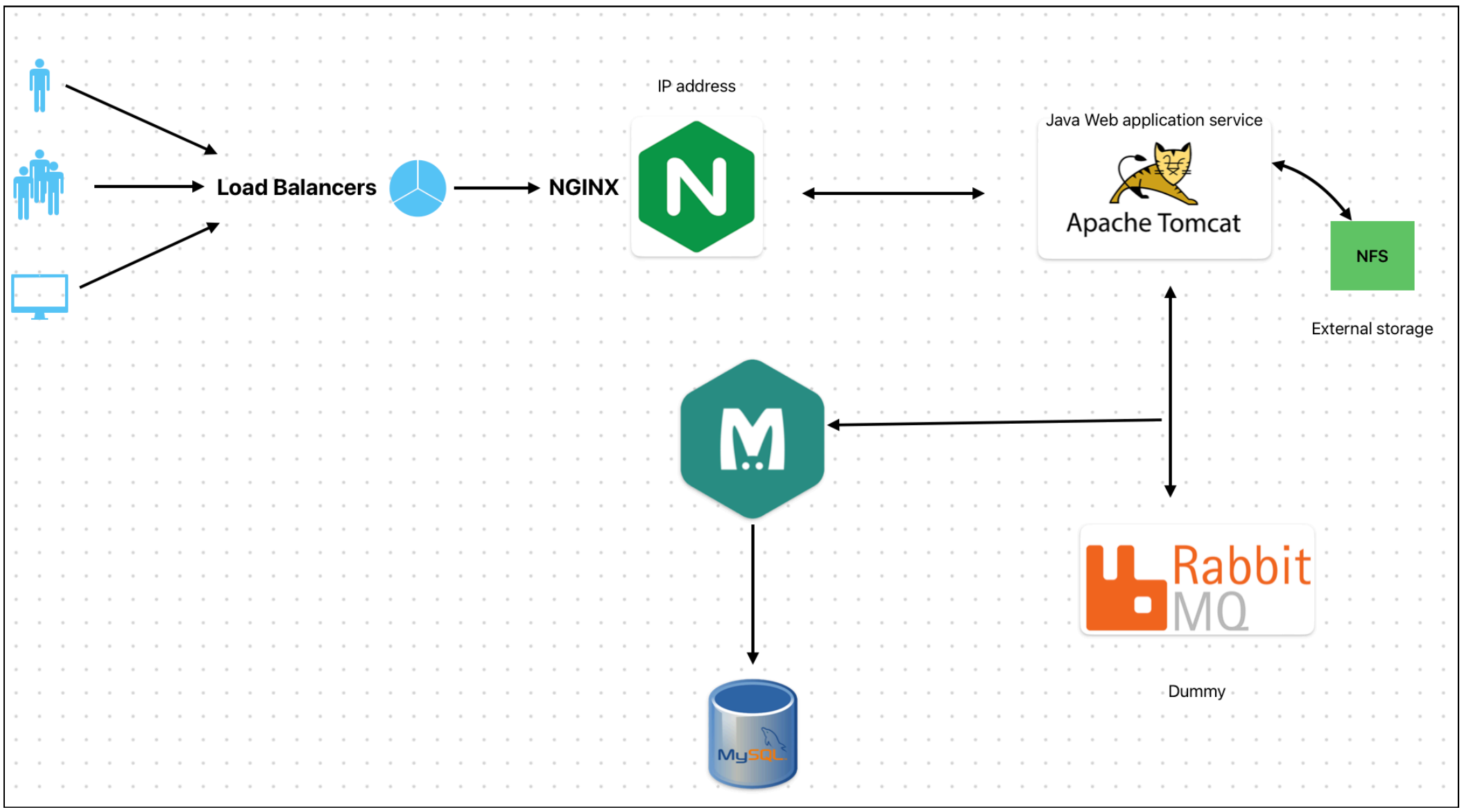
Architecture of automated setup

Vagrant

Virtual box

Git bash / terminal

Overview : so long story short we are setting up a website , web application and this web app is social networking site written by developers written in Java language



So we need to set up all these services in our Virtual machines and configure together.

Whenever user enter url / ip to the browser [ip of load balancer] it is going to router the

When request comes to the load balancer it is going to route the request to the Tomcat server or Apache tomcat service , so the application sitting here and if your application needs an external storage you can use NFS servers, user get the page and login details now login details will be save in mySQL database , RabbitMQ is dummy here , whenever user login our application will run a SQL query to access the user information stored in SQL

database , before it goes to database it will goes to memcached whenever user login second time.

some commands for vagrant status

$vagrant global-status → to check the global status of VM's

$vagrant up → to bring up the VM

$vagrant ssh → checking the status where the vagrant-file is present

$vagrant ssh db01 → login to the VM

$vagrant reload

$cat /etc/hosts → to check matching IP for VM , the output you see created by vagrant host manager plugin

So in multi machine environment where one machines connects to other machines the way of connecting is through IP addresses , but IP addresses may change and are so complicated , so. we are always go with the **hostname** , In configuration files you can see names mentioned not IP addresses

---

#!/bin/bash → this is to open the bash shell interpreter

The #!/bin/bash line at the beginning of a script file is known as a shebang (or hashbang). It indicates that the script should be run using the Bash shell. When you execute the script, the operating system uses the specified interpreter (in this case, /bin/bash) to run the commands within the script.

following are the bashscript files for automated provisioning using vagrant.

go to your project folder and run $vagrant up and it will start bringing up all the VMs

so you see we are not logging into VMs , setting up services vagrant is doing automatically

Vagrantfile
Every VM has it own shell script

```
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 = "jacobw/fedora35-arm64"
    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 = "jacobw/fedora35-arm64"
    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 = "jacobw/fedora35-arm64"
  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 = "jacobw/fedora35-arm64"
    app01.vm.hostname = "app01"
    app01.vm.network "private_network", ip: "192.168.56.12"
    app01.vm.provision "shell", path: "tomcat.sh"
    app01.vm.provider "vmware_desktop" do |vb|
     vb.memory = "1024"
   end
   end
```

```
### Nginx VM ###
  config.vm.define "web01" do |web01|
    web01.vm.box = "spox/ubuntu-arm"
    web01.vm.hostname = "web01"
  web01.vm.network "private_network", ip: "192.168.56.11"
  web01.vm.provision "shell", path: "nginx.sh"
end


end
```

mysql.sh

```
#!/bin/bash

sudo mv /etc/yum.repos.d/fedora-updates.repo /tmp/

sudo mv /etc/yum.repos.d/fedora-updates-modular.repo /tmp/

sudo yum clean all

sudo yum update -y

DATABASE_PASS='admin123'

#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 main 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') WHERE User='root'"

sudo mysql -u root -p"$DATABASE_PASS" -e "DELETE FROM mysql.user WHERE User='root' AND Host NOT IN ('localhost', '127.0.0.1', '::1')"

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' identified by 'admin123'"

sudo mysql -u root -p"$DATABASE_PASS" -e "grant all privileges on accounts.* TO 'admin'@'%' identified by 'admin123'"

sudo mysql -u root -p"$DATABASE_PASS" accounts < /tmp/vprofile-project/src/main/resources/db_backup.sql

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 stop firewalld

sudo systemctl disable firewalld

sudo systemctl restart mariadb
```

memcache.sh

```
#!/bin/bash

mv /etc/yum.repos.d/fedora-updates.repo /tmp/

mv /etc/yum.repos.d/fedora-updates-modular.repo /tmp/

yum clean all

yum update

sudo yum install epel-release -y

sudo yum install memcached -y

sudo systemctl start memcached

sudo systemctl enable memcached

sudo systemctl status memcached

firewall-cmd --add-port=11211/tcp --permanent

firewall-cmd --reload

sed -i 's/OPTIONS="-l 127.0.0.1"/OPTIONS=""/' /etc/sysconfig/memcached

sudo systemctl restart memcached


sudo memcached -p 11211 -U 11111 -u memcached -d
```

rabbitmq.sh

```
#!/bin/bash

sudo mv /etc/yum.repos.d/fedora-updates.repo /tmp/

sudo mv /etc/yum.repos.d/fedora-updates-modular.repo /tmp/

sudo yum clean all

sudo yum update -y

echo "SElinux changes."

sed -i 's/SELINUX=enforcing/SELINUX=disabled/' /etc/selinux/config

setenforce 0

echo

echo

curl -s https://packagecloud.io/install/repositories/rabbitmq/erlang/script.rpm.sh | sudo bash

sudo yum clean all

sudo yum makecache

sudo yum install erlang -y

curl -s https://packagecloud.io/install/repositories/rabbitmq/rabbitmq-server/script.rpm.sh | sudo bash

sudo yum install rabbitmq-server -y

rpm -qi rabbitmq-server

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

firewall-cmd --add-port=5671/tcp --permanent

firewall-cmd --add-port=5672/tcp --permanent

firewall-cmd --reload

sudo systemctl restart rabbitmq-server

nohup sleep 30 &&  reboot &

echo "going to reboot now"
```

tomcat.sh

```
sudo mv /etc/yum.repos.d/fedora-updates.repo /tmp/

sudo mv /etc/yum.repos.d/fedora-updates-modular.repo /tmp/

sudo yum clean all

#sudo yum update

TOMURL="https://archive.apache.org/dist/tomcat/tomcat-9/v9.0.75/bin/apache-tomcat-9.0.75.tar.gz"

yum install java-11-openjdk java-11-openjdk-devel -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/tomcat/

chown -R tomcat.tomcat /usr/local/tomcat


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/tomcat


#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/tomcat

Environment=CATALINE_BASE=/usr/local/tomcat


ExecStart=/usr/local/tomcat/bin/catalina.sh run

ExecStop=/usr/local/tomcat/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 main https://github.com/devopshydclub/vprofile-project.git

cd vprofile-project

mvn install

systemctl stop tomcat

sleep 60

rm -rf /usr/local/tomcat/webapps/ROOT*

cp target/vprofile-v2.war /usr/local/tomcat/webapps/ROOT.war

systemctl start tomcat

firewall-cmd --add-port=8080/tcp --permanent

firewall-cmd --reload

systemctl restart tomcat
```

tomcat-ubuntu.sh

```
#!/bin/bash

sudo apt update

sudo apt upgrade -y

sudo apt install openjdk-8-jdk -y

sudo apt install tomcat8 tomcat8-admin tomcat8-docs tomcat8-common git -y
```

after setting up all the stack we can verify it from browser the the ip of nginx from browwser or vagrantfile

httlp://web01 → route the request to tomcat

so how we have provisioned entire stack with just one command we just did vagrant up and all heavy lifting done by vagrant

$vagrant halt → to bring down all the VMs

$vagrant status

$vagrant global-status