VProfile Project Setup Manual and Automated

Application: A social networking site written by developers in Java language

Introduction

Scenario

Working in a project, we may be having varieties of services like database services, MySQL, PostgresSQL Web Services, Apache Engine X application services, etc that powers our project runtime.

We may have a RunBook or a Setup document to set up our project stack.

Problems

- We may not be comfortable in making changes in real servers
- Local setup is complex
- It is time consuming
- It is not repeatable

Solutions

- We have a local setup that is automated, repeatable
- We have a code to set up the entire stack locally (Infrastructure as a code - IAAC) -> we can do it as many time as we want

Tools

- Hypervisor: Oracle VM Virtualbox
- Automation: Vagrant
- CLI: GitBash
- IDE: VSCode

Objectives

- VM Automation locally
- Baseline for other projects
- Real world project setup locally for R&D

Architecture of Project Services

- NGINX
- TOMCAT
- RABBITMQ
- MEMCACHED
- MYSQL

Architecture of Automated Setup

- Vagrant
- VirtualBox
- GitBash

Flow of stack

- Stack = collection of services working together to create an experience (Web App in this case)

1. User opens the browser or the application
2. Type a URL (converted to IP address via DNS) or IP address (IP of load balancers in this case)

3. NGINX is used to create load balancing experience. It is a (frontend) web service/server just like Apache, httpd. It is also serving static content and acting as a reverse proxy to forward requests to application servers. It is very commonly used. Request is then routed to Apache TOMCAT by the load balancers
4. Apache TOMCAT is a Java Web Application service/server designed for hosting and running Java-based web applications. The user will get a response of a web page. The user will log in. The login details will be stored in a MySQL database service.
5. If the application needs an external storage, we can use NFS servers. If we have a cluster of servers and if we need a centralized storage, we can store it in NFS
6. RabbitMQ service is connected to TOMCAT. It is a message broker or also called a Queuing agent or message queue to connect to application together. We can use it to stream data
7. If the Apache TOMCAT request any database queries. The queries will be delivered to MEMCAT services (Database Caching). If the same query was performed and nothing changed to the databases, the MEMCAT will respond to TOMCAT with the cached data. If not, MEMCAT will communicate to MySQL Servers.
8. MEMCAT Services are connected to MySQL Servers that store data

Automation Stack

1. Vagrant to automate the setup of our virtual machine
2. Vagrant communicates with Oracle VM VirtualBox which is the hypervisor and then create virtual machines automatically
3. Bash Scripts/Commands to set up our services ie. NGINX, TOMCAT, MEMCAT, RabbitMQ, MySQL (each service uses 1 virtual machine)

Flow of Execution

1. Setup tools
2. Clone source code
3. `cd` into the vagrant directory in the source code
4. Bring up VMs from the Vagrantfile
5. Validate all the VMs to make sure they are up and running and able to interact with each other
6. Setup all the services. MySQL -> Memcached -> RabbitMQ -> TOMCAT -> NGINX -> App Build and Deploy
7. Verify from the browser

VM Setup

- Install `hostmanager` plugin: `vagrant plugin install vagrant-hostmanager`

Manual Setup

1. Create a `Vagrantfile` with the following content. Each service will have a corresponding virtual machine. We need to install them in the order as `MySQL` -> `Memcache` -> `RabbitMQ` -> `TOMCAT` -> `NGINX`

```
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 = "eurolinux-vagrant/centos-stream-9"
    db01.vm.hostname = "db01"
    db01.vm.network "private_network", ip: "192.168.56.15"
    db01.vm.provider "virtualbox" do |vb|
```

```
      vb.memory = "600"
    end

  end

### Memcache vm  ####
  config.vm.define "mc01" do |mc01|
    mc01.vm.box = "eurolinux-vagrant/centos-stream-9"
    mc01.vm.hostname = "mc01"
    mc01.vm.network "private_network", ip: "192.168.56.14"
    mc01.vm.provider "virtualbox" do |vb|
     vb.memory = "600"
   end
  end

### RabbitMQ vm  ####
  config.vm.define "rmq01" do |rmq01|
    rmq01.vm.box = "eurolinux-vagrant/centos-stream-9"
  rmq01.vm.hostname = "rmq01"
    rmq01.vm.network "private_network", ip: "192.168.56.13"
    rmq01.vm.provider "virtualbox" do |vb|
     vb.memory = "600"
   end
  end

### tomcat vm ###
   config.vm.define "app01" do |app01|
    app01.vm.box = "eurolinux-vagrant/centos-stream-9"
    app01.vm.hostname = "app01"
    app01.vm.network "private_network", ip: "192.168.56.12"
    app01.vm.provider "virtualbox" do |vb|
     vb.memory = "800"
   end
   end


### Nginx VM ###
  config.vm.define "web01" do |web01|
    web01.vm.box = "ubuntu/jammy64"
    web01.vm.hostname = "web01"
  web01.vm.network "private_network", ip: "192.168.56.11"
  web01.vm.provider "virtualbox" do |vb|
     vb.gui = true
     vb.memory = "800"
   end
end

end
```

2. Bring up all the VMs: `vagrant up`
3. Login into VMs: `vagrant ssh <vm name>`

4. Verify hosts entry, if entries are missing, update them with IP and hostname: `cat /etc/hosts`

MySQL Setup

5. Check if VMs are connected. Here `db01` is connected to `mc01` , `mc01` to `db01` & `TOMCAT` , `RabbitMQ` to `TOMCAT` , `TOMCAT` to `RabbitMQ` & `Memcache` & `NGINX` , `NGINX` to `TOMCAT` using: `ping <connected vm> -c 4` to check the connection
6. Update OS with latest patches: `yum update -y`
7. Set Repository: `yum install epel-release -y`
8. Install maria DB Package to `db01` VM: `yum install git mariadb-server -y`
9. Starting & Enabling `mariadb-server` : `systemctl start mariadb` and `systemctl enable mariadb`
10. Run MySQL secure installation script: `mysql_secure_installation` (command created by mariadb). Answer all the questions. Password of the database (`change the root password`) to `admin123` (it should be a strong password in real life setup)
11. Set DB name and users: `mysql -u root -padmin123` -> `create database accounts;` -> `grant all privileges on accounts.* TO 'admin'@'%' identified by 'admin123';` -> `FLUSH PRIVILEGES;` -> `exit;` . Here `%` = remotely, so admin of the database should be able to connect from a remote machine with the password of `admin123` (this password can be different to the root database password)
12. Clone the existing project into the repository: `git clone -b main https://github.com/hkhcoder/vprofile-project.git` -> `cd vprofile-project` -> `mysql -u root -padmin123 accounts < src/main/resources/db_backup.sql` (`<` for input redirection -> `src/main/resources/db_backup.sql` will be run in the `accounts` database) -> `mysql -u root -padmin123 accounts` -> `show tables;` -> `exit;` -> restart the mariadb `systemctl restart mariadb`
13. Start the firewall and allowing the mariadb to access from port no. 3306

```
systemctl start firewalld
systemctl enable firewalld
firewall-cmd --get-active-zones
firewall-cmd --zone=public --add-port=3306/tcp --permanent
firewall-cmd --reload
systemctl restart mariadb
```

Memcache Setup

13. Login into the memcache `vagrant ssh mc01`
14. Install, start & enable memcache on port 11211

```
dnf install epel-release -y
dnf install memcached -y
systemctl start memcached
systemctl enable memcached
systemctl status memcached
sed -i 's/127.0.0.1/0.0.0.0/g' /etc/sysconfig/memcached # search for 127.0.0.1 and
replace it with `0.0.0.0` in the /etc/sysconfig/memcached file. Some services
listens only for local connection. That means if the service runs on a different
machine. And there are some other services (like application services) connecting to
that machine/service, it will reject those connections because the IP addresses are
```

```
not whitelisted. So we set to 0.0.0.0 (all IPv4) to listen on all available network
interfaces instead of just the loopback interface (127.0.0.1)
systemctl restart memcached
```

15. Starting the firewall and allowing the port 11211 to access memcache

```
systemctl start firewalld
systemctl enable firewalld
firewall-cmd --add-port=11211/tcp
firewall-cmd --runtime-to-permanent
firewall-cmd --add-port=11111/udp
firewall-cmd --runtime-to-permanent
memcached -p 11211 -U 11111 -u memcached -d
```

RabbitMQ Setup

16. Login into the RabbitMQ `vagrant ssh rmq01`
17. `yum update -y`
18. Set Repository: `yum install epel-release -y`
19. Install dependencies:

```
yum install wget -y
cd /tmp/
dnf -y install centos-release-rabbitmq-38
dnf --enablerepo=centos-rabbitmq-38 -y install rabbitmq-server
systemctl enable --now rabbitmq-server
```

20. Setup access to user test and make it admin

```
# The configuration of RabbitMQ (a message broker software) and specifies that there
are no loopback users allowed in the RabbitMQ configuration.
sh -c 'echo "[{rabbit, [{loopback_users, []}]}]." > /etc/rabbitmq/rabbitmq.config'
# echo "[{rabbit, [{loopback_users, []}]}]." = create a string and output it to the
standard output the string is "[{rabbit, [{loopback_users, []}]}]."
# > /etc/rabbitmq/rabbitmq.config = redirects the standard output (the string
generated by echo) to a file named /etc/rabbitmq/rabbitmq.config. It uses the >
operator to overwrite the contents of the file with the output from echo. In this
case, it will create or replace the /etc/rabbitmq/rabbitmq.config file with the
specified content.
rabbitmqctl add_user test test # user = test and password  = test
rabbitmqctl set_user_tags test administrator # set administrator tag to test user
systemctl restart rabbitmq-server
```

21. Starting the firewall and allowing the port 5672 tp access rabbitmq

```
systemctl start firewalld
systemctl enable firewalld
firewall-cmd --runtime-to-permanent
systemctl start rabbitmq-server
```

```
systemctl enable rabbitmq-server
systemctl status rabbitmq-server
```

TOMCAT Setup

22. Login into the web application server TOMCAT `vagrant ssh app01`
23. `yum update -y`
24. Set Repository: `yum install epel-release -y`
25. Install dependencies:

```
dnf -y install java-11-openjdk java-11-openjdk-devel
dnf install git maven wget -y
cd /tmp/
wget https://archive.apache.org/dist/tomcat/tomcat-9/v9.0.75/bin/apache-tomcat-9.0.75.tar.gz
tar xzvf apache-tomcat-9.0.75.tar.gz
```

26. Add tomcat user: `useradd --home-dir /usr/local/tomcat --shell /sbin/nologin tomcat`
27. Copy data to tomcat home dir: `cp -r /tmp/apache-tomcat-9.0.75/* /usr/local/tomcat/`. Normal services, all the folder structure will be distributed to the corresponding Linux directories like log files going to `var/log/`, startup scripts going to `systemd`, configurations going to `/etc/`. However, some services like the apache-tomcat will not be extracted like that. Normally, DevOps engineers will just extract the package and start the service in the extracted folder without using the `systemctl`. Best practice is using the `systemctl` as the following steps
28. Make tomcat user owner of the tomcat home dir: `chown -R tomcat.tomcat /usr/local/tomcat`
29. Setup systemctl command for tomcat `vi /etc/systemd/system/tomcat.service`. And update the file with the following content

```
[Unit]
Description=Tomcat
After=network.target

[Service]
User=tomcat
WorkingDirectory=/usr/local/tomcat
Environment=JRE_HOME=/usr/lib/jvm/jre
Environment=JAVA_HOME=/usr/lib/jvm/jre
Environment=CATALINA_HOME=/usr/local/tomcat
Environment=CATALINA_BASE=/usr/local/tomcat
ExecStart=/usr/local/tomcat/bin/catalina.sh run
ExecStop=/usr/local/tomcat/bin/shutdown.sh
SyslogIdentifier=tomcat-%i

[Install]
WantedBy=multi-user.target
```

30. Reload systemd files: `systemctl daemon-reload`
31. Start & enable server

```
systemctl start tomcat
systemctl enable tomcat
```

32. Enable the firewall and allow port 8080 to access the tomcat

```
systemctl start firewalld
systemctl enable firewalld
firewall-cmd --get-active-zones
firewall-cmd --zone=public --add-port=8080/tcp --permanent
firewall-cmd --reload
```

Code Build and Deploy

33. Download Java project source code: `git clone -b main https://github.com/hkhcoder/vprofile-project.git` (still in the `/temp` directory of the app01 VM)
34. Update configuration

```
cd vprofile-project
vim src/main/resources/application.properties # application.properties is used by tomcat server to find information of all the backend services
mysql, memcache, RabbitMQ, Elasticsearch
update file # update file with backend server details if needed in case we use Amazon, Kubernetes, etc.
mvn install # make sure we are in the correct folder with the pom.xml file
```

35. Deploy the artifact:

```
systemctl stop tomcat
rm -rf /usr/local/tomcat/webapps/ROOT
cp target/vprofile-v2.war /usr/local/tomcat/webapps/ROOT.war
systemctl start tomcat
# chown tomcat.tomcat usr/local/tomcat/webapps -R
# systemctl restart tomcat
```

NGINX Setup

36. Login into the `web01` : `vagrant ssh web01`
37. Update OS with latest patches: `apt update` -> `apt upgrade`
38. Install NGINX: `apt install nginx -y`
39. Create NGINX conf file: `vi /etc/nginx/sites-available/vproapp` and update with the below content

```
upstream vproapp {
    server app01:8080;
    # server app02:8080; if we have multiple tomcat servers
}
server {
    listen 80;
```

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

40. Remove default NGINX conf: `rm -rf /etc/nginx/sites-enabled/default`
41. Create link to activate the website: `ln -s /etc/nginx/sites-available/vproapp /etc/nginx/sites-enabled/vproapp`
42. Restart NGINX: `systemctl restart nginx`
43. Now the web application can be accessed via browser in the `192.168.56.11` of the `web01` (check `ip addr show` to check the correct IP)

Automated Setup

Vagrantfile

```
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 = "eurolinux-vagrant/centos-stream-9"
    db01.vm.hostname = "db01"
    db01.vm.network "private_network", ip: "192.168.56.15"
    db01.vm.provider "virtualbox" do |vb|
     vb.memory = "600"
   end
    db01.vm.provision "shell", path: "mysql.sh"  # path to run the mysql.sh scripts located in the same folder

  end

### Memcache vm  ####
  config.vm.define "mc01" do |mc01|
    mc01.vm.box = "eurolinux-vagrant/centos-stream-9"
    mc01.vm.hostname = "mc01"
    mc01.vm.network "private_network", ip: "192.168.56.14"
    mc01.vm.provider "virtualbox" do |vb|
     vb.memory = "600"
   end
    mc01.vm.provision "shell", path: "memcache.sh"  # path to run the memcache.sh scripts located in the same folder

  end

### RabbitMQ vm  ####
  config.vm.define "rmq01" do |rmq01|
    rmq01.vm.box = "eurolinux-vagrant/centos-stream-9"
  rmq01.vm.hostname = "rmq01"
    rmq01.vm.network "private_network", ip: "192.168.56.16"
```

```
    rmq01.vm.provider "virtualbox" do |vb|
     vb.memory = "600"
   end
    rmq01.vm.provision "shell", path: "rabbitmq.sh"  # path to run the rabbitmq.sh
scripts located in the same folder
  end

### tomcat vm ###
   config.vm.define "app01" do |app01|
    app01.vm.box = "eurolinux-vagrant/centos-stream-9"
    app01.vm.hostname = "app01"
    app01.vm.network "private_network", ip: "192.168.56.12"
    app01.vm.provision "shell", path: "tomcat.sh"  # path to run the tomcat.sh
scripts located in the same folder
    app01.vm.provider "virtualbox" do |vb|
     vb.memory = "800"
   end
   end


### Nginx VM ###
  config.vm.define "web01" do |web01|
    web01.vm.box = "ubuntu/jammy64"
    web01.vm.hostname = "web01"
  web01.vm.network "private_network", ip: "192.168.56.11"
#  web01.vm.network "public_network"
  web01.vm.provider "virtualbox" do |vb|
     vb.gui = true
     vb.memory = "800"
   end
  web01.vm.provision "shell", path: "nginx.sh"  # path to run the nginx.sh scripts
located in the same folder
end

end
```

Bash scripts of mysql

```
#!/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 main https://github.com/hkhcoder/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 "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 restart mariadb
```

Bash scripts of memcache

```
#!/bin/bash
sudo dnf install epel-release -y
sudo dnf install memcached -y
sudo systemctl start memcached
sudo systemctl enable memcached
sudo systemctl status memcached
sed -i 's/127.0.0.1/0.0.0.0/g' /etc/sysconfig/memcached
sudo systemctl restart memcached
firewall-cmd --add-port=11211/tcp
firewall-cmd --runtime-to-permanent
firewall-cmd --add-port=11111/udp
firewall-cmd --runtime-to-permanent
sudo memcached -p 11211 -U 11111 -u memcached -d
```

Bash scripts of RabbitMQ

```
#!/bin/bash
sudo yum install epel-release -y
sudo yum update -y
```

```
sudo yum install wget -y
cd /tmp/
dnf -y install centos-release-rabbitmq-38
 dnf --enablerepo=centos-rabbitmq-38 -y install rabbitmq-server
 systemctl enable --now rabbitmq-server
 firewall-cmd --add-port=5672/tcp
 firewall-cmd --runtime-to-permanent
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
```

Bash scripts of tomcat

```
TOMURL="https://archive.apache.org/dist/tomcat/tomcat-9/v9.0.75/bin/apache-tomcat-
9.0.75.tar.gz"
dnf -y install java-11-openjdk java-11-openjdk-devel
dnf 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/hkhcoder/vprofile-project.git
cd vprofile-project
mvn install
systemctl stop tomcat
sleep 20
rm -rf /usr/local/tomcat/webapps/ROOT*
cp target/vprofile-v2.war /usr/local/tomcat/webapps/ROOT.war
systemctl start tomcat
sleep 20
systemctl stop firewalld
systemctl disable firewalld
#cp /vagrant/application.properties /usr/local/tomcat/webapps/ROOT/WEB-
INF/classes/application.properties
systemctl restart tomcat
```

Bash scripts of Nginx

```
# 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
```