Virtual profile project setup [Locally] Manual provisioning

[Multi tier web application setup (locally)](#)

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

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.

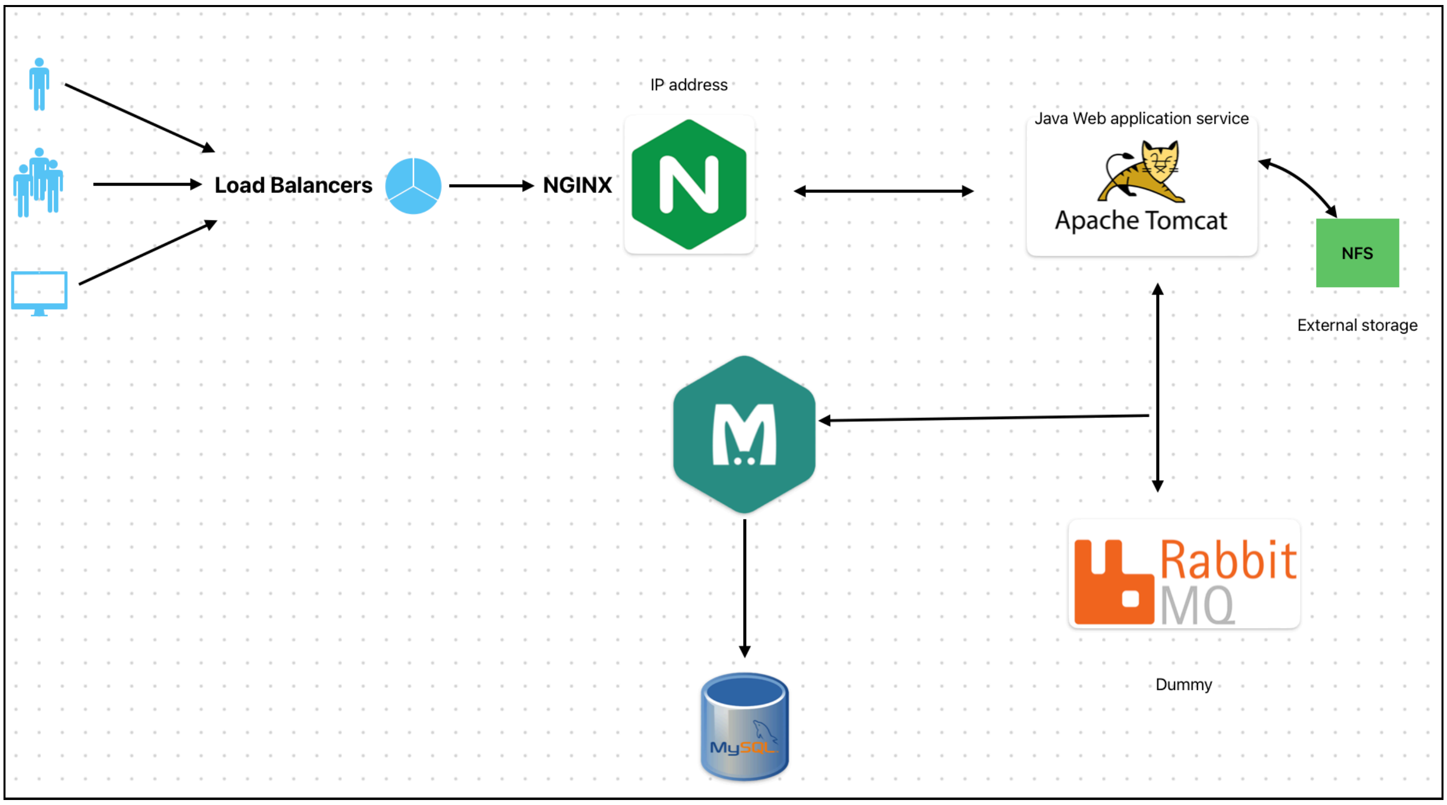
Architecture of automated setup

Vagrant

Virtual box

Git bash

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.

[Vagrantfile]()

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

to check the connection use $ping <hostname> -c 4

in our case $ping web01 -c 4

Setup should be done in the mentioned order

- MySQL
- Memeached
- RabbitMQ
- Tomcat
- Nginx

1. MySQL setup

Login to the db VM

$ vagrant ssh db01

$ sudo -i

Verify Hosts entry, if entries missing update it with IP and hostnames

# cat /etc/hosts

Install MariaDB Package

# yum install git mariadb-server -y

Starting & enabling mariadb-server

# systemctl start mariadb

# systemctl enable mariadb

RUN mysql secure installation script.

# mysql_secure_installation

Set DB name and users.

# mysql -u root -padmin123

mysql> create database accounts;

mysql> grant all privileges on accounts. TO 'admin'@'%' identified by 'admin123';*

mysql> FLUSH PRIVILEGES;

mysql> exit;

Download Source code & Initialize Database.

# git clone https://github.com/purveshshende2/multi-tier-web-application

# cd vprofile-project

# mysql -u root -padmin123 accounts < src/main/resources/db_backup.sql

# mysql -u root -padmin123 accounts

mysql> show tables;

mysql> exit;

Restart mariadb-server

# systemctl restart mariadb

Starting 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

2. MEMCACHE SETUP

Install, start & enable memcache on port 11211

# sudo yum 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

Starting the firewall and allowing the port 11211 to access memcache

# 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

3. RABBITMQ SETUP

Login to the RabbitMQ VM

$ vagrant ssh rmq01

$ sudo -i

Verify Hosts entry, if entries missing update it with IP and hostnames

# cat /etc/hosts

Disable SELINUX on fedora

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

# setenforce 0

Install Dependencies

# 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

Install Rabbitmq Server

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

# sudo yum install rabbitmq-server -y

Start & Enable RabbitMQ

# sudo systemctl start rabbitmq-server

# sudo systemctl enable rabbitmq-server

# sudo systemctl status rabbitmq-server

Config Change

# sudo sh -c 'echo "[{rabbit, [{loopback_users, []}]}]." > /etc/rabbitmq/rabbitmq.config'

# sudo rabbitmqctl add_user test test

# sudo rabbitmqctl set_user_tags test administrator

FEDORA Changes

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

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

# firewall-cmd --reload

# sudo systemctl restart rabbitmq-server

# reboot

Restart RabbitMQ service

# systemctl restart rabbitmq-server

4. TOMCAT SETUP

Login to the tomcat VM

$ vagrant ssh app01

Verify Hosts entry, if entries missing update it with IP and hostnames

# cat /etc/hosts

Update OS with latest patches

# yum update -y

Set Repository

# yum install epel-release -y

Install Dependencies

# dnf -y install java-11-openjdk java-11-openjdk-devel

# dnf install git maven wget -y

Change dir to /tmp

# cd /tmp/

Download & Tomcat Package

# 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

Add tomcat user

# useradd --home-dir /usr/local/tomcat --shell /sbin/nologin tomcat

Copy data to tomcat home dir

# cp -r /tmp/apache-tomcat-9.0.75/ /usr/local/tomcat/*

Make tomcat user owner of tomcat home dir

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

Setup systemctl command for tomcat

Create tomcat service file

# vi /etc/systemd/system/tomcat.service

Update the file with below 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=CATALINE_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

Reload systemd files

# systemctl daemon-reload

Start & Enable service

# systemctl start tomcat

# systemctl enable tomcat

Enabling the firewall and allowing 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 & DEPLOY (app01)

Download Source code

# git clone https://github.com/purveshshende2/multi-tier-web-application

Update configuration

# cd vprofile-project

# vim src/main/resources/application.properties

Update file with backend server details

Build code

Run below command inside the repository folder

# mvn install

# systemctl stop tomcat

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

# systemctl start tomcat

NOTE: For auto deployment create a script and execute using systemd. Below is the sample content for script

# vim /usr/local/bin/deploy.sh

# chmod +x /usr/local/bin/deploy.sh

# vi /etc/systemd/system/deploy.service

Add below content in deploy.sh

```
#!/bin/bash

cd /home/vagrant/vprofile-project

git pull

mvn clean install

systemctl stop tomcat

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

systemctl start tomcat
```

Add below content in deploy.service

```
[Unit]

Description=Tomcat

[Service]

Type=oneshot

User=root

ExecStart=/bin/bash /usr/local/bin/deploy.sh

[Install]
```

WantedBy=multi-user.target

Enable service

# systemctl daemon-reload

# systemctl enable deploy

Execute script

# systemctl start deploy

5. NGINX SETUP

Login to the nginx VM

$ vagrant ssh web01

$ sudo -i

Verify Hosts entry, if entries missing update it with IP and hostnames

# cat /etc/hosts

Update OS with latest patches

# yum update -y

Set Repository

# yum install epel-release -y

Install Nginx

# yum install nginx -y

Starting & Enabling the nginx service

# systemctl start nginx

# systemctl enable nginx

Take a backup of default configuration file

# cp /etc/nginx/nginx.conf /etc/nginx/nginx.conf-bkp

Update the configuration

# vim /etc/nginx/nginx.conf

Add below content in nginx.conf

user nginx;

```
worker_processes auto;

error_log /var/log/nginx/error.log;

pid /run/nginx.pid;


events {

worker_connections 1024;

}


http {

upstream vproapp {

   server app01:8080;

}


server {

   listen 80;


   location / {

       proxy_pass http://vproapp;

   }

}

}
```

Starting the firewall and allowing port 80 to access the nginx

```
# systemctl start firewalld

# systemctl enable firewalld

# firewall-cmd --get-active-zones
```

# firewall-cmd --zone=public --add-port=80/tcp --permanent

# firewall-cmd --reload

# systemctl restart nginx

after all setup you can verify from browser get the IP of web01 by running the command $ ip addr show

http:// ip : 80

to clean up

$vagrant destroy --force