

# VProfile Project Deployment Guide with Vagrant

## Overview

The VProfile application is a Java-based web app (often using Spring Boot) that simulates a virtual profile management system

## Prerequisites

Before we begin, make sure you have the following tools installed:

1. **VirtualBox** - <https://www.virtualbox.org/>
  2. **Vagrant** - <https://developer.hashicorp.com/vagrant/downloads>
  3. **Git** (optional but useful)
  4. **System Requirements**
    - At least **16 GB RAM**
    - Minimum **50 GB free disk space**
    - Virtualization enabled in BIOS
-

## Project Folder Structure

Make sure your directory looks like this:

markdown

CopyEdit

```
deploy-vagrant-multi-tier/
```

```
|— src
```

```
|— Vagrantfile
```

```
|— provisions/
```

```
    |— mariadb.sh
```

```
    |— memcached.sh
```


```
    |— rabbitmq.sh
```

```
    |— tomcat.sh
```

```
    |— nginx.sh
```

```
    |— prometheus.sh
```

```
    |— grafana.sh
```

 Ensure each `.sh` file is executable and properly sets up the respective service.

---

## Step-by-Step Deployment

### **1** Clone the Repository

Change directory into your root folder, create a new folder `src`, to clone the project.

```
cd deploy-vagrant-multi-tier
```

```
mkdir src
```

```
cd src
```

```
git clone https://github.com/Adutoby/vprofile-project/tree/master/src
```

---

## 2 Boot Up the Environment


Run this command in your project directory:

```
vagrant up
```

This will:

- Download the CentOS Stream 9 base box (if not already present)
- Create **7 virtual machines**:
  - `mariadb` (192.168.50.10)
  - `memcached` (192.168.50.11)
  - `rabbitMQ` (192.168.50.12)
  - `tomcat` (192.168.50.13)
  - `nginx` (192.168.50.14)
  - `prom` (192.168.50.15)
  - `graf` (192.168.50.16)

Each VM will be provisioned using the corresponding shell script inside the `provisions` folder.

 The setup might take some time, especially the first time.

---

## 3 Check Each Service

You can SSH into any VM with:

```
vagrant ssh <vm-name>
```

Examples:

```
vagrant ssh mariadb
```

```
vagrant ssh tomcat
```

Or check their respective services using browser access via **forwarded ports**:

Service	IP Address	Host Port	Guest Port
MariaDB	192.168.50.10	3306	3306
Memcached	192.168.50.11	11211	11211
RabbitMQ	192.168.50.12	5672	5672
Tomcat	192.168.50.13	8080	8080
Nginx	192.168.50.14	80	80
Prometheus	192.168.50.15	9090	9090
Grafana	192.168.50.16	3000	3000

Open your browser and go to:

- <http://localhost:8080> — Tomcat
- <http://localhost> — Nginx
- <http://localhost:9090> — Prometheus
- <http://localhost:3000> — Grafana

---

## 4 Deploy the vProfile Application


Your `tomcat.sh` script should:

- Install Tomcat
- Deploy the `vprofile.war` file into the `/webapps` folder
- Configure `application.properties` to connect with:
  - DB at `192.168.50.10`

- Memcached at 192.168.50.11
- RabbitMQ at 192.168.50.12

Ensure your `application.properties` includes:

```
db.host=192.168.50.10
db.port=3306
memcached.host=192.168.50.11
rabbitmq.address=192.168.50.12
```

 If you're deploying the `.war` manually, place it in `webapps/` inside the Tomcat server.

---

## 5 Validate the Application

Go to <http://localhost:8080/vprofile> or <http://localhost> (if routed via Nginx).

You should see the vProfile app login screen or dashboard.

---

## 6 Monitoring Setup (Prometheus & Grafana)

- Prometheus should be auto-configured to scrape metrics from each service. This should be handled in `prometheus.sh`.
- Grafana should be configured to use Prometheus as a data source.

You can access:

- Prometheus: <http://localhost:9090>
  - Grafana: <http://localhost:3000>
    - Default login: `admin / admin`
-

## Useful Vagrant Commands

```
vagrant status      # Check VM status
vagrant ssh <vm>    # SSH into a specific VM
vagrant halt        # Stop all VMs
vagrant destroy      # Remove all VMs
vagrant reload       # Restart VMs and apply changes
vagrant provision    # Re-run provisioning scripts
```

---

## Tips for Students

- Double-check IPs and service ports.
  - If you get errors on `vagrant up`, try running `vagrant destroy -f` and start fresh.
  - Make sure the provisioning scripts are correct and tested.
  - Use `tail -f /var/log/<service>.log` inside VMs to debug.
-