# Practice M8: Exam Preparation (AWS + Vagrant)

## Task

We are presented with a two-component Docker-ized application - **php+apache** and **mysql**

We are expected to build a complete infrastructure that includes the following hosts:

* **Ansible**
* **Docker**
* **Jenkins**
* **Nagios**

Application code is hosted on **GutHub** and available on the following URL:

<https://github.com/shekeriev/two-docker-images.git>

Our solution should do periodic checks (every two minutes) for code changes. If a change is registered, a new pair of images must be produced and then two new containers must be run out of those images

**Nagios** must be used to monitor all hosts by **PING** and **SSH**. Additionally, it must track if the containers are working as expected

## Possible Solution

Automatization and host provisioning will be implemented with the help of **Ansible** for all hosts except the **ansible.dob.lab** host which will be provisioned with **Vagrant**

After the basic provisioning is done, the **ansible** host will execute all playbooks automatically to prepare the rest of the infrastructure including adding the **docker** host as a slave node and adding the build task in **Jenkins**

All of the above will be orchestrated by Vagrant in AWS

### Preparation

#### Host

Extract the **M8-Practice-Exam-Prep-5-AWS-and-Vagrant.zip** file in a folder of your choice and navigate there

Explore the provided files

Main changes, compared to the files in variant #4, are in the **nagios** and **nrpe** roles, **ansible.sh** and **jenkins-add-credentials.sh** files

Please note, that all the necessary steps are adjusted to match the new situation (Red Hat-based AMI in AWS), but the main folder is still named **/vagrant** to retain maximum backward-compatibility with variants #1 to #4

Before we continue forward, we must adjust some of the extracted files

* Generate **new key pair** for AWS
* Change the **ansible/aws-key-pair.pem** with the new one
* Edit the **ansible/hosts** file to set the path to the new key
* Change the key in the **jenkins/credentials.xml** file
* Change both **access** and **security** keys in the **Vagrantfile**
* Change both the **name** and **path** to the **key pair** in the **Vagrantfile**

Then archive the **ansible**, **docker**, **jenkins** and **nagios** folders as for example **data.zip** file

Upload the file to a file share service which can be accessible publicly

Use for example the **Google Drive**

Next, change the path to the uploaded file in the **provision-ansible.sh** file

Note that because we want to have static addresses, we put all machines in a single availability zone and subnet

Note also that because we have not specified a security group, all machines will be attached to the default security group. Thus, we must allow the following set of ports **22/tcp**, **80/tcp**, **5666/tcp**, **8080/tcp** and **icmp**

Other specifics to this approach in comparison to others are:

* No modification of **firewall** rules on the machines as the **AMI** that we use does not have a **firewall** installed
* During the installation of **Jenkins** plugin **SSH** **Slaves** a particular version has been chosen as there is a known issue with **Jenkins 2.152** and **SSH Slaves 1.29** (<https://issues.jenkins-ci.org/browse/JENKINS-54708>)
* This may be obsolete with the recent (**2.299** at the time of writing this) version

### Execution

#### Host

We start the whole provisioning process with

**vagrant up --no-parallel**

After around 10 minutes we should have a working solution

We can monitor the whole process to some extent by establishing a session to the **ansible** machine

**vagrant ssh ansible**

And executing either one of the following commands - **ps ax** or **top** or follow the log with

**tail -f /tmp/ansible.log**

#### Job for the Actual Task (Solution)

In the **EC2** (AWS Console) console we must find the public IP address of the **Jenkins** machine

Open a browser tab and navigate to

http://<JENKINS-EXTERNAL-IP>:8080/

Use the credentials pair **admin** with password **admin**

Explore the created configuration

*We can commit a change to the* ***GitHub*** *project and see the result*

#### Host (check the Nagios monitoring)

In the **EC2** console we must find the public IP address of the **Nagios** host

Open a browser tab and navigate to

http://<NAGIOS-EXTERNAL-IP>/nagios

Use the credentials **nagiosadmin** with password **Password1**

Explore the configured hosts and services

#### Host (check the Docker application)

In the **EC2** console we must find the public IP address of the **Docker** host

Open a browser tab and navigate to

http://<DOCKER-EXTERNAL-IP>

We must see our containerized application

#### Clean Up

We can execute the following command

**vagrant destroy --force**

Then we must check in the **EC2** console and delete any volumes (section **Volumes**) of the deleted machines that may have left