# Practice M8: Exam Preparation (Alternative Approach)

## 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**

### Preparation

#### Host

Extract the **M8-Practice-Exam-Prep-2-Alt.zip** file in a folder of your choice and navigate there

Explore the provided files and adjust them if needed

Start the environment provisioning process with

**vagrant up**

Enter the **ansible** host

**vagrant ssh ansible**

#### Ansible

Navigate to the folder that contains all provision configurations

**cd /playbooks**

Explore the files

Start the installation process

**time ansible-playbook install-all.yml**

Once done, exit the **ansible** host

**exit**

#### Host

Enter the **docker** host

**vagrant ssh docker**

#### Docker

Check if we have connection to the **NRPE** plugin via the **localhost**:

**sudo /usr/lib64/nagios/plugins/check\_nrpe -H localhost**

Check if we have connection to the **NRPE** plugin via the internal **IP** address

**sudo /usr/lib64/nagios/plugins/check\_nrpe -H 192.168.99.103**

Let us test with an arbitrary image

**docker container run -d --name dob-web -p 80:80 php:7.0-apache**

And now, check if by using the **NRPE** plugin we can see the working containers

**sudo /usr/lib64/nagios/plugins/check\_nrpe -H 192.168.99.103 -c check-docker-container -a dob-web**

Stop and remove the container

**docker container rm -f dob-web**

Exit the **docker** host

**exit**

#### Host

Enter the **nagios** host

**vagrant ssh nagios**

#### Nagios Host

First, we must check if can connect to the **NRPE** plugin that is working on the **docker** host

**sudo /usr/lib64/nagios/plugins/check\_nrpe -H 192.168.99.103**

Everything should be fine

Exit the **nagios** host

**exit**

#### Host (working with Jenkins)

Open a browser tab on the host and navigate to

<http://localhost:8080>

Use the **admin** / **admin** credentials pair if asked

Usually, **Jenkins** is configured to be open when installed in an automated fashion

#### Add Credentials

Click on **Manage Jenkins** in the left menu

Then click on **Manage Credentials**

Next, click on **Jenkins** and then on **Global credentials (unrestricted)**

Then click on **Add Credentials** and fill in the following

* Kind: **Username with password**
* Scope: **Global**
* Username: **vagrant**
* Password: **vagrant**

Confirm with the **OK** button

#### Add Host

Return in the home page

Click on **Manage Jenkins** in the left menu and then on **Configure System**

Scroll down to the **SSH remote hosts** section and click on the **Add** button

Fill in the following data:

* Hostname: **docker.dob.lab**
* Port: **22**
* Credentials: **vagrant**

Test the connection with the **Check connection** button

Confirm and save the changes with the **Save** button

#### Test Job

Return to the home page

Select the **New Item** option in the left menu

Enter the following:

* Name: **Docker-Hello**
* Type: **Freestyle Project**

Confirm with the **OK** button

Scroll down to the **Build** section

Select **Execute shell script on remote host using ssh**

Enter **docker container run shekeriev/welcome-dob:2021** in the **Command** text box

Confirm with the **Save** button

Next, select the **Build Now** option in the left menu

Check the execution result

#### Add Slave Host

Return to the home page

Select the **Manage Jenkins** option in the left menu and then the **Manage Nodes and Clouds** command

Then click on the **New Node** link in the left menu

Enter **docker** in the **Node name** field and select the **Permanent Agent** option

Confirm with the **OK** button

Enter the following values:

* # of executors: **2**
* Remote root directory: **/vagrant**
* Labels: **docker**
* Usage: **Only build jobs with label expressions matching this node**
* Launch method: **Launch agents via SSH**
* Host: **docker.dob.lab**
* Credentials: **vagrant**
* Host Key Verification Strategy: **Non verifying Verification Strategy**

Confirm with the **Save** button

We can monitor the process of **Jenkins** agent deployment

#### Job Similar to the Target Job

Return to the home page

Click on the **New Item** option in the left menu

Enter or select the following:

* Name: **Docker-GitHub-Test**
* Type: **Freestyle Project**

Confirm with the **OK** button

Make the following adjustments:

* Set the project as **GitHub project**
* For **Project url** enter <https://github.com/shekeriev/simple-docker-image.git>
* Click on the second **Advanced** button
* Select the **Use custom workspace** option and enter **/vagrant/www-static**
* Select the **Restrict where this project can be run** option and for label set **docker**

Go to the **Source Code Management** section, select the **Git** option, and enter:

* <https://github.com/shekeriev/simple-docker-image.git>

Scroll down to the **Build** section and select **Execute shell**

In the **Command** text field enter:

* **cd /vagrant/www-static**
* **docker image build -t img-static-site .**

While in the **Build** section, click on the **Add build step** button and select **Execute shell**

Enter the following in the **Command** text box:

* **docker container rm -f co-static-site || true**
* **docker container run -d -p 80:80 --name co-static-site img-static-site**

Confirm with the **Save** button

Start the process by clicking on the **Build Now** link in the left menu

Open a browser tab on the host and navigate to the following URL:

<http://localhost:8082>

We should see a simple but working web app

#### Clean Up the Docker Host

Enter the **docker** host

**vagrant ssh docker**

Stop and remove the container we ran during the test

**docker container rm -f co-static-site**

Exit the **docker** host

**exit**

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

Return to the home page of **Jenkins**

Click on the **New Item** option in the left menu

Enter or select the following:

* Name: **Docker-GitHub-Final**
* Type: **Freestyle Project**

Confirm with the **OK** button

Do the following adjustments:

* Set the project type to **GitHub project**
* Enter the following URL: <https://github.com/shekeriev/two-docker-images.git>
* Click on the second button named **Advanced**
* Select the **Use custom workspace** option and enter: **/vagrant/www-dynamic**
* Turn on the **Restrict where this project can be run** option and for label set **docker**

Scroll down to the **Source Code Management** section, select **Git**, and enter:

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

Scroll down to the **Build** section and select **Execute shell**

In the **Command** text box enter:

* **cd /vagrant/www-dynamic/php**
* **docker image build -t img-php .**

While still in the **Build** section, click on the **Add build step** button, and select **Execute shell**

Enter the following:

* **cd /vagrant/www-dynamic/mysql**
* **docker image build -t img-mysql .**

While still in the **Build** section, click on the **Add build step** button, and select **Execute shell**

Enter the following:

* **docker container rm -f dob-http || true**
* **docker container run -d --net dob-network -p 80:80 --name dob-http -v /vagrant/www-dynamic/site:/var/www/html img-php**

Once more, click on the **Add build step** button and select **Execute shell**

Then enter the following:

* **docker container rm -f dob-mysql || true**
* **docker container run -d --net dob-network --name dob-mysql -e MYSQL\_ROOT\_PASSWORD=12345 img-mysql**

Confirm with the **Save** button

Start the building process with the **Build Now** option in the left menu

Monitor the execution process

There is one final touch missing - we have to check periodically for changes

We can click on the **Configure** option in the left menu to go to the project settings

Scroll down to the **Build Triggers** section

Turn on the **Poll SCM** option

In the **Schedule** text field enter **H/2 \* \* \* \*** to check every two minutes

Confirm with the **Save** button

*We can make a change in the* ***GitHub*** *project to see the effect*

#### Host (check the Nagios monitoring)

Open a browser tab on the host and navigate to the following URL:

<http://localhost:8081/nagios>

When asked for credentials use **nagiosadmin** and **Password1**

Explore the hosts, services, and the map

#### Host (check the Docker application)

Open a browser tab on the host and navigate to the following URL:

<http://localhost:8082>

We can see our working application