# Практика: DOB Модул 8 (алтернативен подход)

## Задание

Двукомпонентно Docker-изирано приложение - php+apache и mysql.

Да се построи цялостна инфраструктура, включваща следните хостове:

* Ansible
* Docker
* Jenkins
* Nagios

Проектът за приложението е в GutHub:

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

Да се прави проверка периодично (на всеки 2 мин.) за промени и при наличие на такива, да се генерират нови шаблони и от тях нови контейнери.

Nagios да наблюдава базови показатели като PING и SSH на всички хостове. За контейнерите - да следи дали работят.

## Възможни стъпки

Автоматизацията на провизирането и конфигурирането на хостовете ще бъде изпълнена с помощта на Ansible за всички хостове без хоста ansible.sulab.local, чието провизиране ще бъде извършено с помощта на Vagrant.

### Подготовка

#### Host

Разархивираме изтегления архив и влизаме в съответната папка, примерно:

**cd /home/devops/DOB-Practice-Prep-Alt**

Стартираме процеса по създаване на машините и първоначалното им базово конфигуриране:

**time vagrant up**

Влизаме в **ansible** хоста:

**vagrant ssh ansible**

#### Ansible

Отиваме в папката, съдържаща всички инструкции за провизиране и конфигуриране на другите хостове:

**cd /playbooks**

Стартираме процеса по инсталация:

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

Излизаме от **ansible** хоста:

**exit**

#### Host

Влизаме на **docker** хоста:

**vagrant ssh docker**

#### Docker

Проверяваме дали имаме връзка към **NRPE** плъгина през **localhost**:

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

Проверяваме дали имаме връзка към **NRPE** плъгина през външното **IP**:

**sudo /usr/lib64/nagios/plugins/check\_nrpe -H 192.168.82.103**

Изтегляме шаблон и стартираме контейнер:

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

Проверяваме дали през **NRPE** пъгина виждаме работещия контейнер:

**sudo /usr/lib64/nagios/plugins/check\_nrpe -H 192.168.82.103 -c check-docker-container -a dob-web**

Спираме контейнера:

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

Излизаме от **docker** хоста:

**exit**

#### Host

Влизаме на **nagios** хоста:

**vagrant ssh nagios**

#### Nagios Host

Проверяваме дали имаме връзка към **NRPE** плъгина, работещ на **docker** хоста:

**sudo /usr/lib64/nagios/plugins/check\_nrpe -H 192.168.82.103**

Излизаме от **nagios** хоста:

**exit**

#### Host (работа с Jenkins)

В браузър на локалния хост, отваряме сесия към:

<http://localhost:8080>

Удостоверяваме се с **admin** / **admin**

#### Добавяне на данни за удостоверяване

От менюто вляво избираме **Credentials**

След това кликваме върху **System** и най-накрая върху **Global credentials (unrestricted)**

После кликваме върху **Add Credentials** и попълваме следното:

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

И потвърждаваме с бутона **OK**.

#### Добавяне на данни за хост

Отиваме в главното меню и оттам избираме **Manage Jenkins** и после **Configure System**

В секцията **SSH remote hosts** кликваме върху бутона **Add**

Попълваме следните данни:

* Hostname: **docker.sulab.local**
* Port: **22**
* Credentials: **vagrant**

Тестваме връзката с бутона **Check connection**

Запазваме промените с бутона **Save**

#### Създаване на задача за проверка

От главното меню избираме **New Item**

Като тип и име на проекта избираме:

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

Потвърждаваме с бутона **OK**

В секцията **Build Environment** маркираме **Execute shell script on remote host using ssh** и избираме хоста **docker.sulab.local**

В секцията Build избираме **Execute shell script on remote host using ssh** и като команда въвеждаме **docker container run hello-world**

Потвърждаваме с бутона **Save**

След това от менюто вляво избираме опцията **Build Now**

Проверяваме резултата от изпълнението

#### Добавяне на подчинен хост

От главното меню избираме опцията **Manage Jenkins** и после **Manage Nodes**

След това в лявото меню кликваме върху командата **New Node**

Като име попълваме **docker** и маркираме опцията **Permanent Agent**

Потвърждаваме с бутона **OK**

Попълваме следните параметри:

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

Потвърждаваме с бутона **Save**

Можем да наблюдаваме процеса на инсталиране на **Jenkins** агента

#### Тестово решение на подобна на поставената задача

От главното меню избираме **New Item**

Като тип и име на проекта избираме:

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

Потвърждаваме с бутона **OK**

Правим следните допълнения/настройки:

* Маркираме проекта като **GitHub project**
* Като URL: <https://github.com/shekeriev/simple-docker-image.git>
* Натискаме втория бутон **Advanced**, в опцията **Use custom workspace** и въвеждаме: **/vagrant/www-static**
* Включваме опцията **Restrict where this project can be run** и като етикет избираме **docker**

В секцията **Source Code Management** маркираме **Git** и въвеждаме:

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

В секцията **Build** избираме **Execute shell** и като команди въвеждаме:

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

Отново в секцията **Build**, кликваме върху бутона **Add build step** и избираме **Execute shell**. Въвеждаме:

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

Запазваме с бутона **Save**

Стартираме с бутона **Build Now**

Стартираме браузър и въвеждаме адреса <http://localhost:8082> , за да проверим дали всичко работи

#### Почистваме docker хоста

Влизаме в **docker** хоста

**vagrant ssh docker**

Спираме и изтриваме създадения по-рано контейнер

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

Излизаме от **docker** хоста

**exit**

#### Решение на поставената задача

От главното меню избираме **New Item**

Като тип и име на проекта избираме:

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

Потвърждаваме с бутона **OK**

Правим следните допълнения/настройки:

* Маркираме проекта като **GitHub project**
* Като URL: <https://github.com/shekeriev/two-docker-images.git>
* Натискаме втория бутон **Advanced**, в опцията **Use custom workspace** и въвеждаме: **/vagrant/www-dynamic**
* Включваме опцията **Restrict where this project can be run** и като етикет избираме **docker**

В секцията **Source Code Management** маркираме **Git** и въвеждаме:

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

В секцията **Build** избираме **Execute shell** и като команди въвеждаме:

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

Отново в секцията **Build**, кликваме върху бутона **Add build step** и избираме **Execute shell**. Въвеждаме:

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

Отново в секцията **Build**, кликваме върху бутона **Add build step** и избираме **Execute shell**. Въвеждаме:

* **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**

Отново в секцията **Build**, кликваме върху бутона **Add build step** и избираме **Execute shell**. Въвеждаме:

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

Запазваме с бутона **Save**

Стартираме с бутона **Build Now**

Наблюдаваме процеса на изпълнение

Сега можем от опцията **Configure** в лявото меню да се върнем в настройките на проекта и да добавим проверката по график

В секцията **Build Triggers** включваме опцията **Poll SCM**

В полето **Schedule** задаваме **H/2 \* \* \* \***

Запазваме промените с бутона **Save**

*Можем да направим промяна в* ***GitHub*** *и да наблюдаваме ефекта*

#### Host (преглед на Nagios)

Отваряме сесия в браузъра на локалния хост със следния адрес:

<http://localhost:8081/nagios>

Въвеждаме потребител **nagiosadmin** и парола **Password1**

Преглеждаме наблюдаваните услуги и хостове

#### Host (преглед на docker приложението)

Отваряме сесия в браузъра на локалния хост със следния адрес:

<http://localhost:8082>

Проверяваме работещото уеб приложение