# Практика: DOB Модул 5

## Подготовка на средата

Създаваме папка на хоста и влизаме в нея:

**mkdir DOB-Jenkins**

**cd DOB-Jenkins**

Разархивираме изтегленият файл тук и изпълняваме командата:

**cp Vagrantfile-1 Vagrantfile**

За да подготвим средата за първата част от упражненията. Можем да разгледаме съдържанието му:

**cat Vagrantfile**

Същинското вдигане на средата започва с изпълняването на командата:

**vagrant up**

След като се стартира машината, изпълняваме:

**vagrant ssh master**

## Първа част

#### Инсталация на Jenkins

За да изтеглим описание на инсталационното хранилище, трябва да изпълним командата:

**sudo wget https://pkg.jenkins.io/redhat/jenkins.repo -O /etc/yum.repos.d/jenkins.repo**

След това трябва да изпълним:

**sudo rpm --import https://pkg.jenkins.io/redhat/jenkins.io.key**

Преди да пристъпим към инсталация трябва да опресним дефинициите:

**sudo yum update**

*Бихме могли да изберем друго хранилище, съдържащо само т.нар. стабилни или версии с удължена поддръжка. Това става като в горните два адреса подменим* ***redhat*** *с* ***redhat-stable****. По аналогичен начин, ако искаме да направим инсталация върху друга фамилия от дистрибуции, трябва да заменим* ***redhat*** *с* ***debian*** *или* ***opensuse****.*

*Алтернативен метод на инсталация е варианта да изтеглим и инсталираме конкретен пакет от следния адрес:* ***https://jenkins.io/download/***

*Ако искаме да не последната, а конкретна версия на* ***Jenkins****, можем да проверим, кои са наличните версии, като изпълним:* ***sudo yum list --show-duplicates jenkins***

*И след това да инсталираме желаната версия, примерно:* ***sudo yum install jenkins-2.60-1.1***

Тъй като ние искаме да работим с последната версия на **Jenkins**, ще изпълним:

**sudo yum install jenkins**

*Добра практика е след инсталация да се забрани дефиницията за инсталационното хранилище на* ***Jenkins****, за да не се окажем изненадани от някой ъпдейт. Така контролът върху това, коя версия ще се използва остава у нас. Забраната става с командата:* ***sudo yum-config-manager --disable jenkins***

След като приключи инсталацията, можем да инсталираме и **Java**:

**sudo yum install java-1.8.0-openjdk**

Сега вече можем да пристъпим към стартиране на **Jenkins** услугата:

**sudo systemctl start jenkins**

Нека проверим дали се е стартирала успешно:

**sudo systemctl status jenkins**

И накрая да разрешим стартирането ѝ при стартиране на системата:

**sudo systemctl enable jenkins**

За да можем да работим с **Jenkins** от браузъра на нашия хост, трябва да разрешим порт **8080** да минава през защитната стена. Това става като изпълним последователно:

**sudo firewall-cmd --permanent --zone=public --add-port=8080/tcp**

**sudo firewall-cmd --reload**

*В зависимост от настройките и режима на работа на* ***SELinux*** *на нашата машина, може да се наложи да изпълним следната поредица от команди:*

***sudo yum install selinux-policy-devel setroubleshoot-server***

***sepolicy network -t http\_port\_t***

***sudo semanage port -a -t http\_port\_t -p tcp 8080***

***sepolicy network -t http\_port\_t***

Сега можем да стартираме браузъра на нашата станция и да въведем следния адрес:

**http://localhost:8080**

От заредената страница разбираме, че преди да бъде възможно да работим с **Jenkins**, трябва да го отключим. Това става като копираме и поставим съдържанието на указания файл в празното поле и потвърдим с бутона **Continue**. Нека в конзолата на **master** машината да изпълним:

**su -**

Паролата е **vagrant**. Сега нека да въведем:

**cd /var/lib/jenkins/secrets**

И след това нека да изпълним:

**cat initialAdminPassword**

И да излезем от **root** сесията с:

**exit**

Разбира се горните три стъпки могат да бъдат заменени с изпълнението на следната команда:

**sudo cat /var/lib/jenkins/secrets/initialAdminPassword**

Независимо по кой път, копираме кода и го поставяме в полето на отворената в браузъра страница. Потвърждаваме с бутона **Continue**.

На следващия екран избираме опцията „**Install suggested plugins**”.

След като приключи процеса на инсталация на добавките, ни се показва екран, в който трябва да настроим потребителя, който ще има пълни права върху инсталацията. Нека да въведем следните данни:

- Username: **doadmin**

- Password: **Password1**

- Full name: **DevOps Administrator**

- E-mail address: **doadmin@sulab.local**

И да потвърдим с бутона **Save and Finish**. След това нека да натиснем бутона **Start using Jenkins**.

Можем да разгледаме системата, след което да излезем, тъй като трябва да се направят още няколко настройки.

#### Допълнителни настройки на Jenkins

Нека да изпълним следната команда в конзолата на **master** машината, за да видим как е създаден служебния потребител:

**cat /etc/passwd | grep jenkins**

Виждаме, че обвивката му е установена по начин, който ще ни попречи да използваме този потребител за изпълнение на задачи. Нека да коригираме това, като изпълним:

**sudo vi /etc/passwd**

И да сменим **/bin/flase** на **/bin/bash** след което да запазим промените и да излезем. Сега нека стартираме **root** сесия:

**su -**

Нека да изпълним следната команда, за да установим парола (**Password1**) за потребителя **jenkins**:

**passwd jenkins**

Сега можем да излезем от **root** сесията и да влезем в сесия на **jenkins**:

**exit**

**su - jenkins**

Нека последователно да изпълним следните команди:

**pwd**

**ssh-keygen**

**ssh-copy-id jenkins@localhost**

По време на генерирането на ключове за целите на теста, ще оставим всичко по подразбиране и няма да въвеждаме парола. При инсталирането на ключовете, трябва да въведем паролата на **jenkins**, която зададохме малко по-рано. Нека сега да проверим дали връзка по **ssh** ще работи без парола:

**ssh jenkins@localhost**

Не би трябвало да ни бъде искана парола. Сега нека да излезем от **ssh** сесията и после от **jenkins** сесията:

**exit**

**exit**

Сега нека да дадем **sudo** права на **jenkins** потребителя, като изпълним:

**sudo visudo**

И после въведем:

**jenkins ALL=(ALL) NOPASSWD: ALL**

След като запазим и излезем, нека да рестартираме **jenkins** услугата:

**sudo systemctl restart jenkins**

#### Настройка на сигурността и добавяне на потребител

Нека да отворим браузър на нашия хост и да въведем адреса на **Jenkins**:

<http://localhost:8080>

След това нека да се удостоверим като **doadmin**. После нека да отидем на **Manage Jenkins**, а след това и на **Configure Global Security**. Нека в секцията **Authorization** да изберем **Matrix-based security**. После трябва да добавим потребителя **doadmin** и да му дадем всички права. Потвърждаваме с бутона **Save**.

Нека сега в **Manage Jenkins** да изберем **Manage Users** и след това **Create User**. В появилия се екран нека да въведем:

- Username: **douser**

- Password: **Password2**

- Full name: **DevOps User**

- E-mail address: **douser@sulab.local**

Потвърждаваме с бутона **Create User**.

Нека от **Manage Jenkins** отново да изберем **Configure Global Security**. Нека сега да добавим потребителя **douser** и да му дадем следните права:

**- Overall - Read**

**- Job - Build**

**- Job - Cancel**

**- Job - Configure**

**- Job - Discover**

**- Job - Read**

**- Job - Workspace**

**- Run - Update**

**- View - Configure**

**- View - Create**

**- View - Read**

Потвърждаваме с бутона **Save**.

Нека сега да излезем и да влезем като новия потребител. Има разлика в опциите, които вижда. Нека да излезем отново.

#### Създаване и изпълнение на локален Build

Нека да отворим браузър на нашия хост и да въведем адреса на **Jenkins**:

[**http://localhost:8080**](http://localhost:8080)

След това нека да се удостоверим като **doadmin**. Нека сега да изберем **New Item**, след което да маркираме **Folder**. Като име задаваме **DOB-Demo** и потвърждаваме с бутона **OK** и после бутона **Save**.

Нека сега да изберем отново **New Item**, да укажем **Freestyle** **project** и като име да зададем **Get-All-Processes**. Потвърждаваме с **OK**.

Скролираме до секцията **Build** и избираме **Execute shell**. В полето **Command** въвеждаме:

**ps ax**

Потвърждаваме с бутона **Save**.

От менюто вляво избираме опцията **Build Now**.

Проверяваме резултата от изпълнението като кликнем на номерчето в секцията **Build History** и после на **Console Output**.

*В зависимост от настройките на дистрибуцията на* ***Jenkins*** *хоста, може да се наложи следната корекция:*

***sudo visudo***

*Сменяме:*

***Defaults !requiretty***

## Втора част

#### Настройка на отдалечен достъп

Излизаме от **ssh** сесията на **master**:

**exit**

Сега подменяме **Vagrantfile**:

**cp Vagrantfile-2 Vagrantfile**

И изпълняваме:

**vagrant up**

Сега вече трябва да имаме две машини - **master** и **client**.

Нека да конфигурираме безпрепятствения достъп на **jenkins** потребителя до клиентската станция. За целта:

**vagrant ssh client**

Сега ще създадем потребител **jenkins** с парола **Password1** и ще му дадем **sudo** привилегии:

**sudo useradd jenkins**

**sudo passwd jenkins**

**sudo visudo**

Добавяме:

**jenkins ALL=(ALL) NOPASSWD: ALL**

Записваме и излизаме. След това излизаме от сесията на клиента и влизаме на **master**:

**exit**

**vagrant ssh master**

Превключваме в сесия на потребителя **jenkins**:

**su - jenkins**

Добавяме ключ за клиентската станция:

**ssh-copy-id jenkins@client.sulab.local**

Тестваме връзката:

**ssh jenkins@client.sulab.local**

Затваряме тестовата връзка:

**exit**

Излизаме от сесията на **jenkins**:

**exit**

#### Инсталиране на добавки (plugins)

Нека да отворим браузър на нашия хост и да въведем адреса на **Jenkins**:

[**http://localhost:8080**](http://localhost:8080)

След това нека да се удостоверим като **doadmin**. Нека сега да изберем **Manage Jenkins** и после **Manage Plugins**. После превключваме на **Available** и в полето за филтър въвеждаме **ssh**. Маркираме **SSH plugin** и потвърждаваме с бутона **Download now and install after restart**.

След като се рестартира **Jenkins** ще трябва да се удостоверим наново. Нека влезем като **doadmin**.

#### Данни за удостоверяване

Нека да дефинираме данни за удостоверяване. Кликваме върху **Manage Jenkins** или отиваме на началната му страница. После избираме **Credentials**, след това **Jenkins** и най-накрая **Global credentials**.

Сега от менюто вляво избираме **Add Credentials** и въвеждаме:

**Username with password**

- Username: **jenkins**

- Password: **Password1**

- Description: **Local user with password**

Потвърждаваме с бутона **OK** и отново избираме **Add Credentials**, като този път въвеждаме:

**SSH Username with private key**

- Username: **jenkins**

- Private Key: **From the Jenkins master ~/.ssh**

- Description: **Credentials from file**

Потвърждаваме с бутона **OK**.

#### Настройка на SSH хостове

Нека отново от началната страница да изберем **Manage Jenkins** и после **Configure System**.

В секцията **SSH remote hosts / SSH sites** кликваме върху бутона **Add** и въвеждаме:

- Hostname: **localhost**

- Port: **22**

- Credentials: **jenkins (Local user with password)**

Проверяваме връзката с **Check connection** и добавяме втори хост с бутона **Add**:

- Hostname: **client.sulab.local**

- Port: **22**

- Credentials: **jenkins (Credentials from file)**

Отново тестваме с **Check connection** и потвърждаваме с бутона **Save**

#### Изпълнение на отдалечена задача

Връщаме се в папката **DOB-Demo**. От менюто вляво избираме **New Item**, после **FreeStyle project**, а като име задаваме **Get-All-Processes-Remote** и потвърждаваме с **OK**.

Скролираме до секцията **Build** и избираме **Execute shell script on remote host using ssh**. В полето **Command** въвеждаме:

**hostname**

**ps ax**

За хост избираме клиентската станция и маркираме опцията **Execute each line**.

Потвърждаваме с бутона **Save**.

От менюто вляво избираме опцията **Build Now**.

Проверяваме резултата от изпълнението като кликнем на номерчето в секцията **Build History** и после на **Console Output**.

#### Подготовка за работа с GitHub

На **master** хоста изпълняваме

**sudo yum install git**

После създаваме папка и сменяме собственика ѝ:

**sudo mkdir -p /projects/www-static**

**sudo chown -R jenkins:jenkins /projects**

Инсталираме Apache уеб сървър на хоста и го стартираме:

**sudo yum install httpd**

**sudo systemctl enable httpd**

**sudo systemctl start httpd**

#### Работа с GitHub

Отиваме в папката **DOB-Demo.** От менюто вляво избираме **New Item**, като тип задаваме **FreeStyle Project**, а като име - **GitHub-Build**. Потвърждаваме с **OK**.

Маркираме опцията **GitHub project**. В полето **Project URL** въвеждаме: **https://github.com/shekeriev/simple-html-page.git**

Натискаме бутона **Advanced** и в полето **Display Name** въвеждаме: **GitHub Static Web Page**

Още веднъж натискаме бутона **Advanced**, маркираме опцията **Use custom workspace** и в полето **Directory** въвеждаме: **/projects/www-static**

В секцията **Source Code Management** маркираме опцията **Git**. После в полето **Repository URL** въвеждаме: **https://github.com/shekeriev/simple-html-page.git**

Потвърждаваме с бутона **Save**

Стартираме задачата с **Build Now** и проверяваме резултатите.

#### Работа с GitHub #2

Отиваме в папката **DOB-Demo.** От менюто вляво избираме **New Item**, в полето **Copy from** **въвеждаме GitHub-Build**, а в полето за ново име **GitHub-Build-Post** и потвърждаваме с **OK**.

Добавяме нова **Build** стъпка - **Execute shell**

В полето **Command** въвеждаме:

**sudo cp /projects/www-static/\*.html /var/www/html && sudo cp /projects/www-static/\*.png /var/www/html**

или

**sudo cp -t /var/www/html /projects/www-static/\*.html /projects/www-static/\*.png**

Потвърждаваме с бутона **Save** и изпълняваме с бутона **Build Now**. Проверяваме резултатите.

#### Работа по график

От главното меню избираме **Manage Jenkin**s, после **Manage Plugins**. След това избираме **Available** и в полето за търсене въвеждаме: **schedule**

Маркираме добавката **Schedule Build Plugin** и потвърждаваме с бутона **Download now and install after restart**

След като се рестартира системата, отиваме в папката **DOB-Demo**, избираме **New Item**, после **FreeStyle Project**. Като име задаваме **Get-Processes-Count-Scheduled** и потвърждаваме с **OK**.

Като описание въвеждаме: **Get the running processes count at a certain point in the future**

Добавяме **Build** стъпка от тип **Execute shell**. В полето за команда въвеждаме: **hostname && ps ax | wc -l**

Потвърждаваме с бутона **Save** и изпълняваме със **Schedule Build**

Избираме момент от времето след около минута спрямо текущото време. Потвърждаваме със **Schedule**. Ако възникне проблем пренасочването трябва да изтрием последната част от адреса.

Наблюдаваме процеса на изпълнение и резултатите.

#### Работа по график #2

Отиваме в секцията **Configure** на предния процес. В секцията **Build Triggers** избираме **Build periodically** и в полето **Schedule** въвеждаме **H/2 \* \* \* \***

Потвърждаваме с бутона **Save** и наблюдаваме какво се случва. След това проверяваме резултатите.

Преди да преминем нататък, трябва да спрем графика. Отиваме в **Configure** и изтриваме съдържанието на полето **Schedule**, след което изключваме опцията **Build periodically**. Потвърждаваме с бутона **Save**.

#### Работа по график с GitHub

Отиваме в папката **DOB-Demo**. Правим копие на проекта **GitHub-Build-Post** като **GitHub-Build-Trigger**

Отваряме новия проект и в секцията **Build Triggers** маркираме **Poll SCM**. В полето **Schedule** въвеждаме **\* \* \* \* \***

Потвърждаваме с бутона **Save**.

Правим промяна в **GitHub** и наблюдаваме.

Преди да преминем нататък, трябва да спрем графика. Отиваме в **Configure** и изтриваме съдържанието на полето **Schedule**, след което изключваме опцията **Poll SCM**. Потвърждаваме с бутона **Save**.

## Трета част

Първо излизаме от **ssh** сесията на **master**:

**exit**

Сега подменяме **Vagrantfile**:

**cp Vagrantfile-3 Vagrantfile**

И изпълняваме:

**vagrant up**

Сега вече трябва да имаме три машини - **master**, **client** и **slave**

Нека да конфигурираме безпрепятствения достъп на **jenkins** потребителя до новата станция. За целта:

**vagrant ssh slave**

Сега ще създадем потребител **jenkins** с парола **Password1** и ще му дадем **sudo** привилегии:

**sudo useradd jenkins**

**sudo passwd jenkins**

**sudo visudo**

Добавяме:

**jenkins ALL=(ALL) NOPASSWD: ALL**

Записваме и излизаме. Създаваме папка в профила на потребителя **jenkins**:

**su - jenkins**

**mkdir workspace**

**exit**

После инсталираме **java**:

**sudo yum install java-1.8.0-openjdk**

След това излизаме от сесията на станцията и влизаме на **master**:

**exit**

**vagrant ssh master**

Превключваме в сесия на потребителя **jenkins**:

**su - jenkins**

Добавяме ключ за клиентската станция:

**ssh-copy-id jenkins@slave.sulab.local**

Тестваме връзката:

**ssh jenkins@slave.sulab.local**

Затваряме тестовата връзка:

**exit**

Излизаме от сесията на **jenkins**:

**exit**

Накрая излизаме и от **master** хоста:

**exit**

Сега вече сме готови да добавим реално хоста в **Jenkins**. За целта отваряме интерфейса на **Jenikins**, избираме **Manage Jenkins** и после **Manage Nodes**. След това **New Node**.

Като име задаваме **slave-node** , маркираме **Check Permanent Agent** и потвърждаваме с **OK**

После въвеждаме като описание **DevOps test slave node** и последователно останалите параметри:

# of executors: **4**

Remote root directory: **/home/jenkins/workspace**

Labels: **slave-node**

Usage: **Only build jobs with label expression matching this node**

Launch method: **Launch slave agents via SSH**

Host: **slave.sulab.local**

Credentials: **jenkins (Credentials from file)**

Host Key Verification Strategy: **Known hosts file**

Потвърждаваме с бутона **Save**. След това кликваме на новия хост, а после и на логовете за него.

#### Изпълнение на задача на подчинен хост (локално)

Отиваме в папката **DOB-Demo**, избираме **New Item**. После избираме **FreeStyle Project** и като име задаваме **Slave-Local**. Потвърждаваме с бутона **OK**.

За описание задаваме **Execute project locally on a slave node**

Маркираме опцията **Restrict where this project can run** и задаваме етикет **slave-node**

Добавяме **Build** стъпка от тип **Execute shell**. В полето **Command** въвеждаме **hostname && ps ax**

Потвърждаваме с бутона **Save**. След това изпълняваме с **Build Now** и проверяваме резултатите.

#### Изпълнение на задача на подчинен хост (отдалечено)

За да подготвим нещата, изпълняваме

**vagrant ssh slave**

**su - jenkins**

**ssh-keygen**

**ssh-copy-id jenkins@master.sulab.local**

Проверяваме

**ssh master.sulab.local**

**exit**

Излизаме от **jenkins** сесията

**Exit**

Излизаме и от **master** хоста.

След това отиваме в браузъра - **Manage Jenkins**, **Configure System**

В секцията **SSH remote hosts /** **SSH sites** добавяме:

Първо:

- hostname: **master.sulab.local**

- port: **22**

- credentials: **jenkins (Credentials from file)**

И после:

- hostname: **slave.sulab.local**

- port: **22**

- credentials: **jenkins (Credentials from file)**

Проверяваме последователно връзките с **Check connection**

Потвърждаваме с бутона **Save**

Вече сме готови за изпълнение и на отдалечената задача.

Отиваме **DOB-Demo**, избираме **New Item** и после създаваме **FreeStyle Project** с име **Slave-Remote**

Като описание задаваме **Execute project on a slave node against remote machine**

Маркираме **Restrict where this project can run и** задаваме етикет **slave-node**

Добавяме нова **Build** стъпка от тип **Execute shell script on remote host using ssh**

Като хост задаваме **master.sulab.local**

В полето **Command** въвеждаме **hostname && ps ax**

Потвърждаваме с бутона **Save**. Изпълняваме с **Build Now** и проверяваме резултата.

#### Изпълнение на задача на подчинен хост (Docker)

За да инсталираме **Docker** на подчинения хост, трябва да:

**vagrant ssh slave**

Превключваме в сесия на потребителя **jenkins:**

**su - jenkins**

След това пристъпваме към изпълнение на познатата вече процедура по подготовка и инсталация на **docker**.

Първо инсталираме съпътстващите компоненти:

**sudo yum install -y yum-utils device-mapper-persistent-data lvm2**

След това добавяме съответното хранилище:

**sudo yum-config-manager \**

**--add-repo \**

**https://download.docker.com/linux/centos/docker-ce.repo**

Инсталираме самия **docker**:

**sudo yum makecache fast**

**sudo yum install docker-ce**

Правим няколко допълнителни настройки:

**sudo mkdir -p /etc/docker**

**sudo touch /etc/docker/daemon.json**

**sudo vi /etc/docker/daemon.json**

Добавяме:

**{**

**"storage-driver": "devicemapper"**

**}**

Стартираме услугата:

**sudo systemctl start docker**

**sudo systemctl status docker**

**sudo systemctl enable docker**

Правим проверка дали всичко работи:

**sudo docker version**

**sudo docker system info**

Правим **jenkins** потребителя част от **docker** групата:

**sudo usermod -aG docker $USER**

Накрая излизаме от сесията на **jenkins**:

**exit**

и после от **slave** машината:

**exit**

След това отиваме на **master** хоста:

**vagrant ssh master**

**sudo systemctl restart jenkins**

Нека сега да създадем същинския проект на **Docker**

Отиваме в папка **DOB-Demo** и създаваме нов **FreeStyle** проект с име **Docker-Hello-World**

Description => **Docker hello world on a slave node**

Check option => **Restrict where this project can be run**

Add build step => **Execute shell**

In Command filed enter => **docker container run shekeriev/welcome-dob-oct-17**

Запазваме със **Save** , стартираме с **Build Now** и проверяваме резултатите

#### Втори Docker проект

Първо подготвяме нещата на подчинения хост:

**sudo yum install git**

**sudo mkdir -p /projects/www-static**

**sudo chown -R jenkins:jenkins /projects**

Отиваме в папката **DOB-Demo** и създаваме нов проект от тип **FreeStyle Project** с име **Docker-GitHub**. Като описание задаваме **Build and run Docker container from GitHub**

Маркираме опцията **GitHub project**

Задаваме **Project url** като **https://github.com/shekeriev/simple-docker-image.git**

Включваме опцията **Restrict where this project can be run** и задаваме етикет **slave-node**

Check option => **Use custom workspace**

Enter => **/projects/www-static**

Check option => **Git (in Source Code Management)**

Set Repository URL => **https://github.com/shekeriev/simple-docker-image.git**

Add Build Step => **Execute shell**

In Command enter:

**cd /projects/www-static**

**docker image build -t img-static-site .**

Add Second Build Step => **Execute shell**

In Command enter:

**docker container rm co-static-site || true**

**docker container run -d -p 80:80 --name co-static-site img-static-site**

Запазваме със **Save** и изпълняваме с **Build Now**

Можем да проверим резултата, като посетим: **http://localhost:8088**