# Practice M1: Introduction to DevOps

The lab environment has the following structure:

Graphical user interface, application

Description automatically generated

The following tasks are executed on different machines. It is stated clearly on which machine a task is being executed

VM templates can be downloaded from this URL: <https://zahariev.pro/go/do1-templates>

Do not hurry to download or install anything, wait until you reach the appropriate section

## Part 1: Introduction

Nothing to do here, just talking/discussing 😊

## Part 2: Manual approach

### Preparation

We must ensure that we have a working and supported hypervisor installed

**VirtualBox** is **the preferred option** as is it both free and well supported by **Vagrant**

Of course, other virtualization solutions like **VMware Workstation** and **Hyper-V** can do the job as well but will require some tweaks

Be sure to fill the **Introduction Form** (<https://zahariev.pro/go/do1-form>) and state at least your configuration – hardware, OS and virtualization solution. This will help to adapt the content to better fit the majority

Until the end of this practice, we will be working with **VirtualBox**

*Please make sure that no other virtualization solutions are installed and active (this includes also Docker Desktop as it relies on an existing hypervisor)*

*If there are any, please uninstall them (or just go with them)*

*If you already have VirtualBox installed, skip the following section*

#### Install VirtualBox (on a Windows host)

Navigate to the download page (<https://www.virtualbox.org/wiki/Downloads>) and download the latest version

Make sure that you have administrative access and double-click on the downloaded file to initiate the installation process

As you will see it is pretty straight-forward

#### Install VirtualBox (on a Red Hat-compatible host)

The steps are more or less the same no matter which **Linux** distribution is installed on the host

*In any case, check the official documentation:* [*https://www.virtualbox.org/wiki/Linux\_Downloads*](https://www.virtualbox.org/wiki/Linux_Downloads)

Open a terminal session

Download the repository configuration file

**wget https://download.virtualbox.org/virtualbox/rpm/el/virtualbox.repo**

*Even though Fedora is part of the same family, for it the above should become*

***wget https://download.virtualbox.org/virtualbox/rpm/fedora/virtualbox.repo***

*So, as advised above, check the official documentation 😉*

Move it the appropriate target folder

**sudo mv virtualbox.repo /etc/yum.repos.d/**

Update currently installed packages

**sudo dnf upgrade -y**

Install any prerequisite packages that may be missing

**sudo dnf install -y gcc make kernel-headers kernel-devel**

Install the **VirtualBox** package

**sudo dnf install -y VirtualBox-7.0**

Add the current user to the appropriate user group

**sudo usermod -a -G vboxusers $USER**

Log-out and log-on again

Check that **VirtualBox** is correctly installed and working

*Should you see any errors when trying to start a VM, you may need to run*

***sudo /sbin/vboxconfig***

*If the above end with an error, check the* ***/var/log/vbox-setup.log*** *file*

*Most probably you will have to install additional packages, for example*

***sudo dnf install elfutils-libelf-devel***

*And re-run the* ***/sbin/vboxconfig*** *command*

*The actual steps vary depending on the distribution in use and if the system is BIOS or UEFI-based*

### Application Deployment

Now, that we have the infrastructure set up, we can continue with the application deployment steps

Let’s imagine that we have to deploy a two-tier application with web-based (PHP) front-end and MariaDB/MySQL used for database

Following established practices, we will deploy the two components on two different machines (you can refer to the picture in the beginning of the document)

#### Import a template (on the host)

*We will be using a template to save some time. Of course, should we want, we can start fresh and install the OS in the virtual machines manually*

For the rest of the lab, we can use either one of the available templates – **CentOS**, **Debian**, or **openSUSE**

Let’s use the **CentOS** one (in fact, it is a **CentOS Stream 9** template). Should you want to experiment with another one, be sure to adjust the steps

Start **VirtualBox**

Import **centos-stream-9.ova** as **WEB** machine

Import **centos-stream-9.ova** as **DB** machine

Set networking mode for the first (and only) network adapter to **NAT \***

Add a second network adapter \*\* to both virtual machines and set it to **Internal Network** mode. Make sure that the name of the internal network is the same for both machines

Start both machines

*\* For easier interaction you can set up a port forwarding rules to port* ***22*** *of both virtual machines. For example, for the* ***WEB*** *machine redirect* ***22*** *to* ***10001*** *on the host and for the* ***DB*** *machine redirect* ***22*** *to* ***10002*** *on the host*

*\*\* Don’t forget to set static address on second network adapters and test the connectivity there. Use for example the* ***192.168.99.0/24*** *network*

#### Install software (on WEB VM)

Log-on to the **WEB** machine with **vagrant** user and **vagrant** for password

*Note: The* ***root*** *password is the same (****vagrant****)*

Set the static IP address on the second network interface. For example, **192.168.99.100/24**

Change the hostname to **web** by executing

**sudo hostnamectl set-hostname web**

Log-out and log-on again

Install **Apache**, **PHP**, **Git**, and other required/related packages

**sudo dnf install -y httpd php php-mysqlnd git**

Configure **Apache** to start automatically on boot and start it

**sudo systemctl enable httpd**

**sudo systemctl start httpd**

Check that we have connectivity both locally and remotely\*

We can adjust the firewall in one of the following two ways

Stop and disable it

**sudo systemctl stop firewalld**

**sudo systemctl disable firewalld**

Or open the appropriate port or service

**sudo firewall-cmd --add-service=http --permanent**

**sudo firewall-cmd --reload**

We can check again

*\* Depending on the network mode of the virtual machines, you may need to set up a port forwarding rule. For example, port* ***80*** *on the* ***VM*** *to port* ***8080*** *on the host*

#### Deploy the application (on WEB VM)

Go to the home folder of the **vagrant** user

Execute the following to clone the existing repository and get the application’s files locally

**git clone https://github.com/shekeriev/bgapp**

Copy all files from **bgapp/web** to **/var/www/html**

**sudo cp -v bgapp/web/\* /var/www/html/**

Try to open the page on the host \*

*\* Depending on the network mode of the virtual machines, you may need to set up a port forwarding rule. For example, port* ***80*** *on the* ***VM*** *to port* ***8080*** *on the host*

#### Install software (on DB VM)

Log-on to **DB** machine with **vagrant** for user and **vagrant** for password

*Note: The* ***root*** *password is the same (****vagrant****)*

Set the static IP address on the second network interface. For example, **192.168.99.101/24**

Change the hostname to **db** by executing

**sudo hostnamectl set-hostname db**

Log-out and log-on again

*If using* ***CentOS Stream 8****, we may need to add the* ***MariaDB*** *repository (adjust version or selection) with:*

***curl -LsS https://downloads.mariadb.com/MariaDB/mariadb\_repo\_setup \***

***| sudo bash -s -- --mariadb-server-version=10.6 --skip-maxscale --skip-tools***

Install **MariaDB** client and server components

**sudo dnf install -y mariadb mariadb-server git**

Enable and start the service

**sudo systemctl enable mariadb**

**sudo systemctl start mariadb**

Do some initial configuration \*

**sudo mysql\_secure\_installation**

*Note: Depending on the circumstances, the above command may have to become* ***sudo mariadb-secure-installation***

#### Configure database (on DB VM)

Go to home folder of the **vagrant** user

Execute the following to clone the existing repository and get the application’s files locally

**git clone https://github.com/shekeriev/bgapp**

Navigate to the **db** folder

Examine the **SQL** script

Execute the script against the database

**mysql -u root -p < db\_setup.sql**

Log-on and check that the data is there

**mysql -u root -p**

**use bulgaria;**

**select \* from cities;**

**quit**

Modify the firewall state in either of the following two ways

Disable it

**sudo systemctl stop firewalld**

**sudo systemctl disable firewalld**

Or open the appropriate port:

**sudo firewall-cmd --add-port=3306/tcp --permanent**

**sudo firewall-cmd --reload**

#### Configure and test the application (on WEB VM)

Check the **config.php** script if there is a need to change connection parameters

Open browser and check the final result. It should be working

In case of connectivity error execute the following on the web server to check where is the issue

**curl http://localhost**

**php /var/www/html/index.php**

Under **RedHat**-based distributions, we will see that in one of the attempts we can see the data and in the other – we cannot

This is caused by the **SELinux** suite, and we can tackle the issue by executing the following

**sudo setsebool -P httpd\_can\_network\_connect=1**

Other option is to modify **SELinux** mode (**/etc/sysconfig/selinux**) to either **permissive** or **disabled**

## Part 3: Vagrant

### Preparation

In order to automate the process, we saw so far, we will need a helper tool

**Vagrant** is a good fit for this

To install it, we must navigate to here: <https://www.vagrantup.com/downloads>

Should you need an earlier version, you can check here: <https://releases.hashicorp.com/vagrant/>

#### Install Vagrant (on Windows host)

Make sure that you have administrative access and double-click on the downloaded file to initiate the installation process

As you will see it is pretty straight-forward

#### Install Vagrant (on Red Hat-compatible host)

Let us install the **Vagrant** tool which will help us automate our infrastructure creation and management to a good extent

The steps are more or less the same no matter which **Linux** distribution is installed on the host

First, we download the package locally

**wget https://releases.hashicorp.com/vagrant/2.3.4/vagrant-2.3.4-1.x86\_64.rpm**

Then we install it

**sudo rpm -ivh vagrant-2.3.4-1.x86\_64.rpm**

Then we check that it is working

**vagrant version**

We can add bash completion should we want it. Download a 3rd party completion file

**wget https://github.com/hashicorp/vagrant/raw/master/contrib/bash/completion.sh -O vagrant-completion.sh**

Move (install) the completion script *(you may need to install the* ***bash-completion*** *package as well)*

**sudo mv vagrant-completion.sh /etc/bash\_completion.d/**

#### Create Vagrant box (on host)

We can create our own boxes or use ones already created by others

The basic steps to create a box of our own is to create a simple **VM** with minimal parameters:

* 1 CPU
* 1 GB RAM
* 16 GB HDD (Dynamic)
* No Audio
* 1 NIC in NAT mode (with a port forwarding rule **SSH/TCP/2222/22**)

Then we must install the OS (**CentOS Stream 9** in our case) again with minimal (or what we find suitable) profile:

* Turn off the **kdump** utility
* Enable the network and set a hostname
* Set the appropriate time zone
* Select the OS profile – usually **Minimal**
* Add the **vagrant** user and set its password to **vagrant**
* Use the same password for the **root** user

For the purpose of the demo, we will reuse one of the available templates

Let’s use again **CentOS**. The template is called **centos-stream-9.ova** *(it is in fact* ***CentOS Stream 9****)*

Import a **VM** from the template to save some time *(don’t forget the port forwarding rule)*

Once, we have the machine up and running, we must log on as the **root** user and do some final adjustments

Make sure that there is **SSH** installed and running

Should we want, we can disable services like the firewall

Upgrade all installed packages

**dnf upgrade**

Add new packages if you see fit. For example, install at least these:

**dnf install kernel-devel gcc make tar bzip2 wget elfutils-libelf-devel**

Mount *(****Devices > Insert Guest Additions image****)* the **VirtualBox** **Guest Additions** media and install them

**mount /dev/sr0 /mnt**

**/mnt/VBoxLinuxAdditions.run**

Add the **vagrant** user to the **vboxsf** group

**usermod -aG vboxsf vagrant**

Set the **GRUB** waiting time to **0** *(this will make our machine boot faster)*

**vi /etc/default/grub**

**grub2-mkconfig -o /boot/grub2/grub.cfg**

Add the **vagrant** user to the **sudoers** list and allow it to **sudo** without entering password

**echo "vagrant ALL=(ALL) NOPASSWD:ALL" | tee /etc/sudoers.d/vagrant**

Close the session and open a new one as the **vagrant** user

Install the **vagrant** insecure key. First, create the target folder

**mkdir -m 0700 -p /home/vagrant/.ssh**

Download the key

**wget --no-check-certificate \**

**https://raw.github.com/mitchellh/vagrant/master/keys/vagrant.pub \**

**-O /home/vagrant/.ssh/authorized\_keys**

Set the permissions of the key

**chmod 0600 /home/vagrant/.ssh/authorized\_keys**

Clean up the **DNF** cache

**sudo dnf clean all**

Make sure that the hard disk is aligned

**sudo dd if=/dev/zero of=/EMPTY bs=1M status=progress**

**sudo rm -f /EMPTY**

Reboot the **VM**

**sudo reboot**

Do not forget to eject *(****Optical Drives > Remove disk from virtual drive****)* the additions media or any other media that may be still attached to the machine

Now, we are ready to build our box *(execute next steps* ***on the host and not in the VM****)*

Create a folder to host the box (on the host machine)

If we are on a **Linux** host, we can execute the following to create and enter the work folder

**mkdir -p /home/$USER/Vagrant/centos && cd /home/$USER/Vagrant/centos**

Once, we are inside the folder, we can start the box creation process

While the **VM** is still running, build the box with *(replace <VM Name> with the name of the VM)*

**vagrant package --base <VM Name>**

Let us test it

Add the box to the local catalog *(replace <Box Name> with the name you want to use for your box)*

**vagrant box add <Box Name> package.box**

Create a configuration based on the local box

**vagrant init <Box Name>**

Power on the machine

**vagrant up**

Connect to the machine

**vagrant ssh**

Explore what's inside the machine

Close the **SSH** session

**exit**

Turn off and delete the machine

**vagrant destroy --force**

Finally, we can publish our box to **Vagrant Cloud**

Navigate to **https://app.vagrantup.com** and login

Click on **New Vagrant Box**

Enter the details and click on **Create** **box**

Enter version and description and click **Create** **version**

Click on **Add a provider**

For provider set **virtualbox** and click on **Continue to upload**

Click on the **Browse** button and navigate to the box file

Once the upload is complete, click on the **Update provider** button

Click on the box name

Click on **Release...**

Click on **Release version**

Now, we can continue either with our own box or the cited in the next steps

#### First run of the box (on host)

Create a new work folder **~/do/m1/test**

Go there and execute *(replace <repo/box-name> with your repository and box name)*

**vagrant init <repo/box-name>**

Instead of your repository and box, you can use for example, either **shekeriev/centos-stream-9** or **shekeriev/debian-11**

Examine the created **Vagrantfile**

Power-on the machine

**vagrant up**

Start **SSH** session to the machine

**vagrant ssh**

Browse the machine by executing a few commands

Exit the session

**exit**

List all local machines

**vagrant global-status**

It should be listed indicating that it is in running state

List all boxes

**vagrant box list**

It should appear in the list together with any other boxes you may have

Destroy the machine

**vagrant destroy --force**

### Application Deployment

For the next two tasks we will need the archive file available on the site of the current module

Download it and extract it somewhere

#### Automation Level 1 (on host)

Let’s assume that the files are extracted to a folder **~/do1/m1/**

Navigate to the folder **3-1**

Examine the **Vagrantfile**

Modify the settings if needed

Check the sub-folders

Execute

**vagrant up**

Open a browser and test the application

Destroy the machines

**vagrant destroy --force**

#### Alternative approach #1 (on host)

Let’s assume that the files are extracted to a folder **~/do/m1/**

Navigate to the folder **3-2**

Examine the **Vagrantfile**

Modify the settings if needed

Check the sub-folders

Execute

**vagrant up**

Open a browser and test the application

Destroy the machines

**vagrant destroy --force**

#### Alternative approach #2 (on host)

Can you think of another alternative approach?

For example, to use **git clone** within the **Vagrantfile** during the machine provision phase

Try to do it 😉