

## Лабораторная работа № 1. Подготовка лабораторного стенда

### 1.1. Цель работы

Целью данной работы является приобретение практических навыков установки Rocky Linux на виртуальную машину с помощью инструмента Vagrant.

### 1.2. Предварительные сведения

*Vagrant* — представляет собой инструмент для создания и управления средами виртуальных машин в одном рабочем процессе.

Этот инструмент, по сути, позволяет автоматизировать процесс установки на виртуальную машину как основного дистрибутива операционной системы, так и настройки необходимого в дальнейшем программного обеспечения.

С проектом Vagrant и документацией по этому инструментальному средству можно ознакомиться на сайте <https://www.vagrantup.com>.

Основные понятия Vagrant:

- провайдер (provider) — система виртуализации, с которой работает Vagrant (например, VirtualBox, VMWare и т.п.);
- box-файл (или Vagrant Box) — сохранённый образ виртуальной машины с развёрнутой в ней операционной системой; по сути, box-файл используется как основа для клонирования виртуальных машин с теми или иными настройками;
- Vagrantfile — конфигурационный файл, написанный на языке Ruby, в котором указаны настройки запуска виртуальной машины.

#### 1.2.1. HCL-файл настроек виртуальной машины

Packer использует язык конфигурации Hashicorp (HCL), разработанный для краткого описания необходимых шагов для доступа к файлу сборки. Язык HCL базируется на идеях языка JSON.

Это специальный файл с описанием метаданных по установке дистрибутива на виртуальную машину. HCL-файл является необязательным компонентом для создания box-файлов Vagrant, но полезен, так как позволяет управлять версиями и типами провайдеров (виртуального окружения) и образов операционных систем из одного файла.

#### 1.2.2. Основные команды Vagrant

С Vagrant можно работать, используя следующие основные команды:

- `vagrant help` — вызов справки по командам Vagrant;
- `vagrant box list` — список подключённых к Vagrant box-файлов;
- `vagrant box add` — подключение box-файла к Vagrant;
- `vagrant destroy` — отключение box-файла от Vagrant и удаление его из виртуального окружения;
- `vagrant init` — создание «шаблонного» конфигурационного файла Vagrantfile для его последующего изменения;
- `vagrant up` — запуск виртуальной машины с использованием инструкций по запуску из конфигурационного файла Vagrantfile;
- `vagrant reload` — перезагрузка виртуальной машины;
- `vagrant halt` — остановка и выключение виртуальной машины;
- `vagrant provision` — настройка внутреннего окружения имеющейся виртуальной машины (например, добавление новых инструкций (скриптов) в ранее созданную виртуальную машину);
- `vagrant ssh` — подключение к виртуальной машине через ssh.

### 1.2.3. Пример конфигурации Vagrantfile

Приведём пример содержимого файла Vagrantfile для понимания принципов его синтаксиса:

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

Vagrant.configure("2") do |config|
  config.vm.box = "BOX_NAME"
  config.vm.hostname = "HOST_NAME"
  config.vm.network "private_network", ip: "192.168.1.1"
  config.vm.define "VM_NAME"
  config.vm.provider "virtualbox" do |vb|
    vb.gui = false
    vb.memory = "1024"
  end
end
```

Первые две строки указывают на режим работы с Vagrantfile и использование языка Ruby. Затем идёт цикл do, заменяющий конструкцию Vagrant.configure далее по тексту на config.

Строка config.vm.box = "BOX\_NAME" задаёт название образа (box-файла) виртуальной машины (обычно выбирается из официального репозитория).

Строка config.vm.hostname = "HOST\_NAME" задаёт имя виртуальной машины.

Конструкция config.vm.network задаёт тип сетевого соединения и может иметь следующие назначения:

- config.vm.network "private\_network", ip: "xxx.xxx.xxx.xxx" — адрес из внутренней сети;
- config.vm.network "private\_network", type: "dhcp" — адрес, назначаемый по протоколу DHCP.

Строка config.vm.define "VM\_NAME" задаёт название виртуальной машины, по которому можно обращаться к ней из Vagrant и VirtualBox.

В конце идёт конструкция, определяющая параметры провайдера, а именно запуск виртуальной машины без графического интерфейса и с выделением 1 ГБ памяти.

### 1.3. Сетевая топология

Внутренняя топология лабораторных использует сервер одновременно как маршрутизатор (рис. 1.1).

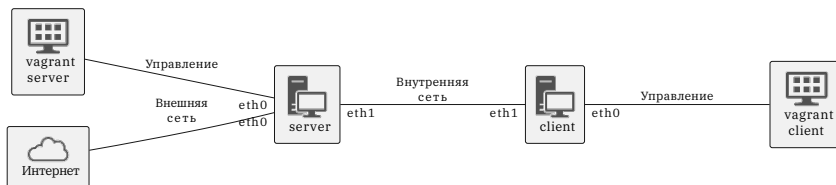


Рис. 1.1. Сетевая топология стенда

Настройка использования сервера как маршрутизатора проводится в лабораторной работе «Расширенные настройки межсетевого экрана». Без этой настройки клиент будет лишён доступа в Интернет.

Кроме того, сервер выступает также в роли сервера адресов DHCP, настройка которого производится в лабораторной работе «Настройка DHCP-сервера». Без этой настройки клиент не будет получать ip-адрес внутренней сети.

## 1.4. Задание

1. Сформируйте box-файл с дистрибутивом Rocky Linux для VirtualBox (см. раздел 1.5.2 или 1.5.3).
2. Запустите виртуальные машины сервера и клиента и убедитесь в их работоспособности.
3. Внесите изменения в настройки загрузки образов виртуальных машин `server` и `client`, добавив пользователя с правами администратора и изменив названия хостов (см. раздел 1.5.4).
4. Скопируйте необходимые для работы с Vagrant файлы и box-файлы виртуальных машин на внешний носитель. Используя эти файлы, вы можете попробовать вернуть виртуальные машины на другом компьютере.

## 1.5. Последовательность выполнения работы

Подготовленный box-файл Vagrant впоследствии можно использовать как в ОС Linux (в дисплейном классе или на собственном компьютере), так и в ОС Windows (только на собственном компьютере). Если вы планируете работать на собственном компьютере, то убедитесь, что:

- у вас достаточно свободного места на диске в разделе, где будет разворачиваться образ виртуальной машины (рекомендуется зарезервировать порядка 15GB);
  - в вашей операционной системе установлены последние версии Vagrant (<https://www.vagrantup.com>) и VirtualBox (<https://www.virtualbox.org/>);
  - для ОС Windows понадобится дополнительно установить Packer (<https://www.packer.io/>) и FAR (<https://www.farmanager.com>) для удобства работы в терминале.
- Далее приведена последовательность действий по подготовке box-файл Vagrant.
1. Перед началом работы с Vagrant создайте каталог для проекта.  
В ОС Linux рекомендуется работать в `/var/tmp`:  

```
mkdir -p /var/tmp/user_name/packer
mkdir -p /var/tmp/user_name/vagrant
```

где `user_name` — идентифицирующее вас имя пользователя, обычно первые буквы инициалов и фамилия.

В ОС Windows, например, `C:\work\user_name\packer` и `C:\work\user_name\vagrant`, где `user_name` — идентифицирующее вас имя пользователя, обычно первые буквы инициалов и фамилия.
  2. В созданном рабочем каталоге в подкаталоге `packer` разместите образ варианта операционной системы Rocky Linux (в этом практикуме будем использовать `Rocky-9.4-x86_64-minimal.iso` — минимальный дистрибутив Rocky Linux, который можно взять с сайта <https://rockylinux.org/download/>). При работе в дисплейном классе университета дистрибутив можно взять из общего каталога `/afs/dk.sci.pfu.edu.ru/common/files/iso/`.
  3. В этом же рабочем каталоге разместите подготовленные заранее для работы с Vagrant файлы:
    - в подкаталоге `packer` файл `vagrant-rocky.pkr.hcl` — специальный файл с описанием метаданных по установке дистрибутива на виртуальную машину (содержание используемого в данном практикуме файла `.hcl` приведено в разделе 1.5.1.1); в частности, в разделе переменных этот файл содержит указание на версию дистрибутива, его хэш-функцию, имя и пароль пользователя по умолчанию; в разделе `builders` указаны специальные синтаксические конструкции для автоматизации работы VirtualBox; в разделе `provisioners` прописаны действия (по сути shell-скрипт) по установке дополнительных пакетов дистрибутива;
    - в подкаталоге `packer` подкаталог `http` с файлом `ks.cfg` — определяет настройки для установки дистрибутива, которые пользователь обычно вводит вручную, в частности настройки языка интерфейса, языковые настройки клавиатуры, тайм-зону, сетевые настройки и т.п.; файл `ks.cfg` должен быть расположен в подкаталоге `http` (содержание используемого в данном практикуме файла `./http/ks.cfg` приведено в разделе 1.5.1.2);

- в подкаталоге vagrant файл Vagrantfile — файл с конфигурацией запуска виртуальных машин — сервера и клиента (содержание используемого в данном практикуме на данном этапе файла Vagrantfile приведено в разделе 1.5.1.3);
- в подкаталоге vagrant файл Makefile — набор инструкций для программы make по работе с Vagrant (содержание используемого в данном практикуме файла Makefile приведено в разделе 1.5.1.5).

Основное назначение Makefile в этом практикуме — применение команд Vagrant в ОС Linux в определённом каталоге — только в каталоге с проектом (в частности в /var/tmp/user\_name/vagrant). Для пользователей, работающих с Vagrant в ОС Windows, Makefile не понадобится.

4. В этом же рабочем каталоге в подкаталоге vagrant создайте каталог provision с подкаталогами default, server и client, в которых будут размещаться скрипты, изменяющие настройки внутреннего окружения базового (общего) образа виртуальной машины, сервера или клиента соответственно.
5. В каталогах default, server и client разместите заранее подготовленный скрипт-заглушку 01-dummy.sh следующего содержания:

```
#!/bin/bash
echo "Provisioning script $0"
```

6. В каталоге default разместите заранее подготовленный скрипт 01-user.sh по изменению названия виртуальной машины следующего содержания:

```
#!/bin/bash

echo "Provisioning script $0"

username=user
userpassword=123456

encpassword=`openssl passwd -1 ${userpassword}`
id -u $username
if [[ $? ]]
then
    adduser -G wheel -p ${encpassword} ${username}
    homedir=`getent passwd ${username} | cut -d: -f6`
    echo "export PS1='[\u\H \W]\$ '" >> ${homedir}/.bashrc
fi
```

В этом скрипте в качестве значения переменной username вместо user укажите имя пользователя, совпадающее с вашим логином, т.е. для Ивана Петровича Сидорова логин должен иметь вид ipsidorov.

7. В каталоге default разместите заранее подготовленный скрипт 01-hostname.sh по изменению названия виртуальной машины следующего содержания:

```
#!/bin/bash

username=user
hostnamectl set-hostname "${HOSTNAME%.*}.${username}.net"
```

В этом скрипте в качестве значения переменной username вместо user укажите имя пользователя, совпадающее с вашим логином, т.е. для Ивана Петровича Сидорова логин должен иметь вид ipsidorov.

8. В каталоге server разместите заранее подготовленный скрипт 02-forward.sh следующего содержания:

```
#!/bin/bash

echo "Provisioning script $0"

echo "Enable forwarding"
echo "net.ipv4.ip_forward = 1" > /etc/sysctl.d/90-forward.conf
sysctl -w net.ipv4.ip_forward=1
```

```
echo "Configure masquerading"
firewall-cmd --add-masquerade --permanent
firewall-cmd --reload
```

```
restorecon -vR /etc
```

Этот скрипт обеспечивает корректную маршрутизацию ip-адресов между сервером и клиентом.

9. В каталоге client разместите заранее подготовленный скрипт 01-routing.sh следующего содержания:

```
#!/bin/bash
```

```
echo "Provisioning script $0"
```

```
nmcli connection modify "System eth1" ipv4.gateway "192.168.1.1"
nmcli connection up "System eth1"
```

```
nmcli connection modify eth0 ipv4.never-default true
nmcli connection modify eth0 ipv6.never-default true
```

```
nmcli connection down eth0
nmcli connection up eth0
```

Этот скрипт обеспечивает корректную работу сетевых интерфейсов клиента.

## 1.5.1. Конфигурационные файлы

### 1.5.1.1. Содержание файла vagrant-rocky.pkr.hcl

```
packer {
  required_plugins {
    vagrant = {
      source = "github.com/hashicorp/vagrant"
      version = "~> 1"
    }
    virtualbox = {
      version = "~> 1"
      source = "github.com/hashicorp/virtualbox"
    }
  }
}

variable "artifact_description" {
  type = string
  default = "Rocky 9.4"
}

variable "artifact_version" {
  type = string
  default = "9.4"
}

variable "disk_size" {
  type = string
  default = "61440"
}
```

```

variable "iso_checksum" {
    type     = string
    default = "ee3ac97fdffab58652421941599902012179c37535aece76824673105169c4a2"
}

variable "iso_checksum_type" {
    type     = string
    default = "sha256"
}

variable "iso_url" {
    type     = string
    default = "Rocky-9.4-x86_64-minimal.iso"
}

variable "redhat_platform" {
    type     = string
    default = "x86_64"
}

variable "redhat_release" {
    type     = string
    default = "9"
}

variable "ssh_password" {
    type     = string
    default = "vagrant"
}

variable "ssh_username" {
    type     = string
    default = "vagrant"
}

source "virtualbox-iso" "virtualbox" {
    boot_command = [
        "<esc>",
        "<wait><esc><esc>",
        "linux inst.ks=http://{var.http_ip}:{var.http_port}/ks.cfg biosdevname=0",
        "net.ifnames=0",
        "<enter>"
    ]
    boot_wait      = "30s"
    disk_size      = "${var.disk_size}"
    export_opts    = [
        "--manifest",
        "--vsys", "0",
        "--description", "${var.artifact_description}",
        "--version", "${var.artifact_version}"
    ]
    guest_additions_path = "VBoxGuestAdditions.iso"
    guest_os_type        = "RedHat_64"
    hard_drive_interface = "sata"
    http_directory       = "${path.root}/http"
    iso_checksum         = "${var.iso_checksum_type}:${var.iso_checksum}"
    iso_url              = "${var.iso_url}"
    output_directory     = "builds"
}

```

```

shutdown_command      = "sudo -S /sbin/halt -h -p"
shutdown_timeout      = "5m"
ssh_password          = "${var.ssh_password}"
ssh_username          = "${var.ssh_username}"
ssh_port              = 22
ssh_pty               = true
ssh_timeout           = "60m"
vboxmanage            = [
  ["modifyvm", "${.Name}", "--memory", "2048"],
  ["modifyvm", "${.Name}", "--cpus", "2"],
  ["modifyvm", "${.Name}", "--nat-localhostreachable1", "on"]
]
virtualbox_version_file = ".vbox_version"
vm_name               = "packer-rocky-virtualbox-vm"
}

build {
  sources = ["source.virtualbox-iso.virtualbox"]

  provisioner "shell" {
    execute_command = "echo 'packer'|${.Vars} sudo -S -E bash '${.Path}
    ↪  ${.Id}'"
    inline          = [
      "sleep 30",
      "sudo dnf -y install epel-release",
      "sudo dnf -y groupinstall 'Development Tools'",
      "sudo dnf -y install kernel-devel kernel-modules",
      "sudo dnf -y install dkms",
      "sudo mkdir /tmp/vboxguest",
      "sudo mount -t iso9660 -o loop /home/vagrant/VBoxGuestAdditions.iso
      ↪ /tmp/vboxguest",
      "cd /tmp/vboxguest",
      "sudo ./VBoxLinuxAdditions.run",
      "cd /tmp",
      "sudo umount /tmp/vboxguest",
      "sudo rmdir /tmp/vboxguest",
      "rm /home/vagrant/VBoxGuestAdditions.iso",
      "sudo systemctl enable --now vboxadd.service",
      "sudo dnf -y install lightdm",
      "sudo dnf -y groupinstall 'Server with GUI'",
      "sudo dnf install -y mc htop tmux",
      "sudo systemctl set-default graphical.target",
      "echo Image Provisioned!"
    ]
  }
}

post-processor "vagrant" {
  compression_level = "6"
  output            = "vagrant-virtualbox-rocky-${var.redhat_release}-${var}
  ↪ .redhat_platform}.box"
}
}

```

### 1.5.1.2. Содержание файла ks.cfg

```
# System bootloader configuration
bootloader --append="no_timer_check console=tty0 console=ttyS0,115200n8
↳ net.ifnames=0 biosdevname=0 elevator=noop" --location=mbr --timeout=1
# Clear the Master Boot Record
zerombr
# Partition clearing information
clearpart --all
# Reboot after installation
reboot
# Use text mode install
text
# Keyboard layouts
keyboard --vckeymap=us,ru --xlayouts='us,ru'
# System language
lang en_US.UTF-8

# Network information
network --bootproto=dhcp --device=link --activate

# System authorization information
authselect select sssd with-sudo with-mkhomedir --force
authselect apply-changes
# Root password
rootpw vagrant
user --name=vagrant --password=vagrant
firstboot --disable
# Do not configure the X Window System
#skipx
# System services
services --enabled="NetworkManager,sshd,chronyd"
# System timezone
timezone UTC --utc
user --name=vagrant --password=vagrant
# Disk partitioning information
part / --fstype="xfs" --size=10239

%post
# configure swap to a file
fallocate -l 2G /swapfile
chmod 600 /swapfile
mkswap /swapfile
echo "/swapfile none swap defaults 0 0" >> /etc/fstab

# sudo
echo "%vagrant ALL=(ALL) NOPASSWD: ALL" > /etc/sudoers.d/vagrant
chmod 0440 /etc/sudoers.d/vagrant

# Fix for https://github.com/CentOS/sig-cloud-instance-build/issues/38
cat > /etc/sysconfig/network-scripts/ifcfg-eth0 << EOF
DEVICE="eth0"
BOOTPROTO="dhcp"
ONBOOT="yes"
TYPE="Ethernet"
PERSISTENT_DHCLIENT="yes"
EOF
```



```
# sshd: disable password authentication and DNS checks
#ex -s /etc/ssh/sshd_config <<EOF
#:%substitute/^\(PasswordAuthentication\) yes$/\1 no/
#:%substitute/^\(UseDNS\) yes$/&\r\1 no/
#:update
#:quit
#EOF
#cat >>/etc/sysconfig/sshd <<EOF

# Decrease connection time by preventing reverse DNS lookups
# (see https://lists.centos.org/pipermail/centos-devel/2016-July/014981.html
# and man sshd for more information)
OPTIONS="-u0"
EOF

# Fix for issue #76, regular users can gain admin privileges via su
ex -s /etc/pam.d/su <<'EOF'
# allow vagrant to use su, but prevent others from becoming root or vagrant
/^account\s\+sufficient\s\+pam_succeed_if.so uid = 0 use_uid quiet$/
:append
account          [success=1 default=ignore] \\\
                  pam_succeed_if.so user = vagrant use_uid quiet
account          required          pam_succeed_if.so user notin root:vagrant

:update
:quit
EOF

# systemd should generate a new machine id during the first boot, to
# avoid having multiple Vagrant instances with the same id in the local
# network. /etc/machine-id should be empty, but it must exist to prevent
# boot errors (e.g. systemd-journald failing to start).
:>/etc/machine-id

#echo 'vag' > /etc/yum/vars/infra

# Blacklist the floppy module to avoid probing timeouts
echo blacklist floppy > /etc/modprobe.d/nofloppy.conf
chcon -u system_u -r object_r -t modules_conf_t /etc/modprobe.d/nofloppy.conf

# Customize the initramfs
pushd /etc/dracut.conf.d
# There's no floppy controller, but probing for it generates timeouts
echo 'omit_drivers+=" floppy "' > nofloppy.conf
popd
# Fix the SELinux context of the new files
restorecon -f - <<EOF
/etc/sudoers.d/vagrant
#/etc/dracut.conf.d/vmware-fusion-drivers.conf
#/etc/dracut.conf.d/hyperv-drivers.conf
/etc/dracut.conf.d/nofloppy.conf
EOF

# Rerun dracut for the installed kernel (not the running kernel):
KERNEL_VERSION=$(rpm -q kernel --qf '%{version}-%{release}.%{arch}\n')
dracut -f /boot/initramfs-${KERNEL_VERSION}.img ${KERNEL_VERSION}

# Seal for deployment
```

```

rm -rf /etc/ssh/ssh_host_*
hostnamectl set-hostname localhost.localdomain
rm -rf /etc/udev/rules.d/70-*
%end

%packages --inst-langs=en
bash-completion
bzip2
chrony
man-pages
rsync
-dracut-config-rescue
-iwl100-firmware
-iwl1000-firmware
-iwl105-firmware
-iwl135-firmware
-iwl2000-firmware
-iwl2030-firmware
-iwl3160-firmware
-iwl3945-firmware
-iwl4965-firmware
-iwl5000-firmware
-iwl5150-firmware
-iwl6000-firmware
-iwl6000g2a-firmware
-iwl6050-firmware
-iwl7260-firmware
-microcode_ctl
-plymouth

%end

%addon com_redhat_kdump --disable --reserve-mb='128'

%end

```

### 1.5.1.3. Содержание файла Vagrantfile

```

# -*- mode: ruby -*-
# vi: set ft=ruby :

Vagrant.configure("2") do |config|

  ## Common configuration
  config.vm.provision "common dummy",
    type: "shell",
    preserve_order: true,
    path: "provision/default/01-dummy.sh"

  config.vm.provision "common hostname",
    type: "shell",
    preserve_order: true,
    run: "always",
    path: "provision/default/01-hostname.sh"

  config.vm.provision "common user",
    type: "shell",
    preserve_order: true,

```

```
        path: "provision/default/01-user.sh"

## Server configuration
config.vm.define "server", autostart: false do |server|
  server.vm.box = "rocky9"
  server.vm.hostname = 'server'

  server.vm.boot_timeout = 1440

  server.ssh.insert_key = false
  server.ssh.username = 'vagrant'
  server.ssh.password = 'vagrant'

  server.vm.network :private_network,
    ip: "192.168.1.1",
    virtualbox____intnet: true

  server.vm.provision "server dummy",
    type: "shell",
    preserve_order: true,
    path: "provision/server/01-dummy.sh"

server.vm.provider :virtualbox do |v|
  v.linked_clone = true
  # Customize the amount of memory on the VM
  v.memory = 1024
  v.cpus = 1
  v.name = "server"
  # Display the VirtualBox GUI when booting the machine
  v.gui = true
  # Set the video memory to 12Mb
  v.customize ["modifyvm", :id, "--vram", "12"]
  v.customize ["modifyvm", :id, "--natdnshostresolver1", "on"]
  v.customize ["modifyvm", :id, "--clipboard", "bidirectional"]
  v.customize ["modifyvm", :id, "--draganddrop", "bidirectional"]
  v.customize ["modifyvm", :id, "--graphicscontroller", "vmsvga"]
  v.customize ["modifyvm", :id, "--accelerate3d", "on"]
end
end

## Client configuration
config.vm.define "client", autostart: false do |client|
  client.vm.box = "rocky9"
  client.vm.hostname = 'client'

  client.vm.boot_timeout = 1440

  client.ssh.insert_key = false
  client.ssh.username = 'vagrant'
  client.ssh.password = 'vagrant'

  client.vm.network :private_network,
    type: "dhcp",
    virtualbox____intnet: true

  client.vm.provision "client dummy",
    type: "shell",
    preserve_order: true,
```

```

        path: "provision/client/01-dummy.sh"

client.vm.provision "client routing",
  type: "shell",
  preserve_order: true,
  run: "always",
  path: "provision/client/01-routing.sh"

client.vm.provider :virtualbox do |v|
  v.linked_clone = true
  # Customize the amount of memory on the VM
  v.memory = 1024
  v.cpus = 1
  v.name = "client"
  # Display the VirtualBox GUI when booting the machine
  v.gui = true
  # Set the video memory to 12Mb
  v.customize ["modifyvm", :id, "--vram", "12"]
  v.customize ["modifyvm", :id, "--natdnshostresolver1", "on"]
  v.customize ["modifyvm", :id, "--clipboard", "bidirectional"]
  v.customize ["modifyvm", :id, "--draganddrop", "bidirectional"]
  v.customize ["modifyvm", :id, "--graphicscontroller", "vmsvga"]
  v.customize ["modifyvm", :id, "--accelerate3d", "on"]
end
end
end

```

#### 1.5.1.4. Содержание файла Makefile для packer

```

.PHONY: version

all: help

init: ## Install missing plugins for packer
    @mkdir -p "`pwd`"/.config/packer/plugins"
    @export PACKER_CONFIG_DIR="`pwd`"/.config/packer"; export
    ↪ PACKER_PLUGIN_PATH="`pwd`"/.config/packer/plugins"; packer init
    ↪ vagrant-rocky.pkr.hcl

box:      init      ## Build box for Rocky Linux
    -@VBoxManage setproperty language C
    @VBoxManage setproperty machinefolder "`pwd`"/vm
    @export TMPDIR="`pwd`"; export
    ↪ PACKER_CONFIG_DIR="`pwd`"/.config/packer"; export
    ↪ PACKER_PLUGIN_PATH="`pwd`"/.config/packer/plugins"; packer build
    ↪ -only=virtualbox-iso.virtualbox vagrant-rocky.pkr.hcl
    @VBoxManage setproperty machinefolder default

help:
    @echo 'Usage:'
    @echo '  make <target>'
    @echo
    @echo 'Targets:'
    @grep -E '^[a-zA-Z_0-9-]+:.*?## .*$$' $(MAKEFILE_LIST) | sort | awk
    ↪ 'BEGIN {FS = ".*?## "}; {printf "  \033[36m%-30s\033[0m %s\n",
    ↪ $1, $2}'
    @echo

```

### 1.5.1.5. Содержание файла Makefile для vagrant

```
.PHONY: version

all: help

help:
    @echo 'Usage:'
    @echo '  make <target>'
    @echo
    @echo 'Targets:'
    @grep -E '^[a-zA-Z_0-9.-]+:.*?## .*$$' $(MAKEFILE_LIST) | sort | awk
    ↪ 'BEGIN {FS = ":.*?## "}; {printf " \033[36m%-30s\033[0m %s\n",
    ↪  $$1, $$2}'
    @echo

plugins:      ## Install plugins
    @export VAGRANT_HOME="\`pwd\`"/.vagrant.d; export
    ↪ VAGRANT_DOTFILE_PATH="\`pwd\`"/.vagrant; vagrant plugin install
    ↪ --plugin-clean-sources --plugin-source https://rubygems.org
    ↪ vagrant-vbguest
    touch plugins

addbox:       ## Add the built box to Vagrant
    @export VAGRANT_HOME="\`pwd\`"/.vagrant.d; export
    ↪ VAGRANT_DOTFILE_PATH="\`pwd\`"/.vagrant; vagrant box add rocky9
    ↪ vagrant-virtualbox-rocky-9-x86_64.box

server-up: plugins      ## Start server
    @VBoxManage setproperty machinefolder "\`pwd\`"/vm
    @export VAGRANT_HOME="\`pwd\`"/.vagrant.d; export
    ↪ VAGRANT_DOTFILE_PATH="\`pwd\`"/.vagrant; export
    ↪ VBOX_USER_HOME="\`pwd\`"/.vbox; export
    ↪ VBOX_INSTALL_PATH="\`pwd\`"/vm; vagrant up server
    @VBoxManage setproperty machinefolder default

client-up: plugins      ## Start client
    @VBoxManage setproperty machinefolder "\`pwd\`"/vm
    @export VAGRANT_HOME="\`pwd\`"/.vagrant.d; export
    ↪ VAGRANT_DOTFILE_PATH="\`pwd\`"/.vagrant; export
    ↪ VBOX_USER_HOME="\`pwd\`"/.vbox; export
    ↪ VBOX_INSTALL_PATH="\`pwd\`"/vm; vagrant up client
    @VBoxManage setproperty machinefolder default

server-halt: plugins    ## Stop server
    @VBoxManage setproperty machinefolder "\`pwd\`"/vm
    @export VAGRANT_HOME="\`pwd\`"/.vagrant.d; export
    ↪ VAGRANT_DOTFILE_PATH="\`pwd\`"/.vagrant; export
    ↪ VBOX_USER_HOME="\`pwd\`"/.vbox; export
    ↪ VBOX_INSTALL_PATH="\`pwd\`"/vm; vagrant halt server
    @VBoxManage setproperty machinefolder default

client-halt: plugins    ## Stop client
    @VBoxManage setproperty machinefolder "\`pwd\`"/vm
```

```

@export VAGRANT_HOME="\`pwd\`"/.vagrant.d; export
↪ VAGRANT_DOTFILE_PATH="\`pwd\`"/.vagrant; export
↪ VBox_USER_HOME="\`pwd\`"/.vbox; export
↪ VBox_INSTALL_PATH="\`pwd\`"/vm; vagrant halt client
@VBoxManage setproperty machinefolder default

server-provision: plugins      ## Start and provision server
@VBoxManage setproperty machinefolder "\`pwd\`"/vm
@export VAGRANT_HOME="\`pwd\`"/.vagrant.d; export
↪ VAGRANT_DOTFILE_PATH="\`pwd\`"/.vagrant; export
↪ VBox_USER_HOME="\`pwd\`"/.vbox; export
↪ VBox_INSTALL_PATH="\`pwd\`"/vm; vagrant up server --provision
@VBoxManage setproperty machinefolder default

client-provision: plugins      ## Start and provision client
@VBoxManage setproperty machinefolder "\`pwd\`"/vm
@export VAGRANT_HOME="\`pwd\`"/.vagrant.d; export
↪ VAGRANT_DOTFILE_PATH="\`pwd\`"/.vagrant; export
↪ VBox_USER_HOME="\`pwd\`"/.vbox; export
↪ VBox_INSTALL_PATH="\`pwd\`"/vm; vagrant up client --provision
@VBoxManage setproperty machinefolder default

server-destroy:               ## Destroy server
@VBoxManage setproperty machinefolder "\`pwd\`"/vm
@export VAGRANT_HOME="\`pwd\`"/.vagrant.d; export
↪ VAGRANT_DOTFILE_PATH="\`pwd\`"/.vagrant; export
↪ VBox_USER_HOME="\`pwd\`"/.vbox; export
↪ VBox_INSTALL_PATH="\`pwd\`"/vm; vagrant destroy server
@VBoxManage setproperty machinefolder default

client-destroy:               ## Destroy client
@VBoxManage setproperty machinefolder "\`pwd\`"/vm
@export VAGRANT_HOME="\`pwd\`"/.vagrant.d; export
↪ VAGRANT_DOTFILE_PATH="\`pwd\`"/.vagrant; export
↪ VBox_USER_HOME="\`pwd\`"/.vbox; export
↪ VBox_INSTALL_PATH="\`pwd\`"/vm; vagrant destroy client
@VBoxManage setproperty machinefolder default

```

## 1.5.2. Развёртывание лабораторного стенда на ОС Linux

1. Перейдите в каталог с проектом:  
`cd /var/tmp/user_name/packer`  
 где user\_name — идентифицирующее вас имя пользователя, обычно первые буквы инициалов и фамилия.
2. В терминале наберите  
`make help`  
 Вы увидите перечень указанных в Makefile целей и краткое описание их действий.
3. Для формирования box-файла с дистрибутивом Rocky Linux для VirtualBox в терминале наберите  
`make init`  
`make box`

Начнётся процесс скачивания, распаковки и установки драйверов VirtualBox и дистрибутива ОС на виртуальную машину. Во время автоматического развёртывания дистрибутива можно просматривать выводимую на экран информацию с разных окон, перемещаясь по ним с помощью клавиш `[Alt] + [Tab]`.

После завершения процесса автоматического развёртывания образа виртуальной машины в каталоге `/var/tmp/user_name/packer` временно появится каталог `builds` с промежуточными файлами `.vdi`, `.vmdk` и `.ovf`, которые затем автоматически будут преобразованы в `box`-файл сформированного образа: `vagrant-virtualbox-rocky-9-x86_64.box`.

4. Сохраните файлы `vagrant-virtualbox-rocky-9-x86_64.box`, `vagrant-rocky.pkr.hcl`, `./http/ks.cfg`, `Vagrantfile` и `Makefile` на внешний носитель или другой каталог.
5. Скопируйте или переместите файл `vagrant-virtualbox-rocky-9-x86_64.box` в рабочий каталог в подкаталог `vagrant`.
6. Для регистрации образа виртуальной машины в Vagrant в терминале в каталоге `/var/tmp/user_name/vagrant` наберите  
`make addbox`
7. Это позволит на основе конфигурации, прописанной в файле `Vagrantfile`, сформировать `box`-файлы образов двух виртуальных машин — сервера и клиента с возможностью их параллельной или индивидуальной работы.
8. Запустите виртуальную машину `Server`, введя  
`make server-up`
9. Запустите виртуальную машину `Client`, введя  
`make client-up`
10. Подключитесь к серверу из консоли:  
`vagrant ssh server`
11. Введите пароль `vagrant`.
12. Перейдите к пользователю `user` (вместо `user` должен быть указан ваш логин):  
`su - user`

В случае появления сообщения

```
su: user user does not exist or the user entry does not contain all the
    ^
    required fields
```

необходимо выполнить действия в консоли основной ОС:

- отлогиниться из виртуальной машины: `logout` или `exit`, или комбинация клавиш `Ctrl`+`d`;
  - выключить виртуальную машину:  
`make server-halt`
  - запустить еще раз виртуальную машину, убедившись в отработке скриптов `hostname` и `user`;
  - подключиться к серверу из консоли и перейти под пользователя `user` (вместо `user` должен быть указан ваш логин).
13. Отлогиньтесь.
  14. Выполните тоже самое для клиента.  
`make server-halt`  
`make client-halt`
  15. Убедитесь, что запуск обеих виртуальных машин прошёл успешно, залогиньтесь под пользователем `vagrant` с паролем `vagrant` в графическом окружении.

### 1.5.3. Развёртывание лабораторного стенда на ОС Windows

В данном разделе приведена последовательность действий при развёртывании образа виртуальной машины в ОС Windows на домашнем компьютере в VirtualBox с использованием Vagrant. После установки необходимого программного обеспечения не забудьте перезагрузить систему.

Далее выполните следующие действия:

1. Используя FAR, перейдите в созданный вами рабочий каталог с проектом. В этом же каталоге должен быть размещён файл `packer.exe`. В командной строке введите  
`packer.exe init vagrant-rocky.pkr.hcl`  
`packer.exe build vagrant-rocky.pkr.hcl`

для начала автоматической установки образа операционной системы Rocky Linux в VirtualBox и последующего формирования box-файла с дистрибутивом Rocky Linux для VirtualBox. По окончании процесса в рабочем каталоге сформируется box-файл с названием `vagrant-virtualbox-rocky-9-x86_64.box`.

2. Для регистрации образа виртуальной машины в `vagrant` в командной строке введите  
`vagrant box add rocky9 vagrant-virtualbox-rocky-9-x86_64.box`
3. Для запуска виртуальной машины `Server` введите в консоли  
`vagrant up server`
4. Для запуска виртуальной машины `Client` введите в консоли  
`vagrant up client`
5. Убедитесь, что запуск обеих виртуальных машин прошёл успешно, залогиньтесь под пользователем `vagrant` с паролем `vagrant` в графическом окружении.
6. Подключитесь к серверу из консоли:  
`vagrant ssh server`
7. Введите пароль `vagrant`.
8. Перейдите к пользователю `user` (вместо `user` должен быть указан ваш логин):  
`su - user`
9. Отлогиньтесь.
10. Выполните тоже самое для клиента.
11. Выключите обе виртуальные машины:  
`vagrant halt server`  
`vagrant halt client`

#### 1.5.4. Внесение изменений в настройки внутреннего окружения виртуальной машины

1. Для отработки созданных скриптов во время загрузки виртуальных машин убедитесь, что в конфигурационном файле `Vagrantfile` до строк с конфигурацией сервера имеется следующая запись:

```
# Common configuration
config.vm.provision "common user",
  type: "shell",
  preserve_order: true,
  path: "provision/default/01-user.sh"

config.vm.provision "common hostname",
  type: "shell",
  preserve_order: true,
  run: "always",
  path: "provision/default/01-hostname.sh"
```

2. Зафиксируйте внесённые изменения для внутренних настроек виртуальных машин, введя в терминале:

```
make server-provision
Затем
make client-provision
```

Для работающих под ОС Windows вместо инструкций `Makefile` следует последовательно ввести в командной строке:

```
vagrant up server --provision
vagrant up client --provision
```

3. Залогиньтесь на сервере и клиенте под созданным пользователем. Убедитесь, что в терминале приглашение отображается в виде `user@server.user.net` на сервере и в виде `user@client.user.net` на клиенте, где вместо `user` указан ваш логин.
4. Выключите виртуальные машины.



5. После выключения виртуальных машин скопируйте необходимые для работы с Vagrant файлы и box-файлы виртуальных машин на внешний носитель или в другой каталог вашей ОС. Используя эти файлы, вы можете развернуть виртуальные машины на другом компьютере.

## 1.6. Содержание отчёта

1. Титульный лист с указанием номера лабораторной работы и ФИО студента.
2. Формулировка задания работы.
3. Описание результатов выполнения задания:
  - скриншоты (снимки экрана), фиксирующие выполнение работы;
  - подробное описание настроек служб в соответствии с заданием;
  - полные тексты конфигурационных файлов настраиваемых в работе служб;
  - результаты проверки корректности настроек служб в соответствии с заданием (подтверждённые скриншотами).
4. Выводы, согласованные с заданием работы.
5. Ответы на контрольные вопросы.

## 1.7. Контрольные вопросы

1. Для чего предназначен Vagrant?
2. Что такое box-файл? В чём назначение Vagrantfile?
3. Приведите описание и примеры вызова основных команд Vagrant.
4. Дайте построчные пояснения содержания файлов `vagrant-rocky.pkr.hcl`, `ks.cfg`, `Vagrantfile`, `Makefile`.

При ответах на вопросы рекомендуется ознакомиться с источниками [1–5].

## Список литературы

1. GNU Bash Manual. — 2019. — URL: <https://www.gnu.org/software/bash/manual/> (дата обр. 13.09.2021).
2. GNU Make Manual. — 2016. — URL: <http://www.gnu.org/software/make/manual/> (visited on 09/13/2021).
3. Powers S. Vagrant Simplified [Построено Vagrant] / Пер.: А. Панин // Библиотека сайта rus-linux.net. — 2015. — URL: <http://rus-linux.net/MyLDP/vm/vagrant-simplified.html> (visited on 09/13/2021).
4. Vagrant Documentation. — URL: <https://www.vagrantup.com/docs> (visited on 09/13/2021).
5. Купер М. Искусство программирования на языке сценариев командной оболочки. — 2004. — URL: [https://www.opennet.ru/docs/RUS/bash\\_scripting\\_guide/](https://www.opennet.ru/docs/RUS/bash_scripting_guide/) (дата обр. 13.09.2021).