

# **Отчет по лабораторной работе 1**

Генералов Даниил, НПИбд-01-21, 1032202280

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
<b>4</b>	<b>Выводы</b>	<b>14</b>
<b>5</b>	<b>Контрольные вопросы</b>	<b>15</b>

## Список иллюстраций

3.1	git . . . . .	7
3.2	git . . . . .	8
3.3	packer . . . . .	8
3.4	packer . . . . .	9
3.5	virtualbox . . . . .	10
3.6	virtualbox . . . . .	11
3.7	packer . . . . .	12
3.8	packer . . . . .	12
3.9	vagrant . . . . .	13
3.10	vagrant . . . . .	13

## **Список таблиц**

# 1 Цель работы

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

## 2 Задание

1. Сформируйте box-файл с дистрибутивом Rocky Linux для VirtualBox (см. раздел 1.4.2 или 1.4.3).
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 1.4.4).
4. Скопируйте необходимые для работы с Vagrant файлы и box-файлы виртуальных машин на внешний носитель. Используя эти файлы, вы можете попробовать развернуть виртуальные машины на другом компьютере.

### 3 Выполнение лабораторной работы

Сначала я создал нужные директории для работы. Поскольку Vagrant-файлы – текстовые, я решил положить их в тот же самый git-репозиторий, где также находятся мои отчеты, а именно <https://github.com/danya02/rudn-year3-subsystem-admin>.

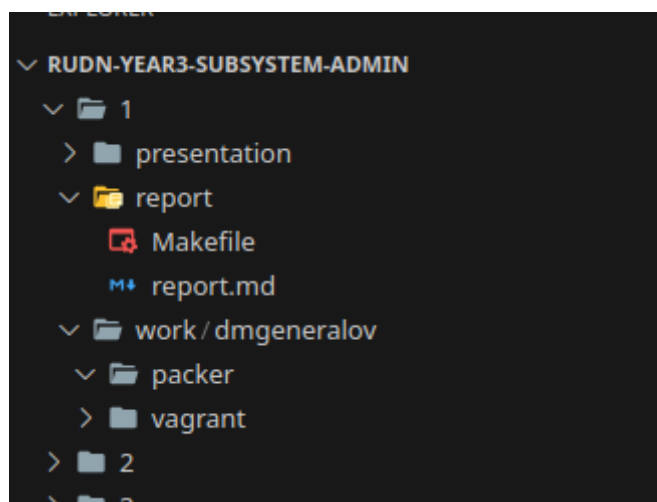


Рис. 3.1: git

Затем, проверив работоспособность VirtualBox, я положил в эту папку файлы из архива.

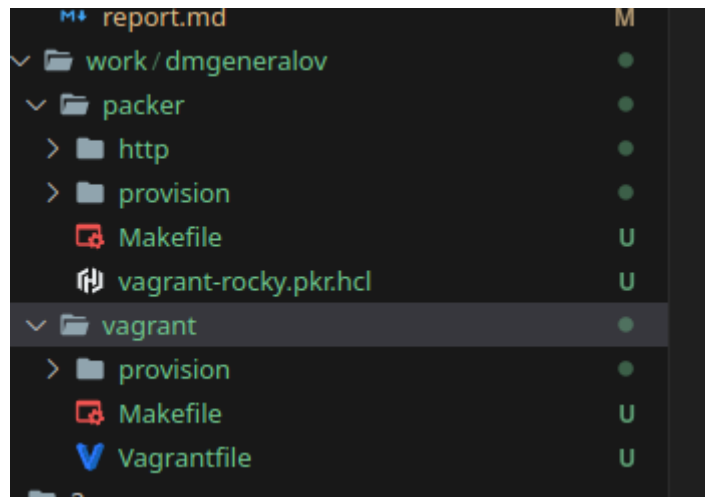


Рис. 3.2: git

Теперь нужно было решить проблему с тем, что plugin vagrant не был установлен для Packer.

```
[danya@archlinux packer]$ make init
Installed plugin github.com/hashicorp/vagrant v1.1.1 in "/home/danya/Documents/university/rudn-year3-subsystem-admin/1/work/dmgeneralov/packer/.config/packer/plugins/github.com/hashicorp/vagrant/packer-plugin-vagrant_v1.1.1_x5.0_linux_amd64"
[danya@archlinux packer]$ make box
Warning: Bundled plugins used

This template relies on the use of plugins bundled into the Packer binary.
The practice of bundling external plugins into Packer will be removed in an
upcoming version.

To remove this warning, add the following section to your template:

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

Then run 'packer init' to manage installation of the plugins

Error: failed loading shell
on vagrant-rocky.pkr.hcl line 101:
(source code not available)

Unrecognized remote plugin message: Error starting plugin server: listen unix
/home/danya/Documents/university/rudn-year3-subsystem-admin/1/work/dmgeneralov/packer/packer-plugin2922764540:
bind: invalid argument

make: *** [Makefile:12: box] Error 1
[danya@archlinux packer]$
```

Рис. 3.3: packer

Как оказалось, эта проблема на самом деле была вызвана тем, что путь к unix-сокету находился слишком глубоко. Я перенес папку packer в /tmp, и проблема исчезла.



```

[pid 16182] +++ exited with 1 +++
[pid 16181] +++ exited with 1 +++
+++ exited with 1 +++
[danya@archlinux packer]$ cd ..
[danya@archlinux dmgeneralov]$ cp -R packer/ /tmp
[danya@archlinux dmgeneralov]$ cd /tmp/packer/
[danya@archlinux packer]$ ls
http Makefile packer-plugin-3759899976.zip provision Rocky-9.2-x86_64-minimal.iso vagrant-rocky.pkr.hcl
[danya@archlinux packer]$ make box
Warning: Bundled plugins used

This template relies on the use of plugins bundled into the Packer binary.
The practice of bundling external plugins into Packer will be removed in an
upcoming version.

To remove this warning, add the following section to your template:

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

Then run 'packer init' to manage installation of the plugins

virtualbox-iso.virtualbox: output will be in this color.

==> virtualbox-iso.virtualbox: Cannot find "Default Guest Additions ISO" in vboxmanage output (or it is empty)
==> virtualbox-iso.virtualbox: Retrieving Guest additions checksums
==> virtualbox-iso.virtualbox: Trying https://download.virtualbox.org/virtualbox/7.0.12/SHA256SUMS
==> virtualbox-iso.virtualbox: Trying https://download.virtualbox.org/virtualbox/7.0.12/SHA256SUMS

```

Рис. 3.4: packer

Теперь запускалась виртуальная машина, и Packer ожидал запуска SSH на ней. После некоторого времени начали устанавливаться пакеты DNF.

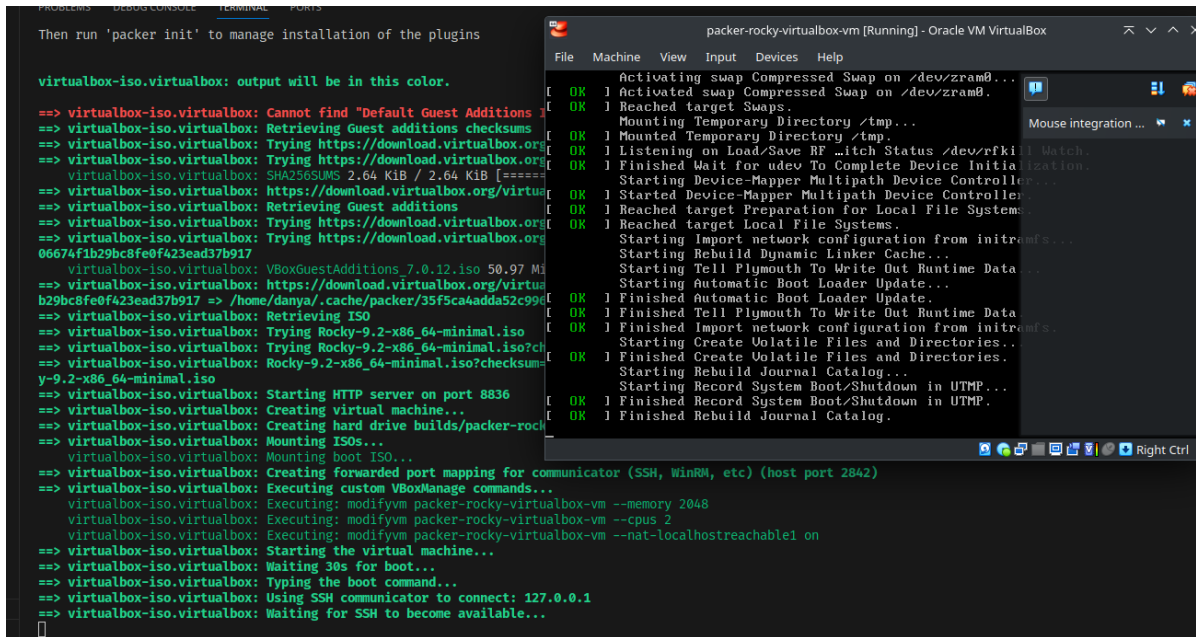


Рис. 3.5: virtualbox

В это время, виртуальная машина выполняла установку системных компонентов вроде загрузчика.

```
Verifying xfsprogs.x86_64 (320/324)
Verifying xz.x86_64 (321/324)
Verifying xz-libs.x86_64 (322/324)
Verifying yum.noarch (323/324)
Verifying zlib.x86_64 (324/324)
.
Installing boot loader
..
Performing post-installation setup tasks
.
Configuring installed system
.....
Writing network configuration
.
Creating users
.....
Configuring addons
.
Generating initramfs
.....
Storing configuration files and kickstarts
.
Running post-installation scripts
.
[anaconda11:main* 2:shell 3:log 4:storage-log >Switch tab: Alt+Tab 1
```

Рис. 3.6: virtualbox

Когда ОС установилась, Packer подключился к виртуальной машине с помощью SSH и выполнил скрипт настройки.

```
==> virtualbox-iso.virtualbox: Cannot find "Default Guest Additions"
==> virtualbox-iso.virtualbox: Retrieving Guest additions checksums
==> virtualbox-iso.virtualbox: Trying https://download.virtualbox.org/virtualbox/7.0.12/VBoxGuestAdditions_7.0.12.iso
virtualbox-iso.virtualbox: SHA256SUMS 2.64 KiB / 2.64 KiB [=====]
==> virtualbox-iso.virtualbox: Trying https://download.virtualbox.org/virtualbox/7.0.12/VBoxGuestAdditions_7.0.12.iso
==> virtualbox-iso.virtualbox: Retrieving Guest additions
==> virtualbox-iso.virtualbox: Trying https://download.virtualbox.org/virtualbox/7.0.12/VBoxGuestAdditions_7.0.12.iso
==> virtualbox-iso.virtualbox: Trying https://download.virtualbox.org/virtualbox/7.0.12/VBoxGuestAdditions_7.0.12.iso
06674f1b29bc8fe0f423ead37b917
virtualbox-iso.virtualbox: VBoxGuestAdditions_7.0.12.iso 50.97 MiB
==> virtualbox-iso.virtualbox: https://download.virtualbox.org/virtualbox/7.0.12/VBoxGuestAdditions_7.0.12.iso => /home/danya/.cache/packer/35f5ca4adda52c996b29bc8fe0f423ead37b917
==> virtualbox-iso.virtualbox: Retrieving ISO
==> virtualbox-iso.virtualbox: Trying Rocky-9.2-x86_64-minimal.iso
==> virtualbox-iso.virtualbox: Trying Rocky-9.2-x86_64-minimal.iso?checksum=y-9.2-x86_64-minimal.iso
==> virtualbox-iso.virtualbox: Starting HTTP server on port 8836
==> virtualbox-iso.virtualbox: Creating virtual machine...
==> virtualbox-iso.virtualbox: Creating hard drive builds/packer-rocky-9.2-x86_64-minimal.vdi
==> virtualbox-iso.virtualbox: Mounting ISOs...
virtualbox-iso.virtualbox: Mounting boot ISO...
==> virtualbox-iso.virtualbox: Creating forwarded port mapping for custom VBoxManage commands...
==> virtualbox-iso.virtualbox: Executing custom VBoxManage commands...
virtualbox-iso.virtualbox: Executing: modifyvm packer-rocky-virtualbox-vm --nat-localhostreachable1 on
virtualbox-iso.virtualbox: Executing: modifyvm packer-rocky-virtualbox-vm --nat-localhostreachable1 on
virtualbox-iso.virtualbox: Executing: modifyvm packer-rocky-virtualbox-vm --nat-localhostreachable1 on
==> virtualbox-iso.virtualbox: Starting the virtual machine...
==> virtualbox-iso.virtualbox: Waiting 30s for boot...
==> virtualbox-iso.virtualbox: Typing the boot command...
==> virtualbox-iso.virtualbox: Using SSH communicator to connect: 127.0.0.1
==> virtualbox-iso.virtualbox: Waiting for SSH to become available...
==> virtualbox-iso.virtualbox: Connected to SSH!
==> virtualbox-iso.virtualbox: Uploading VirtualBox version info (7.0.12)
==> virtualbox-iso.virtualbox: Uploading VirtualBox guest additions ISO...
==> virtualbox-iso.virtualbox: Provisioning with shell script: /tmp/packer/packer-shell13858776556
[ ]
```

Рис. 3.7: packer

После того, как все команды установки были выполнены, виртуальная машина была выключена, а затем экспортирована в box-файл.

```
virtualbox-iso.virtualbox (vagrant): Copying from artifact: builds/packer-rocky-virtualbox-vm-disk001.vmdk
virtualbox-iso.virtualbox (vagrant): Copying from artifact: builds/packer-rocky-virtualbox-vm.mf
virtualbox-iso.virtualbox (vagrant): Copying from artifact: builds/packer-rocky-virtualbox-vm.ovf
virtualbox-iso.virtualbox (vagrant): Renaming the OVF to box.ovf...
virtualbox-iso.virtualbox (vagrant): Compressing: Vagrantfile
virtualbox-iso.virtualbox (vagrant): Compressing: box.ovf
virtualbox-iso.virtualbox (vagrant): Compressing: metadata.json
virtualbox-iso.virtualbox (vagrant): Compressing: packer-rocky-virtualbox-vm-disk001.vmdk
virtualbox-iso.virtualbox (vagrant): Compressing: packer-rocky-virtualbox-vm.mf
Build 'virtualbox-iso.virtualbox' finished after 29 minutes 8 seconds.

==> Wait completed after 29 minutes 8 seconds

==> Builds finished. The artifacts of successful builds are:
--> virtualbox-iso.virtualbox: 'virtualbox' provider box: vagrant-virtualbox-rocky-9-x86_64.box
o [danya@archlinux packer]$ [ ]
```

Рис. 3.8: packer

Теперь мы переходим в папку vagrant, добавляем созданный box-файл, и запускаем две ВМ.

```
[danya@archlinux rudn-year3-subsystem-admin]$ cd 1/work/dmgeneralov/vagrant/
[danya@archlinux vagrant]$ vagrant box add rocky9 ../packer/vagrant-virtualbox-rocky-9-x86_64.box
==> box: Box file was not detected as metadata. Adding it directly...
==> box: Adding box 'rocky9' (v0) for provider:
    box: Unpacking necessary files from: file:///home/danya/Documents/university/rudn-year3-subsystem-admin/1/work/dmgeneralov/packer/vagrant-virtualbox-rocky-9-x86_64.box
==> box: Successfully added box 'rocky9' (v0) for '!'
[danya@archlinux vagrant]$ vagrant up server
Bringing machine 'server' up with 'virtualbox' provider...
==> server: You assigned a static IP ending in ".1" to this machine.
==> server: This is very often used by the router and can cause the
==> server: network to not work properly. If the network doesn't work
==> server: properly, try changing this IP.
==> server: Preparing master VM for linked clones...
    server: This is a one time operation. Once the master VM is prepared,
    server: it will be used as a base for linked clones, making the creation
    server: of new VMs take milliseconds on a modern system.
==> server: Importing base box 'rocky9'...
Progress: 10%
```

Рис. 3.9: vagrant

Обе машины успешно запускаются и настраиваются.

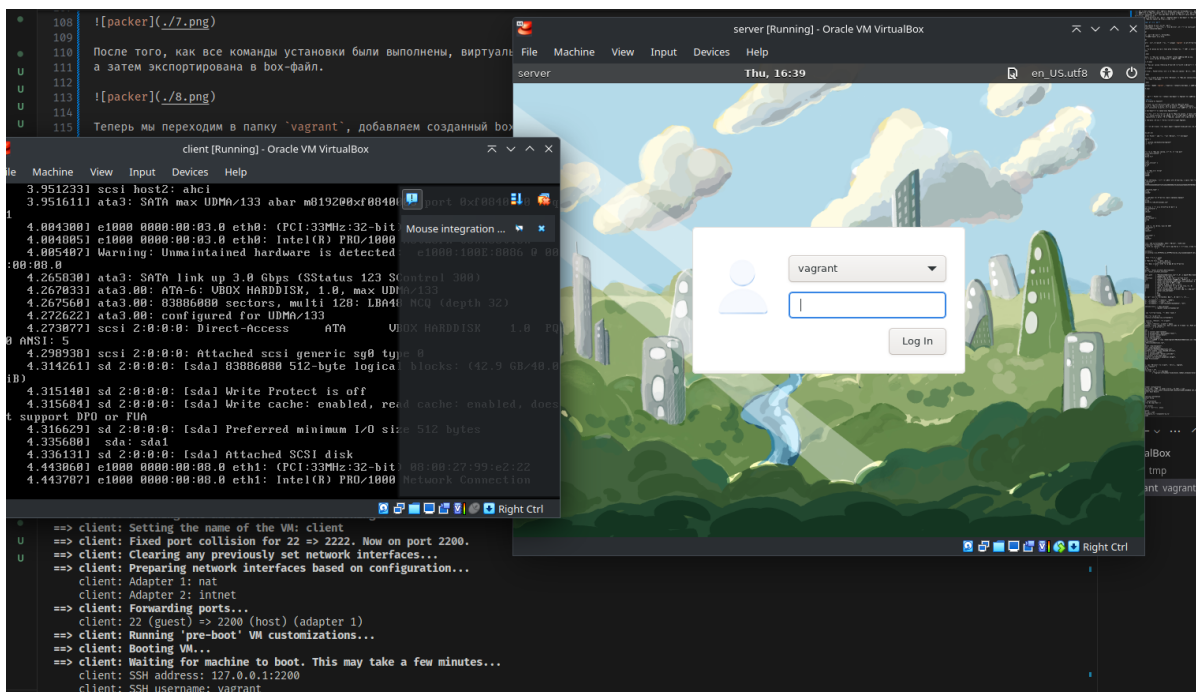


Рис. 3.10: vagrant

## 4 Выводы

Я получил опыт работы с Packer для создания box-файлов и Vagrant для запуска их.

## 5 Контрольные вопросы

### 1. Для чего предназначен Vagrant?

Vagrant – инструмент для автоматической настройки виртуальных машин. Он запускает образ виртуальной машины, затем применяет требуемые изменения, а затем сохраняет измененный образ ВМ, который затем можно запустить на сервере.

### 2. Что такое box-файл? В чём назначение Vagrantfile?

Box-файл – это архив, который содержит образ виртуальной машины, а также метаданные для того, чтобы запустить её с помощью Vagrant. Vagrantfile – это декларативный формат файла, который используется Vagrant для настройки виртуальной машины. Vagrant запускает box-файл, настраивает его с помощью Vagrantfile, и затем эту виртуальную машину можно использовать для серверных задач (как appliance).

### 3. Приведите описание и примеры вызова основных команд Vagrant.

`vagrant box add` – добавить box-файл в контекст Vagrant для использования с виртуальными машинами.

`vagrant up <config>` – запустить виртуальную машину с определенной конфигурацией.

`vagrant up <config> --provision` – запустить виртуальную машину с определенной конфигурацией и выполнить настройку.

`vagrant halt` – остановить все виртуальные машины в контексте  
`vagrant destroy <config>` – удалить виртуальную машину с определенной конфигурацией

#### 4. Дайте построчные пояснения содержания файлов `vagrant-rocky.pkr.hcl`, `ks.cfg`, `Vagrantfile`, `Makefile`

```
# vagrant-rocky.pkr.hcl
```

```
# Какие plugin для Packer требуются, чтобы выполнить этот hcl-файл?
```

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

```
# Как называется та виртуальная машина, которую мы собираем?
```

```
variable "artifact_description" {  
  type    = string  
  default = "Rocky 9.2"  
}
```

```
# Какая версия VM?
```

```
variable "artifact_version" {  
  type    = string  
  default = "9.2"  
}
```



# Какой размер виртуального диска?

```
variable "disk_size" {  
    type    = string  
    default = "40960"  
}
```

# Какая чек-сумма ISO-файла, с которого будет идти установка, и какой тип этой суммы?

```
variable "iso_checksum" {  
    type    = string  
    default = "06505828e8d5d052b477af5ce62e50b938021f5c28142a327d4d5c075f0670dc"  
}
```

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

# Где находится ISO-файл для установки (какое название файла)?

```
variable "iso_url" {  
    type    = string  
    default = "Rocky-9.2-x86_64-minimal.iso"  
}
```

# Это RedHat-система, и это её архитектура процессора

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

```
}
```

```
# Какой релиз RedHat?
```

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

```
# Какой логин и пароль для подключения по SSH?
```

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

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

```
# Чтобы установить ISO на VirtualBox, нужно выполнить следующее
```

```
source "virtualbox-iso" "virtualbox" {
```

```
    # Нажать на следующие клавиши, чтобы ввести загрузчик в состояние, когда он раб  
    файлom.
```

```
    boot_command          = [  
        "<esc>",  
        "<wait><esc><esc>",  
        "linux inst.ks=http://{{.HTTPIP}}:{{.HTTPPort}}/ks.cfg biosdevname=0 net.ifna  
        "<enter>"  
    ]
```

```

# Ждать загрузки столько времени
boot_wait                = "30s"

# Создать виртуальный диск с таким размером
disk_size                = "${var.disk_size}"

# Передать эти опции команде экспорта образа после установки
export_opts              = [
    "--manifest",
    "--vsys", "0",
    "--description", "${var.artifact_description}",
    "--version", "${var.artifact_version}"
]

guest_additions_path     = "VBoxGuestAdditions.iso" # Путь к файлу VBox Guest Add
guest_os_type            = "RedHat_64" # тип ОС
hard_drive_interface     = "sata" # Как подключается жесткий диск?
http_directory           = "${path.root}/http" # Запустить HTTP-
сервер от этой папки
iso_checksum              = "${var.iso_checksum_type}:${var.iso_checksum}" # чекс
файла
iso_url                  = "${var.iso_url}" # путь к ISO
output_directory         = "builds" # куда сохранять вывод
shutdown_command         = "sudo -S /sbin/halt -h -p" # как выключить ВМ
shutdown_timeout         = "5m" # сколько ждать выключения ВМ
ssh_password             = "${var.ssh_password}" # логин и пароль и параметры дл
ssh_username             = "${var.ssh_username}"
ssh_port                 = 22
ssh_pty                  = true
ssh_timeout              = "60m"

# команды для настройки для Virtualbox: память, процессоры, сеть...
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"
}

```

# После того, как ОС установлена, что нужно делать?

```

build {
    # работать над этим образом VM
    sources = ["source.virtualbox-iso.virtualbox"]

    # С помощью консоли, выполнить эти команды:
    provisioner "shell" {
        # запустить bash и выполнить команды в файле
        execute_command = "echo 'packer'|{{ .Vars }} sudo -S -E bash '{{ .Path }}'"
        # содержимое которого приводится:
        inline           = [
            "sleep 30",
            "sudo dnf -y install epel-release",
            "sudo dnf -y groupinstall 'Development Tools'",
            "sudo dnf -y install kernel-devel",
            "sudo dnf -y install dkms",
            "sudo mkdir /tmp/vboxguest",
            "sudo mount -t iso9660 -o loop /home/vagrant/VBoxGuestAdditions.iso /tmp/vb",
            "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!"
    ]
}

# после того, как выполнены все команды, с помощью vagrant,
post-processor "vagrant" {
    # сжать образ
    compression_level = "6"

    # и экспортировать его в этот box-файл
    output              = "vagrant-virtualbox-rocky-${var.redhat_release}-
${var.redhat_platform}.box"
}
}

# ks.conf

# System bootloader configuration
# Настройка загрузчика: в MBR, серийная консоль, не ждать в меню
bootloader --append="no_timer_check console=tty0 console=ttyS0,115200n8 net.ifname=
location=mbr --timeout=1
# Clear the Master Boot Record
# обнулить MBR

```

```
zerombr

# Partition clearing information
# обнулить разделы диска
clearpart --all

# Reboot after installation
# после установки перезагрузиться
reboot

# Use text mode install
# устанавливаться в текстовом режиме
text

# Keyboard layouts
# раскладка русская и США
keyboard --vckeymap=us,ru --xlayouts='us,ru'

# System language
# язык английский
lang en_US.UTF-8


# Network information
# сеть по DHCP
network --bootproto=dhcp --device=link --activate


# System authorization information
# системный логин через SSSD
authselect select sssd with-sudo with-mkhomedir --force
authselect apply-changes

# Root password
# создать пользователя и не требовать изменить пароль при загрузке
rootpw vagrant

user --name=vagrant --password=vagrant
```

```
firstboot --disable
# Do not configure the X Window System
# не пропускать настройку Xorg
#skipx
# System services
# включить NetworkManager, sshd и chronyd сервисы
services --enabled="NetworkManager,sshd,chronyd"
# System timezone
# время в UTC
timezone UTC --utc
user --name=vagrant --password=vagrant
# Disk partitioning information
# корневой раздел на xfs
part / --fstype="xfs" --size=10239

# после установки, shell-команды
%post
# configure swap to a file
# выделить 2ГБ на swap и монтировать его
fallocate -l 2G /swapfile
chmod 600 /swapfile
mkswap /swapfile
echo "/swapfile none swap defaults 0 0" >> /etc/fstab

# sudo
# разрешить sudo без пароля для vagrant
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
# настраивать eth0 для DHCP
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
# не выключать вход по паролю для SSH
#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
# разрешить su для пользователей, кроме root и vagrant
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).
```

```
# очистить machine-id для генерации
```

```
:>/etc/machine-id
```

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

```
# Blacklist the floppy module to avoid probing timeouts
```

```
# выключить поддержку дискеты
```

```
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):
# сгенерировать новый initramfs
KERNEL_VERSION=$(rpm -q kernel --qf '%{version}-%{release}.%{arch}\n')
dracut -f /boot/initramfs-${KERNEL_VERSION}.img ${KERNEL_VERSION}

# Seal for deployment
# удалить SSH-идентификаторы
rm -rf /etc/ssh/ssh_host_*
hostnamectl set-hostname localhost.localdomain
rm -rf /etc/udev/rules.d/70-*
%end

# При установке, также установить эти пакеты, и не устанавливать какие-
то другие.
%packages --inst-langs=en
bash-completion
bzip2
chrony
man-pages
rsync

```

```

-dracut-config-rescue
-iwl100-firmware
-iwl1000-firmware    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, "--accelerate3d", "on"]
end

-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

# Выключить kdump
%addon com_redhat_kdump --disable --reserve-mb='128'

%end

# Vagrantfile

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

# настройка версии 2
Vagrant.configure("2") do |config|

  ## Common configuration
  # перечисляем скрипты, которые надо выполнять всегда
  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
# конфигурация только для серверов
config.vm.define "server", autostart: false do |server|
    # настройки hostname BM
    server.vm.box = "rocky9"
    server.vm.hostname = 'server'

    # ждать загрузки столько времени
    server.vm.boot_timeout = 1440

    # это логин и пароль для SSH
    server.ssh.insert_key = false
    server.ssh.username = 'vagrant'
    server.ssh.password = 'vagrant'

    # виртуальная машина должна запускаться в этой сети
    server.vm.network :private_network,
        ip: "192.168.1.1",
        virtualbox____intnet: true

    # запустить этот скрипт

```

```

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

# для virtualbox-машин, надо выполнить настройки:
server.vm.provider :virtualbox do |v|
    # использовать copy on write
    v.linked_clone = true
    # Customize the amount of memory on the VM
    # настроить количество ОЗУ и процессоров на ВМ
    v.memory = 1024
    v.cpus = 1
    v.name = "server"
    # Display the VirtualBox GUI when booting the machine
    # показывать окно ВМ
    v.gui = true
    # Set the video memory to 12Mb
    # настройки свойств virtualbox
    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, "--accelerate3d", "on"]
end
end

## Client configuration
# конфигурация только для клиентов

```

```

config.vm.define "client", autostart: false do |client|

  # имя машины
  client.vm.box = "rocky9"
  client.vm.hostname = 'client'

  # ждать загрузки столько
  client.vm.boot_timeout = 1440

  # логин и пароль для входа в систему
  client.ssh.insert_key = false
  client.ssh.username = 'vagrant'
  client.ssh.password = 'vagrant'

  # сеть -- частная, с DHCP
  client.vm.network :private_network,
                    type: "dhcp",
                    virtualbox____intnet: true

  # запустить эти скрипты
  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"

```

```

# для virtualbox-машин, надо выполнить настройки:
server.vm.provider :virtualbox do |v|
  # использовать copy on write
  v.linked_clone = true
  # Customize the amount of memory on the VM
  # настроить количество ОЗУ и процессоров на ВМ
  v.memory = 1024
  v.cpus = 1
  v.name = "server"
  # Display the VirtualBox GUI when booting the machine
  # показывать окно ВМ
  v.gui = true
  # Set the video memory to 12Mb
  # настройки свойств virtualbox
  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, "--accelerate3d", "on"]
end
end
end

# Makefile
.PHONY: version

# по умолчанию, делать init и box
all: init box

```



```
# установить зависимости для packer в локальную папку
```

```
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"
```

```
rocky.pkr.hcl
```

```
# использовать packer, чтобы собрать box-файл
```

```
box: ## 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"
```

```
only=virtualbox-iso.virtualbox vagrant-rocky.pkr.hcl
```

```
@VBoxManage setproperty machinefolder default
```

```
# перечислить команды, которые можно выполнить с этим Makefile
```

```
help:
```

```
@echo 'Usage:'
```

```
@echo '  make <target>'
```

```
@echo
```

```
@echo 'Targets:'
```

```
@grep -E '^[a-zA-Z_0-9.-]+:.*?## .*$$' $(MAKEFILE_LIST) | sort | awk 'BEGIN {FS=":"; OFS="\n";} {printf "%s\n", $1, $2}'
```

```
30s\033[0m %s\n", $$1, $$2}'
```

```
@echo
```