Лабораторная работа №1

Подготовка лабораторного стенда

Жукова Арина Александровна

Содержание

Список иллюстраций

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.
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.
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 Теоретическое введение

3.1 Назначение Vagrant

Vagrant — это инструмент для создания и управления средами виртуальных машин, позволяющий автоматизировать процесс развертывания операционных систем и настройки программного обеспечения. Vagrant обеспечивает:

- Единообразие сред разработки и тестирования
- Автоматизацию настройки виртуальных машин
- Простое тиражирование инфраструктуры

3.2 Основные компоненты Vagrant

3.2.1 Провайдер (Provider)

Система виртуализации, с которой работает Vagrant:

- VirtualBox (наиболее распространенный)
- VMWare
- Hyper-V
- Docker

3.2.2 Box-файл (Vagrant Box)

Готовый образ виртуальной машины с предустановленной ОС, используемый как шаблон для создания новых ВМ. Box-файлы могут быть:

- Официальными (из каталога Vagrant Cloud)
- Пользовательскими (собственной сборки)

3.2.3 Vagrantfile

Конфигурационный файл на языке Ruby, содержащий:

- Настройки виртуальной машины
- Параметры сети
- Скрипты провижининга
- Настройки провайдера

3.3 Ключевые команды Vagrant

Таблица 3.1: Ключевые команды

Команда	Назначение
`vagrant up`	Запуск ВМ
`vagrant halt`	Остановка ВМ
`vagrant destroy`	Удаление ВМ
`vagrant ssh`	Подключение по SSH
`vagrant provision`	Применение скриптов настройки
`vagrant reload`	Перезагрузка ВМ

3.4 Сетевая топология лабораторного стенда

Стенд состоит из двух виртуальных машин:

3.4.1 Сервер

- **Роли**: маршрутизатор, DHCP-сервер
- **Сетевые интерфейсы**:
  - eth0: управление + внешняя сеть
  - eth1: внутренняя сеть (192.168.1.1)

3.4.2 Клиент

- **Сетевые интерфейсы**:
  - eth0: управление
  - eth1: внутренняя сеть (DHCP)

3.5 Вспомогательные инструменты

3.5.1 Packer

Инструмент для создания образов виртуальных машин, использующий HCL-файлы для описания процесса сборки.

3.5.2 HCL (Hashicorp Configuration Language)

Декларативный язык конфигурации, основанный на JSON, используемый для описания инфраструктуры.

3.5.3 Kickstart (ks.cfg)

Файл автоматической установки Rocky Linux, содержащий все необходимые настройки для бесприсутственного развертывания ОС.

3.6 Преимущества использования Vagrant

1. **Повторяемость** — идентичная среда на разных машинах
2. **Изоляция** — независимость от основной системы
3. **Автоматизация** — минимизация ручных операций
4. **Документирование** — конфигурация как код
5. **Масштабируемость** — простое создание сложных стендов

4 Выполнение лабораторной работы

Для работы была выбрана операционная система Windows.

4.1 Подготовка каталога проекта

1. Создание папок `C:\work\aazhukova\packer` и `C:\work\aazhukova\vagrant` (рис. 4.1).

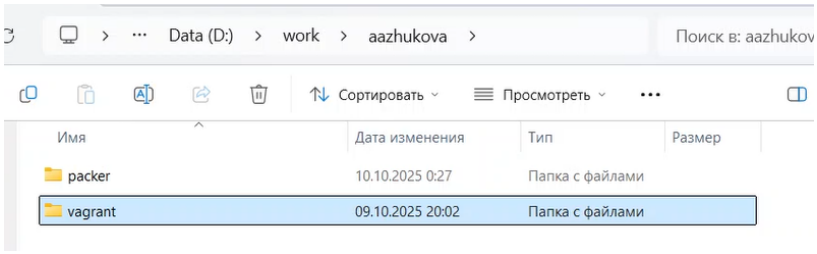

Рисунок 4.1: Создание папок

2. В созданном рабочем каталоге в подкаталоге packer разместила образ варианта операционной системы Rocky Linux, vagrant-rocky.pkr.hcl (рис. 4.2).

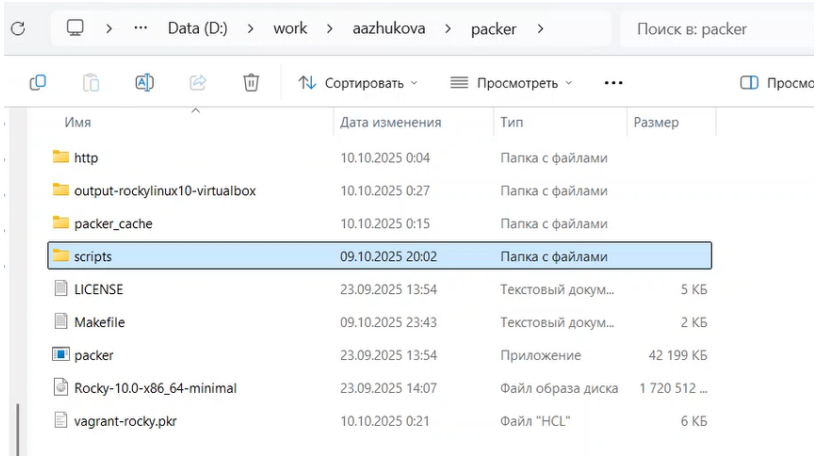

Рисунок 4.2: Подкаталог packer

Создаем подкаталог http с файлом ks.cfg (рис. 4.3).

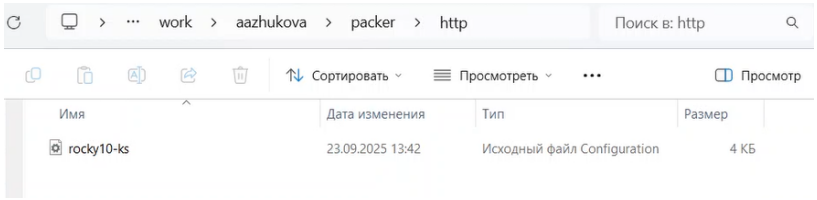

Рисунок 4.3: подкаталог http

В созданном рабочем каталоге в подкаталоге vagrant файл Vagrantfile, файл Makefile.(рис. 4.4).

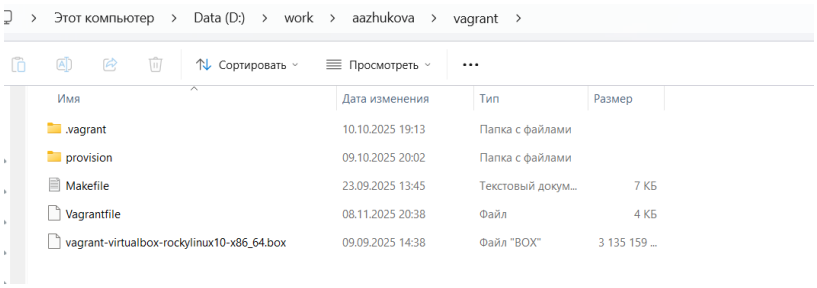

Рисунок 4.4: подкаталог vagrant

Создаём каталог provision с подкаталогами default, server и client (рис. 4.5).

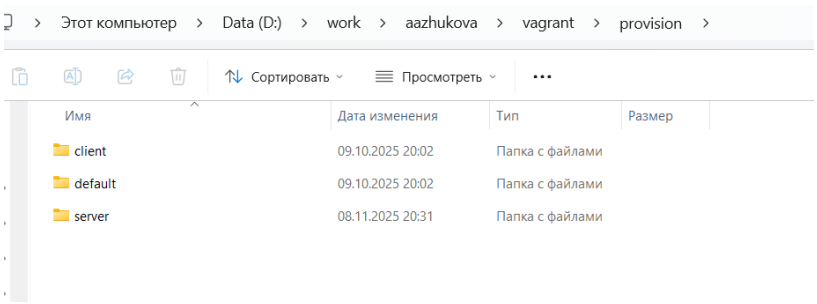

Рисунок 4.5: каталог provision

В каталоге default размещаем скрипт 01-user.sh, скрипт-заглушку 01-dummy.sh, скрипт 01-hostname.sh (рис. 4.6).

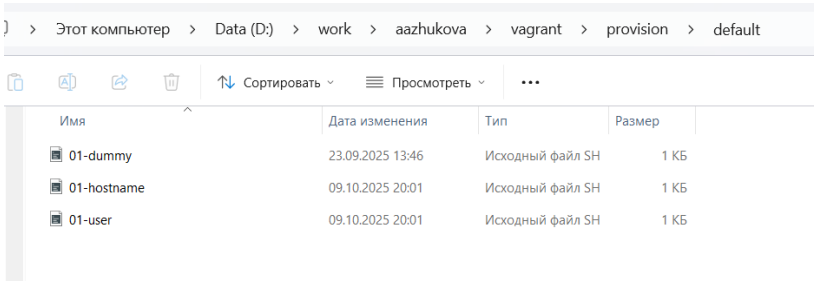

Рисунок 4.6: каталог default

В каталоге server размещаем скрипт-заглушку 01-dummy.sh, скрипт 02-forward.sh (рис. 4.7).

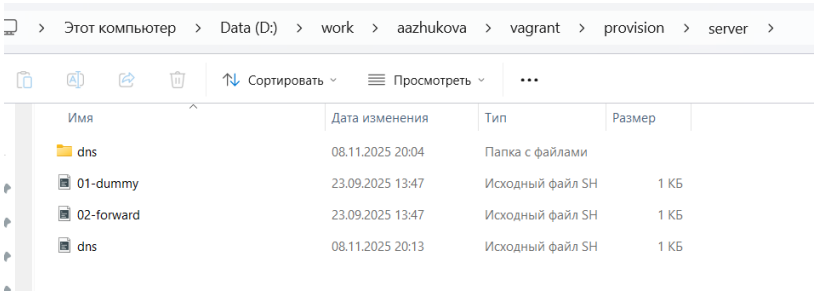

Рисунок 4.7: каталог server

В каталоге client размещаем скрипт-заглушку 01-dummy.sh, скрипт 01-routing.sh (рис. 4.8).

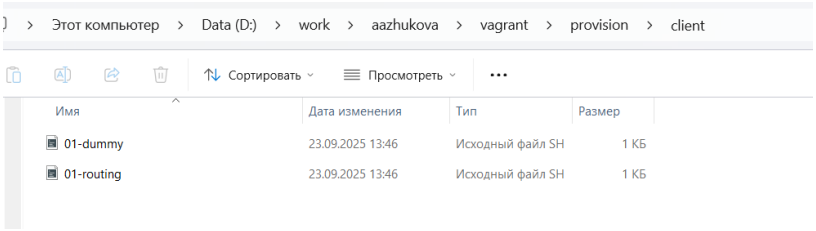

Рисунок 4.8: каталоге client

4.2 Развёртывание лабораторного стенда на ОС Windows

1. Используя FAR, переходим в созданный рабочий каталог с проектом. В этом же каталоге размещён файл packer.exe. В командной строке ввожу `packer.exe init vagrant-rocky.pkr.hcl` и `packer.exe build vagrant-rocky.pkr.hcl` (рис. 4.9).

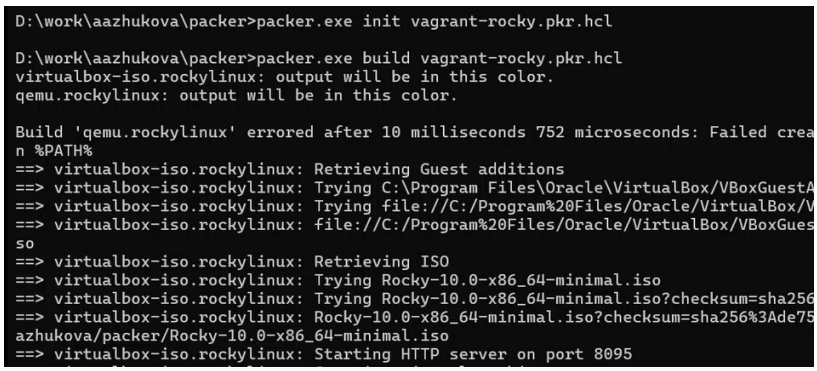

Рисунок 4.9: Автоматическая установка образа ОС

2. Для регистрации образа виртуальной машины в vagrant в командной строке ввожу `vagrant box add rockylinux10 vagrant-virtualbox-rockylinux10-x86_64.box` (рис. 4.10).

```
D:\work\aazhukova\packer>vagrant box add rockylinux10 vagrant-virtualbox-r
==> box: Box file was not detected as metadata. Adding it directly...
==> box: Adding box 'rockylinux10' (v0) for provider: (amd64)
    box: Unpacking necessary files from: file://D:/work/aazhukova/packer/v
agrant-virtualbox-rockylinux10-x86_64.box
    box:
==> box: Successfully added box 'rockylinux10' (v0) for '(amd64)'!
```

Рисунок 4.10: Регистрация образа виртуальной машины

3. Для запуска виртуальной машины Server ввожу в консоли `vagrant up server` (рис. 4.11).

```
D:\work\aazhukova\vagrant>vagrant up server
Bringing machine 'server' up with 'virtualbox' provider...
==> server: You assigned a static IP ending in ".1" or ":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: You assigned a static IP ending in ".1" or ":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.
```

Рисунок 4.11: Запуск Server

Входим в аккаунт пользователя vagrant с паролем vagrant в графическом окружении (рис. 4.12).

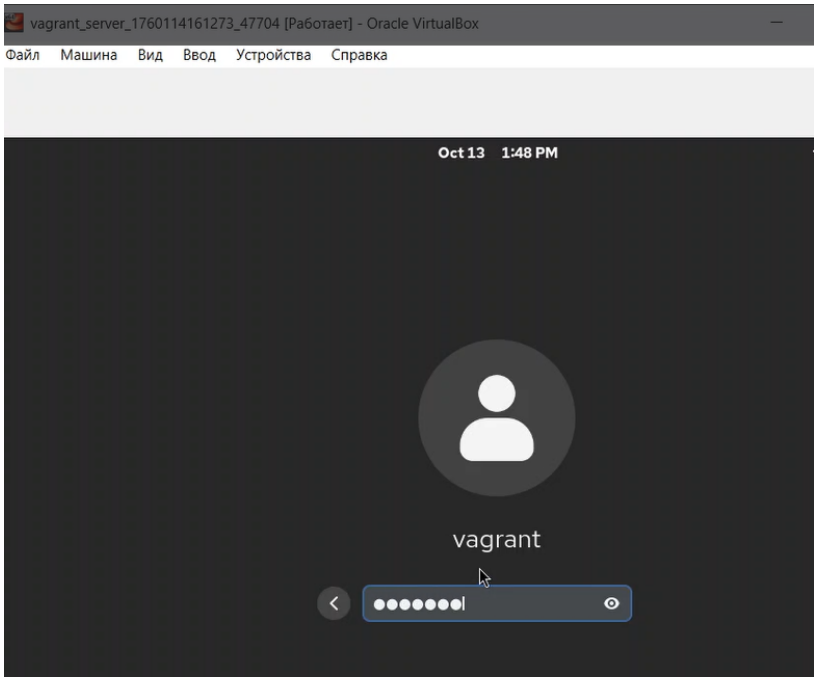

Рисунок 4.12: Пользователь vagrant в Server

4. Для запуска виртуальной машины Client введите в консоли `vagrant up client` (рис. 4.13).

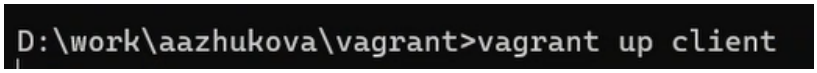

Рисунок 4.13: Запуск Client

Входим в аккаунт пользователя vagrant с паролем vagrant в графическом окружении (рис. 4.14).

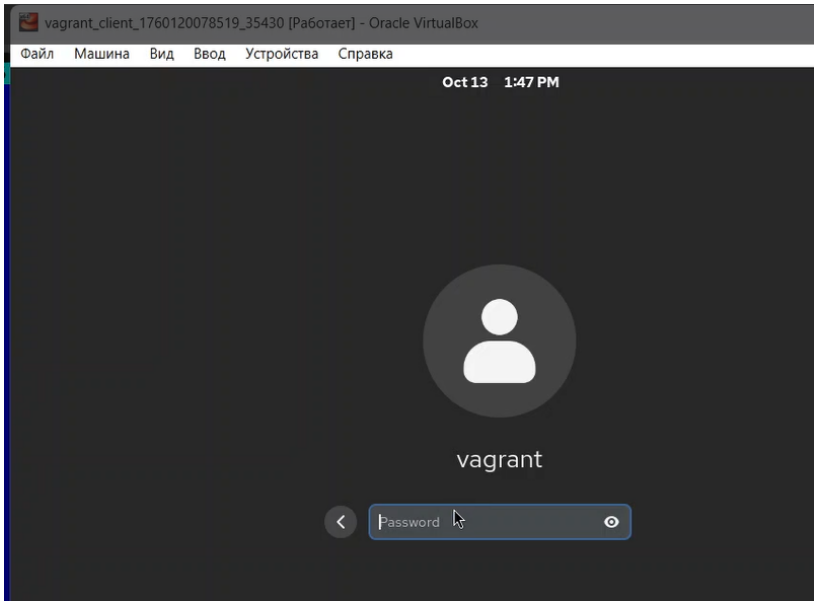

Рисунок 4.14: Пользователь vagrant в Client

5. Подключаемся к серверу из консоли: `vagrant ssh server`, ввожу пароль vagrant. Переходим к пользователю aazhukova: `su - user` (рис. 4.15).

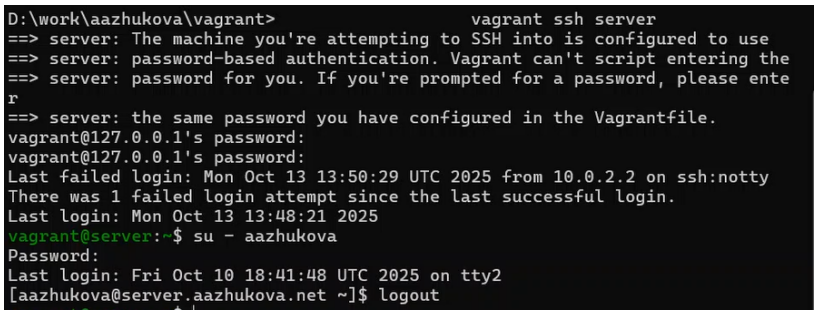

Рисунок 4.15: Подключение к серверу из консоли

6. Подключаемся к клиенту из консоли: `vagrant ssh client`, ввожу пароль vagrant. Переходим к пользователю aazhukova: `su - user` (рис. 4.16).

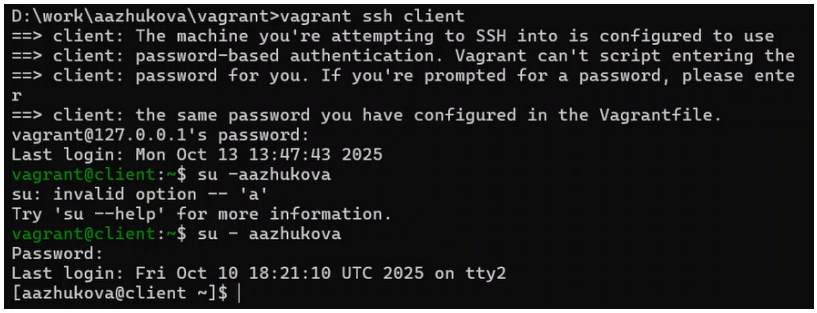

Рисунок 4.16: Подключение к клиенту из консоли

7. Выключаем обе машины (рис. 4.17).

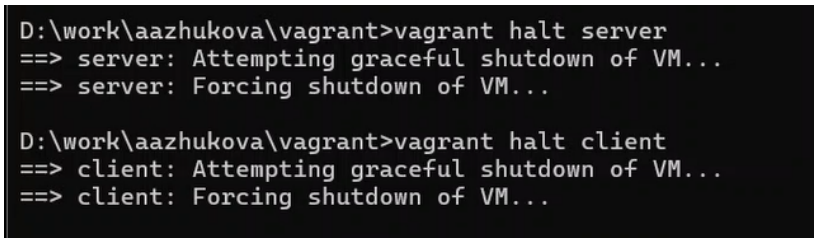

Рисунок 4.17: Выключение машин

4.3 Внесение изменений в настройки внутреннего окружения виртуальной машины

1. Убеждаемся, 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”
```

(рис. 4.18).

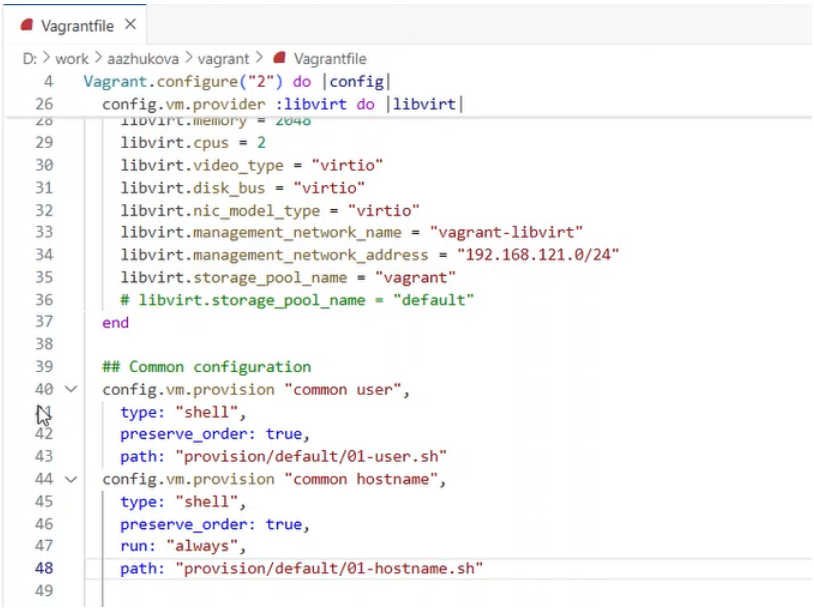

Рисунок 4.18: файл Vagrantfile

2. Фиксируем изменения (рис. 4.19).

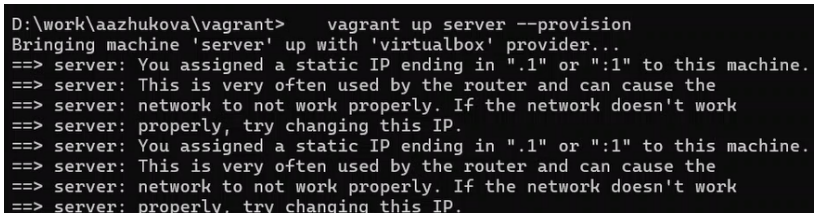

Рисунок 4.19: Фиксация изменений

3. Заходим на сервер и клиент под созданными пользавателями, проверяем корректность имён и выключаем машины (рис. 4.20).

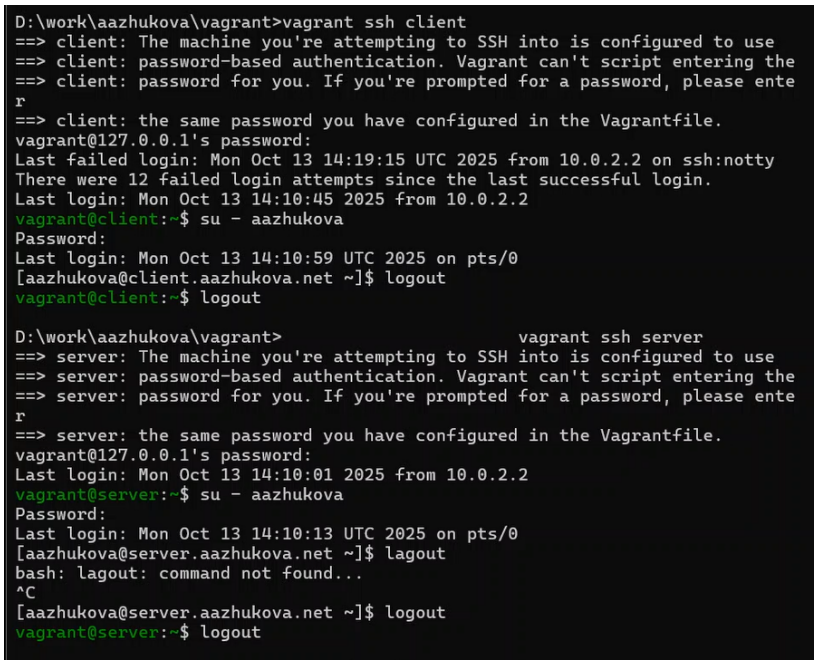

Рисунок 4.20: Проверка имён

4. Копируем файлы для развертывания системы на другом компьютере (рис. 4.21).

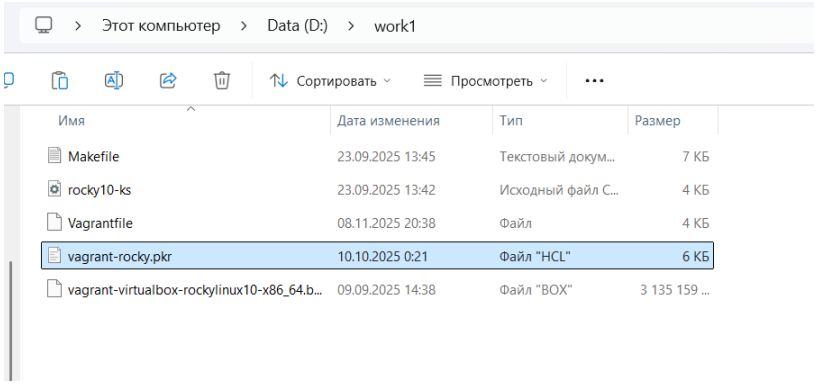

Рисунок 4.21: Файлы для работы

4.4 Конфигурационные файлы

4.4.1 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"
    }
    qemu = {
      version = "~> 1"
      source  = "github.com/hashicorp/qemu"
    }
  }
}
```

```
variable "artifact_description" {
  type    = string
  default = "Rocky 10.0"
}

variable "artifact_version" {
  type    = string
  default = "10.0"
}

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

variable "iso_checksum" {
  type    = string
 default = "de75c2f7cc566ea964017a1e94883913f066c4ebeb1d356964e398ed76cadd12"
}

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

variable "iso_url" {
  type    = string
 # default = "https://download.rockylinux.org/pub/rocky/10/isos/x86_64/Rocky-
```

```
10.0-x86_64-minimal.iso"
  default = "Rocky-10.0-x86_64-minimal.iso"
}

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

variable "redhat_release" {
  type    = string
  default = "10"
}

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

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

variable "http_directory" {
  type    = string
  default = "http"
}
```

```
source "qemu" "rockylinux" {
  boot_command           = [
    "<up>",
    "e",
    "<down><down><end><wait>",
    " inst.ks=http://{{ .HTTPIP }}:{{ .HTTPPort }}/rocky10-ks.cfg ",
    " biosdevname=0 net.ifnames=0 ",
    "<enter><wait><leftCtrlOn>x<leftCtrlOff>"
  ]
  boot_wait               = "10s"
  disk_size               = "${var.disk_size}"
  http_directory          = "${path.root}/${var.http_directory}"
  iso_checksum          = "${var.iso_checksum_type}:${var.iso_checksum}"
  iso_url                 = "${var.iso_url}"
  output_directory       = "output-rockylinux${var.redhat_release}-qemu"
  format                  = "qcow2"
  ssh_password            = "${var.ssh_password}"
  ssh_username            = "${var.ssh_username}"
  ssh_timeout             = "60m"
  vm_name                 = "rockylinux${var.redhat_release}-qemu"
  net_device              = "virtio-net"
  disk_interface          = "virtio"
  # headless                = true

  # Настройки QEMU
  # Параметры процессора
  cpus                  = 2
  memory                = 2048
  accelerator           = "kvm"
```

```
  cpu_model           = "host"         # Использовать характеристики хоста
  machine_type            = "q35"             # Тип системной платы
  firmware          = "/usr/share/edk2-ovmf/OVMF_CODE.fd" # UEFI вместо BIOS. Проверьте

  # Настройки видео
  vga               = "virtio" # Для virtio-vga

  ## Дополнительные флаги процессора
  qemuargs = [
    ["-device", "qemu-xhci"], # Виртуализированные USB-контроллеры
    ["-device", "virtio-tablet"], # Устройства ввода
    ## GPU-passthrough
    # ["-device", "virtio-gpu-pci"], #  3D-акселерация через VirGL
    # ["-vga", "none"]
  ]
}

source "virtualbox-iso" "rockylinux" {
  boot_command = [
    "<up>",
    "e",
    "<down><down><end><wait>",
    " inst.ks=http://{{ .HTTPIP }}:{{ .HTTPPort }}/rocky10-ks.cfg ",
    " biosdevname=0 net.ifnames=0 ",
    "<enter><wait><leftCtrlOn>x<leftCtrlOff>"
  ]
  boot_wait               = "10s"
  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"
  http_directory          = "${var.http_directory}"
  iso_checksum          = "${var.iso_checksum_type}:${var.iso_checksum}"
  iso_url                 = "${var.iso_url}"
   output_directory          = "output-rockylinux${var.redhat_release}-
virtualbox"
  shutdown_command        = "sudo -S /sbin/halt -h -p"
  shutdown_timeout        = "5m"
  ssh_password            = "${var.ssh_password}"
  ssh_username            = "${var.ssh_username}"
  ssh_port                = 22
  ssh_pty                 = true
  ssh_timeout             = "60m"
  iso_interface           = "sata"
  headless                = true
  vboxmanage              = [
    [ "modifyvm", "{{.Name}}", "--memory", "2048" ],
    [ "modifyvm", "{{.Name}}", "--cpus", "2" ],
    [ "modifyvm", "{{.Name}}", "--nat-localhostreachable1", "on" ],
    [ "modifyvm", "{{.Name}}", "--firmware", "EFI" ],
    ["modifyvm", "{{.Name}}", "--vrde", "on"],
    ["modifyvm", "{{.Name}}", "--vrdeport", "3390"]
  ]
```

```
  virtualbox_version_file = ".vbox_version"
  vm_name                 = "rockylinux${var.redhat_release}-virtualbox"
}

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

  provisioner "shell" {
        execute_command  =  "echo  'packer'|{{  .Vars  }}  sudo  -S  -
E bash '{{ .Path }}'"
    scripts          = ["scripts/vagrant.sh", "scripts/software.sh"]
  }

  provisioner "shell" {
    only            = ["virtualbox-iso.rockylinux"]
    script          = "scripts/virtualbox.sh"
  }

  provisioner "shell" {
    script          = "scripts/cleanup.sh"
  }

  post-processor "vagrant" {
    compression_level = "6"
   output         = "vagrant-{{ .Provider }}-rockylinux${var.redhat_release}-
${var.redhat_platform}.box"
```

```
  }
}
```

4.4.2 Содержание файла rocky10-ks.cfg

```
# System bootloader configuration
bootloader --append="no_timer_check console=tty0 console=ttyS0,115200n8 net.ifnames=0 b
-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
bootloader --location=mbr
clearpart --all --initlabel
autopart --type=lvm

%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
@core
openssh-server
sudo
%end

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

4.4.3 Содержание файла Vagrantfile

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

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

  config.vagrant.plugins = "vagrant-libvirt"
  config.vagrant.plugins = "vagrant-vbguest"
```

```
config.vm.provider :virtualbox do |virtualbox|
  virtualbox.linked_clone = true
  # Customize the amount of memory on the VM
  virtualbox.memory = 4096
  virtualbox.cpus = 2
  ## Display the VirtualBox GUI when booting the machine
  virtualbox.gui = false
  ## Set the video memory to 12Mb
  virtualbox.customize ["modifyvm", :id, "--vram", "128"]
 virtualbox.customize ["modifyvm", :id, "--natdnshostresolver1", "on"]
 virtualbox.customize ["modifyvm", :id, "--clipboard", "bidirectional"]
 virtualbox.customize ["modifyvm", :id, "--draganddrop", "bidirectional"]
 virtualbox.customize ["modifyvm", :id, "--graphicscontroller", "vboxsvga"]
  virtualbox.customize ["modifyvm", :id, "--accelerate3d", "on"]
  virtualbox.customize ["modifyvm", :id, "--nested-hw-virt", "on"]
end

config.vm.provider :libvirt do |libvirt|
  libvirt.driver = "kvm"
  libvirt.memory = 2048
  libvirt.cpus = 2
  libvirt.video_type = "virtio"
  libvirt.disk_bus = "virtio"
  libvirt.nic_model_type = "virtio"
  libvirt.management_network_name = "vagrant-libvirt"
  libvirt.management_network_address = "192.168.121.0/24"
  libvirt.storage_pool_name = "vagrant"
  # libvirt.storage_pool_name = "default"
```

```
end

## 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"

## Server configuration
config.vm.define "server", autostart: false do |server|
  server.vm.box = "rockylinux10"
  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.provider :virtualbox do |virtualbox|
```

```
    virtualbox.customize ["modifyvm", :id, "--vrde", "on"]
    virtualbox.customize ["modifyvm", :id, "--vrdeport", "3391"]
  end

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

end

## Client configuration
config.vm.define "client", autostart: false do |client|
  client.vm.box = "rockylinux10"
  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,
                    ip: "192.168.1.2",
                    virtualbox__intnet: true

  client.vm.provider :virtualbox do |virtualbox|
    virtualbox.customize ["modifyvm", :id, "--vrde", "on"]
    virtualbox.customize ["modifyvm", :id, "--vrdeport", "3392"]
```

```
    end

    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"
  end
end
```

4.4.4 Содержание файла Makefile для packer

```
# Цели по умолчанию
.DEFAULT_GOAL := help

.PHONY: version

# Получение текущего каталога
CURDIR := $(realpath .)

init: ## Install missing plugins for packer
    @mkdir -p "$(CURDIR)""/.config/packer/plugins"
   @export PACKER_CONFIG_DIR="$(CURDIR)""/.config/packer"; export PACKER_PLUGIN_PATH="
rocky.pkr.hcl
```

```
virtualbox: init    ## Build Rocky Linux box for Virtualbox
    -@VBoxManage setproperty language C
    -@VBoxManage setproperty machinefolder "$(CURDIR)"/vm
   -@export TMPDIR="$(CURDIR)"; export PACKER_CONFIG_DIR="$(CURDIR)""/.config/packer";
only=virtualbox-iso.rockylinux vagrant-rocky.pkr.hcl
    -@VBoxManage setproperty machinefolder default

qemu:   init    ## Build Rocky Linux box for Qemu
   -@export TMPDIR="$(CURDIR)"; export PACKER_CONFIG_DIR="$(CURDIR)""/.config/packer";
only=qemu.rockylinux vagrant-rocky.pkr.hcl

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

4.4.5 Содержание файла Makefile для vagrant

```
## Конфигурация
BOX_NAME := rockylinux10
VAGRANT_FILE ?= Vagrantfile
## Set your provider
# PROVIDERS := virtualbox libvirt
PROVIDERS := virtualbox
```

```
# Цели по умолчанию
.DEFAULT_GOAL := help

.PHONY: version

# Получение текущего каталога
CURDIR := $(realpath .)

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

check-virtualbox:  ## Check VirtualBox
  @which VBoxManage >/dev/null 2>&1 || (echo "VirtualBox is not installed"; exit 1)

check-libvirt:  ## Check libvirt
  @which virsh >/dev/null 2>&1 || (echo "libvirt is not installed"; exit 1)
  @export VAGRANT_HOME="$(CURDIR)"/.vagrant.d; export VAGRANT_DOTFILE_PATH="$(CURDIR)
q vagrant-libvirt || (echo "The vagrant-libvirt plugin is not installed"; exit 1)

plugins:    ## Install plugins
  @export VAGRANT_HOME="$(CURDIR)"/.vagrant.d; export VAGRANT_DOTFILE_PATH="$(CURDIR)
-plugin-clean-sources  --plugin-source  https://rubygems.org  vagrant-
```

```
libvirt
   @export VAGRANT_HOME="$(CURDIR)"/.vagrant.d; export VAGRANT_DOTFILE_PATH="$(CURDIR)
-plugin-clean-sources   --plugin-source   https://rubygems.org   vagrant-
vbguest
    touch plugins

addbox: plugins ## Add the built box to Vagrant
    @for provider in $(PROVIDERS); do \
     export VAGRANT_HOME="$(CURDIR)"/.vagrant.d; export VAGRANT_DOTFILE_PATH="$(CURDIR
$$provider-$(BOX_NAME)-x86_64.box ; \
       if [[ "$$provider" == "libvirt" ]] then mkdir -p vm ; virsh pool-
define-as --name vagrant --type dir --target "$(CURDIR)"/vm ; virsh pool-
start vagrant ; fi ; \
    done

server-up: plugins  ## Start server
    @VBoxManage setproperty machinefolder "$(CURDIR)"/vm
   @export VAGRANT_HOME="$(CURDIR)"/.vagrant.d; export VAGRANT_DOTFILE_PATH="$(CURDIR)
-provider=virtualbox server
    @VBoxManage setproperty machinefolder default

client-up: plugins  ## Start client
    @VBoxManage setproperty machinefolder "$(CURDIR)"/vm
   @export VAGRANT_HOME="$(CURDIR)"/.vagrant.d; export VAGRANT_DOTFILE_PATH="$(CURDIR)
-provider=virtualbox client
    @VBoxManage setproperty machinefolder default

server-ssh: ## Start server
    @VBoxManage setproperty machinefolder "$(CURDIR)"/vm
```

```
   @export VAGRANT_HOME="$(CURDIR)"/.vagrant.d; export VAGRANT_DOTFILE_PATH="$(CURDIR)
    @VBoxManage setproperty machinefolder default

client-ssh: ## Start client
    @VBoxManage setproperty machinefolder "$(CURDIR)"/vm
   @export VAGRANT_HOME="$(CURDIR)"/.vagrant.d; export VAGRANT_DOTFILE_PATH="$(CURDIR)
    @VBoxManage setproperty machinefolder default

server-halt: plugins    ## Stop server
    @VBoxManage setproperty machinefolder "$(CURDIR)"/vm
   @export VAGRANT_HOME="$(CURDIR)"/.vagrant.d; export VAGRANT_DOTFILE_PATH="$(CURDIR)
    @VBoxManage setproperty machinefolder default

client-halt: plugins    ## Stop client
    @VBoxManage setproperty machinefolder "$(CURDIR)"/vm
   @export VAGRANT_HOME="$(CURDIR)"/.vagrant.d; export VAGRANT_DOTFILE_PATH="$(CURDIR)
    @VBoxManage setproperty machinefolder default

server-provision: plugins   ## Start and provision server
    @for provider in $(PROVIDERS); do \
        VBoxManage setproperty machinefolder "$(CURDIR)"/vm; \
     export VAGRANT_HOME="$(CURDIR)"/.vagrant.d; export VAGRANT_DOTFILE_PATH="$(CURDIF
-provider=$$provider server --provision ; \
        VBoxManage setproperty machinefolder default ;\
    done

client-provision: plugins   ## Start and provision client
    @for provider in $(PROVIDERS); do \
        VBoxManage setproperty machinefolder "$(CURDIR)"/vm ; \
```

```
      export VAGRANT_HOME="$(CURDIR)"/.vagrant.d; export VAGRANT_DOTFILE_PATH="$(CURDIR
-provider=$$provider client --provision ; \
        VBoxManage setproperty machinefolder default ; \
    done

server-destroy: ## Destroy server
    @VBoxManage setproperty machinefolder "$(CURDIR)"/vm
   @export VAGRANT_HOME="$(CURDIR)"/.vagrant.d; export VAGRANT_DOTFILE_PATH="$(CURDIR)
f server
    @VBoxManage setproperty machinefolder default

client-destroy: ## Destroy client
    @VBoxManage setproperty machinefolder "$(CURDIR)"/vm
   @export VAGRANT_HOME="$(CURDIR)"/.vagrant.d; export VAGRANT_DOTFILE_PATH="$(CURDIR)
f client
    @VBoxManage setproperty machinefolder default

status: plugins  ## Show status of all VMs
    @for provider in $(PROVIDERS); do \
        echo "=== Provider: $$provider ==="; \
      export VAGRANT_HOME="$(CURDIR)"/.vagrant.d; export VAGRANT_DOTFILE_PATH="$(CURDIR
-machine-readable --provider=$$provider | awk -F, '$$2 == "default" && $$3 == "state" {pr
    done

destroy: server-destroy client-destroy  ## Destroy all VM

clean: destroy  ## Full clean
    rm -rf .vagrant
    rm -f plugins
```

```
  @export VAGRANT_HOME="$(CURDIR)"/.vagrant.d; export VAGRANT_DOTFILE_PATH="$(CURDIR)
-all --force || true
    @for provider in $(PROVIDERS); do \
            if [[ "$$provider" == "libvirt" ]] ; then virsh pool-
delete vagrant ; virsh pool-undefine vagrant ; fi ; \
    done
```

5 Выводы

Во время выполнения лабораторной работы я получила навыки установки Rocky Linux на виртуальную машину с помощью инструмента Vagrant.

6 Ответы на контрольные вопросы

1. Для чего предназначен Vagrant?

   Vagrant — инструмент для создания и управления виртуальными машинами, автоматизации их развертывания и настройки.

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

   Box-файл — образ виртуальной машины, используемый как шаблон.

   Vagrantfile — конфигурационный файл, описывающий параметры ВМ (память, сеть, провайдер и т.д.).

3. Приведите описание и примеры вызова основных команд Vagrant

`vagrant up` — запуск ВМ;
`vagrant halt` — остановка ВМ;
`vagrant ssh` — подключение по SSH;
`vagrant destroy` — удаление ВМ;
`vagrant provision` — применение провижининга.

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

   1. Файл `vagrant-rocky.pkr.hcl`

- **Строки 1-11**: Объявление необходимых плагинов Packer для работы с Vagrant и VirtualBox

- **Строки 13-16**: Переменная с описанием артефакта
- **Строки 23-26**: Контрольная сумма ISO-образа для проверки целостности
- **Строки 39-46**: Команды загрузки с указанием Kickstart файла и параметров сети
- **Строки 60-61**: Учетные данные для SSH-подключения
- **Строки 66-69**: Настройки VirtualBox (память 2 ГБ, 2 CPU)
- **Строки 79-85**: Настройка провижининга - установка дополнительных пакетов
- **Строки 90-93**: Установка VirtualBox Guest Additions

2. Файл `ks.cfg`

- **Строка 2**: Настройка загрузчика с параметрами ядра
- **Строки 4-5**: Очистка MBR и всех разделов
- **Строки 11-12**: Настройка раскладки клавиатуры и системного языка
- **Строка 15**: Настройка сети через DHCP
- **Строки 21-22**: Установка пароля root и создание пользователя vagrant
- **Строка 27**: Включение необходимых служб
- **Строка 31**: Создание корневого раздела размером 10 ГБ
- **Строки 35-40**: Постинсталляционные действия - создание swap-файла
- **Строка 43**: Предоставление прав sudo пользователю vagrant

3. Файл `Vagrantfile`

- **Строка 1**: Объявление конфигурации для Vagrant версии 2
- **Строки 6-10**: Настройка общего провижининга для создания пользователя
- **Строки 14-17**: Определение виртуальной машины «server» с указанием box и имени хоста
- **Строки 24-27**: Настройка приватной сети с фиксированным IP-адресом
- **Строки 35-39**: Настройки провайдера VirtualBox (память, CPU, имя, GUI)
- **Строки 57-60**: Настройка сети клиента с получением IP через DHCP
- **Строки 64-69**: Провижининг для настройки маршрутизации на клиенте

4. Файл `Makefile`

- **Строки 1-2**: Объявление phony-целей и цели по умолчанию
- **Строки 4-7**: Цель для инициализации Packer и установки плагинов
- **Строки 9-12**: Цель для сборки box-файла Rocky Linux
- **Строки 14-21**: Цель help для отображения доступных целей с описанием
- **Строки 29-33**: Цель для запуска виртуальной машины сервера
- **Строки 47-51**: Цель для запуска сервера с применением провижининга

Список литературы

1. GNU Bash Manual. — 2019. — URL: https : / / www . gnu . org / software / bash / manual/
2. GNU Make Manual. — 2016. — URL: http : / / www . gnu . org / software / make / manual/
3. Powers S. Vagrant Simplified [Просто о Vagrant] / Пер.: А. Панин // Библиотека сайта ruslinux.net. — 2015. — URL: http://rus-linux.net/MyLDP/vm/vagrant-simplified.html
4. Vagrant Documentation. — URL: https : / / www . vagrantup . com / docs
5. Купер М. Искусство программирования на языке сценариев командной оболочки. — 2004. — URL: https://www.opennet.ru/docs/RUS/bash_scripting_guide/