Лабораторная работа №1

Подготовка лабораторного стенда

Гурылев Артем Андреевич

Содержание

Список иллюстраций

1 Цель работы

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

2 Выполнение лабораторной работы

В данной лабораторной работе устанавливается версия Rocky Linux 10, поэтому в ходе её выполнения был использован пример кода.

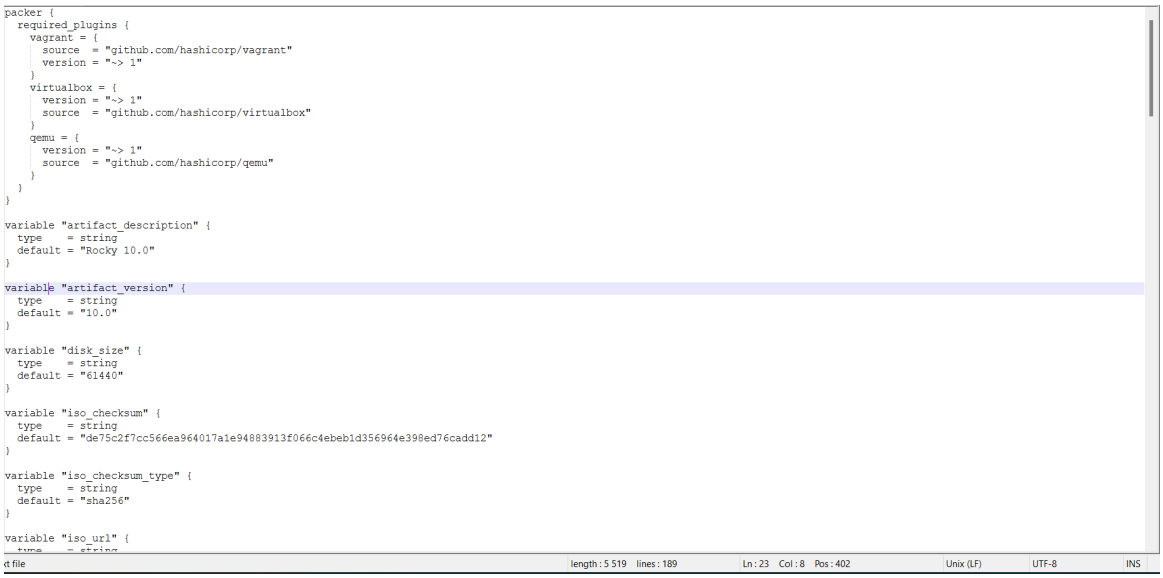
Рисунок 2.1: Конфигурация папок Vagrant и Packer

```
# -*- 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 = 2048
    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", "32"]
    virtualbox.customize ["modifyvm", :id, "--natdnshostresolver1", "on"]
    virtualbox.customize ["modifyvm", :id, "--clipboard", "bidirectional"]
    virtualbox.customize ["modifyvm", :id, "--draganddrop", "bidirectional"]
    virtualbox.customize ["modifyvm", :id, "--graphicscontroller", "vmsvga"]
    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 dummy",
                      type: "shell",
                      preserve_order: true,
                      path: "provision/default/01-dummy.sh"
```

Рисунок 2.2: Конфигурационный файл Vagrantfile

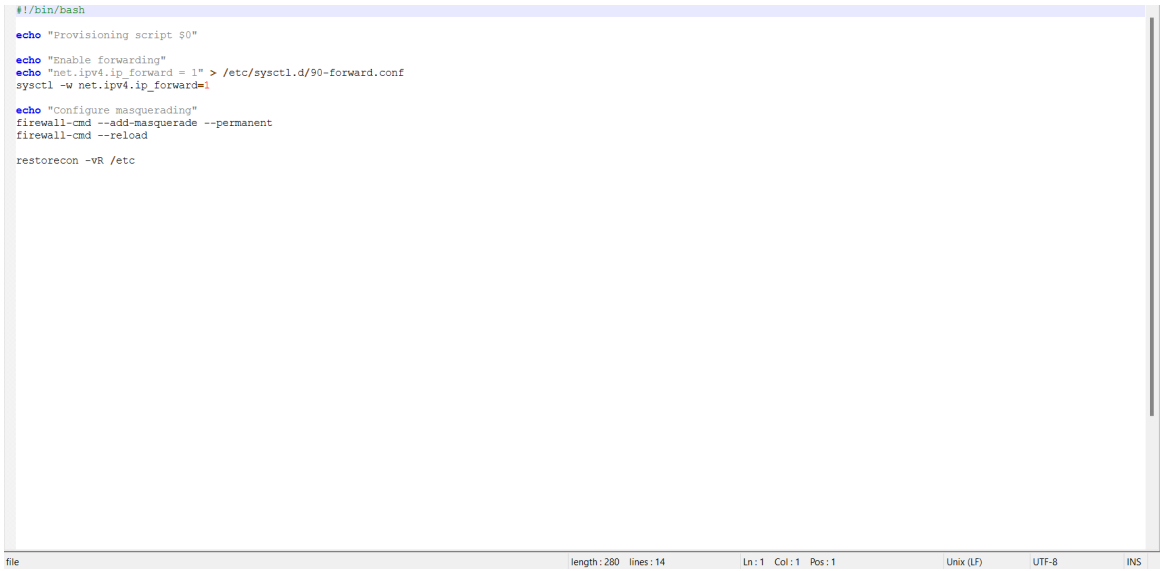
```
# System bootloader configuration
bootloader --append="no_timer_check console=tty0 console=ttyS0,115200n8 net.ifnames=0 biosdevname=0 elevator=noop" --location=mbr --timeout=1
# Clear the Master Boot Record
zerombr
# Partition clearing information
clearpart --all
# Reboot after installation
reboot
# 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
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
```

Рисунок 2.3: Конфигурационный файл ks.cfg



```
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
```


Рисунок 2.4: Конфигурационный факл vagrant-rocky.pkr.hcl



```
#!/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
```


Рисунок 2.5: Скрипт виртуальной машины server 02-forward.sh

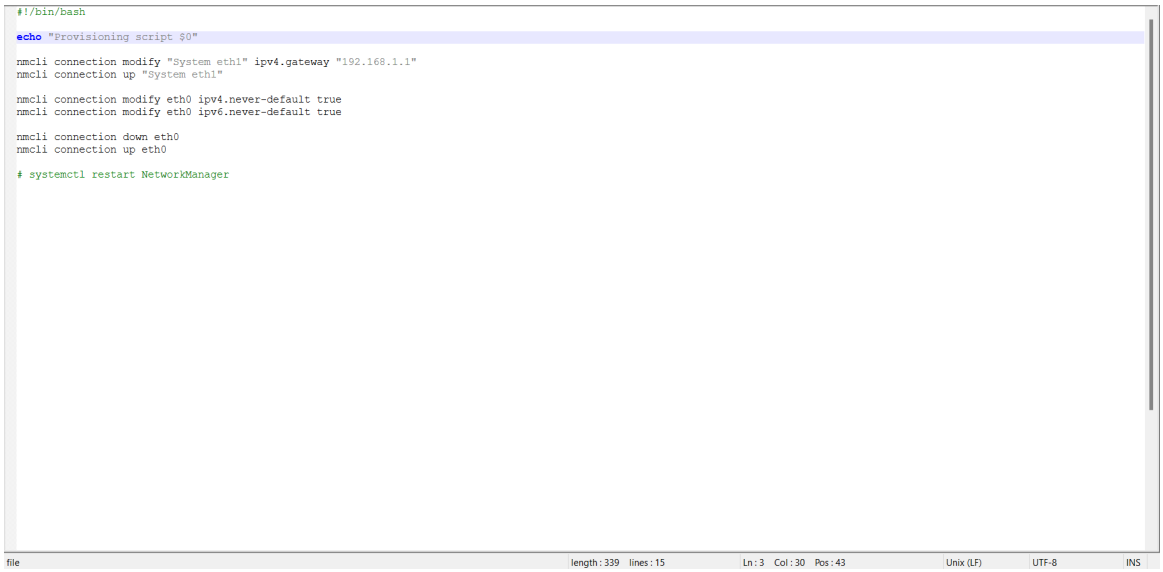


Рисунок 2.6: Скрипт виртуальной машины client 01-routing.sh

Теперь приступим к основной части работы. Для начала нам нужно сформировать box-файл виртуальной машины - для этого используем программу packer. Он работает с конфигурационным файлом vagrant-rocky.pkr.hcl, в котором содержатся настройки и команды, которые выполняться во время создания базовой виртуальной машины. (рис. 2.7, рис. 2.8)

```
C:\Work\aagurihlev\packer>packer.exe init vagrant-rocky.pkr.hcl

C:\Work\aagurihlev\packer>packer.exe build vagrant-rocky.pkr.hcl
virtualbox-iso.rockylinux: output will be in this color.
qemu.rockylinux: output will be in this color.

Build 'qemu.rockylinux' errored after 11 milliseconds 111 microseconds: Failed creating Qemu driver: exec: "qemu-system-x86_64": executable file not found in
%PATH%
==> virtualbox-iso.rockylinux: Retrieving Guest additions
==> virtualbox-iso.rockylinux: Trying C:\Program Files\Oracle\VirtualBox/VBoxGuestAdditions.iso
==> virtualbox-iso.rockylinux: Trying file://C:/Program%20Files/Oracle/VirtualBox/VBoxGuestAdditions.iso
==> virtualbox-iso.rockylinux: file://C:/Program%20Files/Oracle/VirtualBox/VBoxGuestAdditions.iso => C:/Program Files/Oracle/VirtualBox/VBoxGuestAdditions.iso
==> virtualbox-iso.rockylinux: Retrieving ISO
==> virtualbox-iso.rockylinux: Trying Rocky-10.0-x86_64-minimal.iso
==> virtualbox-iso.rockylinux: Trying Rocky-10.0-x86_64-minimal.iso?checksum=sha256%3Ade75c2f7cc566ea964017a1e94883913f066c4ebeb1d356964e398ed76cadd12
==> virtualbox-iso.rockylinux: Rocky-10.0-x86_64-minimal.iso?checksum=sha256%3Ade75c2f7cc566ea964017a1e94883913f066c4ebeb1d356964e398ed76cadd12 => C:/Work/aag
urihlev/packer/Rocky-10.0-x86_64-minimal.iso
==> virtualbox-iso.rockylinux: Starting HTTP server on port 8468
==> virtualbox-iso.rockylinux: Creating virtual machine...
==> virtualbox-iso.rockylinux: Creating hard drive output-rockylinux10-virtualbox\rockylinux10-virtualbox.vdi with size 61440 MiB...
==> virtualbox-iso.rockylinux: Mounting ISOs...
==> virtualbox-iso.rockylinux: Mounting boot ISO...
==> virtualbox-iso.rockylinux: Creating forwarded port mapping for communicator (SSH, WinRM, etc) (host port 2278)
==> virtualbox-iso.rockylinux: Executing custom VBoxManage commands...
==> virtualbox-iso.rockylinux: Executing: modifyvm rockylinux10-virtualbox --memory 2048
==> virtualbox-iso.rockylinux: Executing: modifyvm rockylinux10-virtualbox --cpus 2
==> virtualbox-iso.rockylinux: Executing: modifyvm rockylinux10-virtualbox --nat-localhostreachable1 on
==> virtualbox-iso.rockylinux: Executing: modifyvm rockylinux10-virtualbox --firmware EFI
==> virtualbox-iso.rockylinux: Executing: modifyvm rockylinux10-virtualbox --vrde on
==> virtualbox-iso.rockylinux: Executing: modifyvm rockylinux10-virtualbox --vrdeport 3390
==> virtualbox-iso.rockylinux: Starting the virtual machine...
==> virtualbox-iso.rockylinux: The VM will be run headless, without a GUI. If you want to
==> virtualbox-iso.rockylinux: view the screen of the VM, connect via VRDP without a password to
==> virtualbox-iso.rockylinux: rdp://127.0.0.1:5952
==> virtualbox-iso.rockylinux: Waiting 10s for boot...
==> virtualbox-iso.rockylinux: Typing the boot command...
==> virtualbox-iso.rockylinux: Using SSH communicator to connect: 127.0.0.1
==> virtualbox-iso.rockylinux: Waiting for SSH to become available...
```

Рисунок 2.7: Результат работы packer

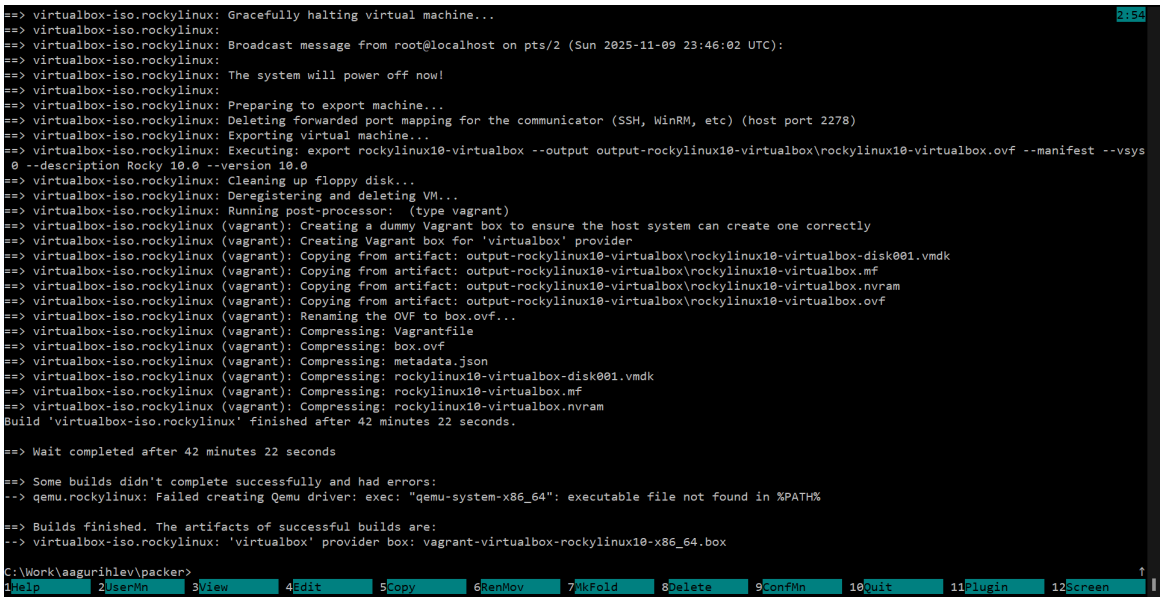


Рисунок 2.8: Результат работы packer

Далее необходимо зарегистрировать созданный box-файл в программе Vagrant. (рис. 2.9)

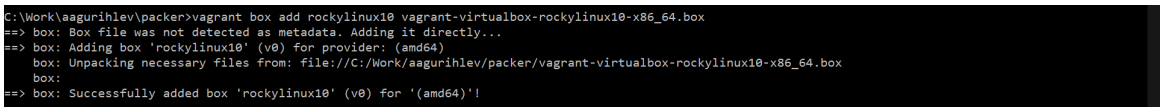


Рисунок 2.9: Регистрация box-файла в Vagrant

Теперь запустим виртуальные машины server и клиент - они основаны на box-файле, который мы создали с помощью packer. Начнем с сервера, выполнив команду vagrant up server. (рис. 2.10, рис. 2.11)

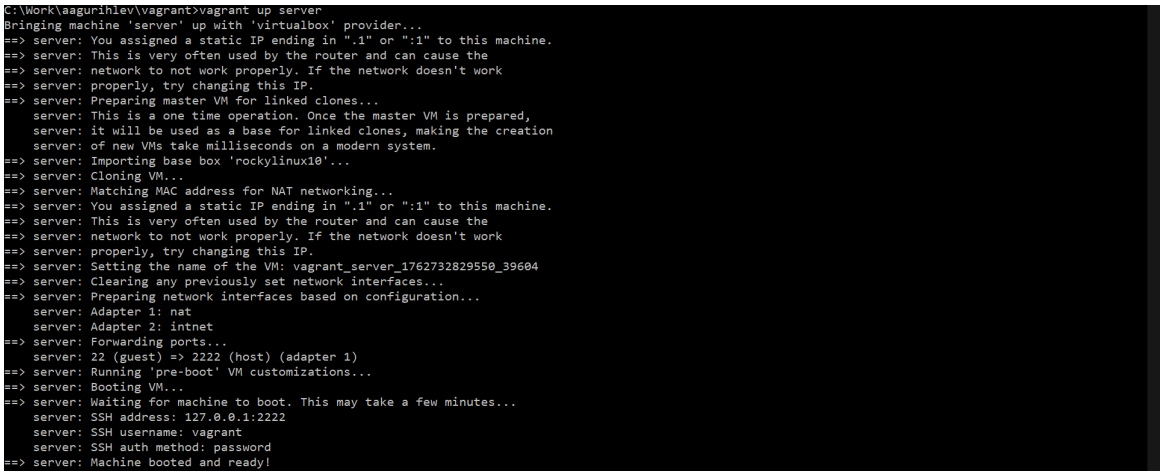


Рисунок 2.10: Запуск виртуальной машины server

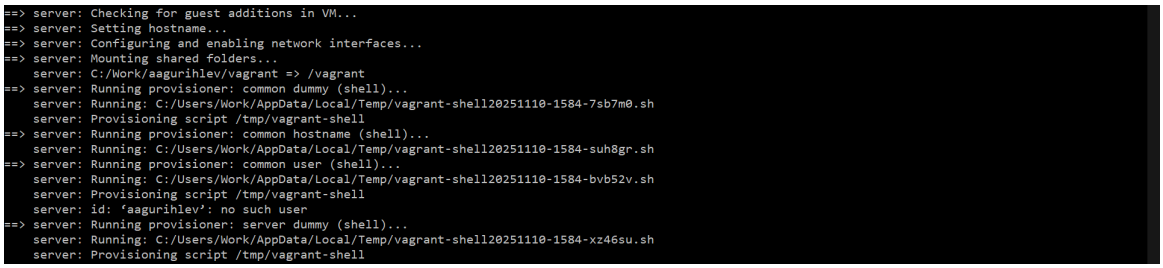


Рисунок 2.11: Запуск виртуальной машины server

После запуска и проверки server запустим виртуальную машину client командой vagrant up client. (рис. 2.12)

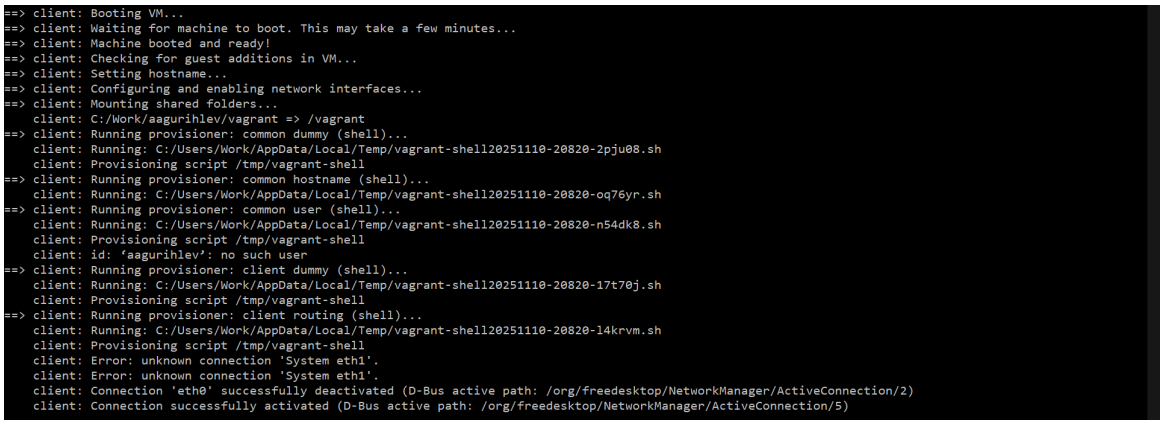


Рисунок 2.12: Запуск виртуальной машины client

Запустив обе виртуальные машины, зайдем в пользователя vagrant в обеих системах, а затем проверим ssh-соединение с ними. Вписав в консоли команду vagrant ssh server, подключимся к серверу и залогинимся в собственного пользователя, затем выйдем. (рис. 2.13, рис. 2.14)

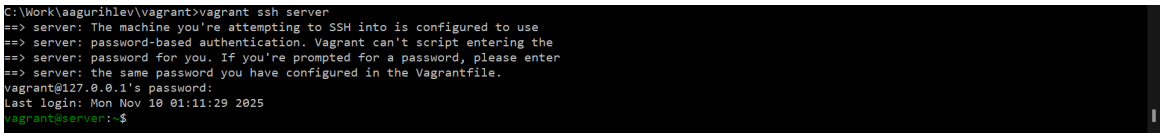


Рисунок 2.13: Логин под пользователем vagrant виртуальной машины server

Рисунок 2.14: Логин под собственным пользователем виртуальной машины server

Тоже самое проделаем и с клиентом - посредством команды vagrant ssh client. (рис. 2.15)

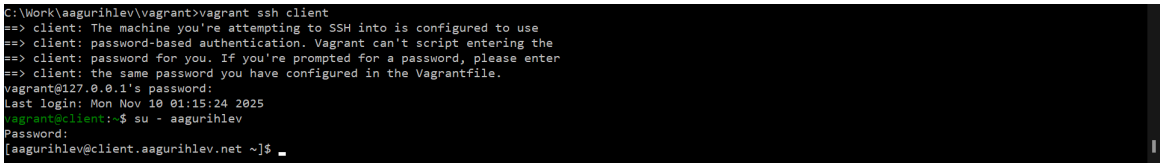


Рисунок 2.15: Логин под пользователем vagrant и под собственным пользователем виртуальной машины client

Выключим обе виртуальные машины командами vagrant halt server и vagrant halt client.

3 Выводы

В результате выполнения лабораторной работы была настроена система виртуальных машин Linux server и client, в которых будут выполняться следующие лабораторные работы, а также получены навыки работы с программой vagrant.