Методическое пособие по выполнению домашнего задания по курсу «Администратор Linux. Professional»

Vagrant-стенд с сетевой лабораторией

Цель домашнего задания

Научится менять базовые сетевые настройки в Linux-based системах.

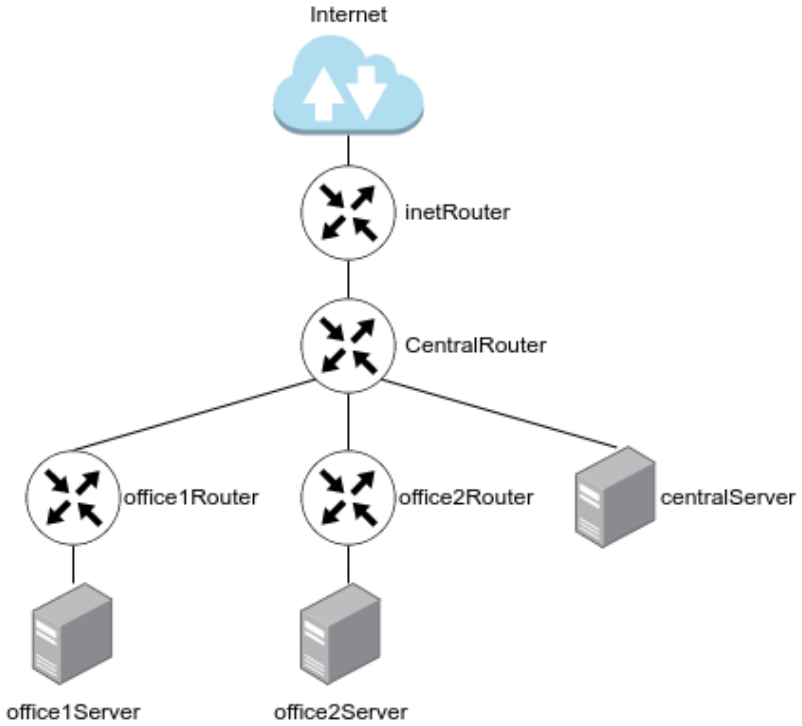
Описание домашнего задания

1. Скачать и развернуть Vagrant-стенд https://github.com/erlong15/otus-linux/tree/network
2. Построить следующую сетевую архитектуру:

Сеть office1
- 192.168.2.0/26 - dev
- 192.168.2.64/26 - test servers
- 192.168.2.128/26 - managers
- 192.168.2.192/26 - office hardware

Сеть office2
- 192.168.1.0/25 - dev
- 192.168.1.128/26 - test servers
- 192.168.1.192/26 - office hardware

Сеть central
- 192.168.0.0/28 - directors
- 192.168.0.32/28 - office hardware
- 192.168.0.64/26 - wifi



Итого должны получиться следующие сервера:
- inetRouter
- centralRouter
- office1Router

- office2Router
- centralServer
- office1Server
- office2Server

Задание состоит из 2-х частей: теоретической и практической.
В теоретической части требуется:
- Найти свободные подсети
- Посчитать количество узлов в каждой подсети, включая свободные
- Указать Broadcast-адрес для каждой подсети
- Проверить, нет ли ошибок при разбиении

В практической части требуется:
- Соединить офисы в сеть согласно логической схеме и настроить роутинг
- Интернет-трафик со всех серверов должен ходить через inetRouter
- Все сервера должны видеть друг друга (должен проходить ping)
- У всех новых серверов отключить дефолт на NAT (eth0), который vagrant поднимает для связи
- Добавить дополнительные сетевые интерфейсы, если потребуется

Рекомендуется использовать **Vagrant + Ansible** для настройки данной схемы.

Введение

Сеть — очень важная составляющая в работе серверов. По сети сервера взаимодействуют между собой.

В данном домашнем задании мы рассмотрим технологии маршрутизации и NAT.

Маршрутизация — выбор оптимального пути передачи пакетов. Для маршрутизации используется таблица маршрутизации.

Основная задача маршрутизации — доставить пакет по указанному IP-адресу.

Если одно устройство имеет сразу несколько подсетей, например в сервере есть 2 порта с адресами:
1. 192.168.1.10/24
2. 10.10.12.72/24

то, такие сети называются непосредственно подключенными (Directly connected networks). Маршрутизация между Directly Connected сетями происходит автоматически. Дополнительная настройка не требуется.

Если необходимая сеть удалена, маршрутизатор будет искать через какой порт она будет доступна, если такой порт не найден, то трафик уйдет на шлюз по умолчанию.

Маршрутизация бывает статическая и динамическая.

При использовании статической маршрутизации администратор сам создает правила для маршрутов. Плюсом данного метода является безопасность, так как статические маршруты не обновляются по сети, а минусом — сложности при работе с сетями больших объёмов…

Динамическая маршрутизация подразумевает автоматическое построение маршрутов с помощью различных протоколов (RIP,OSPF,BGP, и.т.д.). Маршрутизаторы сами обмениваются друг с другом информацией о сетях и автоматически прописывают маршруты.

NAT — это процесс, используемый для преобразования сетевых адресов.

Основные цели NAT:

- Экономия публичных IPv4-адресов
- Повышение степени конфиденциальности и безопасности сети.

NAT обычно работает на границе, где локальная сеть соединяется с сетью Интернет. Когда устройству сети потребуется подключение к устройству вне его сети (например в Интернете), пакет пересылается маршрутизатору с NAT, а маршрутизатор преобразовывает его внутренний адрес в публичный.

Функциональные и нефункциональные требования

- ПК на Unix с 12ГБ ОЗУ или виртуальная машина с включенной Nested Virtualization.

Предварительно установленное и настроенное следующее ПО:

- Hashicorp Vagrant (https://www.vagrantup.com/downloads)
- Oracle VirtualBox (https://www.virtualbox.org/wiki/Linux_Downloads).
- Ansible (версия 2.7 и выше) - https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html
- Любой редактор кода, например Visual Studio Code, Atom и т.д.

Инструкция по выполнению домашнего задания

Все дальнейшие действия были проверены при использовании Vagrant 2.3.7, VirtualBox Version 7.0.8 r156879. Серьёзные отступления от этой конфигурации могут потребовать адаптации с вашей стороны.

1. **Теоретическая часть**

В теоретической части нам необходимо продумать топологию сети, а также:

- Найти свободные подсети
- Посчитать количество узлов в каждой подсети, включая свободные
- Указать Broadcast-адрес для каждой подсети
- Проверить, нет ли ошибок при разбиении

Первым шагом мы рассмотрим все сети, указанные в задании. Посчитаем для них количество узлов, найдём Broadcast-адрес, проверим, нет ли ошибок при разбиении.

Расчеты можно выполнить вручную, или воспользоваться калькулятором сетей. Для примера, посчитаем одну подсеть вручную:

У нас есть сеть directors 192.168.0.0/28
192.168.0.0 — это сама сеть, 28 — это маска. Маска показывает нам, границы сети 192.168.0.0. Маска может быть записана в 2-х видах:

1) /28
2) 255.255.255.240

Пример перевода маски /28 в формат 255.255.255.240:

Маска Ipv4-адреса — это 4 октета, т.е. 4 блока по 8 цифр (1 или 0).

/28 — это 28 единиц с начала маски: **11111111.11111111.11111111.11110000**
Всегда при разбиении сетей, после знака / указывается количество единиц с начала маски.

11111111.11111111.11111111.11110000 — это двоичный код маски, если мы переведем данное значение в десятичную систему счисления, то получим **255.255.255.240**

Далее посчитаем количество устройств в сети:

Количество устройств в сети рассчитывается по формуле = $2^{(32-\text{маска})} - 2$
Таким образом, количество устройств для подсети /28 будет = $2^{(32-28)} - 2 = 16 - 2 = 14$

Цифра 2 вычитается, так как:

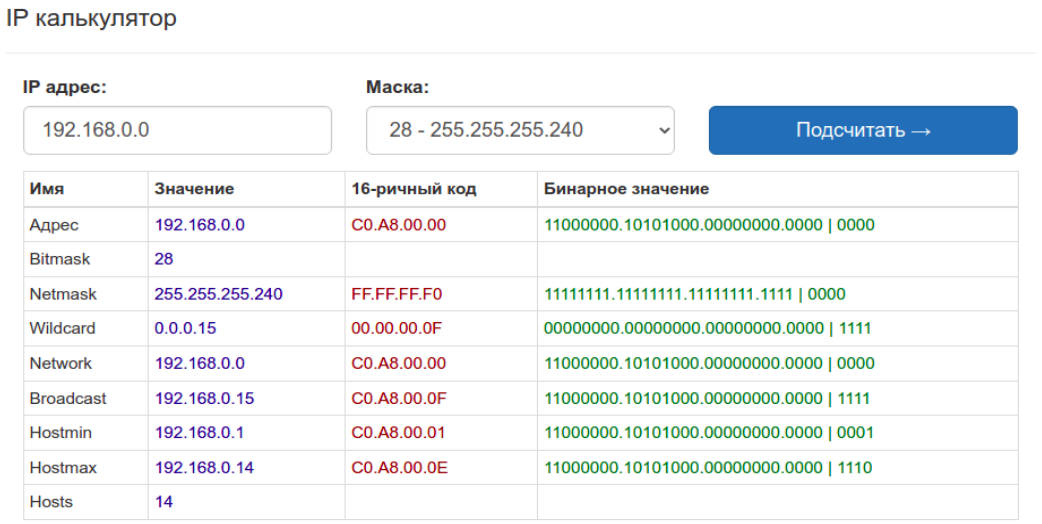
- Первый адрес (192.168.0.0) — это наименование подсети, его нельзя задать устройству
- Последний адрес (192.168.0.15) — это всегда broadcast-адрес. Broadcast-адрес нужен для рассылки всем устройствам сети.

Таким образом мы можем сформировать таблицу топологии нашей сети

Сеть	**Маска**	**Кол-во адресов**	**Первый адрес в сети**	**Последний адрес в сети**	**Broadcast — адрес**
192.168.0.0/28	255.255.255.240	14	192.168.0.1	192.168.0.14	192.168.0.15

По такому примеру нужно рассчитать остальные сети. Для проверки себя можно использовать калькулятор масок, например, этот - https://ip-calculator.ru/

Для того, чтобы получить всю информацию по сети, потребуется указать ip-адрес и маску, например:



После расчета всех сетей, мы должны получить следующую таблицу топологии

Name	**Network**	**Netmask**	**N**	**Hostmin**	**Hostmax**	**Broadcast**
Central Network						
Directors	192.168.0.0/28	255.255.255.240	14	192.168.0.1	192.168.0.14	192.168.0.15
Office hardware	192.168.0.32/28	255.255.255.240	14	192.168.0.33	192.168.0.46	192.168.0.47
Wifi(mgt network)	192.168.0.64/26	255.255.255.192	62	192.168.0.65	192.168.0.126	192.168.0.127
Office 1 network						
Dev	192.168.2.0/26	255.255.255.192	62	192.168.2.1	192.168.2.62	192.168.2.63

Test	192.168.2.64/26	255.255.255.192	62	192.168.2.65	192.168.2.126	192.168.2.127
Managers	192.168.2.128/26	255.255.255.192	62	192.168.2.129	192.168.2.190	192.168.2.191
Office hardware	192.168.2.192/26	255.255.255.192	62	192.168.2.193	192.168.2.254	192.168.2.255
Office 2 network						
Dev	192.168.1.0/25	255.255.255.128	126	192.168.1.1	192.168.1.126	192.168.1.127
Test	192.168.1.128/26	255.255.255.192	62	192.168.1.129	192.168.1.190	192.168.1.191
Office	192.168.1.192/26	255.255.255.192	62	192.168.1.193	192.168.1.254	192.168.1.255
InetRouter — CentralRouter network						
Inet — central	192.168.255.0/30	255.255.255.252	2	192.168.255.1	192.168.255.2	192.168.255.3

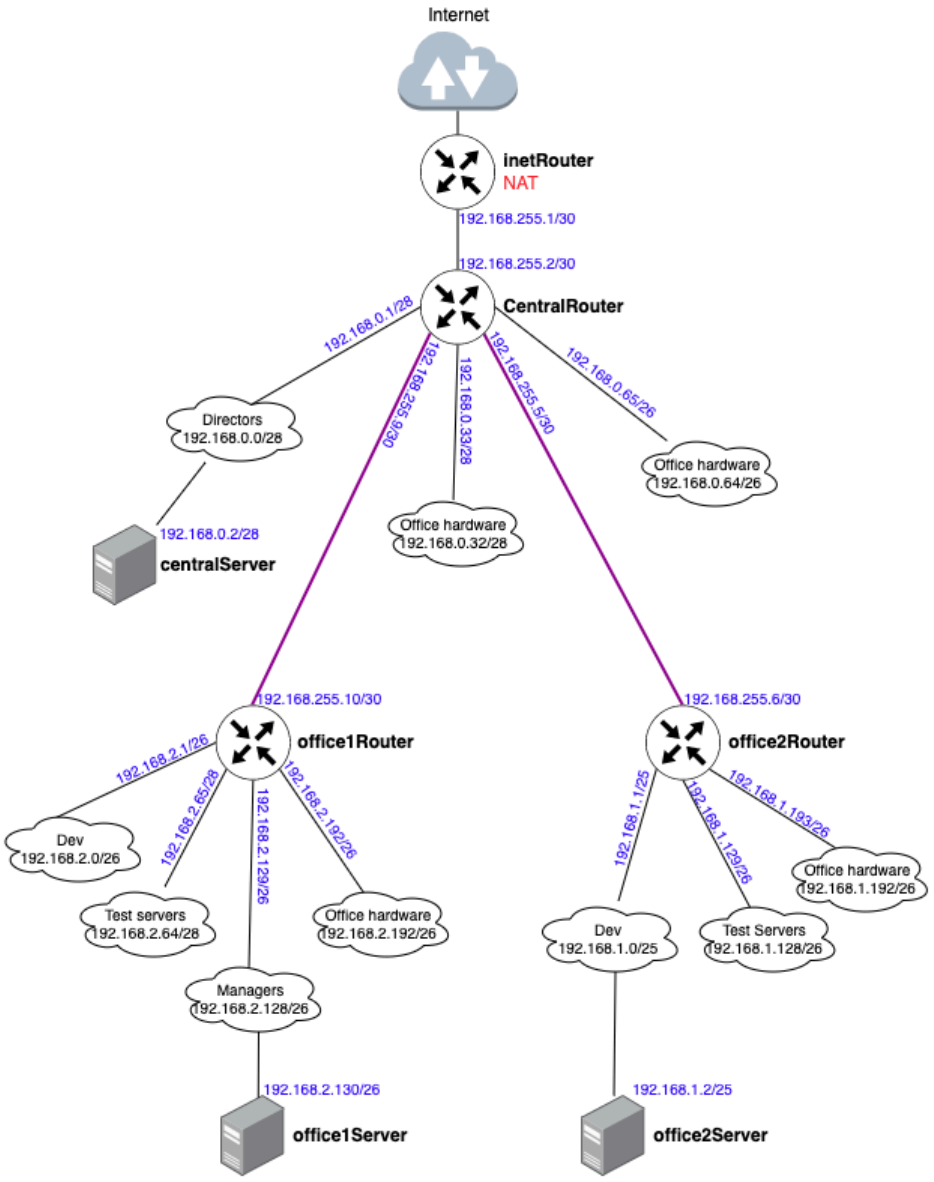
После создания таблицы топологии, мы видим, что ошибок в задании нет, также мы сразу видим следующие свободные сети:

- 192.168.0.16/28
- 192.168.0.48/28
- 192.168.0.128/25
- 192.168.255.64/26
- 192.168.255.32/27
- 192.168.255.16/28
- 192.168.255.8/29
- 192.168.255.4/30

На этом теоретическая часть заканчивается. Можем приступать к выполнению практической части.

2. Практическая часть

Изучив таблицу топологии сети и Vagrant-стенд из задания, мы можем построить полную схему сети:



Давайте рассмотрим схему:

- Знак облака означает сеть, которую необходимо будет настроить на сервере
- Значки роутеров и серверов означают хосты, которые нам нужно будет создать.

На схеме, мы сразу можем увидеть, что нам потребуется создать дополнительно 2 сети (на схеме обозначены полужирными фиолетовыми линиями):

- Для соединения office1Router с centralRouter
- Для соединения office2Router с centralRouter

На основании этой схемы мы получаем готовый список серверов.

Server	**IP and Bitmask**
inetRouter	Default-NAT address VirtualBox
	192.168.255.1/30

centralRouter	192.168.255.2/30
	192.168.0.1/28
	192.168.0.33/28
	192.168.0.65/26
	192.168.255.9/30
	192.168.255.5/30
centralServer	192.168.0.2/28
office1Router	192.168.255.10/30
	192.168.2.1/26
	192.168.2.65/26
	192.168.2.129/26
	192.168.2.193/26
office1Server	192.168.2.130/26
office2Router	192.168.255.6/30
	192.168.1.1/26
	192.168.1.129/26
	192.168.1.193/26
office2Server	192.168.1.2/26

Все виртуальные машины у нас будут работать на Ubuntu 22.04.

Рекомендация
При выполнении дальнейших действий, постоянно пользуйтесь схемой, она поможет быстрее понять некоторые моменты руководства.

Скачаем Vagrantfile из репозитория
https://github.com/erlong15/otus-linux/tree/network

В полученном vagrant file в :box_name меняем образы CentOS на образы ubuntu 22.04 и добавляем оставшиеся хосты. Также нужно удалить все команды, выполняемые на хостах:

```
MACHINES = {
  :inetRouter => {
        :box_name => "generic/ubuntu2204",
        :vm_name => "inetRouter",
        #:public => {:ip => "10.10.10.1", :adapter => 1},
        :net => [
                    #ip, adpter, netmask, virtualbox__intnet
                    ["192.168.255.1", 2, "255.255.255.252",  "router-net"],
                    ["192.168.50.10", 8, "255.255.255.0"],
                ]
  },

  :centralRouter => {
        :box_name => "generic/ubuntu2204",
        :vm_name => "centralRouter",
        :net => [
                    ["192.168.255.2",  2, "255.255.255.252",  "router-net"],
                    ["192.168.0.1",    3, "255.255.255.240",  "dir-net"],
                    ["192.168.0.33",   4, "255.255.255.240",  "hw-net"],
                    ["192.168.0.65",   5, "255.255.255.192",  "mgt-net"],
                    ["192.168.255.9",  6, "255.255.255.252",  "office1-central"],
                    ["192.168.255.5",  7, "255.255.255.252",  "office2-central"],
```

```
                    ["192.168.50.11",  8, "255.255.255.0"],
                ]
  },

  :centralServer => {
        :box_name => "generic/ubuntu2204",
        :vm_name => "centralServer",
        :net => [
                    ["192.168.0.2",    2, "255.255.255.240",  "dir-net"],
                    ["192.168.50.12",  8, "255.255.255.0"],
                ]
  },

  :office1Router => {
        :box_name => "generic/ubuntu2204",
        :vm_name => "office1Router",
        :net => [
                    ["192.168.255.10",  2,  "255.255.255.252",  "office1-central"],
                    ["192.168.2.1",     3,  "255.255.255.192",  "dev1-net"],
                    ["192.168.2.65",    4,  "255.255.255.192",  "test1-net"],
                    ["192.168.2.129",   5,  "255.255.255.192",  "managers-net"],
                    ["192.168.2.193",   6,  "255.255.255.192",  "office1-net"],
                    ["192.168.50.20",   8,  "255.255.255.0"],
                ]
  },

  :office1Server => {
        :box_name => "generic/ubuntu2204",
        :vm_name => "office1Server",
        :net => [
                    ["192.168.2.130",  2,  "255.255.255.192",  "managers-net"],
                    ["192.168.50.21",  8,  "255.255.255.0"],
                ]
  },

  :office2Router => {
       :box_name => "generic/ubuntu2204",
       :vm_name => "office2Router",
       :net => [
                    ["192.168.255.6",  2,  "255.255.255.252",  "office2-central"],
                    ["192.168.1.1",    3,  "255.255.255.128",  "dev2-net"],
                    ["192.168.1.129",  4,  "255.255.255.192",  "test2-net"],
                    ["192.168.1.193",  5,  "255.255.255.192",  "office2-net"],
                    ["192.168.50.30",  8,  "255.255.255.0"],
                ]
  },

  :office2Server => {
       :box_name => "generic/ubuntu2204",
       :vm_name => "office2Server",
       :net => [
                    ["192.168.1.2",    2,  "255.255.255.128",  "dev2-net"],
                    ["192.168.50.31",  8,  "255.255.255.0"],
                ]
  }
}

Vagrant.configure("2") do |config|
  MACHINES.each do |boxname, boxconfig|
    config.vm.define boxname do |box|
      box.vm.box = boxconfig[:box_name]
```

```
      box.vm.host_name = boxconfig[:vm_name]

      box.vm.provider "virtualbox" do |v|
        v.memory = 768
        v.cpus = 1
       end

      boxconfig[:net].each do |ipconf|
        box.vm.network("private_network", ip: ipconf[0], adapter: ipconf[1], netmask:
ipconf[2], virtualbox__intnet: ipconf[3])
      end

      if boxconfig.key?(:public)
        box.vm.network "public_network", boxconfig[:public]
      end

      box.vm.provision "shell", inline: <<-SHELL
        mkdir -p ~root/.ssh
        cp ~vagrant/.ssh/auth* ~root/.ssh
      SHELL
    end
  end
end
```

В данный Vagrantfile мы добавили 4 новых сервера, также к старым серверам добавили 2 интерфейса для соединения сетей офисов.

Дополнительно в коде вы видите сетевые устройства из подсети 192.168.50.0/24 — они не обязательны и потребуются, если вы планируете настраивать хосты с помощью Ansible.

После того, как все 7 серверов у нас развернуты, нам нужно настроить маршрутизацию и NAT таким образом, чтобы доступ в Интернет со всех хостов был через inetRouter и каждый сервер должен быть доступен с любого из 7 хостов.

Настройка NAT

Для того, чтобы на всех серверах работал интернет, на сервере inetRouter должен быть настроен NAT. В нашем Vagrantfile он настраивался с помощью команды:

iptables -t nat -A POSTROUTING ! -d 192.168.0.0/16 -o eth0 -j MASQUERADE

При настройке NAT таким образом, **правило удаляется после перезагрузки сервера.** Для того, чтобы правила применялись после перезагрузки, в Ubuntu 22.04 нужно выполнить следующие действия:

1)Подключиться по SSH к хосту: *vagrant ssh inetRouter*

2)Проверить, что отключен другой файервол: *systemctl status ufw*

```
root@inetRouter:~# systemctl status ufw
● ufw.service - Uncomplicated firewall
```

```
     Loaded: loaded (/lib/systemd/system/ufw.service; enabled; vendor preset:
enabled)
     Active: active (exited) since Sat 2023-10-14 15:48:04 UTC; 7min ago
       Docs: man:ufw(8)
   Main PID: 517 (code=exited, status=0/SUCCESS)
        CPU: 3ms

Oct 14 15:48:04 inetRouter systemd[1]: Starting Uncomplicated firewall...
Oct 14 15:48:04 inetRouter systemd[1]: Finished Uncomplicated firewall.
root@inetRouter:~#
```

Если служба будет запущена, то нужно её отключить и удалить из автозагрузки:
systemctl stop ufw
systemctl disable ufw

3) Создаём файл **/etc/iptables_rules.ipv4***:*

vi /etc/iptables_rules.ipv4

```
# Generated by iptables-save v1.8.7 on Sat Oct 14 16:14:36 2023
*filter
:INPUT ACCEPT [90:8713]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [54:7429]
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT
COMMIT
# Completed on Sat Oct 14 16:14:36 2023
# Generated by iptables-save v1.8.7 on Sat Oct 14 16:14:36 2023
*nat
:PREROUTING ACCEPT [1:44]
:INPUT ACCEPT [1:44]
:OUTPUT ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
-A POSTROUTING ! -d 192.168.0.0/16 -o eth0 -j MASQUERADE
COMMIT
# Completed on Sat Oct 14 16:14:36 2023
```

4) Создаём файл, в который добавим скрипт автоматического восстановления правил при перезапуске системы:
vi /etc/network/if-pre-up.d/iptables

```
#!/bin/sh
/sbin/iptables-restore < /etc/iptables_rules.ipv4
```

5) Добавляем права на выполнение файла /etc/network/if-pre-up.d/iptables
sudo chmod +x /etc/network/if-pre-up.d/iptables

6) Перезагружаем сервер: reboot
7) После перезагрузки сервера проверяем правила iptables: iptables-save

Настройка NAT через Ansible

Выполним идентичные действия с помощью Ansible, для этого в playbook добавим следующие команды:

```
  - name: Set up NAT on inetRouter
    template:
      src: "{{ item.src }}"
      dest: "{{ item.dest }}"
      owner: root
      group: root
      mode: "{{ item.mode }}"
    with_items:
      - { src: "iptables_rules.ipv4", dest: "/etc/iptables_rules.ipv4", mode:
"0644" }
      - { src: "iptables_restore", dest: "/etc/network/if-pre-up.d/iptables",
mode: "0755" }
    when: (ansible_hostname == "inetRouter")
```

Модуль template копирует 2 файла, которые были указаны выше. Для файла */etc/network/if-pre-up.d/iptables уже установлен атрибут выполнения файла.*

Маршрутизация транзитных пакетов (IP forward)

Важным этапом настройки сетевой лаборатории, является маршрутизация транзитных пакетов. Если объяснить простыми словами — это возможность сервера Linux пропускать трафик через себя к другому серверу. По умолчанию эта функция отключена в Linux. Включить её можно командой:
echo "net.ipv4.conf.all.forwarding = 1" >> /etc/sysctl.conf
sysctl -p

Посмотреть статус форвардинга можно командой: *sysctl net.ipv4.ip_forward*
Если параметр равен 1, то маршрутизация транзитных пакетов включена, если 0 — отключена.

В нашей схеме необходимо включить данную маршрутизацию на всех роутерах.

Настройка маршрутизации транзитных пакетов с помощью Ansible

В Ansible есть специальный блок для внесений изменений в параметры ядра:

```
  - name: set up forward packages across routers
    sysctl:
      name: net.ipv4.conf.all.forwarding
      value: '1'
      state: present
    when: "'routers' in group_names"
```

В условии указано, что изменения будут применяться только для группы «routers», группа routers создана в hosts-файле:

```
[routers]
inetRouter ansible_host=192.168.50.10 ansible_user=vagrant
ansible_ssh_private_key_file=.vagrant/machines/inetRouter/virtualbox/private_key
centralRouter ansible_host=192.168.50.11 ansible_user=vagrant
ansible_ssh_private_key_file=.vagrant/machines/centralRouter/virtualbox/private_ke
y
office1Router ansible_host=192.168.50.20 ansible_user=vagrant
ansible_ssh_private_key_file=.vagrant/machines/office1Router/virtualbox/private_ke
y
```

office2Router *ansible_host=192.168.50.30 ansible_user=vagrant ansible_ssh_private_key_file=.vagrant/machines/office2Router/virtualbox/private_key*

Файл hosts — это файл инвентаризации, в нем указан список серверов, их адреса, группы и способы доступа на сервер.

Отключение маршрута по умолчанию на интерфейсе eth0

При разворачивании нашего стенда Vagrant создает в каждом сервере свой интерфейс, через который у сервера появляется доступ в интернет. Отключить данный порт нельзя, так как через него Vagrant подключается к серверам. Обычно маршрут по умолчанию прописан как раз на этот интерфейс, данный маршрут нужно отключить:

Для отключения маршрута по умолчанию в файле **/etc/netplan/00-installer-config.yaml** добавляем отключение маршрутов, полученных через DHCP:

```
# This is the network config written by 'subiquity'
network:
  ethernets:
    eth0:
      dhcp4: true
      dhcp4-overrides:
          use-routes: false
      dhcp6: false
  version: 2
```

После внесения данных изменений перезапускаем сетевую службу:
netplan try

Отключение дефолтного маршрута требуется настроить на всех хостах кроме inetRouter

Отключение маршрута по умолчанию с помощью Ansible

Для выполнения идентичных изменений с помощью Ansible, можно воспользоваться следующим блоком:

```
- name: disable default route
  template:
    src: 00-installer-config.yaml
    dest: /etc/netplan/00-installer-config.yaml
    owner: root
    group: root
    mode: 0644
  when: (ansible_hostname != "inetRouter")
```

Настройка статических маршрутов

Для настройки статических маршрутов используется команда **ip route**.

Давайте рассмотрим пример настройки статического маршрута на сервере office1Server. Исходя из схемы мы видим, что трафик с данного сервера будет идти через office1Router. Office1Server и office1Router у нас соединены через сеть managers (192.168.2.128/26). **В статическом маршруте нужно указывать адрес следующего хоста.** Таким образом мы должны указать на сервере office1Server маршрут, в котором доступ к любым IP-адресам у нас будет происходить через адрес 192.168.2.129, который расположен на сетевом интерфейсе office1Router. Команда будет выглядеть так: *ip route add 0.0.0.0/0 via 192.168.2.129*

Посмотреть список всех маршрутов: *ip route*

```
root@office1Server:~# ip r
default via 192.168.2.129 dev eth1
10.0.2.0/24 dev eth0 proto kernel scope link src 10.0.2.15 metric 100
10.0.2.2 dev eth0 proto dhcp scope link src 10.0.2.15 metric 100
10.0.2.3 dev eth0 proto dhcp scope link src 10.0.2.15 metric 100
192.168.2.128/26 dev eth1 proto kernel scope link src 192.168.2.130
192.168.50.0/24 dev eth2 proto kernel scope link src 192.168.50.21
root@office1Server:~#
```

Удалить маршрут: *ip route **del** 0.0.0.0/0 via 192.168.2.129*

Важно помнить, что маршруты, настроенные через команду ip route удаляются после перезагрузки или перезапуске сетевой службы.

Для того, чтобы маршруты сохранялись после перезагрузки нужно их указывать непосредственно в файле конфигурации сетевых интерфейсов:

В современных версиях Ubuntu, для указания маршрута нужно поправить netplan-конфиг. Конфиги netplan хранятся в виде YAML-файлов и обычно лежат в каталоге */etc/netplan*
В нашем стенде такой файл - **/etc/netplan/50-vagrant.yaml**

Для добавления маршрута, после раздела addresses нужно добавить блок:

```
routes:
 - to: <сеть назначения>/<маска>
   via: <Next hop address>
```

Пример файла /etc/netplan/50-vagrant.yaml

```
---
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s8:
      addresses:
      - 192.168.2.130/26
      routes:
      - to: 0.0.0.0/0
        via: 192.168.2.129
    enp0s19:
      addresses:
      - 192.168.50.21/24
```

В YAML-файле очень важно следить за правильными отступами, ошибка в один пробел не даст сохранить изменения.

Применение изменений:

```
root@office1Server:~# netplan try
Do you want to keep these settings?

Press ENTER before the timeout to accept the new configuration

Changes will revert in 120 seconds
Configuration accepted.
root@office1Server:~#
```

При настройке интерфейсов и маршрутов в любой из ОС можно оставлять комментарии в файле. Перед комментарием должен стоять знак «#»

Настройте самостоятельно все маршруты на серверах. *Важно помнить, что помимо маршрутов по умолчанию, вам нужно будет использовать обратные маршруты.*

Давайте разберем пример такого маршрута: допустим мы хотим отправить команду ping с сервера office1Server (192.168.2.130) до сервера centralRouter (192.168.0.1)
Наш трафик пойдёт следующим образом: **office1Server — office1Router — centralRouter — office1Router — office1Server**

Office1Router знает сеть (192.168.2.128/26), в которой располагается сервер office1Server, а сервер centralRouter, когда получит запрос от адреса 192.168.2.130 не будет понимать, куда отправить ответ. Решением этой проблемы будет добавление обратного маршрута.

Обратный маршрут указывается также как остальные маршруты. Изучив схему мы видим, что связь между сетями 192.168.2.0/24 и 192.168.0.0/24 осуществляется через сеть 192.168.255.8/30. Также мы видим что сети office1 подключены к centralRouter через порт eth5. На основании этих данных мы можем добавить маршрут в файл /etc/netplan/50-vagrant.yaml

```
    eth5:
      addresses:
      - 192.168.255.9/30
      routes:
      - to: 192.168.2.0/24
        via: 192.168.255.8
```

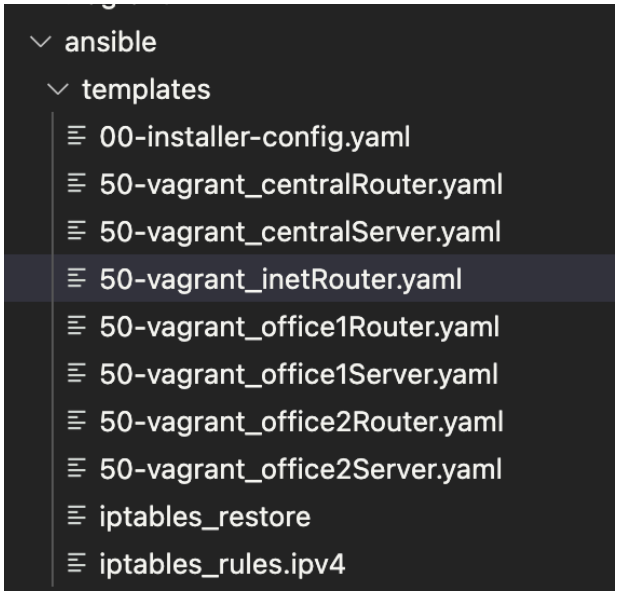
Настройка маршрутов с помощью Ansible

Для настройки маршрутов с помощью Ansible, нам необходимо подготовить файлы 50-vagrant.yaml с маршрутами для всех серверов. Далее с помощью модуля template мы можем их добавлять:

```
  - name: add default gateway for centralRouter
    template:
      src: "50-vagrant_{{ansible_hostname}}.yaml"
      dest: /etc/netplan/50-vagrant.yaml
      owner: root
      group: root
      mode: 0644
```

```
  - name: restart all hosts
    reboot:
      reboot_timeout: 600
```

Для того, чтобы не писать 7 идентичных модулей, можно сделать один модуль, в котором будут перечисляться все хосты. Для этого у template-файлов должны быть идентичные имена формата **50-vagrant_<имя сервера>.yaml**
Например:



После копирования файлов, для применения сетевых настроек, перезагрузим все хосты

Также модуль template позволяет использовать jinja2-шаблоны, с помощью которых файл может генерироваться автоматически, собирая информацию из данных о хосте (Ansible facts) и файлов переменных.

Более подробно про jinja2-шаблоны можно прочитать тут - https://docs.ansible.com/ansible/2.9/user_guide/playbooks_templating.html

На данном этапе настройка серверов закончена. После настройки серверов рекомендуется перезагрузить все хосты, чтобы проверить, что правила не удаляются после перезагрузки.

Помимо этого, рекомендуется на все хосты установить утилиту traceroute, для проверки нашего стенда.

Установка traceroute: *apt install -y traceroute*

Пример проверки выхода в Интернет через сервер inetRouter с хоста office1Server:

```
root@office1Server:~# traceroute 8.8.8.8
traceroute to 8.8.8.8 (8.8.8.8), 30 hops max, 60 byte packets
 1  _gateway (192.168.2.129)  1.736 ms  1.587 ms  1.524 ms
 2  192.168.255.9 (192.168.255.9)  2.052 ms  1.943 ms  1.862 ms
 3  192.168.255.1 (192.168.255.1)  4.252 ms  3.549 ms  3.197 ms
 4  * * *
 5  * * *
 6  * * *
 7  * * *
 8  77.37.250.221 (77.37.250.221)  7.821 ms  5.616 ms  5.558 ms
 9  77.37.250.249 (77.37.250.249)  5.530 ms  6.958 ms  6.913 ms
10  72.14.209.81 (72.14.209.81)  6.878 ms  6.842 ms  6.954 ms
```

```
11  * * *
12  108.170.250.129 (108.170.250.129)  7.244 ms 108.170.250.33
(108.170.250.33)  7.180 ms 108.170.250.129 (108.170.250.129)  7.151 ms
13  108.170.250.51 (108.170.250.51)  9.975 ms 108.170.250.113
(108.170.250.113)  9.927 ms 108.170.250.51 (108.170.250.51)  9.842 ms
14  142.251.49.24 (142.251.49.24)  18.530 ms  21.204 ms 216.239.51.32
(216.239.51.32)  25.922 ms
15  74.125.253.94 (74.125.253.94)  35.244 ms 172.253.66.110
(172.253.66.110)  48.365 ms 74.125.253.94 (74.125.253.94)  24.055 ms
16  142.250.238.181 (142.250.238.181)  24.700 ms 172.253.51.243
(172.253.51.243)  33.739 ms 209.85.254.135 (209.85.254.135)  26.701 ms
17  * * *
18  * * *
19  * * *
20  * * *
21  * * *
22  * * *
23  * * *
24  * * *
25  * * *
26  * dns.google (8.8.8.8)  23.135 ms  24.084 ms
root@office1Server:~#
```

В данном примере, в первых трёх переходах мы видим что запрос идет через сервера: **office1Router — centralRouter — inetRouter**

Для того, чтобы Ansible запускался сразу после развертывания серверов командой vagrant up, в текущий Vagrantfile нужно добавить блок запуска Ansible. Данный блок рекомендуется добавить после блока развертывания виртуальных машин:

```
if boxconfig[:vm_name] == "office2Server"
 box.vm.provision "ansible" do |ansible|
  ansible.playbook = "ansible/provision.yml"
  ansible.inventory_path = "ansible/hosts"
  ansible.host_key_checking = "false"
  ansible.limit = "all"
 end
end
```

При добавлении блока Ansible в Vagrant, Ansible будет запускаться после создания каждой ВМ, это создаст ошибку в разворачивании стенда и стенд не развернется. Для того, чтобы Ansible запустился после создания виртуальных машин, можно добавить условие, которое будет сравнивать имя виртуальной машины, и, когда условный оператор увидит имя последней созданной ВМ (office2Server), он запустит Ansible.

Разворачивание 7 виртуальных машин — процесс достаточно затратный для ресурсов компьютера. При разворачивании стенда с помощью Ansible иногда могут вылетать ошибки из-за сетевой связности. Это не страшно, после того как ВМ будут созданы, можно просто ещё раз запустить процесс настройки через ansible с помощью команды: `vagrant provision`

Критерии оценивания

Статус «Принято» ставится при выполнении следующих условий:

1. Ссылка на репозиторий GitHub.
2. Vagrantfile, который будет разворачивать виртуальные машины
3. Настройка виртуальных машин либо вручную, либо с помощью Ansible.
4. Документация по каждому заданию:

Создайте файл README.md и снабдите его следующей информацией:

- название выполняемого задания;
- текст задания;
- схема сети;
- описание команд и их вывод;
- особенности проектирования и реализации решения,
- заметки, если считаете, что имеет смысл их зафиксировать в репозитории.

Рекомендуемые источники

- Статья «Маршрутизация в linux» - https://losst.ru/marshrutizatsiya-v-linux
- Статья «NAT для новичков» - https://habr.com/ru/post/583172/
- Статья «Templating(Jinja2)» - https://docs.ansible.com/ansible/2.9/user_guide/playbooks_templating.html
- Статья «Ansible Provisioner» - https://www.vagrantup.com/docs/provisioning/ansible