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

**Vagrant-стенд c VLAN и LACP**

**Цель домашнего задания**

Научиться настраивать VLAN и LACP.

**Описание домашнего задания**

в Office1 в тестовой подсети появляется сервера с доп интерфейсами и адресами

в internal сети testLAN:

- testClient1 - 10.10.10.254

- testClient2 - 10.10.10.254

- testServer1- 10.10.10.1

- testServer2- 10.10.10.1

Равести вланами:

testClient1 <-> testServer1

testClient2 <-> testServer2

Между centralRouter и inetRouter "пробросить" 2 линка (общая inernal сеть) и объединить их в бонд, проверить работу c отключением интерфейсов

Формат сдачи ДЗ - **vagrant + ansible**

**Введение**

Иногда требуется разделить сеть на несколько подсетей, для этого отлично подходит технология VLAN`ов.

**VLAN (*Virtual Local Area Network,*** *виртуальная локальная компьютерная сеть*) - это виртуальные сети, которые работают на втором уровне модели OSI. Протокол VLAN разделяет хосты на подсети, путём добавления тэга к каждому кадру (Протокол 802.1Q).

**Принцип работы VLAN:**

Группа устройств в сети VLAN взаимодействует так, будто устройства подключены с помощью одного кабеля…

**Преимущества использования VLAN:**

* Безопасность
* Снижение издержек
* Повышение производительности (уменьшение лишнего трафика)
* Сокращение количества доменов широковещательной рассылки
* Повышение производительности ИТ-отдела

Пакеты между VLAN могут передаваться только через маршрутизатор или коммутатор 3-го уровня.

Если через один порт требуется передавать сразу несколько VLAN`ов, то используются Trunk-порты.

Помимо VLAN иногда требуется объединить несколько линков, это делается для увеличения отказоустойчивости.

**Агрегирование каналов (англ. link aggregation)** — технологии объединения нескольких параллельных каналов передачи данных в сетях Ethernet в один логический, позволяющие увеличить пропускную способность и повысить надёжность. В различных конкретных реализациях агрегирования используются альтернативные наименования: транкинг портов (англ. port trunking), связывание каналов (link bundling), склейка адаптеров (NIC bonding), сопряжение адаптеров (NIC teaming).

**LACP (англ. link aggregation control protocol)** — открытый стандартный протокол агрегирования каналов, описанный в документах IEEE 802.3ad и IEEE 802.1aq.

Главное преимущество агрегирования каналов в том, что потенциально повышается полоса пропускания: в идеальных условиях полоса может достичь суммы полос пропускания объединенных каналов. Другое преимущество — «горячее» резервирование линий связи: в случае отказа одного из агрегируемых каналов трафик без прерывания сервиса посылается через оставшиеся, а после восстановления отказавшего канала он автоматически включается в работу

**Функциональные и нефункциональные требования**

* ПК на Unix c 10ГБ 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 2.8 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 2.2.19, VirtualBox v6.1.26 r145957. В лабораторной работе используются Vagrant boxes с CentOS 8 Stream и Ubuntu 22.04. Серьёзные отступления от этой конфигурации могут потребовать адаптации с вашей стороны.

# -\*- mode: ruby -\*-

# vim: set ft=ruby :

MACHINES **=** **{**

:inetRouter **=>** **{**

:box\_name **=>** "centos/stream8"**,**

:box\_version **=>** "20210210.0"**,**

:vm\_name **=>** "inetRouter"**,**

:net **=>** **[**

**{**adapter: 2**,** auto\_config: **false,** virtualbox\_\_intnet: "router-net"**},**

**{**adapter: 3**,** auto\_config: **false,** virtualbox\_\_intnet: "router-net"**},**

**{**ip: '192.168.56.10'**,** adapter: 8**},**

**]**

**},**

:centralRouter **=>** **{**

:box\_name **=>** "centos/stream8"**,**

:box\_version **=>** "20210210.0"**,**

:vm\_name **=>** "centralRouter"**,**

:net **=>** **[**

**{**adapter: 2**,** auto\_config: **false,** virtualbox\_\_intnet: "router-net"**},**

**{**adapter: 3**,** auto\_config: **false,** virtualbox\_\_intnet: "router-net"**},**

**{**ip: '192.168.255.9'**,** adapter: 6**,** netmask: "255.255.255.252"**,** virtualbox\_\_intnet: "office1-central"**},**

**{**ip: '192.168.56.11'**,** adapter: 8**},**

**]**

**},**

:office1Router **=>** **{**

:box\_name **=>** "centos/stream8"**,**

:box\_version **=>** "20210210.0"**,**

:vm\_name **=>** "office1Router"**,**

:net **=>** **[**

**{**ip: '192.168.255.10'**,** adapter: 2**,** netmask: "255.255.255.252"**,** virtualbox\_\_intnet: "office1-central"**},**

**{**adapter: 3**,** auto\_config: **false,** virtualbox\_\_intnet: "vlan1"**},**

**{**adapter: 4**,** auto\_config: **false,** virtualbox\_\_intnet: "vlan1"**},**

**{**adapter: 5**,** auto\_config: **false,** virtualbox\_\_intnet: "vlan2"**},**

**{**adapter: 6**,** auto\_config: **false,** virtualbox\_\_intnet: "vlan2"**},**

**{**ip: '192.168.56.20'**,** adapter: 8**},**

**]**

**},**

:testClient1 **=>** **{**

:box\_name **=>** "centos/stream8"**,**

:box\_version **=>** "20210210.0"**,**

:vm\_name **=>** "testClient1"**,**

:net **=>** **[**

**{**adapter: 2**,** auto\_config: **false,** virtualbox\_\_intnet: "testLAN"**},**

**{**ip: '192.168.56.21'**,** adapter: 8**},**

**]**

**},**

:testServer1 **=>** **{**

:box\_name **=>** "centos/stream8"**,**

:box\_version **=>** "20210210.0"**,**

:vm\_name **=>** "testServer1"**,**

:net **=>** **[**

**{**adapter: 2**,** auto\_config: **false,** virtualbox\_\_intnet: "testLAN"**},**

**{**ip: '192.168.56.22'**,** adapter: 8**},**

**]**

**},**

:testClient2 **=>** **{**

:box\_name **=>** "ubuntu/jammy64"**,**

:box\_version **=>** "20220411.2.0"**,**

:vm\_name **=>** "testClient2"**,**

:net **=>** **[**

**{**adapter: 2**,** auto\_config: **false,** virtualbox\_\_intnet: "testLAN"**},**

**{**ip: '192.168.56.31'**,** adapter: 8**},**

**]**

**},**

:testServer2 **=>** **{**

:box\_name **=>** "ubuntu/jammy64"**,**

:box\_version **=>** "20220411.2.0"**,**

:vm\_name **=>** "testServer2"**,**

:net **=>** **[**

**{**adapter: 2**,** auto\_config: **false,** virtualbox\_\_intnet: "testLAN"**},**

**{**ip: '192.168.56.32'**,** adapter: 8**},**

**]**

**},**

**}**

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**.**box\_version **=** boxconfig**[**:box\_version**]**

config**.**vm**.**provider "virtualbox" **do** **|**v**|**

v**.**memory **=** 1024

v**.**cpus **=** 2

**end**

**if** boxconfig**[**:vm\_name**]** **==** "testServer2"

box**.**vm**.**provision "ansible" **do** **|**ansible**|**

ansible**.**playbook **=** "ansible/provision.yml"

ansible**.**inventory\_path **=** "ansible/hosts"

ansible**.**host\_key\_checking **=** "false"

ansible**.**become **=** "true"

ansible**.**limit **=** "all"

**end**

**end**

boxconfig**[**:net**].**each **do** **|**ipconf**|**

box**.**vm**.**network "private\_network"**,** ipconf

**end**

box**.**vm**.**provision "shell"**,** inline: **<<-**SHELL

mkdir **-**p **~**root**/.**ssh

cp **~**vagrant**/.**ssh**/**auth**\*** **~**root**/.**ssh

SHELL

**end**

**end**

**end**

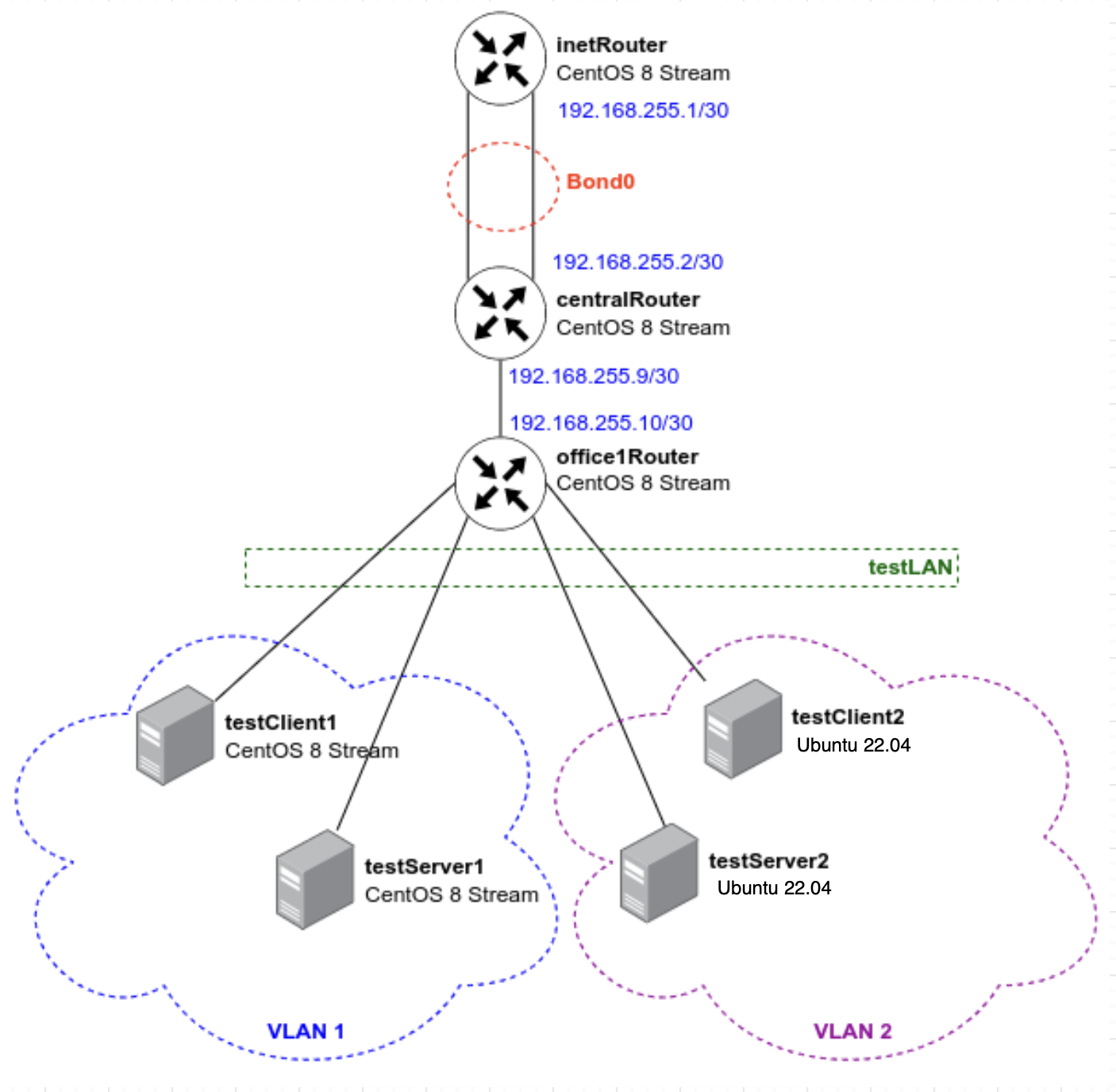
Данный Vagrantfile развернет 7 виртуальных машин:

* 5 ВМ на CentOS 8 Stream
* 2 ВМ на Ubuntu 22.04

Обратите внимание, что хосты testClient1, testServer1, testClient2 и testServer2 находятся в одной сети (testLAN).

Для использования Ansible, каждому хосту выделен ip-адрес из подсети 192.168.56.0/24.

По итогу выполнения домашнего задания у нас должна получиться следующая топология сети:



**Предварительная настройка хостов**

*Перед настройкой VLAN и LACP рекомендуется установить на хосты следующие утилиты:*

* *vim*
* *traceroute*
* *tcpdump*
* *net-tools*

Установка пакетов на CentOS 8 Stream:

*yum install -y vim traceroute tcpdump net-tools*

Установка пакетов на Ubuntu 22.04:

*apt install -y vim traceroute tcpdump net-tools*

**Предварительная настройка хостов с помощью Ansible**

**- name**: Base set up

**#Настройка производится на всех хостах**

**hosts**: all

**become**: **yes**

**tasks**:

**#Установка приложений на RedHat-based системах**

**- name**: install software on CentOS

**yum**:

**name**:

- vim

- traceroute

- tcpdump

- net-tools

**state**: present

**update\_cache**: **true**

**when**: (ansible\_os\_family == "RedHat")

**#Установка приложений на Debian-based системах**

**- name**: install software on Debian-based

**apt**:

**name**:

- vim

- traceroute

- tcpdump

- net-tools

**state**: present

**update\_cache**: **true**

**when**: (ansible\_os\_family == "Debian")

**Настройка VLAN на хостах**

**Настройка VLAN на RHEL-based системах:**

На хосте **testClient1** требуется создать файл ***/etc/sysconfig/network-scripts/ifcfg-vlan1*** *со следующим параметрами:*

VLAN=yes

**#Тип интерфейса - VLAN**

TYPE=Vlan

**#Указываем физическое устройство, через которые будет работать VLAN**

PHYSDEV=eth1

**#Указываем номер VLAN (VLAN\_ID)**

VLAN\_ID=1

VLAN\_NAME\_TYPE=DEV\_PLUS\_VID\_NO\_PAD

PROXY\_METHOD=none

BROWSER\_ONLY=no

BOOTPROTO=none

**#Указываем IP-адрес интерфейса**

IPADDR=10.10.10.254

**#Указываем префикс (маску) подсети**

PREFIX=24

**#Указываем имя vlan**

NAME=vlan1

**#Указываем имя подинтерфейса**

DEVICE=eth1.1

ONBOOT=yes

На хосте **testServer1** создадим идентичный файл с другим IP-адресом (10.10.10.1).

После создания файлов нужно перезапустить сеть на обоих хостах:

***systemctl restart NetworkManager***

Проверим настройку интерфейса, если настройка произведена правильно, то с хоста testClient1 будет проходить ping до хоста testServer1:

[vagrant@testClient1 ~]$ **ip a**

1: lo: <LOOPBACK,UP,LOWER\_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000

link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00

inet 127.0.0.1/8 scope host lo

valid\_lft forever preferred\_lft forever

inet6 ::1/128 scope host

valid\_lft forever preferred\_lft forever

2: eth0: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc fq\_codel state UP group default qlen 1000

link/ether 52:54:00:03:15:fa brd ff:ff:ff:ff:ff:ff

inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute eth0

valid\_lft 80162sec preferred\_lft 80162sec

inet6 fe80::5054:ff:fe03:15fa/64 scope link

valid\_lft forever preferred\_lft forever

3: eth1: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc fq\_codel state UP group default qlen 1000

link/ether 08:00:27:cb:35:ae brd ff:ff:ff:ff:ff:ff

inet6 fe80::e1b1:ec7e:6337:dd54/64 scope link noprefixroute

valid\_lft forever preferred\_lft forever

4: eth2: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc fq\_codel state UP group default qlen 1000

link/ether 08:00:27:91:64:3c brd ff:ff:ff:ff:ff:ff

inet 192.168.56.21/24 brd 192.168.56.255 scope global noprefixroute eth2

valid\_lft forever preferred\_lft forever

inet6 fe80::a00:27ff:fe91:643c/64 scope link

valid\_lft forever preferred\_lft forever

**5: eth1.1@eth1: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000**

**link/ether 08:00:27:cb:35:ae brd ff:ff:ff:ff:ff:ff**

**inet 10.10.10.254/24 brd 10.10.10.255 scope global noprefixroute eth1.1**

**valid\_lft forever preferred\_lft forever**

**inet6 fe80::a00:27ff:fecb:35ae/64 scope link**

**valid\_lft forever preferred\_lft forever**

[vagrant@testClient1 ~]$ **ping 10.10.10.254**

**PING 10.10.10.254 (10.10.10.254) 56(84) bytes of data.**

**64 bytes from 10.10.10.254: icmp\_seq=1 ttl=64 time=0.081 ms**

**64 bytes from 10.10.10.254: icmp\_seq=2 ttl=64 time=0.083 ms**

**64 bytes from 10.10.10.254: icmp\_seq=3 ttl=64 time=0.082 ms**

**64 bytes from 10.10.10.254: icmp\_seq=4 ttl=64 time=0.080 ms**

^C

--- 10.10.10.254 ping statistics ---

4 packets transmitted, 4 received, 0% packet loss, time 3056ms

rtt min/avg/max/mdev = 0.080/0.081/0.083/0.009 ms

[vagrant@testClient1 ~]$

**Настройка VLAN на Ubuntu:**

На хосте **testClient2** требуется создать файл **/etc/netplan/50-cloud-init.yaml** со следующим параметрами:

# This file is generated from information provided by the datasource. Changes

# to it will not persist across an instance reboot. To disable cloud-init's

# network configuration capabilities, write a file

# /etc/cloud/cloud.cfg.d/99-disable-network-config.cfg with the following:

# network: {config: disabled}

network:

version: 2

ethernets:

enp0s3:

dhcp4: true

**#В разделе ethernets добавляем порт, на котором будем настраивать VLAN**

enp0s8: {}

**#Настройка VLAN**

vlans:

**#Имя VLANа**

vlan2:

**#Указываем номер VLAN`а**

id: 2

**#Имя физического интерфейса**

link: enp0s8

**#Отключение DHCP-клиента**

dhcp4: no

**#Указываем ip-адрес**

addresses: [10.10.10.254/24]

На хосте **testServer2** создадим идентичный файл с другим IP-адресом (10.10.10.1).

После создания файлов нужно перезапустить сеть на обоих хостах: ***netplan apply***

После настройки второго VLAN`а ping должен работать между хостами testClient1, testServer1 и между хостами testClient2, testServer2.

*Примечание: до остальных хостов ping работать не будет, так как не настроена маршрутизация.*

**Настройка VLAN с помощью Ansible:**

**Настройка VLAN 1 на хостах testClient1 и testServer1**

**- name**: set up vlan1

**#Настройка будет производиться на хостах testClient1 и testServer1**

**hosts**: testClient1,testServer1

**#Настройка производится от root-пользователя**

**become**: **yes**

**tasks**:

**#Добавление темплейта в файл /etc/sysconfig/network-scripts/ifcfg-vlan1**

**- name**: set up vlan1

**template**:

**src**: ifcfg-vlan1.j2

**dest**: /etc/sysconfig/network-scripts/ifcfg-vlan1

**owner**: root

**group**: root

**mode**: 0644

**#Перезапуск службы NetworkManager**

**- name**: restart network for vlan1

**service**:

**name**: NetworkManager

**state**: restarted

Чтобы не делать 2 отдельных конфигурационных файла, можно сделать **template**, который автоматически поменяет IP-адрес и VLAN\_ID:

VLAN=yes

TYPE=Vlan

PHYSDEV=eth1

VLAN\_ID=**{{ vlan\_id }}**

VLAN\_NAME\_TYPE=DEV\_PLUS\_VID\_NO\_PAD

PROXY\_METHOD=none

BROWSER\_ONLY=no

BOOTPROTO=none

IPADDR=**{{ vlan\_ip }}**

PREFIX=24

NAME=vlan**{{ vlan\_id }}**

DEVICE=eth1.**{{ vlan\_id }}**

ONBOOT=yes

Параметры **vlan\_id** и **vlan\_ip** можно указать в файле hosts:

[nets]

inetRouter ansible\_host=192.168.56.10 ansible\_user=vagrant ansible\_ssh\_private\_key\_file=./.vagrant/machines/inetRouter/virtualbox/private\_key **bond\_ip=192.168.255.1**

centralRouter ansible\_host=192.168.56.11 ansible\_user=vagrant ansible\_ssh\_private\_key\_file=./.vagrant/machines/centralRouter/virtualbox/private\_key **bond\_ip=192.168.255.2**

office1Router ansible\_host=192.168.56.20 ansible\_user=vagrant ansible\_ssh\_private\_key\_file=./.vagrant/machines/office1Router/virtualbox/private\_key

testClient1 ansible\_host=192.168.56.21 ansible\_user=vagrant ansible\_ssh\_private\_key\_file=./.vagrant/machines/testClient1/virtualbox/private\_key **vlan\_id=1 vlan\_ip=10.10.10.254**

testServer1 ansible\_host=192.168.56.22 ansible\_user=vagrant ansible\_ssh\_private\_key\_file=./.vagrant/machines/testServer1/virtualbox/private\_key **vlan\_id=1 vlan\_ip=10.10.10.1**

testClient2 ansible\_host=192.168.56.31 ansible\_user=vagrant ansible\_ssh\_private\_key\_file=./.vagrant/machines/testClient2/virtualbox/private\_key **vlan\_id=2 vlan\_ip=10.10.10.254**

testServer2 ansible\_host=192.168.56.32 ansible\_user=vagrant ansible\_ssh\_private\_key\_file=./.vagrant/machines/testServer2/virtualbox/private\_key **vlan\_id=2 vlan\_ip=10.10.10.1**

**Настройка VLAN 2 на хостах testClient2 и testServer2:**

**- name**: set up vlan2

**hosts**: testClient2,testServer2

**become**: **yes**

**tasks**:

**- name**: set up vlan2

**template**:

**src**: 50-cloud-init.yaml.j2

**dest**: /etc/netplan/50-cloud-init.yaml

**owner**: root

**group**: root

**mode**: 0644

**- name**: apply set up vlan2

**shell**: netplan apply

**become**: **true**

Чтобы не делать 2 отдельных конфигурационных файла, можно сделать **template**, который автоматически поменяет IP-адрес и VLAN\_ID:

# This file is generated from information provided by the datasource. Changes

# to it will not persist across an instance reboot. To disable cloud-init's

# network configuration capabilities, write a file

# /etc/cloud/cloud.cfg.d/99-disable-network-config.cfg with the following:

# network: {config: disabled}

network:

version: 2

ethernets:

enp0s3:

dhcp4: true

enp0s8: {}

vlans:

vlan**{{ vlan\_id }}**:

id: **{{ vlan\_id }}**

link: enp0s8

dhcp4: no

addresses: [**{{ vlan\_ip }}**/24]

Параметры **vlan\_id** и **vlan\_ip** такжеуказаны в файле hosts.

**Настройка LACP между хостами inetRouter и centralRouter**

*Bond интерфейс будет работать через порты eth1 и eth2.*

1) Изначально необходимо на обоих хостах добавить конфигурационные файлы для интерфейсов **eth1** и **eth2**:

***vim /etc/sysconfig/network-scripts/ifcfg-eth1***

**#Имя физического интерфейса**

DEVICE=eth1

**#Включать интерфейс при запуске системы**

ONBOOT=yes

**#Отключение DHCP-клиента**

BOOTPROTO=none

**#Указываем, что порт часть bond-интерфейса**

MASTER=bond0

**#Указываем роль bond**

SLAVE=yes

NM\_CONTROLLED=yes

USERCTL=no

У интерфейса ***ifcfg-eth2*** *идентичный конфигурационный файл, в котором нужно изменить имя интерфейса.*

2) После настройки интерфейсов eth1 и eth2 нужно настроить bond-интерфейс, для этого создадим файл ***/etc/sysconfig/network-scripts/ifcfg-bond0***

***vim /etc/sysconfig/network-scripts/ifcfg-bond0***

*DEVICE=bond0*

*NAME=bond0*

***#Тип интерфейса — bond***

*TYPE=Bond*

*BONDING\_MASTER=yes*

***#Указаваем IP-адрес***

*IPADDR=192.168.255.1*

***#Указываем маску подсети***

*NETMASK=255.255.255.252*

*ONBOOT=yes*

*BOOTPROTO=static*

***#Указываем режим работы bond-интерфейса Active-Backup***

***# fail\_over\_mac=1 — данная опция «разрешает отвалиться» одному интерфейсу***

*BONDING\_OPTS="mode=1 miimon=100 fail\_over\_mac=1"*

*NM\_CONTROLLED=yes*

После создания данных конфигурационных файлов необходимо перезапустить сеть:

***systemctl restart NetworkManager***

*На некоторых версиях RHEL/CentOS перезапуск сетевого интерфейса не запустит bond-интерфейс, в этом случае рекомендуется перезапустить хост.*

После настройки агрегации портов, необходимо проверить работу bond-интерфейса, для этого, на хосте inetRouter (192.168.255.1) запустим ping до centralRouter (192.168.255.2):

[root@inetRouter ~]# **ping 192.168.255.2**

PING 192.168.255.2 (192.168.255.2) 56(84) bytes of data.

64 bytes from 192.168.255.2: icmp\_seq=1 ttl=64 time=1.49 ms

64 bytes from 192.168.255.2: icmp\_seq=2 ttl=64 time=1.00 ms

64 bytes from 192.168.255.2: icmp\_seq=3 ttl=64 time=0.926 ms

64 bytes from 192.168.255.2: icmp\_seq=4 ttl=64 time=0.912 ms

64 bytes from 192.168.255.2: icmp\_seq=5 ttl=64 time=1.04 ms

64 bytes from 192.168.255.2: icmp\_seq=6 ttl=64 time=0.889 ms

Не отменяя ping подключаемся к хосту centralRouter и выключаем там интерфейс eth1:

[root@centralRouter ~]# **ip link set down eth1**

После данного действия ping не должен пропасть, так как трафик пойдёт по-другому порту.

**Настройка LACP между хостами inetRouter и centralRouter с помощью Ansible**

**- name**: set up bond0

**hosts**: inetRouter,centralRouter

**become**: **yes**

**tasks**:

**- name**: set up ifcfg-bond0

**template**:

**src**: ifcfg-bond0.j2

**dest**: /etc/sysconfig/network-scripts/ifcfg-bond0

**owner**: root

**group**: root

**mode**: 0644

**- name**: set up eth1,eth2

**copy**:

**src**: "{{ item }}"

**dest**: /etc/sysconfig/network-scripts/

**owner**: root

**group**: root

**mode**: 0644

**with\_items**:

- templates/ifcfg-eth1

- templates/ifcfg-eth2

**#Перезагрузка хостов**

**- name**: restart hosts for bond0

**reboot**:

**reboot\_timeout**: 3600

Чтобы не создавать 2 конфигурационных файла для bond-интерфейса можно сделать template:

DEVICE=bond0

NAME=bond0

TYPE=Bond

BONDING\_MASTER=yes

IPADDR=**{{ bond\_ip }}**

NETMASK=255.255.255.252

ONBOOT=yes

BOOTPROTO=static

BONDING\_OPTS="mode=1 miimon=100 fail\_over\_mac=1"

NM\_CONTROLLED=yes

USERCTL=no

Параметры **bond\_ip** такжеуказаны в файле hosts.

**Критерии оценивания**

Статус «Принято» ставится при выполнении следующих условий:

1. Сcылка на репозиторий GitHub.

2. Vagrantfile, который будет разворачивать виртуальные машины

3. Настройка виртуальных машин происходит 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 «VLAN» - <https://ru.wikipedia.org/wiki/VLAN>
* Статья «VLAN для чайников» - https://asp24.ru/novichkam/vlan-dlya-chaynikov/
* Статья «Настройка сети в Linux с помощью netplan» - <https://www.dmosk.ru/miniinstruktions.php?mini=network-netplan>
* Статья «Настройка VLAN на Linux CentOS 7» - <https://www.dmosk.ru/miniinstruktions.php?mini=vlan-centos>
* Статья «Агрегиование каналов» - <https://ru.wikipedia.org/wiki/%D0%90%D0%B3%D1%80%D0%B5%D0%B3%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BA%D0%B0%D0%BD%D0%B0%D0%BB%D0%BE%D0%B2>
* Статья «Настройка Bonding в режиме Active-backup на CentOS» - <http://www.zaweel.ru/2016/07/bonding-active-backup-centos-1.html>