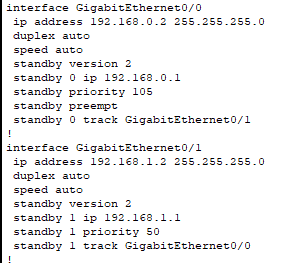
# **Задание 1**

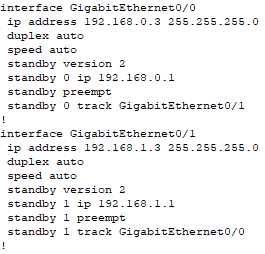
* Дана [схема](https://github.com/netology-code/sflt-homeworks/blob/main/1/hsrp_advanced.pkt) для Cisco Packet Tracer, рассматриваемая в лекции.
* На данной схеме уже настроено отслеживание интерфейсов маршрутизаторов Gi0/1 (для нулевой группы)
* Необходимо аналогично настроить отслеживание состояния интерфейсов Gi0/0 (для первой группы).
* Для проверки корректности настройки, разорвите один из кабелей между одним из маршрутизаторов и Switch0 и запустите ping между PC0 и Server0.
* На проверку отправьте получившуюся схему в формате pkt и скриншот, где виден процесс настройки маршрутизатора.

# **Решение 1**

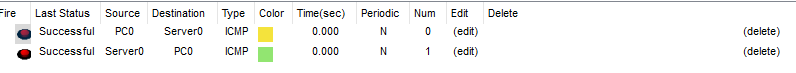
Ro1:



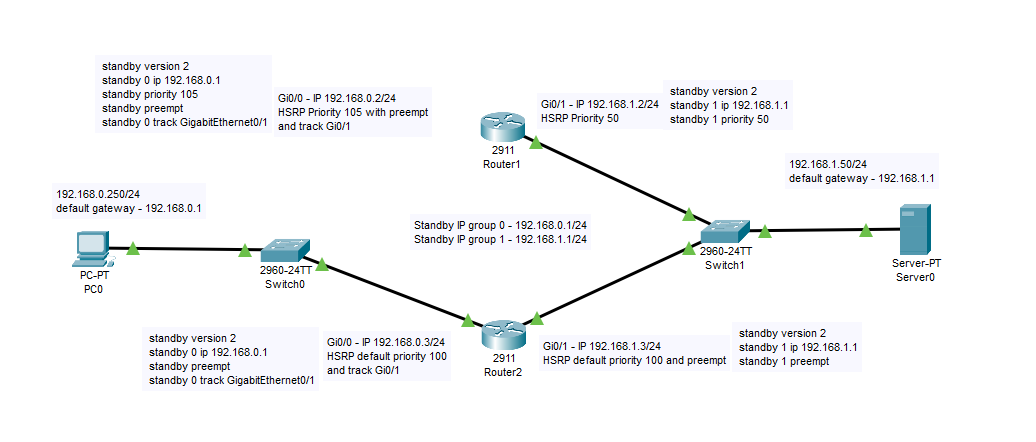
Ro2:



При всех работающих коннектах все ок, ожидаемо:

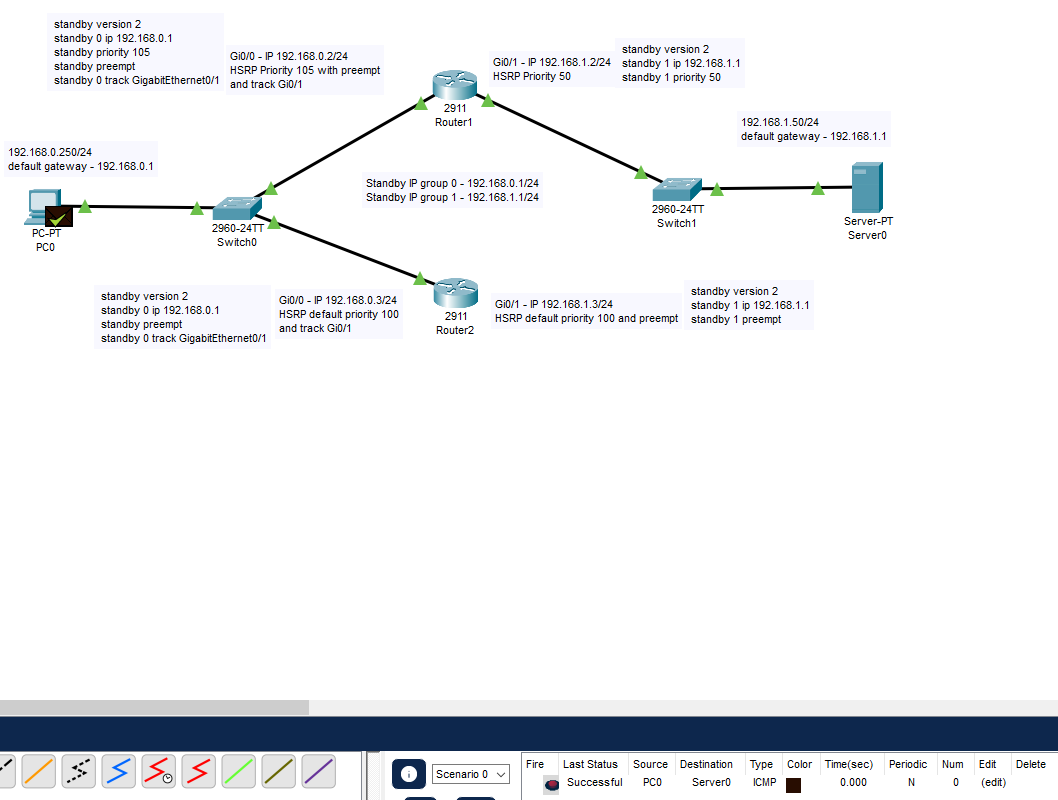


Разорвем один коннект:





Пингуется. Порвем другой:



Тоже пингуется.

# **Задание 2**

* Запустите две виртуальные машины Linux, установите и настройте сервис Keepalived как в лекции, используя пример конфигурационного [файла](https://github.com/netology-code/sflt-homeworks/blob/main/1/keepalived-simple.conf).
* Настройте любой веб-сервер (например, nginx или simple python server) на двух виртуальных машинах
* Напишите Bash-скрипт, который будет проверять доступность порта данного веб-сервера и существование файла index.html в root-директории данного веб-сервера.
* Настройте Keepalived так, чтобы он запускал данный скрипт каждые 3 секунды и переносил виртуальный IP на другой сервер, если bash-скрипт завершался с кодом, отличным от нуля (то есть порт веб-сервера был недоступен или отсутствовал index.html). Используйте для этого секцию vrrp\_script
* На проверку отправьте получившейся bash-скрипт и конфигурационный файл keepalived, а также скриншот с демонстрацией переезда плавающего ip на другой сервер в случае недоступности порта или файла index.html

# **Решение 2**

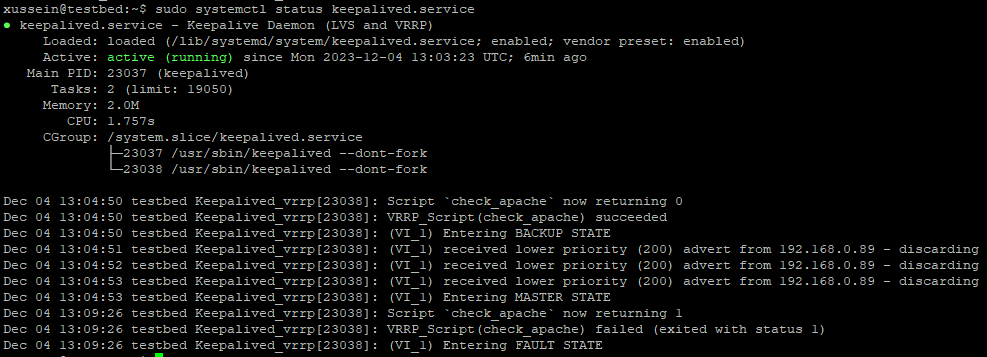
Конфигурационный файл keepalived:

1. vrrp\_script check\_apache {
2. script "/home/xussein/netology/fault\_tolerance/check\_apache.sh"
3. interval 3
4. fall 1
5. rise 1
6. }
7. vrrp\_instance VI\_1 {
8. state MASTER
9. interface ens192
10. virtual\_router\_id 15
11. priority 255
12. advert\_int 1
13. virtual\_ipaddress {
14. 192.168.0.15/23
15. }
16. track\_script {
17. check\_apache
18. }
19. }

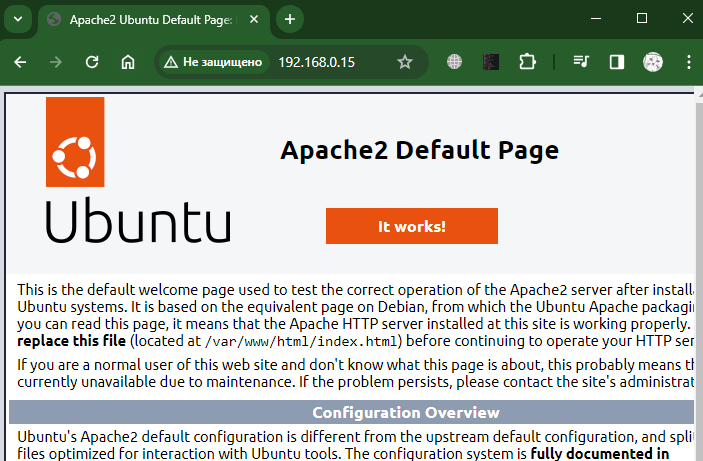
Скрипт проверки доступности сервиса:

1. #!/bin/bash
2. nc -zv localhost 80 > /dev/null 2>&1
3. if [[ $? -eq 1 ]]; then
4. exit 1
5. elif [[ ! -e /var/www/html/index.html ]]; then
6. exit 2
7. else
8. exit 0
9. fi

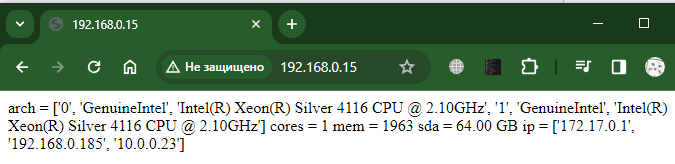
Собственно переезд при отключении апача:



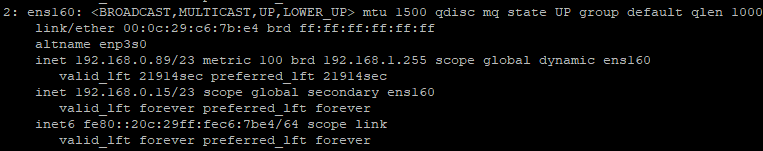
На мастере стоял чистый апач сервер:

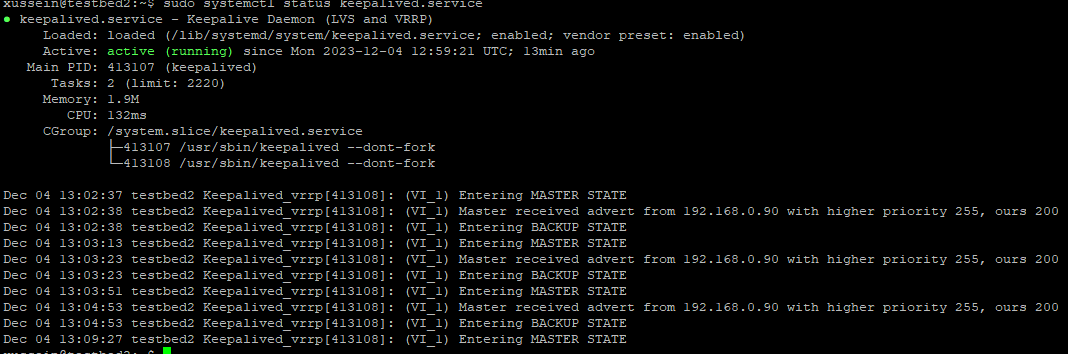


После отключения на бэке запускается сервер с одной из предыдущих домашек:



Вывод данных о платформе. На бэке запускается виртуальный айпишник:





Тут я уже поигрался, несколько раз включал-выключал апач, как видно на скрине. Бэк несколько раз переходил из слейва в мастер.

# **Задание 3**

* Изучите дополнительно возможность Keepalived, которая называется vrrp\_track\_file
* Напишите bash-скрипт, который будет менять приоритет внутри файла в зависимости от нагрузки на виртуальную машину (можно разместить данный скрипт в cron и запускать каждую минуту). Рассчитывать приоритет можно, например, на основании Load average.
* Настройте Keepalived на отслеживание данного файла.
* Нагрузите одну из виртуальных машин, которая находится в состоянии MASTER и имеет активный виртуальный IP и проверьте, чтобы через некоторое время она перешла в состояние SLAVE из-за высокой нагрузки и виртуальный IP переехал на другой, менее нагруженный сервер.
* Попробуйте выполнить настройку keepalived на третьем сервере и скорректировать при необходимости формулу так, чтобы плавающий ip адрес всегда был прикреплен к серверу, имеющему наименьшую нагрузку.
* На проверку отправьте получившийся bash-скрипт и конфигурационный файл keepalived, а также скриншоты логов keepalived с серверов при разных нагрузках

# **Решение 3**

Файл скрипта:

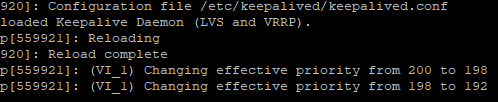
1. #!/bin/bash
2. loadavg=$(sudo cat /proc/loadavg | awk '{print int($1 \* 10)}')
3. echo "-$loadavg" > /home/xussein/netology/fault\_tolerance/loadavg

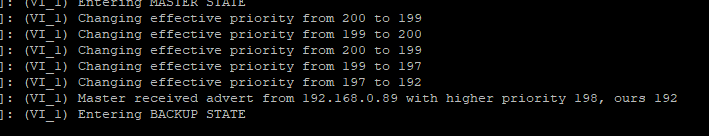
На этом этапе хотелось убить себя, так как напрочь забыл что cron будет писать куда попало, если оставить строку echo "-$loadavg" > loadavg в таком виде. Как мне показалось сделать значение приоритета значением 200 - loadavg \* 10 наиболее оптимальным. Сами же стартовые приоритеты нужно высчитывать исходя из производительности разных устройств.

Файл конфигурации keepalived:

1. vrrp\_script check\_apache {
2. script "/home/xussein/netology/fault\_tolerance/check\_apache.sh"
3. interval 3
4. fall 1
5. rise 1
6. }
7. vrrp\_track\_file check\_loadavg {
8. file "/home/xussein/netology/fault\_tolerance/loadavg"
9. }
10. vrrp\_instance VI\_1 {
11. state MASTER
12. interface ens192
13. virtual\_router\_id 15
14. priority 200
15. advert\_int 1
16. virtual\_ipaddress {
17. 192.168.0.15/23
18. }
19. track\_script {
20. check\_apache
21. }
22. track\_file {
23. check\_loadavg weight 1
24. }
25. }

Загружаем сервера для изменения loadavg (dd if=/dev/zero of=/dev/null):





Стартовые приортитеты - 200. Как видно, при их изменении сервера меняют друг друга. Более нагруженный становится backup, менее - master.