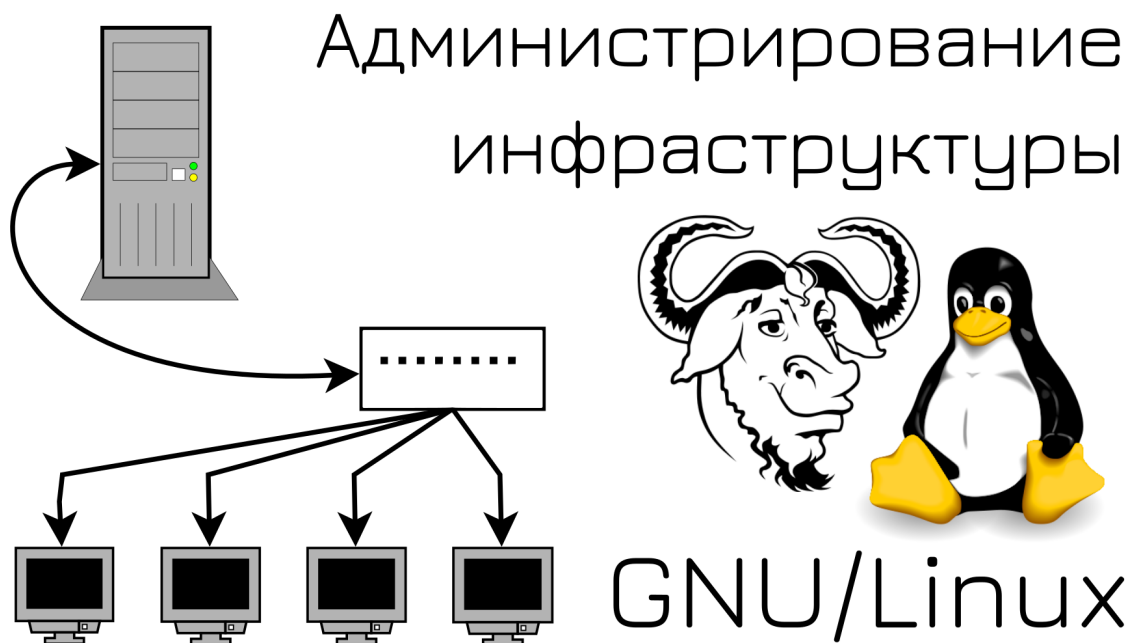

INFRA

Выпуск 17/11/2021

GNU Linux Pro

янв. 24, 2024

1	Вступление	1
1.1	Содержимое	2
2	Курс	3
2.1	01. Как устроена IT инфраструктура	3
2.2	02. Подготовка лабораторной	22
2.3	03. Деление на VLAN-ы	41
2.4	04. Объединение портов	67
2.5	05. Деление на подсети	91
2.6	06. Как попасть в сеть через SSH	118
2.7	07. Деление на зоны	148
2.8	08. Политики Firewall	175
2.9	09. Плавающий IP	194
2.10	10. DHCP сервер ч.1	218
2.11	11. DHCP сервер ч.2	239
2.12	12. Основы DNS ч.1	269
2.13	13. Основы DNS ч.2	286



Всем привет!

Это начало курса по администрированию инфраструктуры. Здесь мы разберём всё что полезно понимать системному администратору - от поднятия инфраструктуры с нуля до настройки различных отказоустойчивых сетевых сервисов. Данный курс является продолжением курса «Основы GNU/Linux» и требует понимания многих тем, разобранных там.

Особенностью курса является то, что вся наша инфраструктура будет строиться на операционных системах семейства GNU/Linux. Реальная инфраструктура зачастую состоит из множества готовых проприетарных решений, я же буду акцентировать внимание не на инструментах, а на задачах ад-

министратора. Т.е. мы научимся практикам, которые применимы везде, независимо от программного обеспечения. И тем не менее, умение работать с open-source решениями также будет вам полезно.

1.1 Содержимое

1.1.1 Ссылки

1.1.2 Команды

Команды

03. L2 - VLAN

```
nmcli con add type vlan con-name vlan2 dev enp0s3 id 2
```

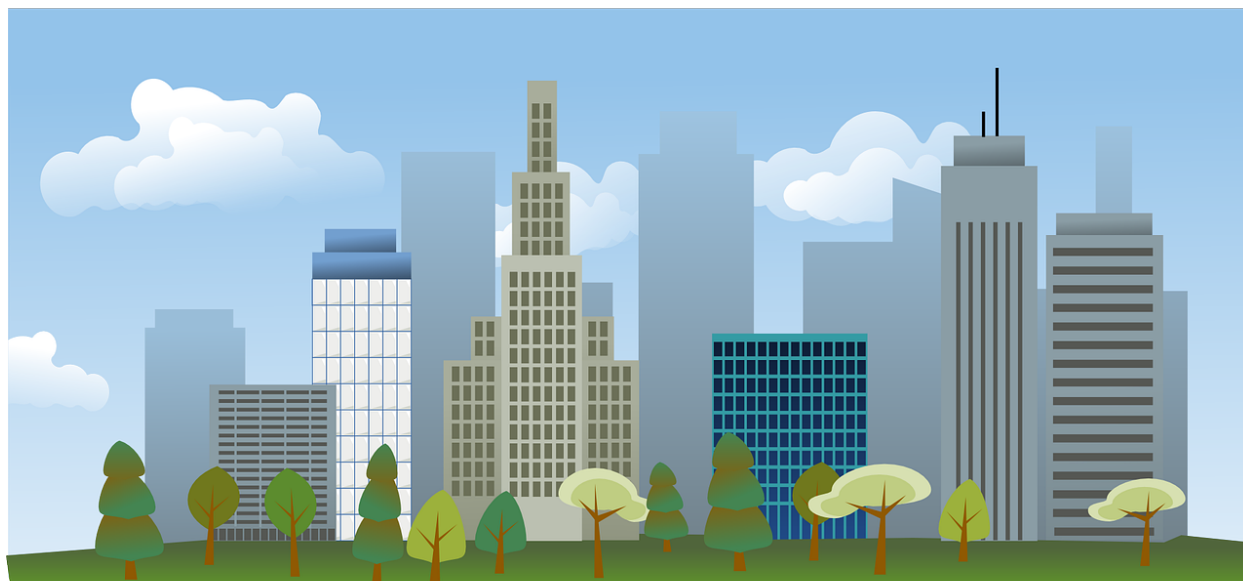
04. L2 - Агрегирование каналов

```
nmcli con sh
nmcli con del con1 con2
nmcli con add type team con-name teams3s8 ifname team0 team.runner lacp
nmcli con add type team-slave ifname enp0s3 master team0
nmcli con add type team-slave ifname enp0s8 master team0
```

2.1 01. Как устроена IT инфраструктура

2.1.1 01. Как устроена IT инфраструктура

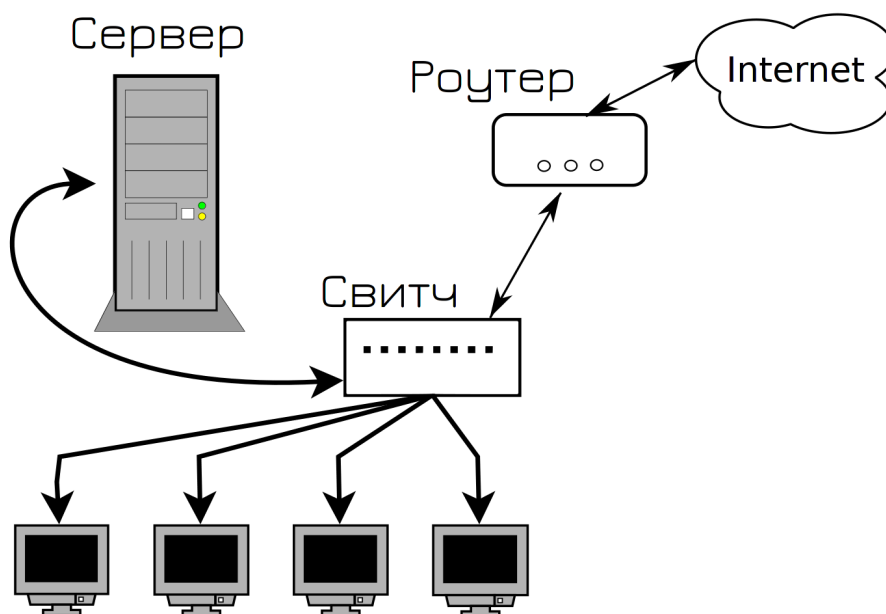
IT инфраструктура



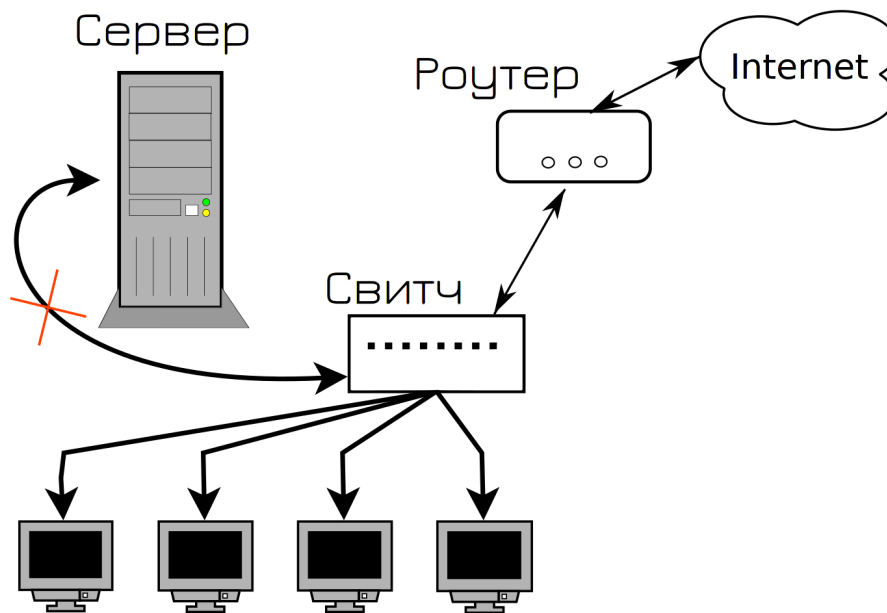
Начнём с вопроса - кто такой системный администратор? Человек, который занимается настройкой и отвечает за штатную работу компьютерной техники, сети и программного обеспечения. В совокупности это называется IT-инфраструктурой. Её можно сравнить с инфраструктурой города - есть системы, вроде водоснабжение, электричество, дороги, вывоз мусора и прочее - и они нужны, чтобы люди могли

спокойно жить и работать в городах. Также и в IT - без нормальной инфраструктуры пользователи не смогут спокойно работать. Хотя инфраструктура есть во всех городах, но в каждом она уникальна. Так и IT инфраструктура - везде что-то по разному, но база одна и та же. Поэтому ответ на вопрос «Как устроена инфраструктура?» - зависит от компании.

Сеть



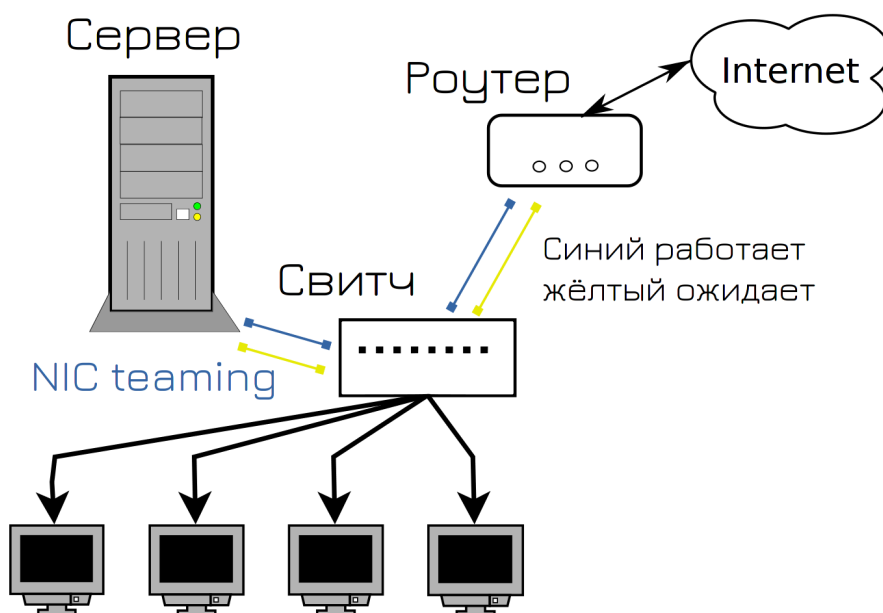
Давайте мы создадим свою компанию со своей инфраструктурой. Говорим с провайдером, он нам даёт роутер и интернет. Берём какой-нибудь системный блок и называем его сервером. Ставим на него Linux, поднимаем какой-нибудь сервис, выдаём IP адрес. Чтобы связать между собой пользователей, сервер и роутер, покупаем какой-нибудь простой свитч и подключаем всё между собой медными патч-кордами. Вот и готова наша инфраструктура.



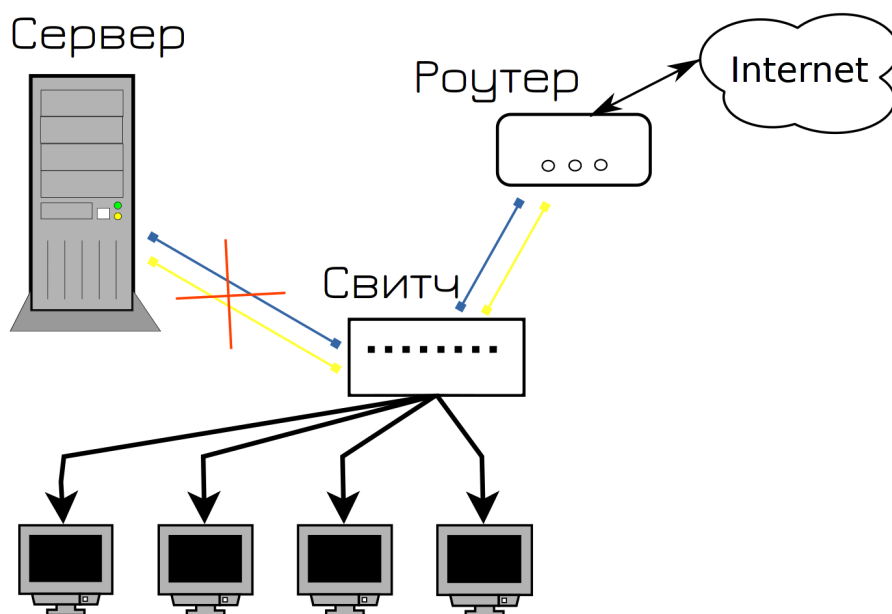
И всё работает, пока в один из дней кто-то случайно не положит что-то тяжелое на кабель, идущий к серверу. Кабель испортится и сервер становится недоступным. А вы в этот день взяли отгул. Пользователи не могут зайти на сервер, клиенты не довольны и вам в срочном порядке нужно ехать в офис и подключать новый кабель.

Вся эта история знакомит нас с таким термином, как SPOF - single point of failure - единая точка отказа. Это такой элемент системы, выход из строя которого приводит к остановке работы сервиса. Сервис не в плане программы в системе, а в плане услуги для потребителя. Так вот, в нашей инфраструктуре кабель, соединяющий сервер со свитчем, был единой точкой отказа - он вышел из строя и сервис стал недоступен. Даже если операционка работает, программа внутри работает - для пользователей она недоступна.

SPOF - это движущая сила ИТ, это один из главных терминов, отвечающих на большинство вопросов, в том числе - почему инфраструктура устроена так, а не иначе. Впрочем, вы всё поймёте, давайте продолжим.

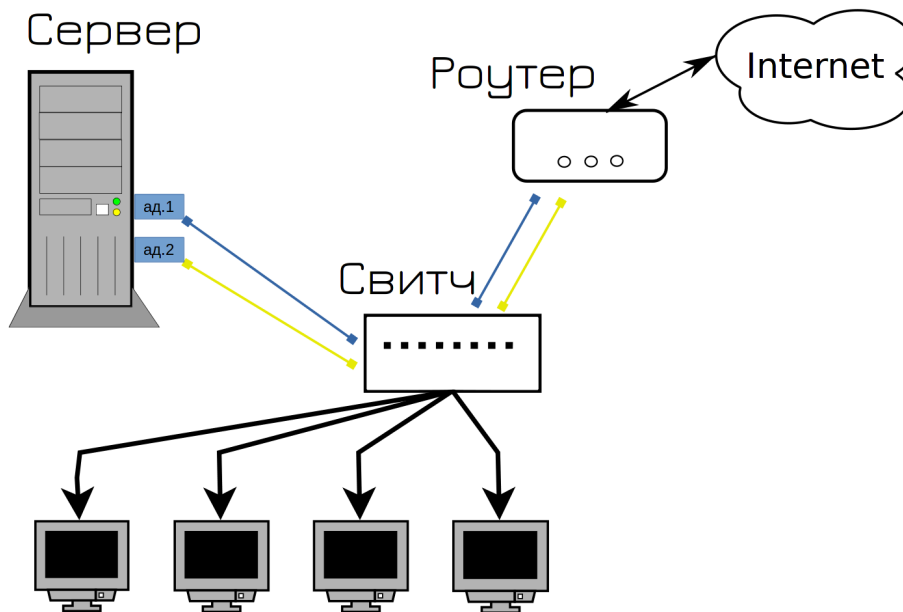


Конечно, мы могли бы просто заменить кабель и надеяться, что это больше не повторится, но мы поступим умнее и попытаемся предотвратить эту ситуацию. И в интернете мы нашли ответ, как это сделать - NIC teaming. Мы можем объединить в системе два порта, один будет работать, а второй ждать. Если с первым портом или кабелем что-то случится, всё перейдёт на второй порт. Так мы соединили сервер со свитчем двумя портами, ну и заодно решили также поступить с роутером, мало ли, и там что-то случится с кабелем. Можно было бы и с пользовательскими компьютерами заморочиться, но у нас портов на свитче не так много, а если у юзера повредится кабель - он может и на wifi пересесть.

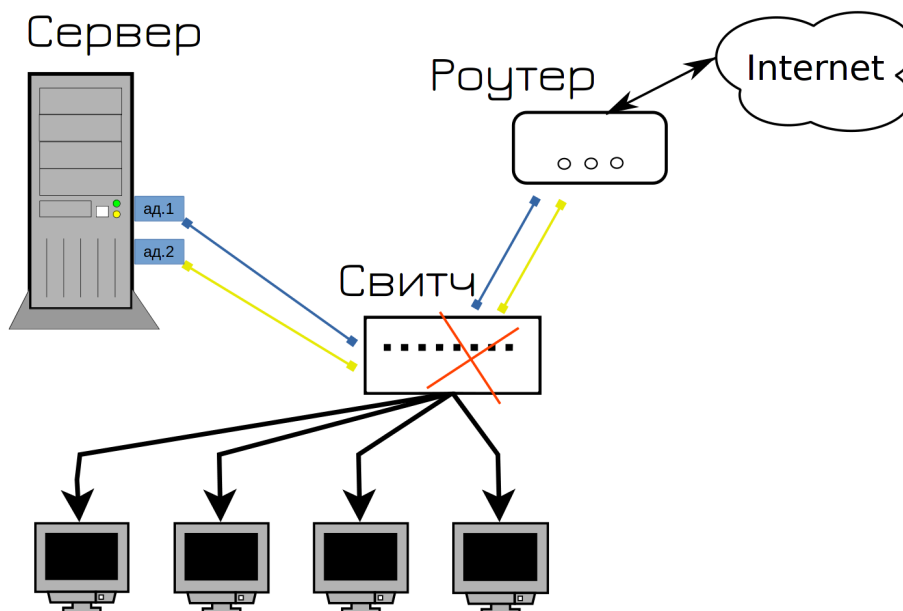


И всё вроде нормально, пока в один из дней на сервере не выходит из строя сетевой адаптер. А так

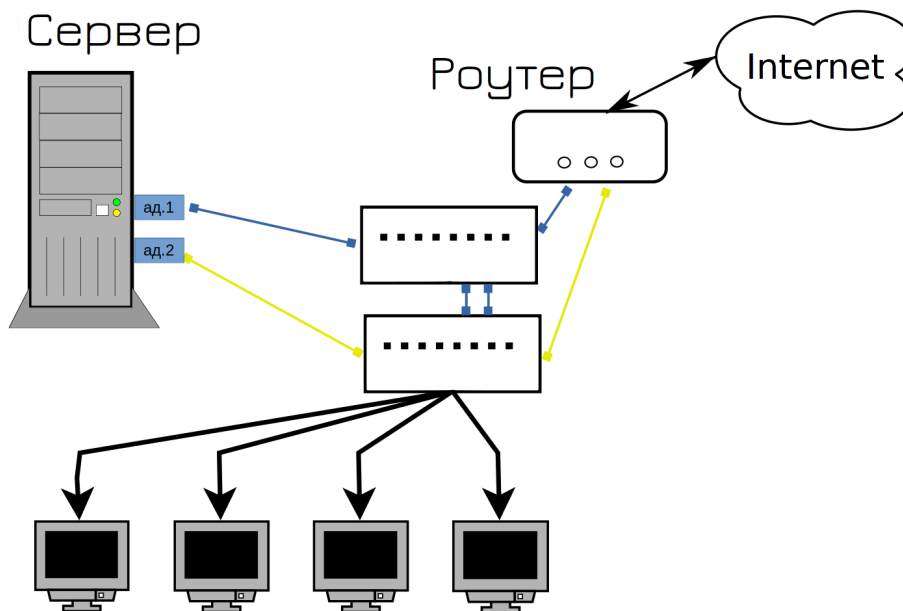
как вышел из строя адаптер, к которому были подключены оба кабеля, опять сервис стал недоступен, опять пользователи недовольны.



И чтобы решить эту проблему и предотвратить в будущем, мы покупаем два сетевых адаптера. Выйдет из строя один - всё автоматом перейдёт на второй, а мы в это время купим новый адаптер и заменим.



Знаете что потом вышло из строя? Свитч. И как мы это решим и предотвратим?



Правильно, купим второй свитч, одни кабели воткнём в первый свитч, вторые во второй, ну и свяжем между собой свитчи. Теперь, какой бы из кабелей не вышел из строя, сеть продолжит работать. Да? Ну почти, у нас сеть ляжет в первую же минуту.

Петля коммутации (Bridging loop, Switching loop) - состояние в сети, при котором происходит бесконечная пересылка фреймов между коммутаторами, подключенными в один и тот же сегмент сети.

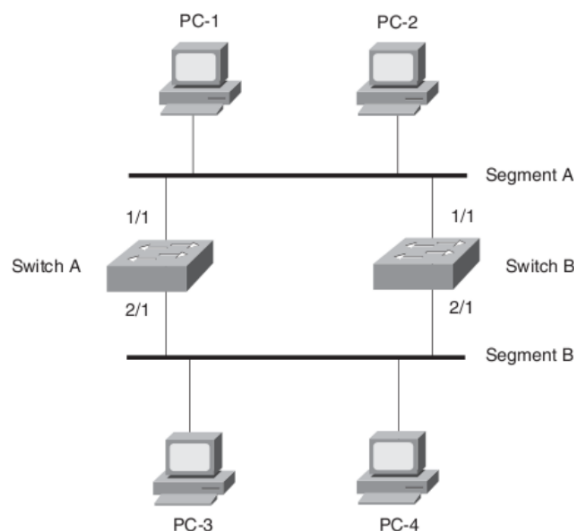
Описание

[\[править\]](#)

Состояние петли формируется следующим образом:

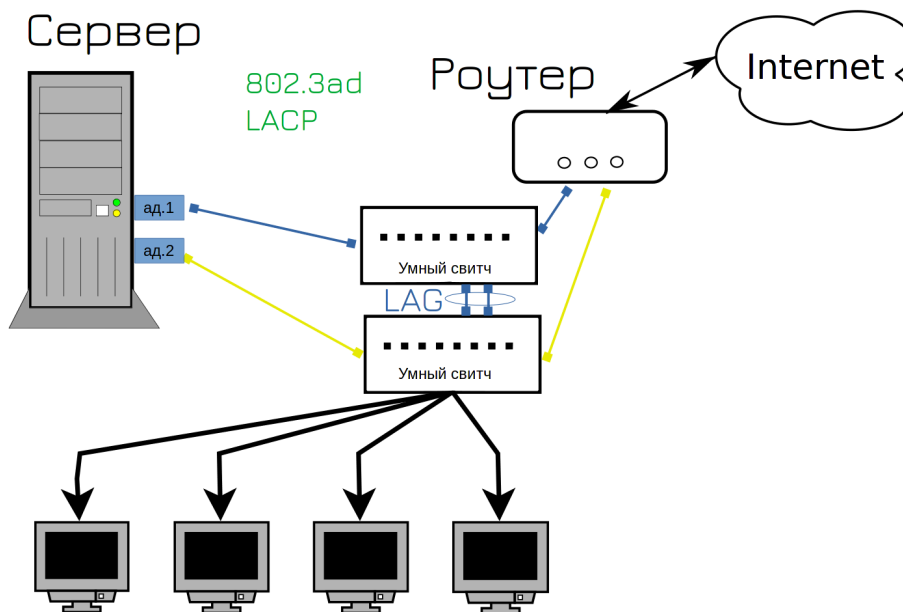
1. Компьютер PC-1 отправляет фрейм компьютеру PC-4;
2. Коммутатор A первым получает фрейм и заносит в таблицу коммутации адрес компьютера PC-1 с исходящим портом 1/1;
3. Так как коммутатор A не знает местоположение(порт подключения) получателя фрейма, он рассылает фрейм через все свои активные порты, кроме порта, из которого этот фрейм был получен;
4. Коммутатор B получает фрейм от компьютера PC-1 и производит аналогичные манипуляции;
5. Компьютер PC-4 получает две копии фрейма - один от коммутатора A, другой от B;
6. Одновременно с этим, копию фрейма от коммутатора A через сегмент B получает коммутатор B. Так как для B полученная копия является "новым" фреймом, то он производит стандартный процесс коммутации фрейма:
 1. Так как MAC-адрес источника идентичен предыдущему фрейму из сегмента A, он удаляет из своей таблицы коммутации запись для компьютера PC-1 с портом 1/1 и добавляет новую запись для PC-1 с портом 2/1;
 2. Рассылает фрейм по всем активным портам, кроме порта, из которого фрейм был получен(2/1);
7. Аналогичная ситуация происходит на коммутаторе A, когда он получает копию фрейма от коммутатора B;

Тем самым происходит бесконечное циркулирование фрейма между сегментами сети.

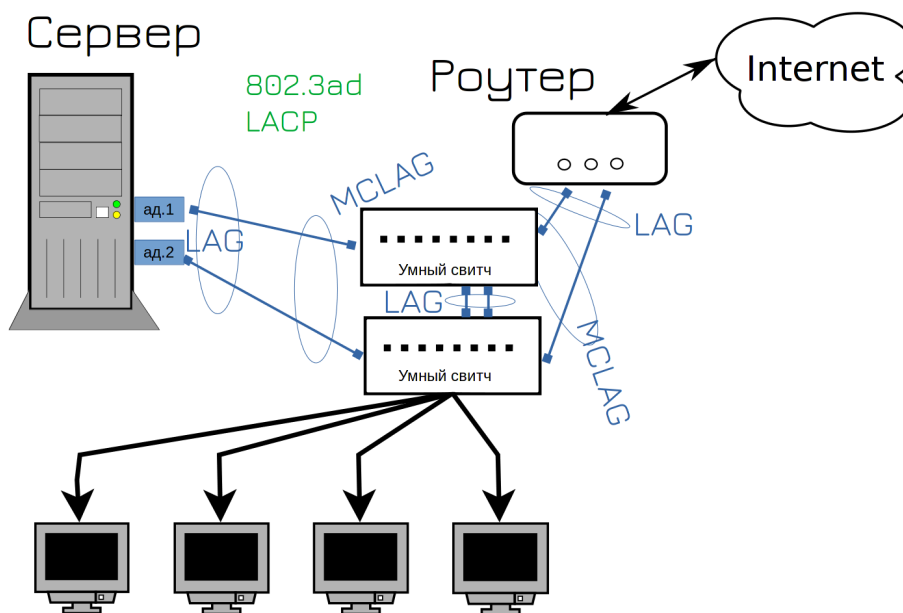


Подключив два свитча между собой двумя кабелями, мы создали **петлю**. Если вкратце, свитчи начнут

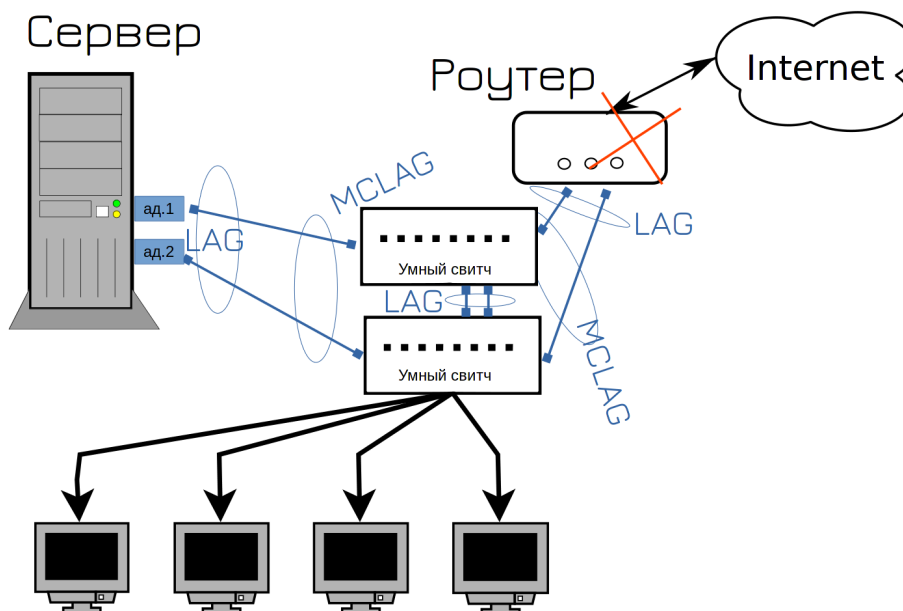
пересылать друг другу одни и те же пакеты, которые будут лавинообразно множиться и в итоге забьют всю сеть, из-за чего вся сеть станет недоступна.



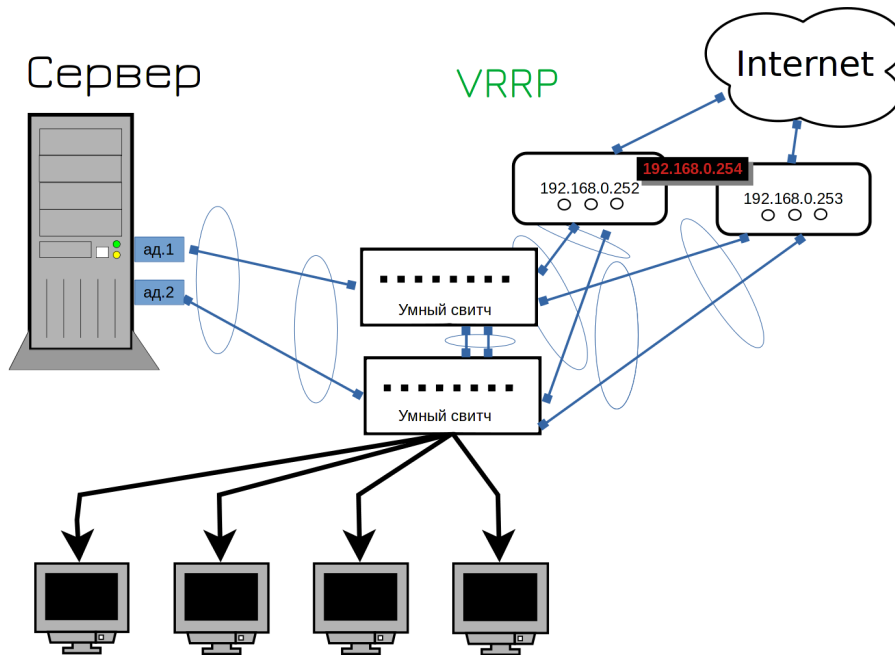
И мы находим решение этой проблемы - нужны свитчи поумнее. Они ещё называются управляемыми свитчами, т.е. managed switch. В них много полезного функционала, который бы нам пригодился. Например - агрегация портов - link aggregation - LAG. Это тот же самый NIC teaming, но со стороны свитча. Но если с тупыми свитчами нам приходилось один из кабелей держать в режиме ожидания, то теперь мы можем использовать оба кабеля. Мало того, что будет отказоустойчиво, ещё и будут использоваться оба кабеля одновременно. Раньше у нас использовался тип агрегации портов «активный и бэкапный», т.е. один из портов был в режиме ожидания, то теперь оба порта будут активны. Этот тип NIC-тиминга называется LACP, его ещё часто называют 802.3ad - под таким названием он описан в стандартах. НО LACP позволяет соединять несколькими портами только два устройства. Т.е. мы по LACP можем соединить два свитча. А как нам LACP использовать между сервером и двумя свитчами, чтобы оба порта были активны, при том что получается 3 устройства: два свитча и один сервер?



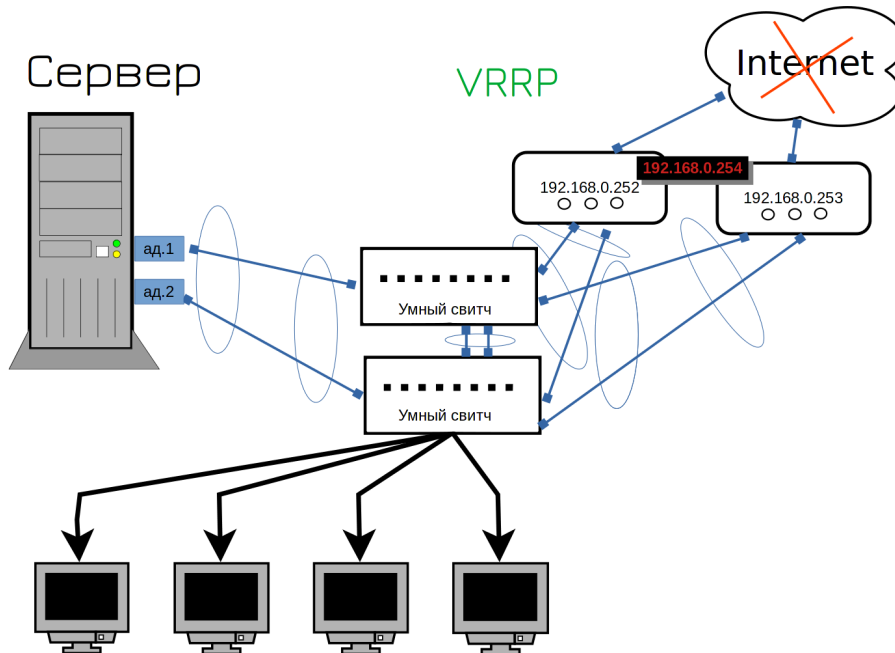
Для этого современные свитчи поддерживают так называемый MCLAG - multi-chassis lag - агрегация портов между несколькими устройствами. Так мы можем объединить в одну группу порты на первом и втором свитче. Сервер будет думать, что с другой стороны одно устройство, один свитч, поэтому на нём получится настроить NIC-teaming с поддержкой LACP. И теперь у нас выросла скорость сети и её отказоустойчивость. Ну теперь то у нас сеть не упадёт, так?



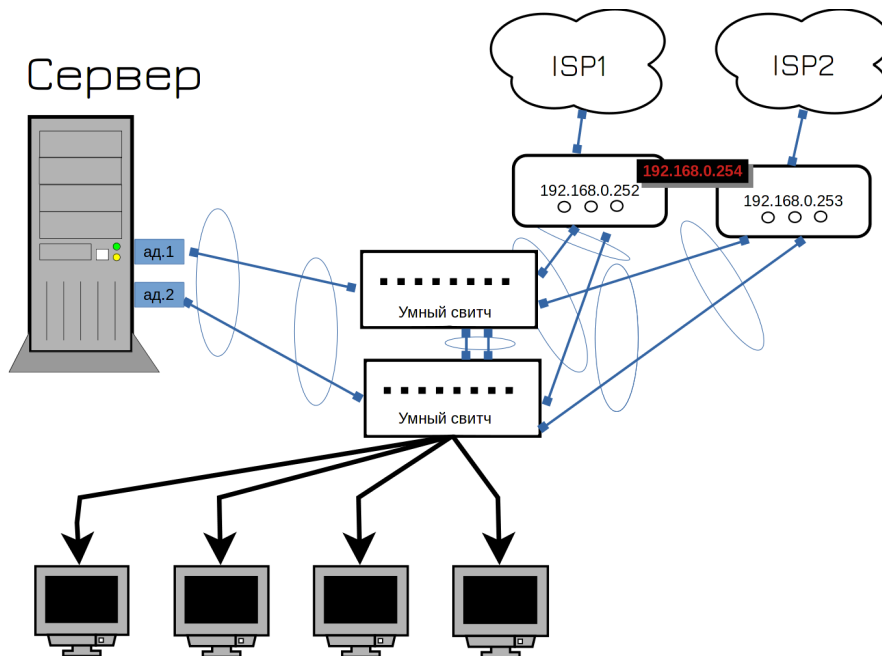
Рано радуетесь. Теперь вышел из строя роутер. Ну вот не везёт нам. Что делать дальше? Правильно, покупать второй роутер.



Окей, мы подключили второй роутер, но ведь у него другой адрес. А наши пользователи для выхода в интернет используют в качестве гейтвея только один адрес. Что, нам теперь бегать и везде менять гейтвей? Не. Как же нам на обоих роутерах поставить один и тот же адрес? Технически, так нельзя делать, у каждого устройства должен быть свой IP адрес. Но есть протокол VRRP, который позволяет разделить IP на два устройства. Сначала этим адресом пользуется одно устройство, но если оно перестанет работать - IP начнёт работать на другом. Скажем, если всё работает, то адрес 254 будет на первом роутере. А если первый роутер выйдет из строя - второй роутер возьмёт себе этот адрес. Ну теперь то точно всё?



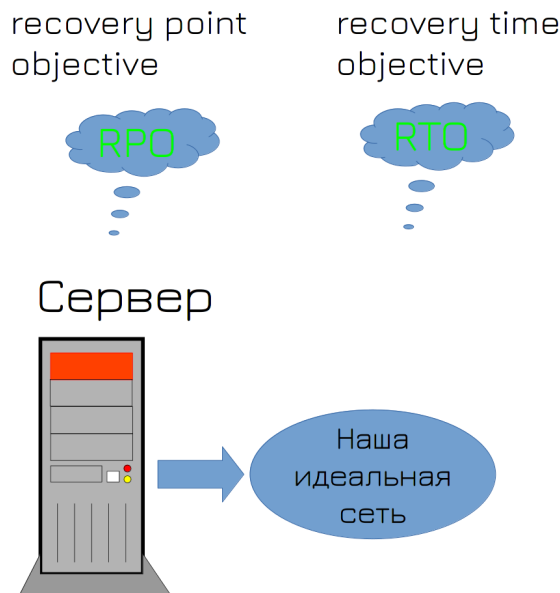
А тут сюрприз от провайдера - интернет лёг. Кто-то что-то копал и повредил линию, интернет будет через пару часов. Ну ё-моё, что нам теперь, два интернет провайдера покупать?



А вы думали нет? Да, нужен второй провайдер. Кстати, ISP - это internet service provider - т.е. интернет провайдер. На этом этапе начинаются танцы с бубном, потому что нередко один провайдер даёт только один кабель, т.е. на два роутера его не подключить. И тут надо либо подключать один роутер к одному провайдеру, либо подключать провайдеров к свитчу - но тогда свитч становится единой точкой отказа. Можно воткнуть один кабель в один свитч, а второй кабель во второй свитч, но тогда при падении свитча переключится и интернет канал. Что обычно не критично, но будет неприятно. Два разных провайдера дают два разных публичных адреса, соответственно, один из ваших адресов станет недоступен. И те, кто по нему подключались, скажем, на ваши вебсервера, потеряют доступ. Ну и это можно обойти, если у DNS провайдера настроить проверку доступности IP, но давайте такие дебри оставим на потом.

Ладно, сеть у нас нормализовалась. На неё теперь можно положиться - вряд ли с ней возможны проблемы. В реальной инфраструктуре свитчей будет побольше, но это просто количество, суть останется та же. Роутеры по разному можно объединять, мы с вами разобрали VRRP, который является стандартным открытым протоколом. А нередко в компаниях два роутера программно работают в одном кластере и переход IP адресов реализован иначе, через внутренний софт этого роутера.

Системы

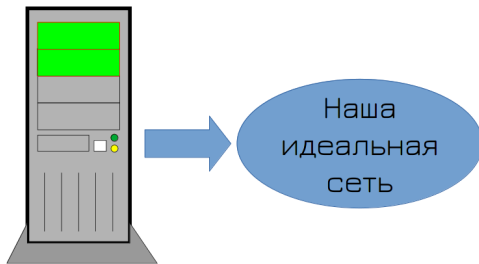


И после того, как вы с трудом объяснили начальству все проблемы и с трудом выбрали бюджет, у вас ложится сервер - вышел из строя жёсткий диск. В этот момент у вас в голове зарождается много мыслей, среди которых есть термины RPO и RTO.

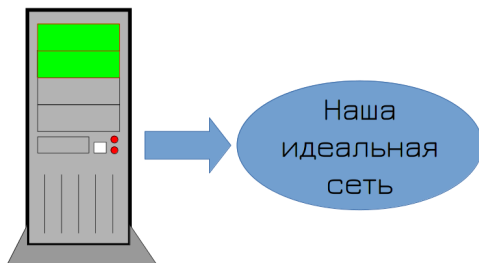
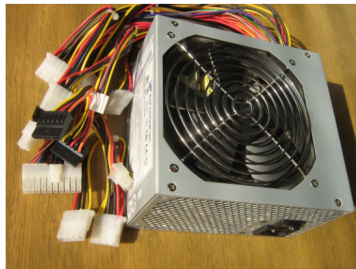
- RPO - recovery point objective - это максимальный период времени, за который могут быть потеряны данные. Никто не хочет терять данные, но и делать бэкап каждую секунду и хранить кучу бэкапов выйдет нереально дорого. Поэтому вы с начальством соглашаетесь, что если что-то пойдёт не так, вы готовы потерять данные, например, за два последних часа. Это ваш RPO. Соответственно, бэкапы должны делаться примерно с такой периодичностью.
- RTO - recovery time objective - промежуток времени, в течение которого система может оставаться недоступной. Т.е. сколько времени вам нужно будет, чтобы вернуть всё как было? Предположим, час, вы за это время успеете поставить новый диск, восстановить на него бэкап и всё запустить.

RAID

Сервер



Но чтобы избежать повторения ситуации, вы решаете приобрести второй диск и настроить RAID. У вас два сетевых адаптера, два диска, что ещё может пойти не так?

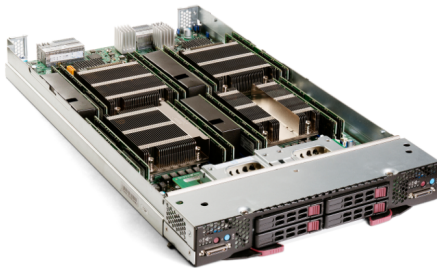


Например, может сгореть блок питания. Но второй блок питания на комп не поставишь.



Наша
идеальная
сеть

Зато мы наконец-то можем избавиться от этого хлама и купить настоящий сервер. У него есть рейд контроллер, процессор и оперативка, рассчитанные на постоянную работу и много всего полезного, например, отслеживание состояния всего оборудования. И целых два блока питания! Теперь то что может пойти не так?



Наша
идеальная
сеть

Почему бы не выйти из строя операционной системе? Или не сгореть материнке? Как насчёт второй материнки? А вот никак.

Кластер серверов



Что ж, придётся раскошелиться. Нужно купить второй сервер и сделать кластер. Кластер можно сделать на основе гипервизора, чтобы всё было в виртуалках и они запускались на разных серверах. Но делать кластер из двух серверов очень плохая затея, потому что если сервера потеряют связь друг с другом, то каждый из них посчитает, что упал второй сервер. И каждый попытается запустить виртуалки. Тогда получится каша и с большой вероятностью всё приведёт к порче данных.

Кластер серверов



Поэтому кластер нужно делать из 3 серверов. Если выйдет из строя один сервер - другие два продолжают работать. Если какой-то сервер увидит, что потерял связь с двумя - то он посчитает, что проблема в нём и не станет запускать виртуалки.

Окей, предположим, помер один из серверов. Как запустить виртуалку с того сервера на другом? Файлы виртуалки то лежат на сервере, который недоступен. Что, постоянно бэкапить виртуалки с одного сервера на другой? Тогда при падении сервера мы будем терять часть данных, а мы этого не хотели бы. Нам нужно данные держать как-то централизованно, чтобы они были одновременно доступны на всех трёх серверах.



Для этого есть готовые решения - системы хранения данных, часто называемые стораджами. Это эдакий сервер, где куча дисков, который нужен для раздачи пространства на сервера. Сервера подключаются к специальному свитчу, называемому SAN-свитчем, а он подключен к стораджу. На нём вы создаёте виртуальные диски, а сервера видят эти диски по сети. Только это не обычная сеть, которую мы изучали, а именно стораджовая сеть. Т.е. одновременно у вас работает и обычная сеть, где ходят сетевые пакеты, и также есть пара адаптеров оптических, которые выделены для доступа к дискам по сети. Такая сеть называется SAN - storage area network.

Сторадж, сам по себе, имеет два контроллера - т.е. что-то вроде двух материнок, поэтому для отказоустойчивости нет необходимости покупать второй сторадж. Внутри стораджа есть и рейд, и дедупликация, и снапшоты, и куча других технологий, полезных для хранения данных.

Засчёт того, что все три сервера будут видеть одни и те же виртуальные диски, виртуалки будут спокойно перемещаться с одного сервера на другой.

Такая инфраструктура, когда есть сервера и системы хранения данных, называется классической. Есть и другие, но это отдельная тема.



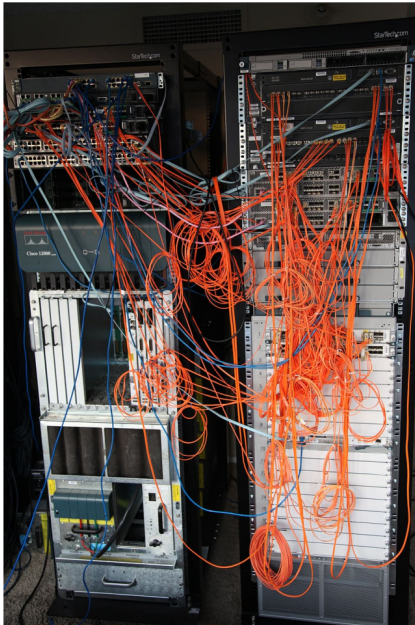
И так, у нас накопились сервера, системы хранения данных и свитчи. Всё это оборудование сильно греется, выделяет много тепла и очень чувствительно к окружению. Стандартная температура, при которой такое оборудование чувствует себя в порядке - где-то 20 градусов. Скажем, при 30-35 градусах могут начаться проблемы и даже что-то испортиться. Также влажность должна быть где-то 40%. Такое окружение в обычных условиях организовать сложно, да и шум от серверов не даст вам спокойно работать. Поэтому всё это оборудование нужно хранить в специальных комнатах с подходящими условиями - в серверных комнатах.



Ну и чтобы всё это оборудование не ставить на пол, есть специальные металлические шкафы, назы-

ваемые рэками. Они стандартизированы - в основном высота и ширина у многих шкафов одинаковая. В рэках есть промежутки пространств, называемые Unit-ами, или просто U, по русски они звучат как «монтажная единица». И у неё высота где-то 4,5 сантиметра. У серверов, свитчей и прочего оборудования, как правило, высота соответствует этим юнитам. Скажем, обычно свитчи занимают 1U, сервера - 1, 2, иногда 4U, в зависимости от мощностей и древности этих серверов.

Есть определённые требования к серверным - откуда должен поступать воздух, как он должен циркулировать, должны быть специальные кондиционеры для регулирования температуры и влажности в серверной, специальные системы мониторинга окружения, чтобы следить за температурой, влажностью и другими параметрами, а также специальные системы пожаротушения. В общем, к серверным комнатам огромное количество требований.



Что, думаете потратив столько денег и собрав такую серверную, вы избавились от единой точки отказа? Ха, как бы не так. В один день у вас просто вырубает электричество. Это ж не ваша вина, что можно с этим поделать? Что-то я не слышал, чтобы у гугла свет отрубили, поэтому он стал недоступен.



Так что берём деньги в руки и покупаем UPS-ы. Но домашние не подойдут, вы в курсе сколько электричества жрёт всё это оборудование? Нужны UPS-ы помощнее. Если у вас один-два шкафа оборудования, то сойдут UPS-ы для рэков. Да и на один UPS рассчитывать не стоит, поэтому нужно минимум два. А бывает, что оборудования много, да и вы не хотите, чтобы по всему зданию пропадал свет. Поэтому можно под UPS-ы хоть отдельную комнату выделить. Но это уже не совсем касается IT, нам важно, чтобы сервера работали, чтобы сервис был доступен.

А что, если электричество пропадёт надолго? Разбиваем последнюю копилку и проводим вторую линию электричества. Один блок питания UPS-ов сажаем на одну линию, второй блок - на вторую. Ляжет электричество с одной стороны - будет работать на другой.



Что, думаете на этом всё? А если серверную затопит? Если по всему городу пропадёт электричество? Если на город упадёт метеорит? Что, из-за этого бизнес должен перестать работать? Как бы не так.



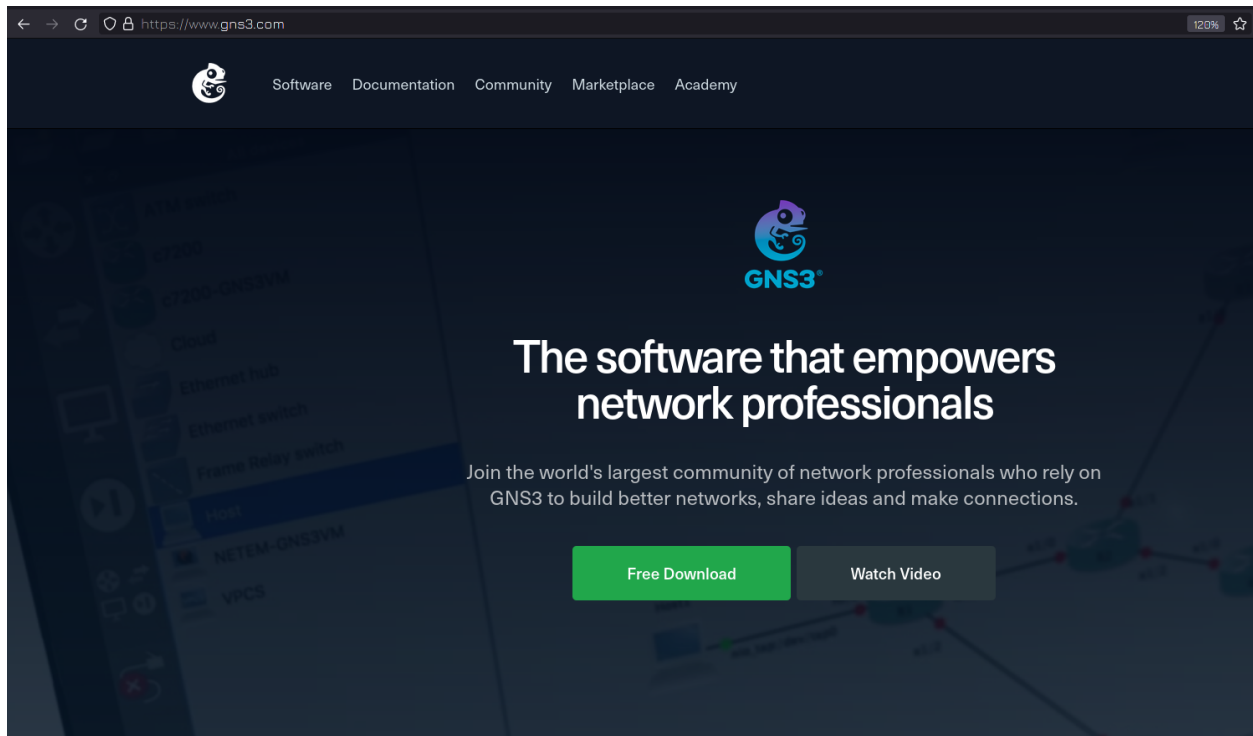
Строим копию нашего датацентра в другом здании, другом городе, другой стране, другом континенте, другой планете. Чтобы ни случилось, наша инфраструктура должна работать. Вы думаете Илон Маск осваивает Марс для людей? Поверьте, первое что там появится - датацентр.

Теперь вы понимаете, почему SPOF - главный термин в IT и почему IT такое дорогое удовольствие? Есть много других нюансов, которые я не упомянул, но со временем вы многому научитесь. Да, не

каждая компания строит запасной датацентр, но вы растёте как специалист, когда вместо одного кабеля подключаете два.

2.2 02. Подготовка лабораторной

2.2.1 02. Подготовка лабораторной



Теперь, когда мы знаем, как должна выглядеть инфраструктура, давайте попытаемся её построить. Ну и самое главное - разберём программную часть этого всего. Нам понадобится много виртуалок и мы будем их связывать с помощью виртуальной сети. Всё это можно сделать и в VirtualBox-е, но это не будет наглядно. Поэтому нам нужна программа **GNS3**.

← → ↻ 🔍 https://docs.gns3.com/docs/ 130% ☆

GNS3 Documentation Appliances Commun

Getting Started ▾

- What is GNS3?
- Installation >
- Setup wizard for local server
- Setup wizard for GNS3 VM
- Your first GNS3 topology
- Your first Cisco topology

Using GNS3 >

Emulators >

Contribute >

How-to Guides >

Troubleshooting & FAQ >

This is the client part of GNS3 and is graphical user interface (GUI). You install the all-in-one software on your local PC (Windows, MAC, Linux) and create your topologies using this software. This is what you usually see show in screenshots such as the following:

Мы всё ещё будем использовать VirtualBox, чтобы работали виртуалки с линуксами, но за сеть и за наглядность будет отвечать GNS3. Сразу предупрежу, что она немного требовательная и рекомендуемый минимум по оперативке где-то 8 гигабайт. На самом деле, если ваша система использует мало памяти, то можно обойтись меньшей памятью. Я буду строить большую сетку с кучей виртуалок, вам же не обязательно так делать. Для разбора большинства тем хватит 3-4 одновременно запущенных виртуалок.

GNS3 Documentation Appliances Community 🔍

Getting Started ▾

- What is GNS3?
- Installation** ▾
 - Windows
 - Mac
 - Linux**
 - ESXi
 - Remote Server
 - Download the GNS3 VM
 - Download for Windows 32-bit
 - One GNS3 server, multiple clients

GNS3 Linux Install

Installation from Packages

Ubuntu-based distributions (64-bit only)

These instructions are for Ubuntu and all distributions based on it (like Linux Mint).

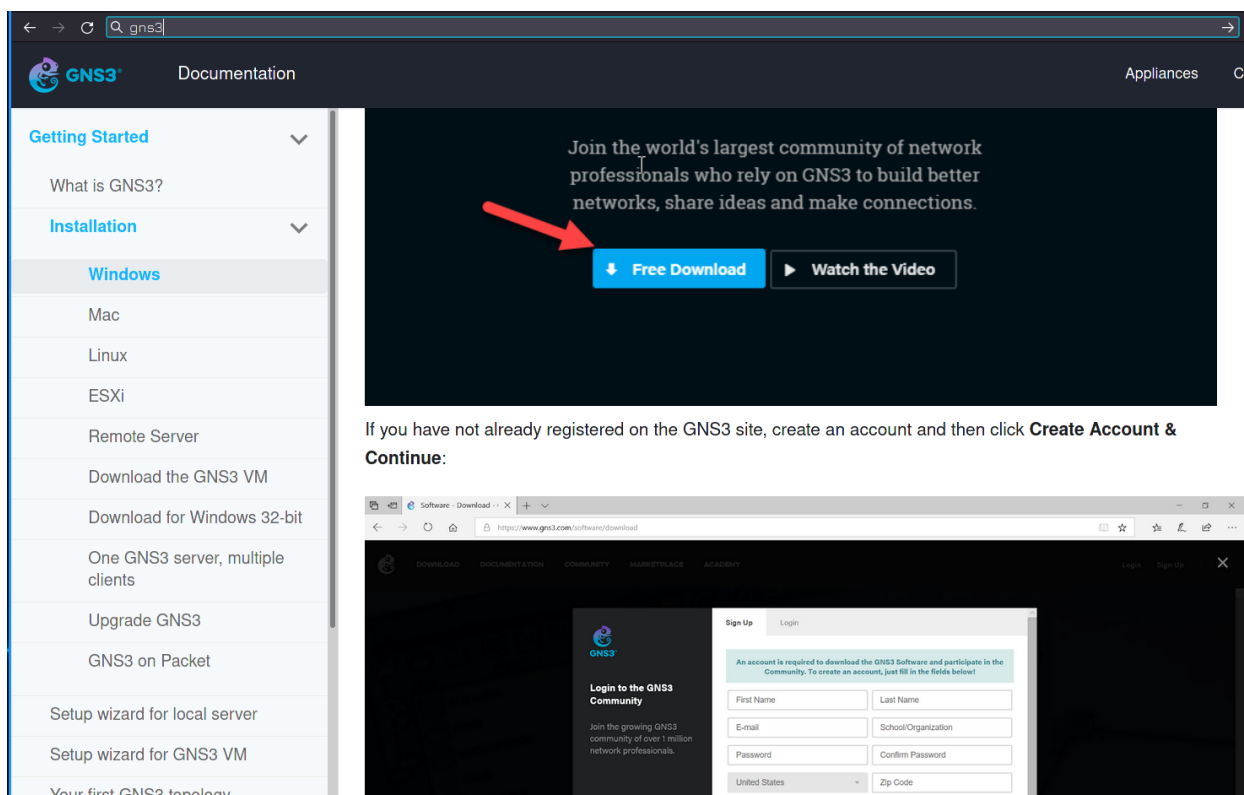
```
sudo add-apt-repository ppa:gns3/ppa
sudo apt update
sudo apt install gns3-gui gns3-server
```

(when prompted whether non-root users should be allowed to use wireshark and ubridge, select 'Yes' both times)

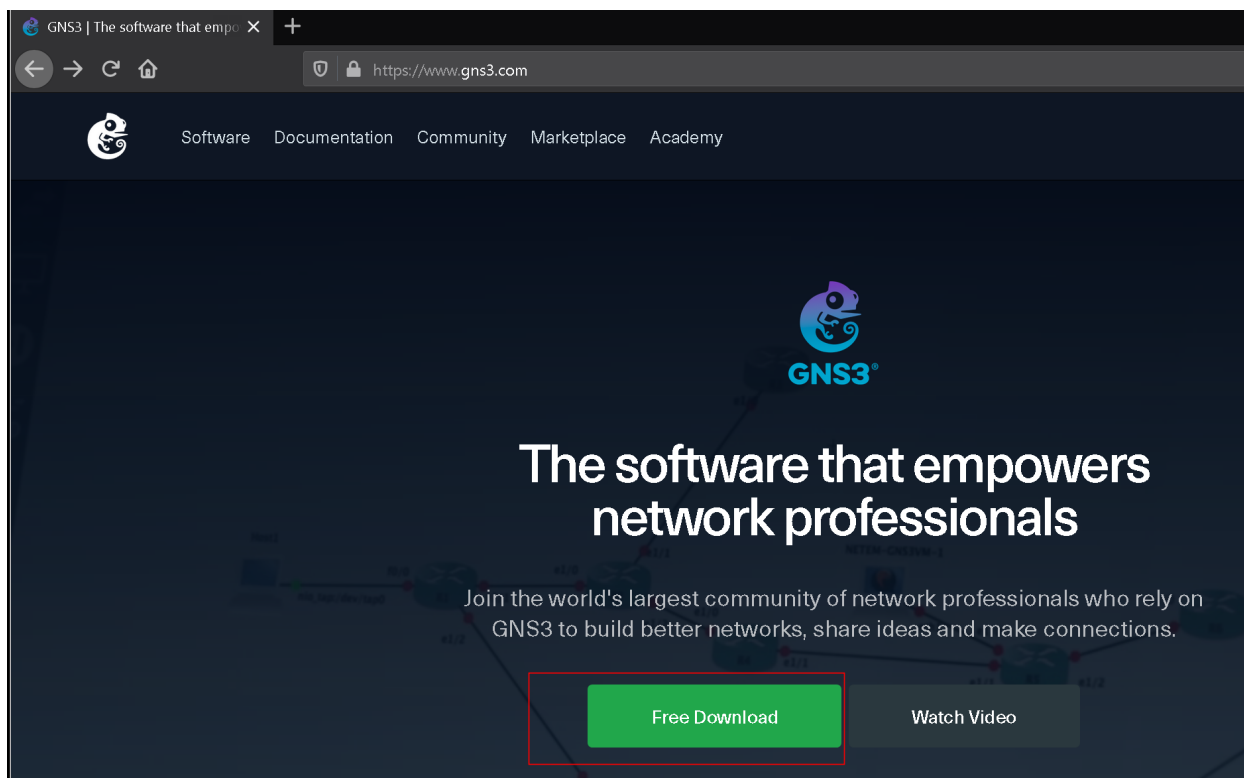
[Installation from Packages](#)

- Ubuntu-based distributions (64-bit only)
- Debian-based distributions (64-bit only)

Если у вас основная система GNU/Linux, то вам для установки нужно будет добавить репозиторий, установить несколько пакетов и добавить вашего пользователя в несколько групп. Всё зависит от вашего дистрибутива, для убунту и дебиан инструкцию можно найти в документации на официальном сайте, а для других дистрибутивов есть статьи в интернете.

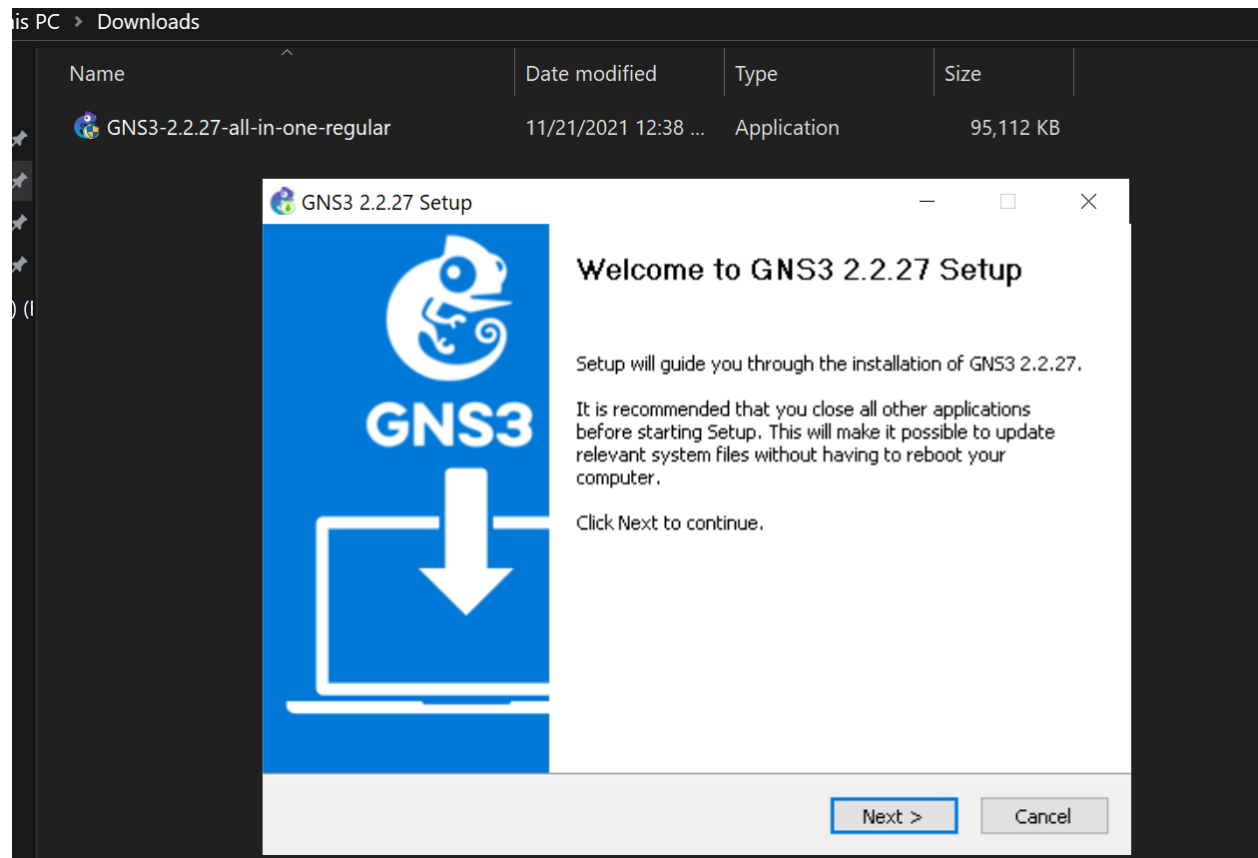


Что касается Windows, в официальной документации также есть инструкция, но там много лишнего, поэтому я покажу сам.

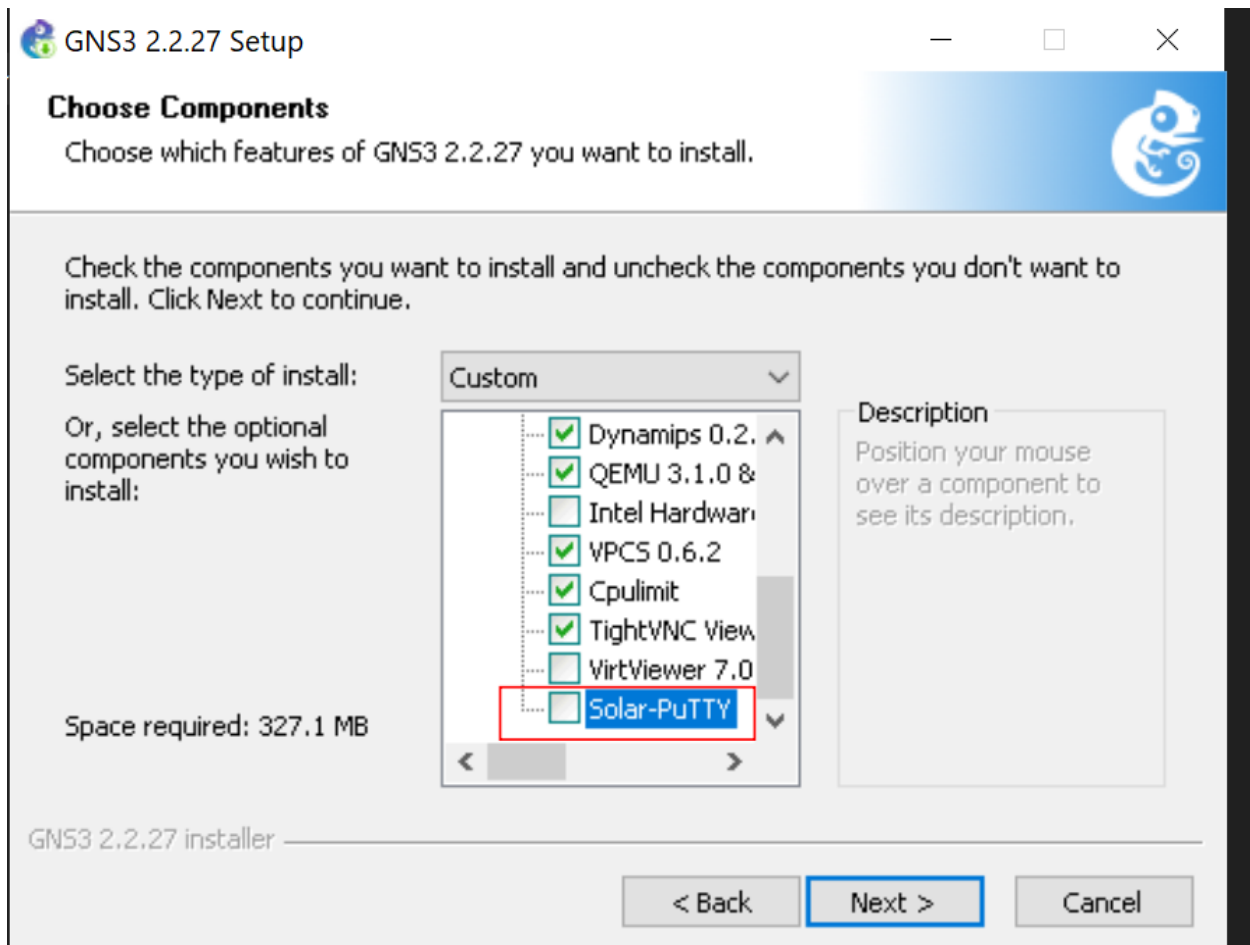


На официальном сайте GNS3 нажимаете Free Download, после чего вам нужно будет зарегистриро-

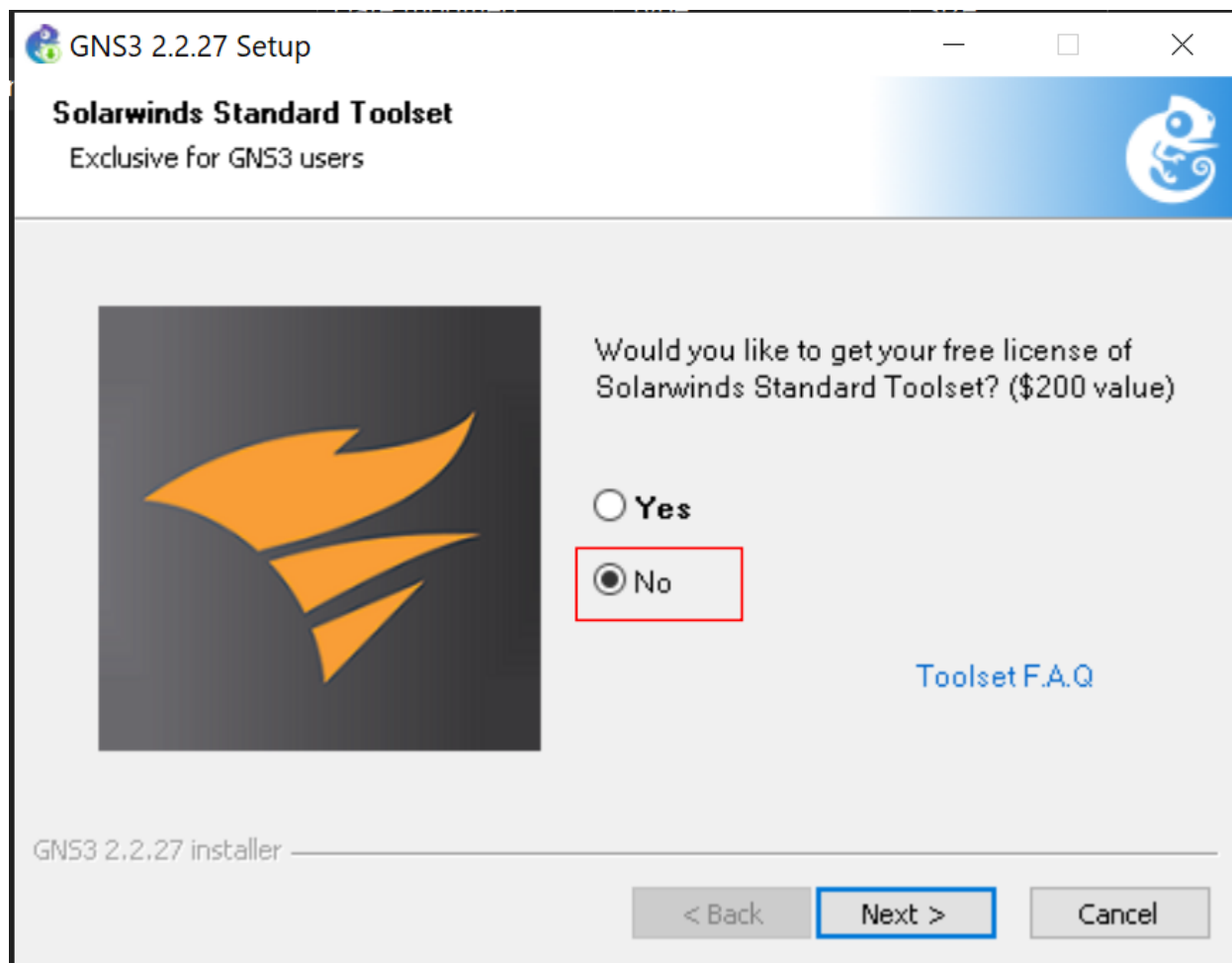
ваться, чтобы скачать.



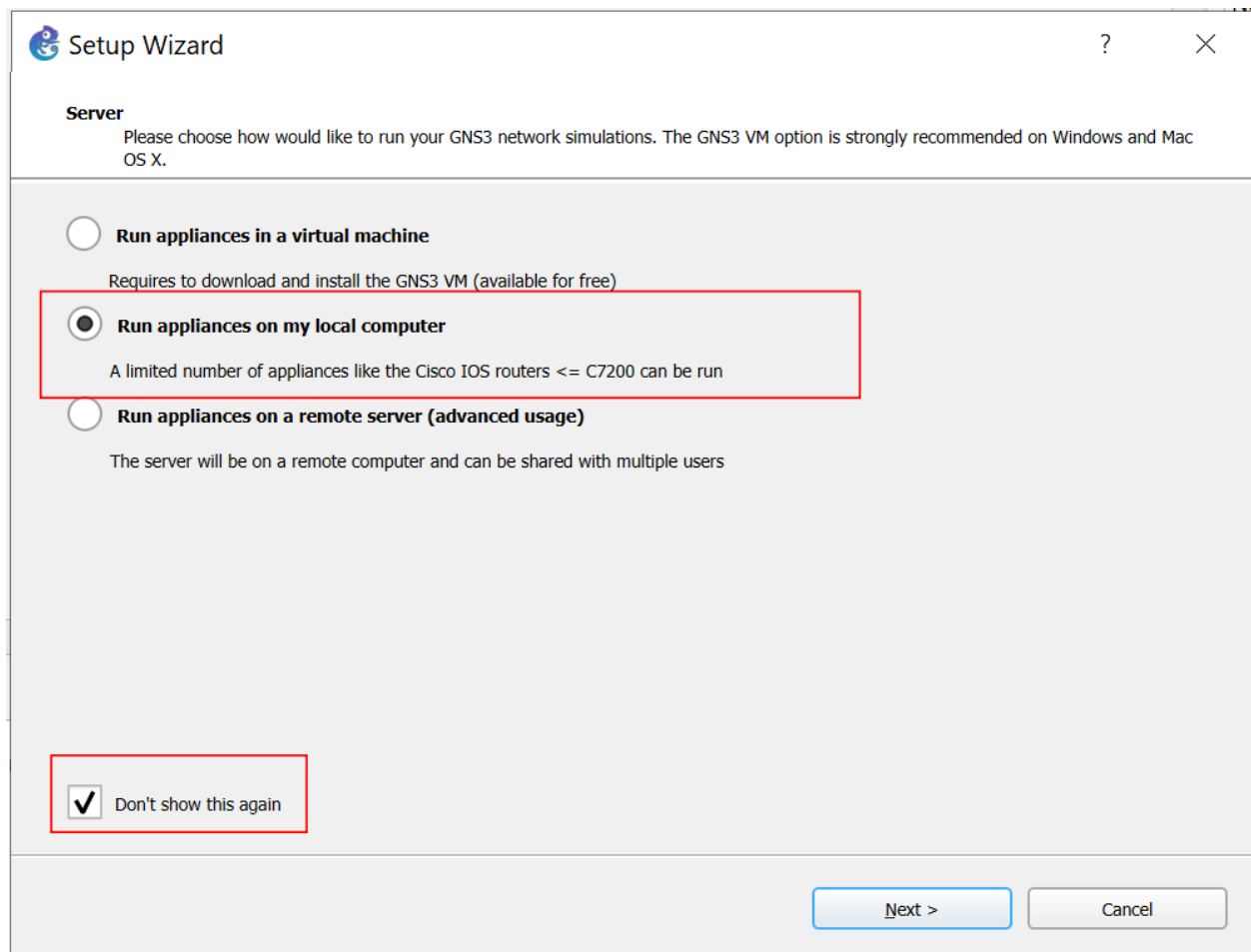
Дальше запускаете установщик, нажимаете Next, соглашаетесь с лицензией и ещё раз «Next».



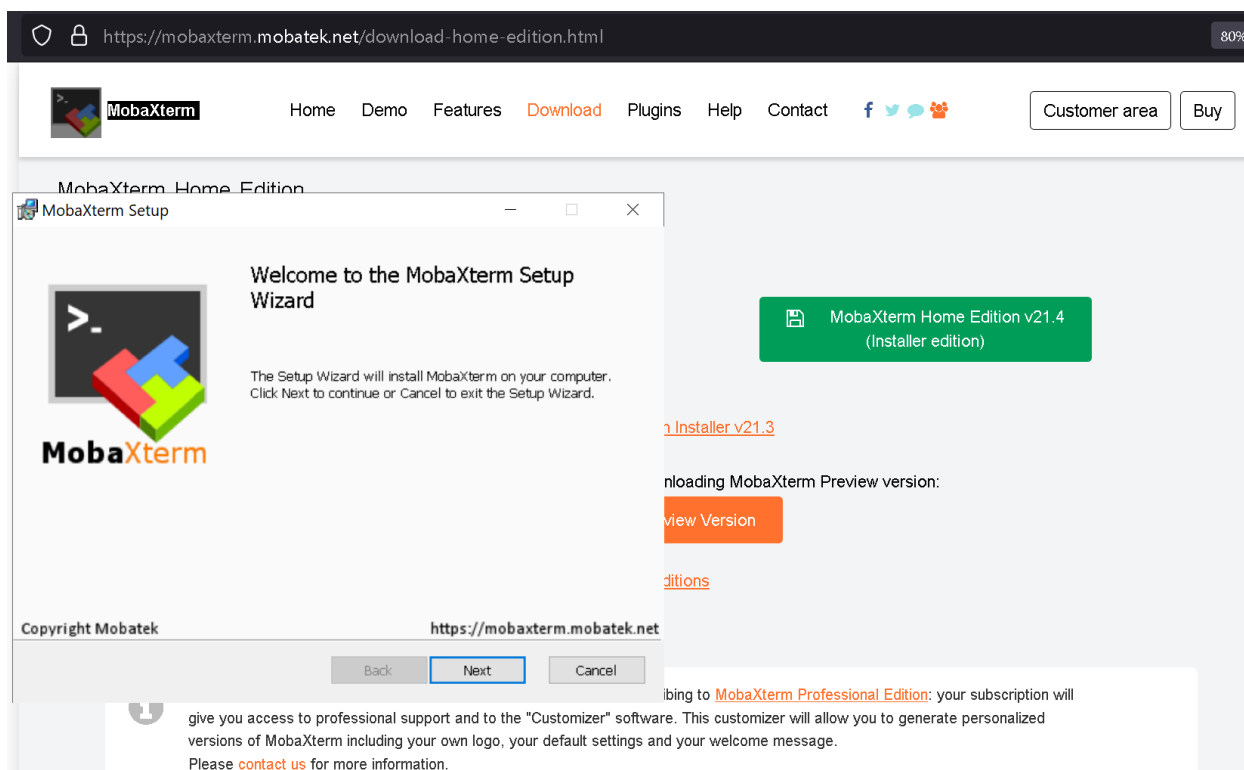
Во вкладке выбора компонентов раскрываете Tools и убираете галочку с Solar-PuTTY. Это ssh клиент, чтобы мы могли подключаться к оборудованию. Мы же всё равно будем использовать другую утилиту - MobaXterm. После чего нажимаем Next.



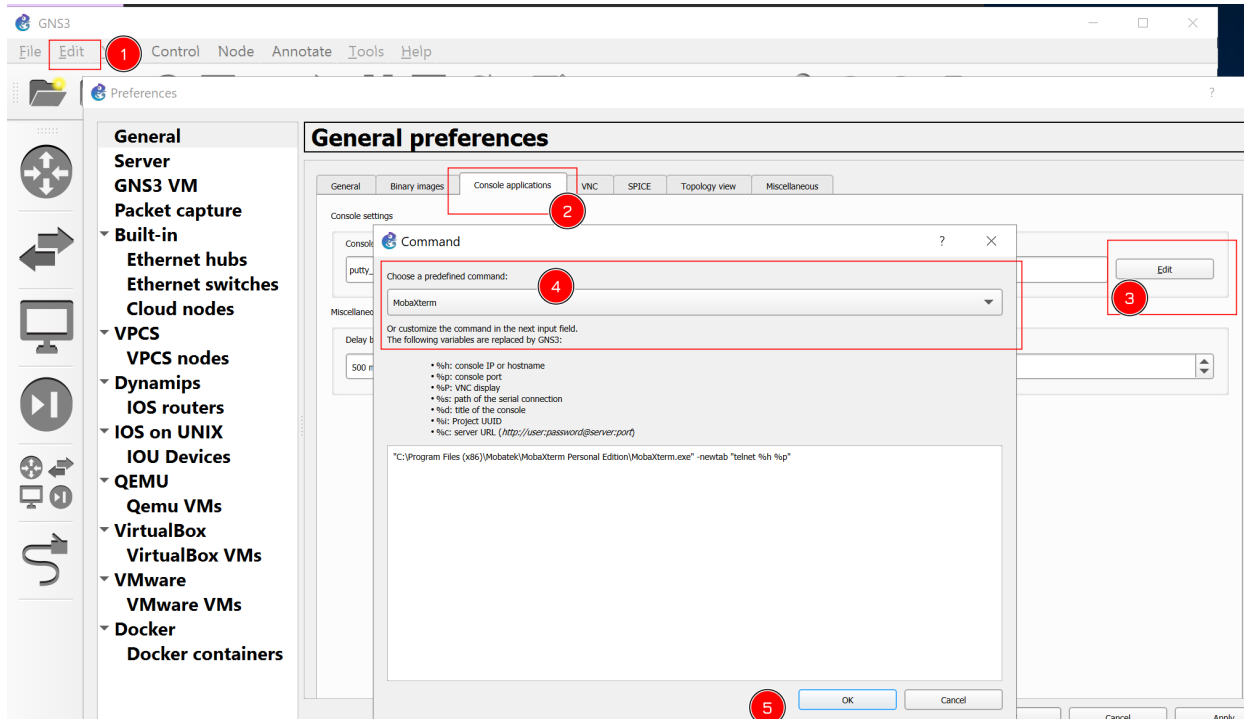
Пока опять не появится окно с предложением бесплатно получить лицензию для SolarWinds Standard Toolset - это набор утилит для мониторинга сети. Нам это не нужно, мы будем делать свой мониторинг сети, поэтому нажимаем No. На этом завершаем установку и запускаем GNS3.



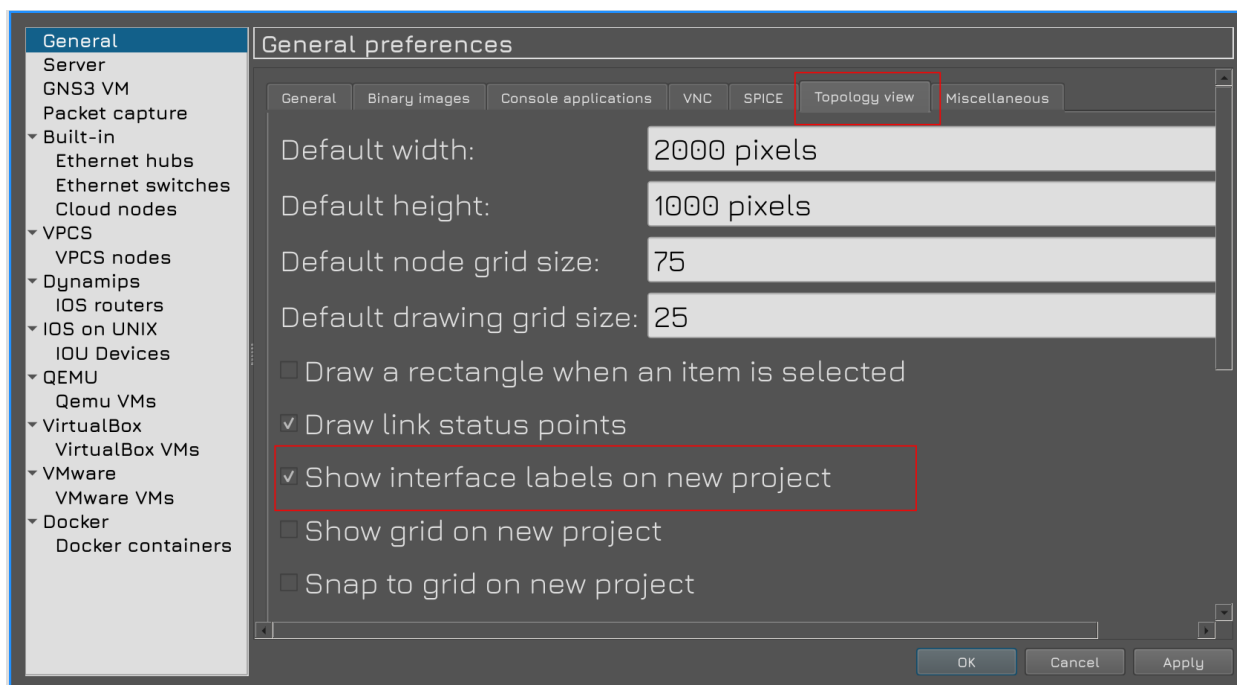
Нас встретит окно с начальными настройками и спросит, где мы будем запускать виртуальное оборудование. У GNS3 есть своя готовая виртуалка с линуксом, но мы всё равно все наши системы будем запускать через VirtualBox, и, чтобы не ставить лишнюю виртуалку, выбираем второй пункт и ставим галочку на «Больше не показывать». Затем Next, Next и finish.



Также скачиваем и ставим **MobaXterm**. С помощью неё мы будем подключаться к нашим виртуалкам по SSH. Установка довольно простая - просто next next и всё.



После чего в GNS3 заходите в Edit - Preferences - General - Console Applications - Edit и выбираете MobaXterm. Затем нажимаете OK и Apply.



Также во вкладке Topology ставите галочку на «Show interface labels on new project». Это будет показывать нам названия интерфейсов, подключенных к сети, что будет полезно в будущем.

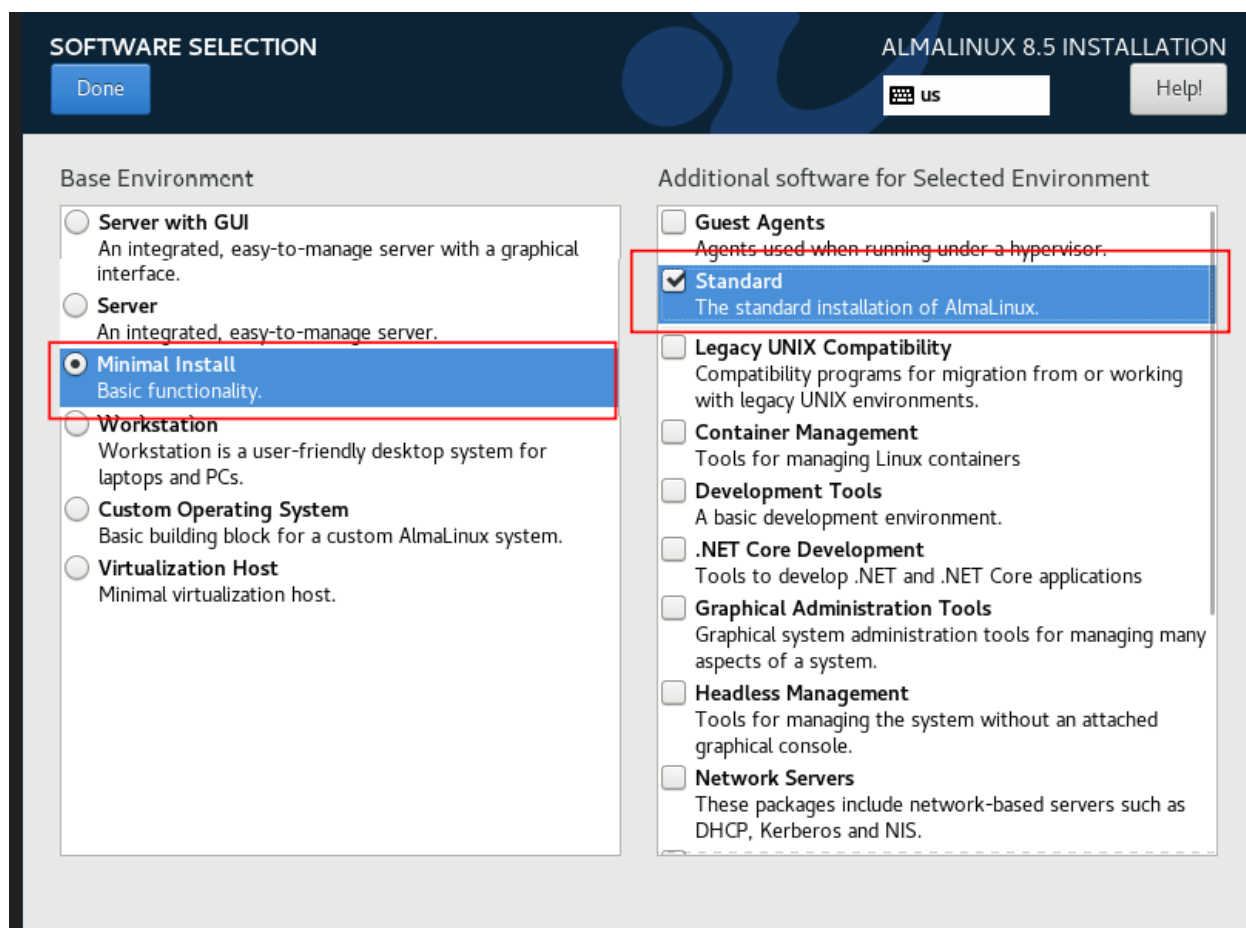
← → ↻ 🔍 mirrorhosting.com.tr/almalinux/8.5/isos/x86_64/

Index of /almalinux/8.5/isos/x86_64

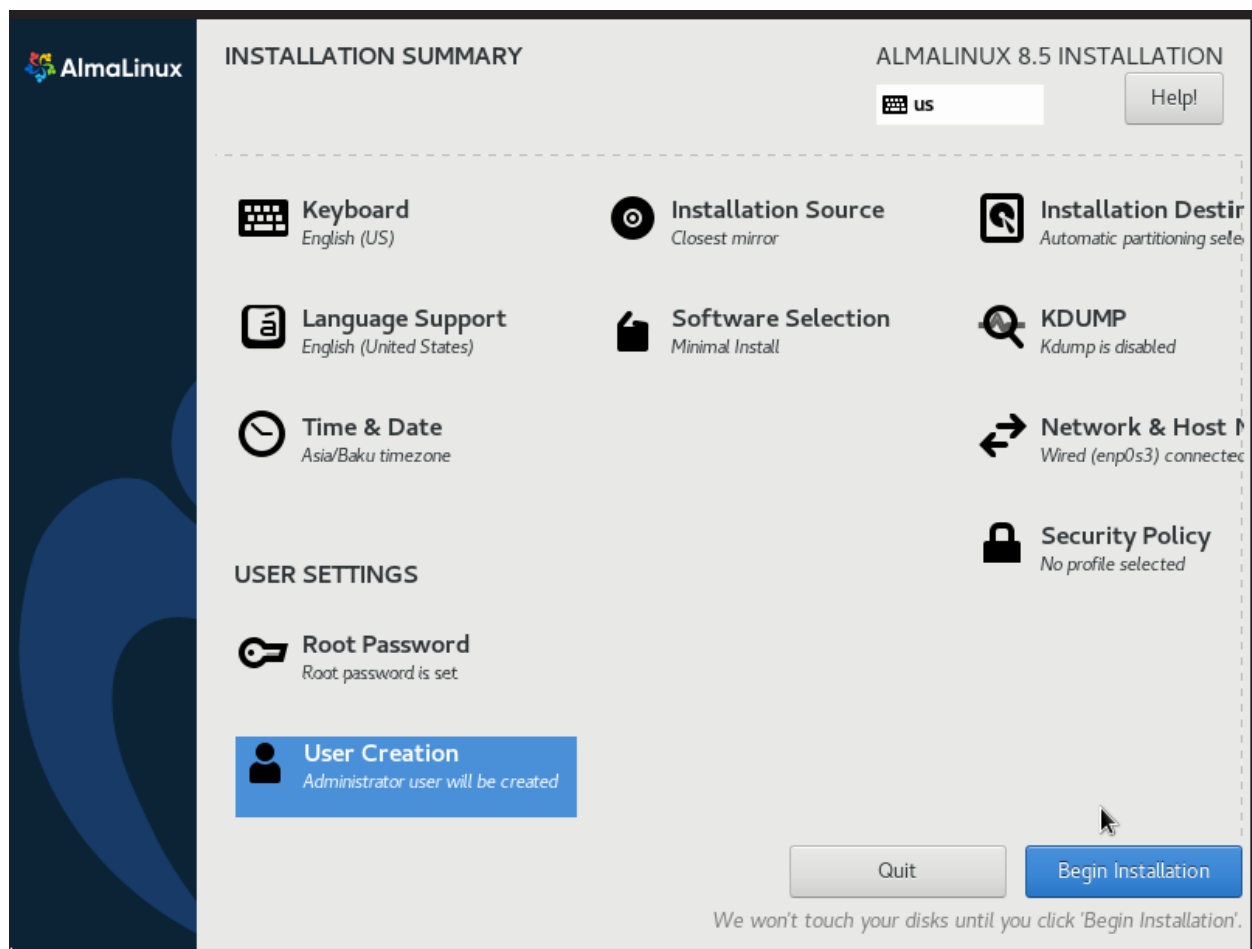
	Name	Last modified	Size	Description
📁	Parent Directory	-		
📄	AlmaLinux-8.5-x86_64-boot.iso	11-Nov-2021 19:35	738M	
📄	AlmaLinux-8.5-x86_64-boot.iso.manifest	11-Nov-2021 19:37	626	
📄	AlmaLinux-8.5-x86_64-dvd.iso	11-Nov-2021 19:42	9.9G	
📄	AlmaLinux-8.5-x86_64-dvd.iso.manifest	11-Nov-2021 19:42	501K	
📄	AlmaLinux-8.5-x86_64-minimal.iso	11-Nov-2021 19:16	1.9G	
📄	AlmaLinux-8.5-x86_64-minimal.iso.manifest	11-Nov-2021 19:16	94K	
📄	AlmaLinux-8.5-x86_64.torrent	12-Nov-2021 12:21	64K	
📄	CHECKSUM	11-Nov-2021 20:28	1.2K	

Apache/2.2.15 (CentOS) Server at mirrorhosting.com.tr Port 80

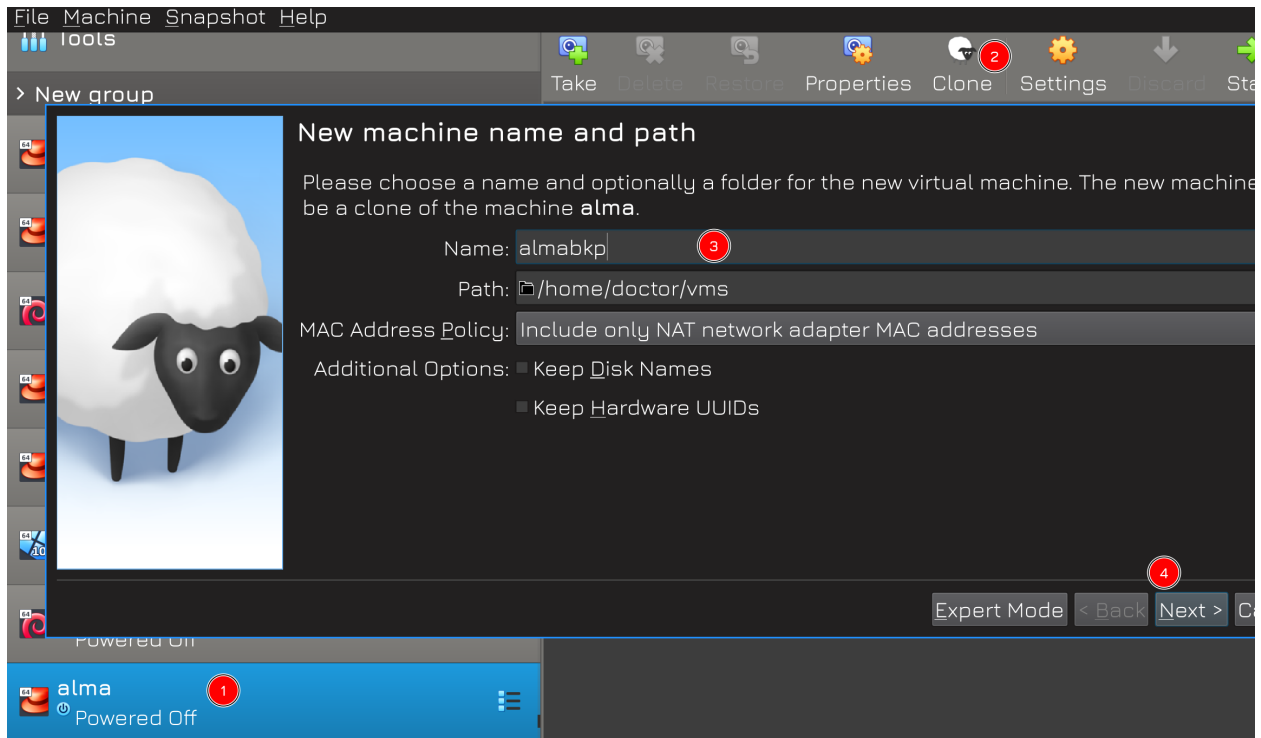
Также, из-за того, что мы часто будем создавать и удалять виртуалки, будет неудобно их постоянно регистрировать, поэтому вместо RHEL используем AlmaLinux - это один из дистрибутивов, который появился на замену Centos. Скачать его можно с официального сайта по [ссылке](#). Архитектура x86_64, а версию берите последнюю. Я не буду разбирать установку, она ничем не отличается от установки RHEL или Centos. Вы можете использовать любой другой дистрибутив, но желательно всё таки Alma, чтобы избежать путаницы в названиях пакетов и прочих мелочах.



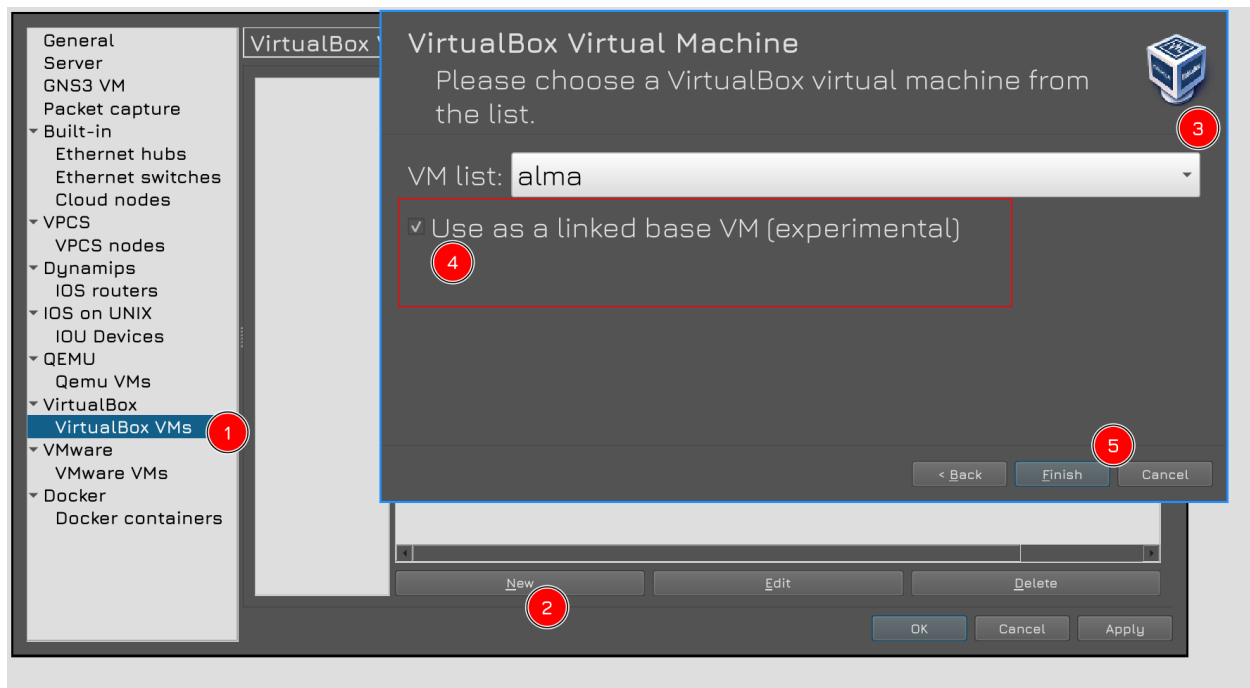
Что касается параметров виртуалки, назовите виртуалку просто «alma». Для установки можете выделить 1 гигабайт оперативки и 16 гигабайт дискового пространства. Среди пакетов выберите минимальную установку со стандартными пакетами.



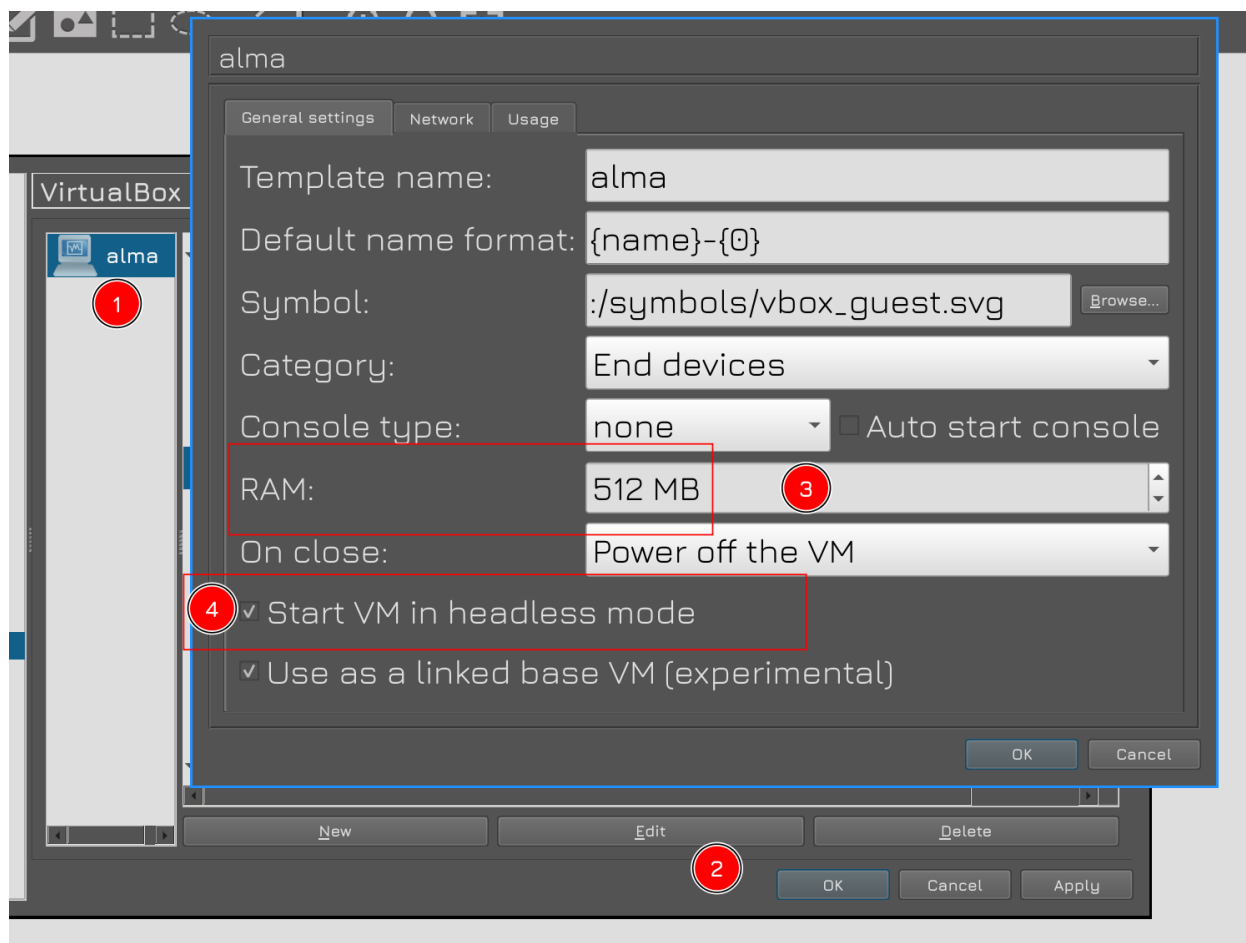
Всё остальное на ваше усмотрение.



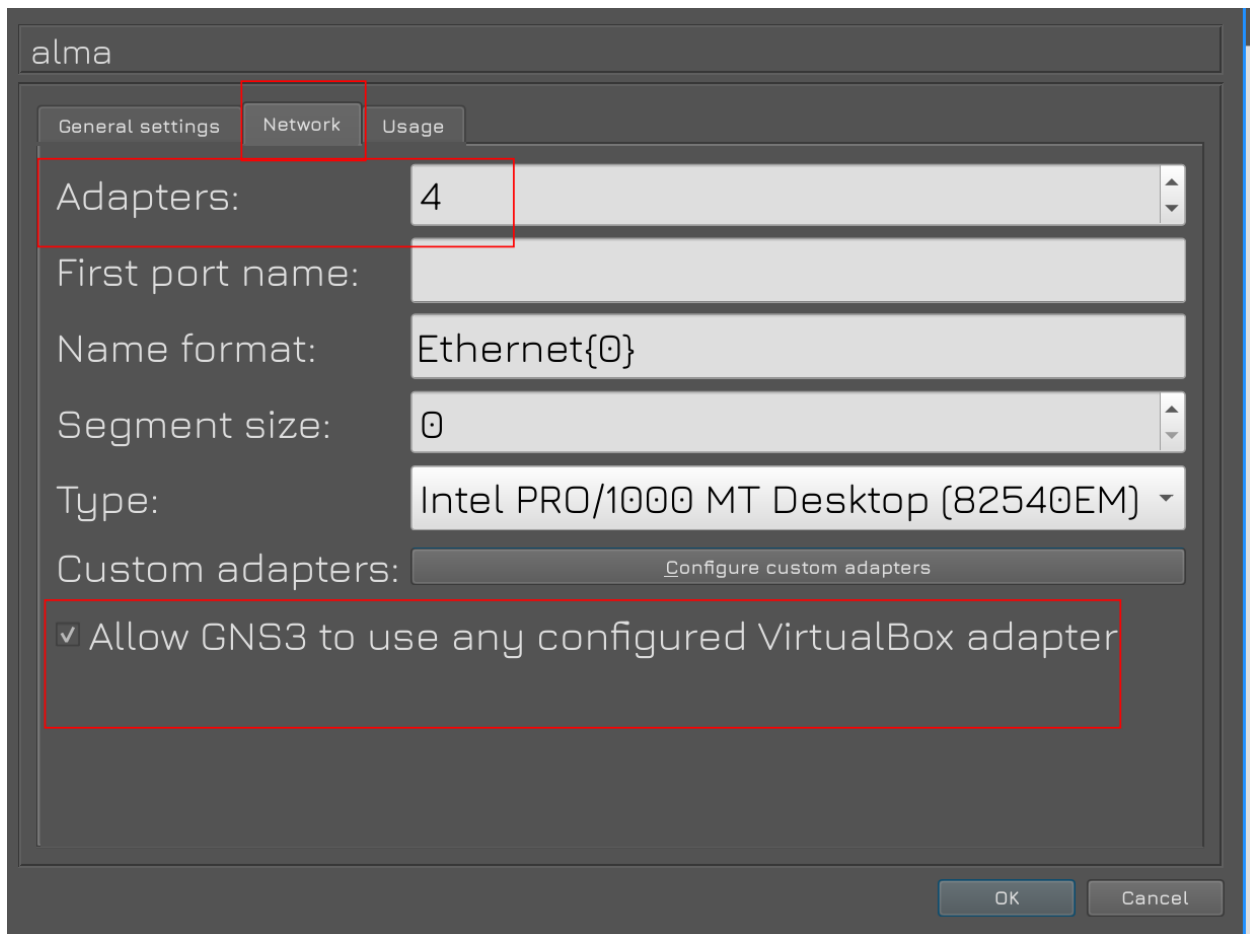
После создания виртуалки выключите её и создайте её клон. Он вам понадобится, чтобы быстро восстановить виртуалку, если вдруг что-то с ней случится. Всё дело в том, что GNS немного капризный в работе и можно случайно попортить связь между GNS3 и VirtualBox. Поэтому и создадим клон на всякий случай.



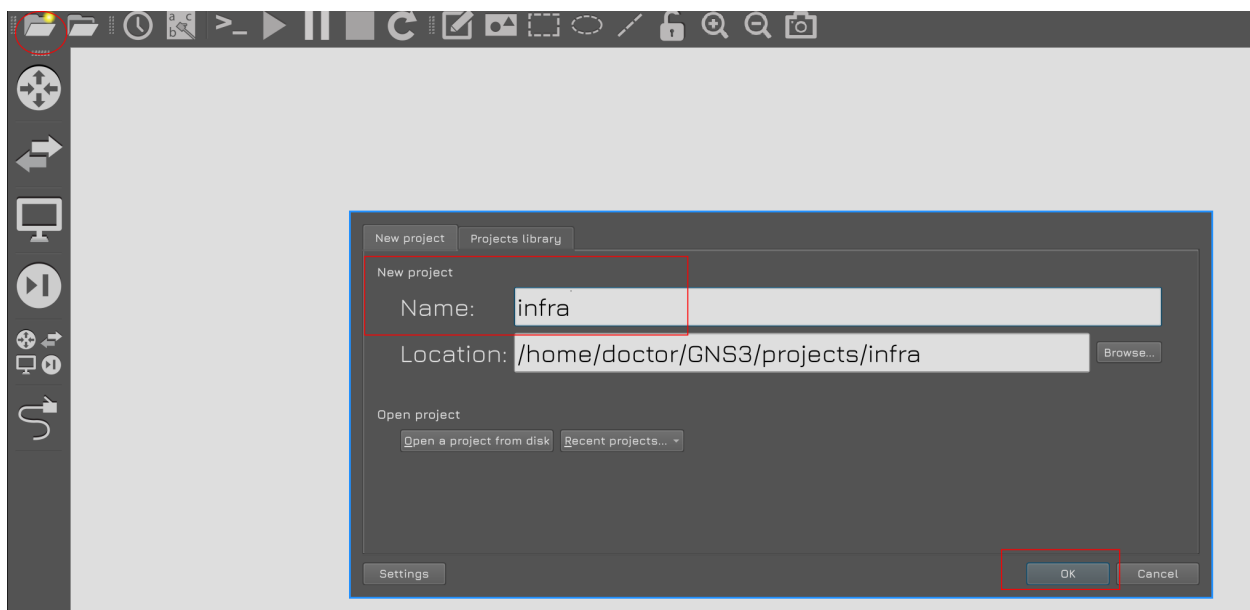
После создания виртуалки возвращаемся в настройки GNS. Теперь во вкладке VirtualBox VMs нажимаем New и из списка виртуалок выбираем alma. Не забываем поставить галочку «Use as a linked base VM», что позволит нам простым перетягиванием создавать новые виртуалки, используя эту как образ.



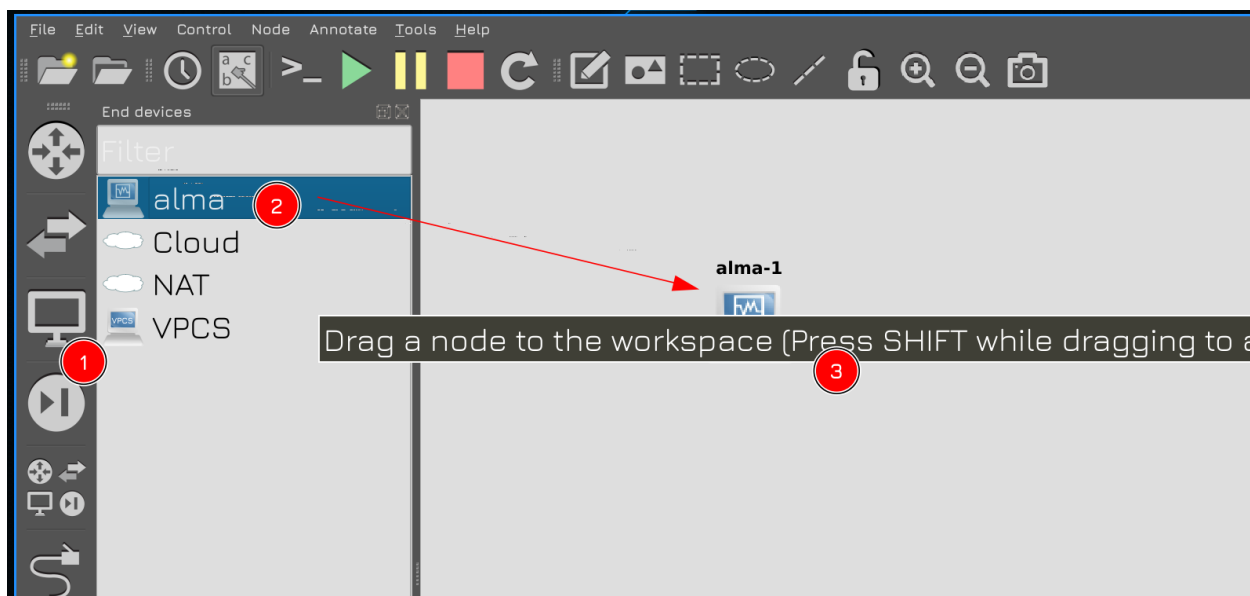
Затем в списке слева появится alma. Нажимаем на неё - Edit - меняем количество оперативки на 512MB и ставим галочку «Start VM in Headless mode». Это позволит запускать виртуалку без нового окна с консолью, так как мы всё равно будем подключаться по SSH.



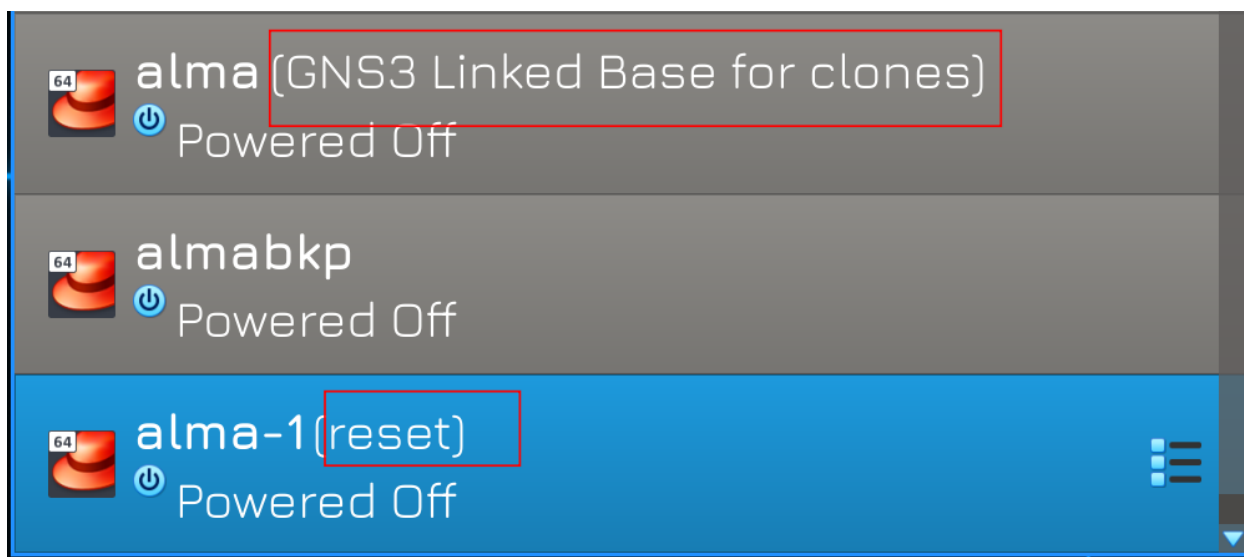
Затем во вкладке Network меняем количество адаптеров на 4 и ставим галочку внизу, чтобы разрешить GNS3 управлять сетевыми адаптерами виртуалбокса. После чего нажимаем OK и Apply.



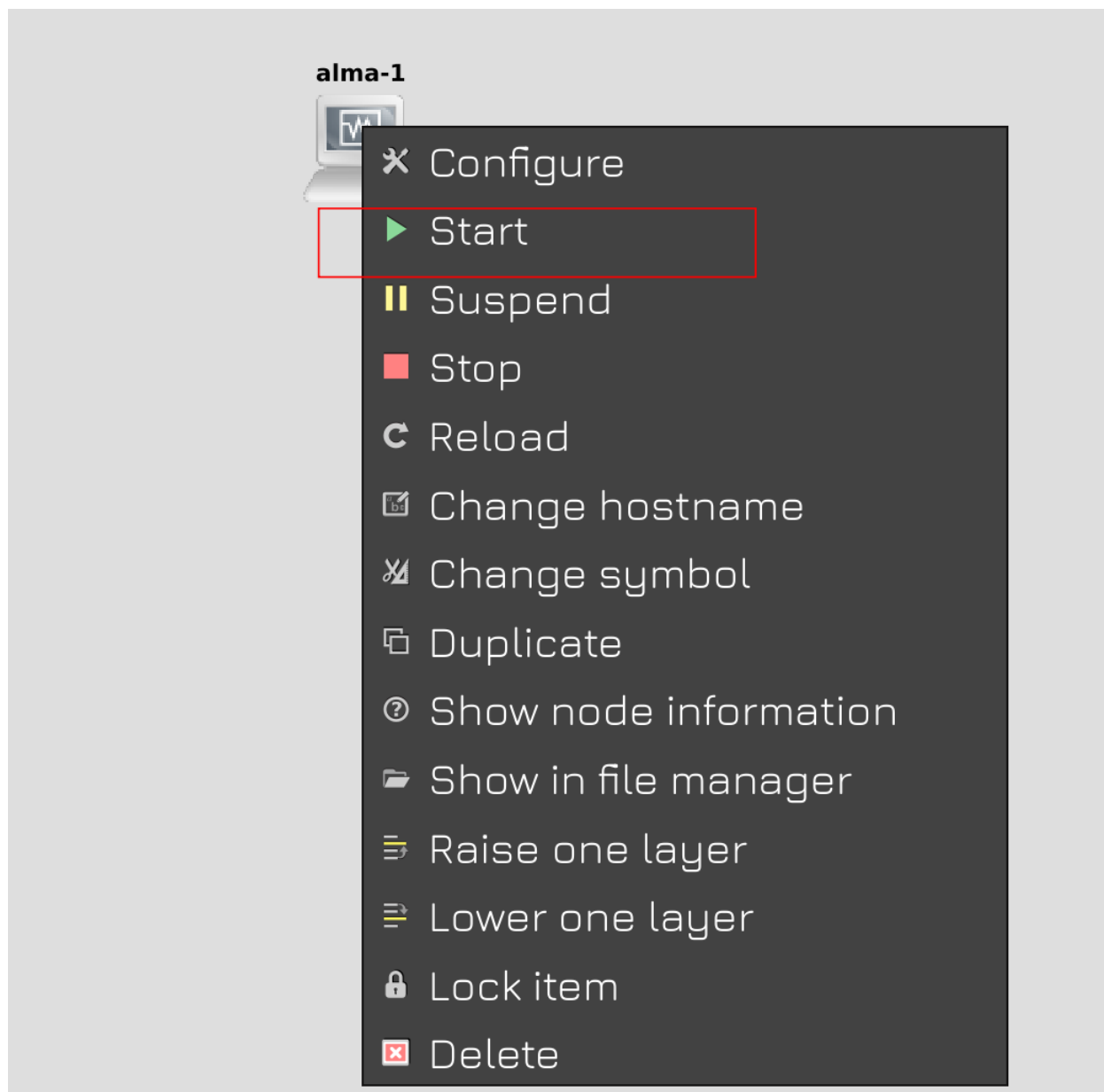
Закрываем окно настроек и нажимаем на иконку «New blank project» в верхней панели. В появившемся окне даём имя нашему проекту, по необходимости выбираем директорию и нажимаем OK.



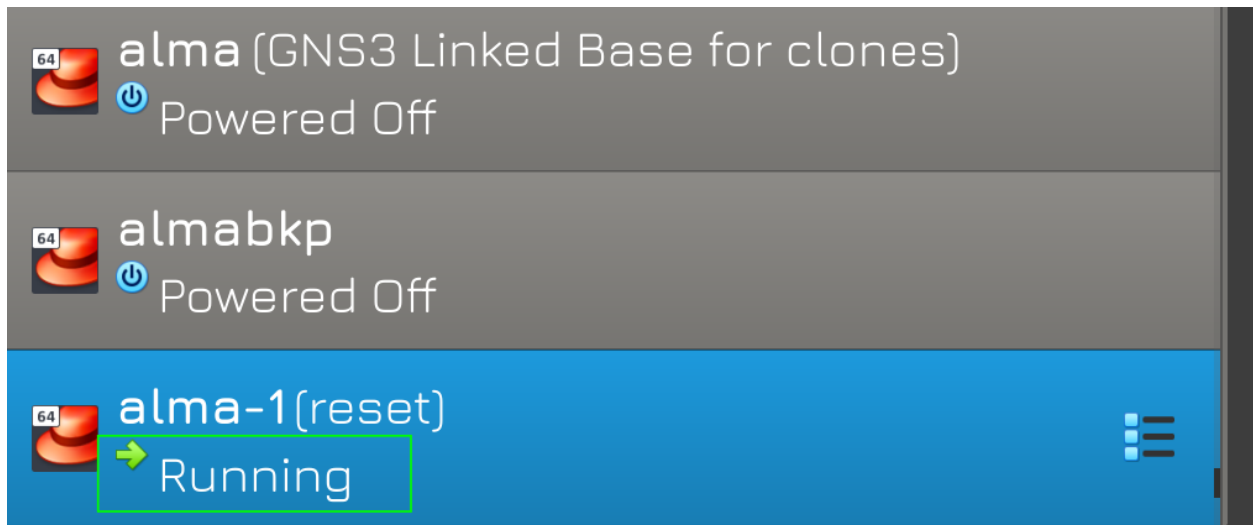
Затем нажимаем на иконку монитора в левой панели - откроется панель с устройствами, среди которых наша alma. Попробуем перетянуть её в центральную часть.



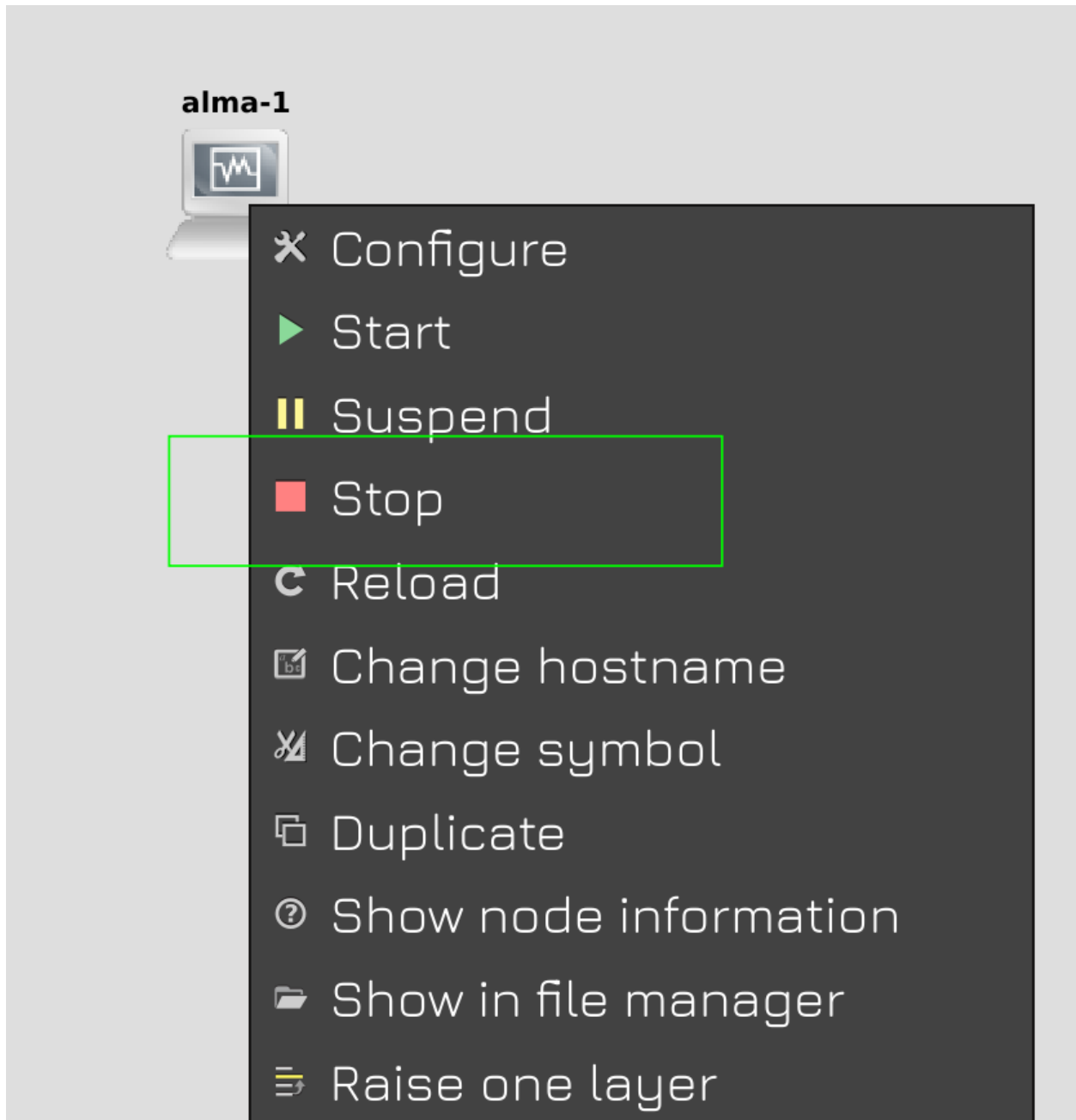
При этом создаётся копия виртуалки. В самом виртуалбоксе можно будет увидеть эту виртуалку, ну и у этих двух виртуалок в названии появятся метки. Не трогайте эти виртуалки в VirtualBox - не удаляйте и ничего не меняйте. Ими управляет GNS, и, как я говорил, GNS очень капризный. Если вы случайно удалите виртуалку или что-то в ней испортите, то весь ваш проект в GNS перестанет запускаться, а для его восстановления придётся не мало повозиться. Когда вы будете закрывать GNS, виртуалки из проекта будут пропадать из виртуалбокса, а при запуске проекта - появляться.



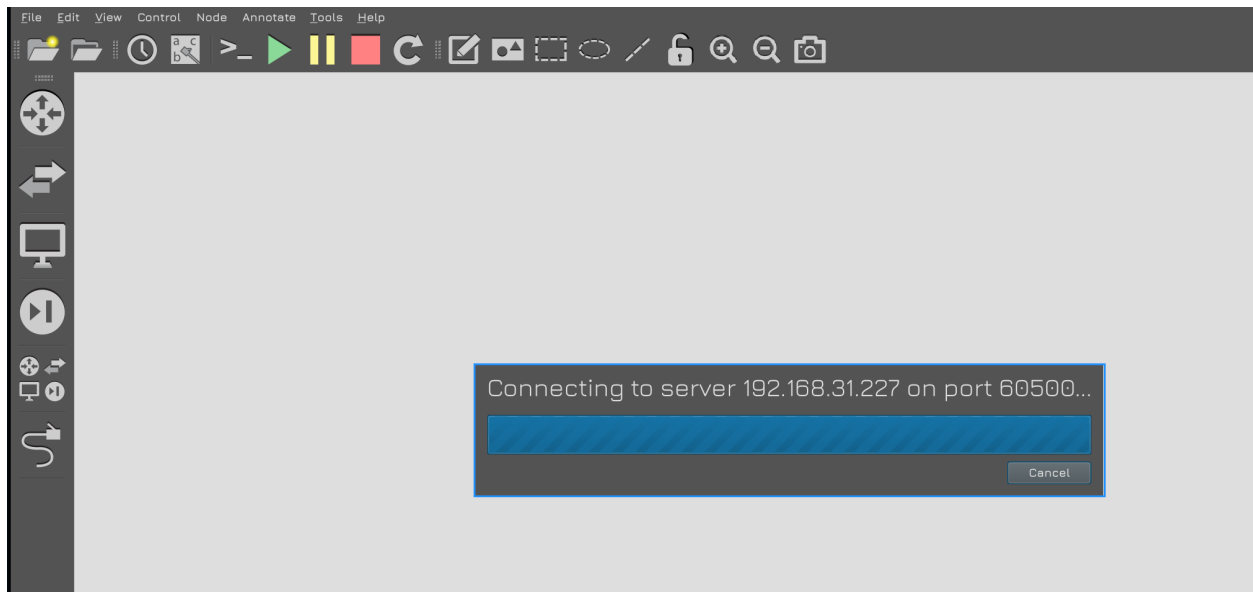
Теперь попробуем запустить виртуалку. Для этого в GNS нажмите правой на ней и нажмите Start.



После чего в виртуалбоксе видно, что виртуалка запустилась. Значит всё окей.



Чтобы выключить виртуалку, опять же, в GNS нажимаем правой кнопкой мыши на виртуалке и нажимаем Stop. Всегда перед выходом из GNS предварительно выключайте все виртуалки.

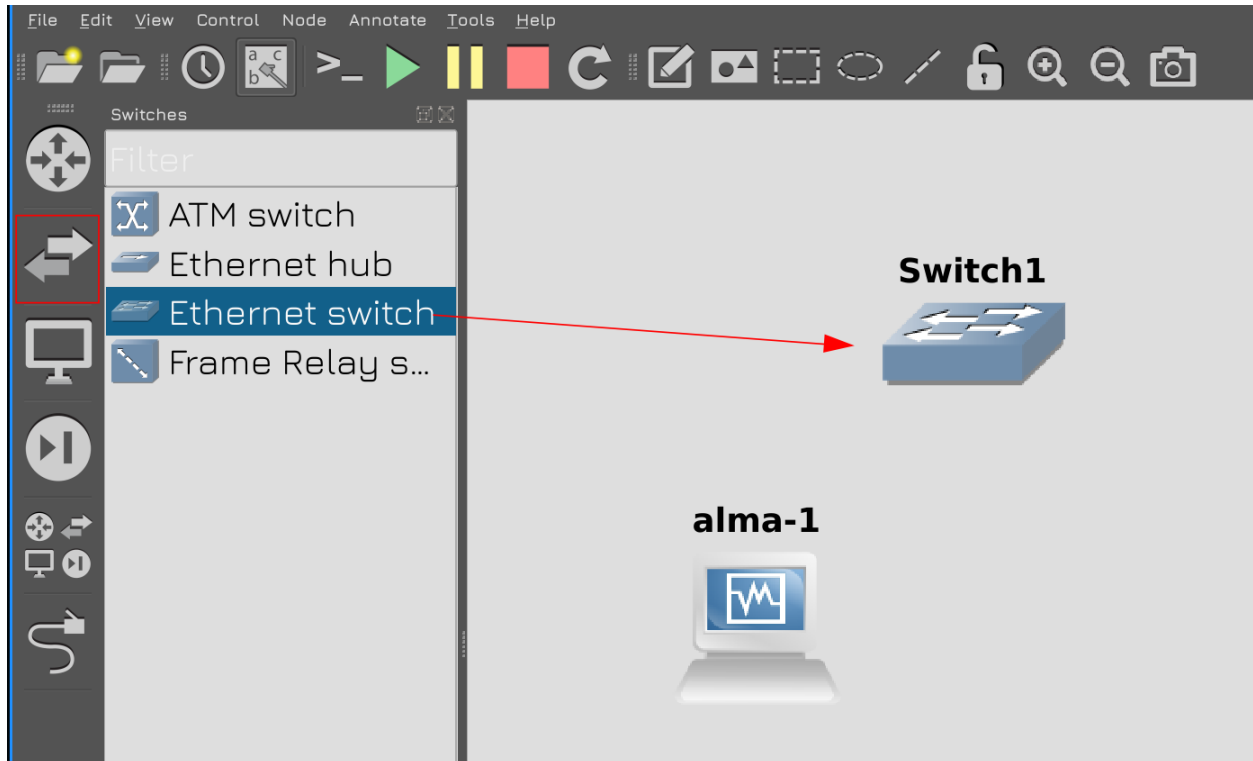


Ну и всегда при запуске GNS3 дожидайтесь, чтобы отработал его запуск и появилось окно с выбором проектов. Иногда это может занять время, но лучше подождать.

Есть другие программы по типу GNS3, есть другие гипервизоры, другие ssh клиенты, другие дистрибутивы - и вы вольны использовать всё что угодно. Но я работал именно с такой связкой, поэтому, если у вас возникнут проблемы - с этими инструментами мне будет легче вам помочь.

2.3 03. Деление на VLAN-ы

2.3.1 03. Деление на VLAN-ы

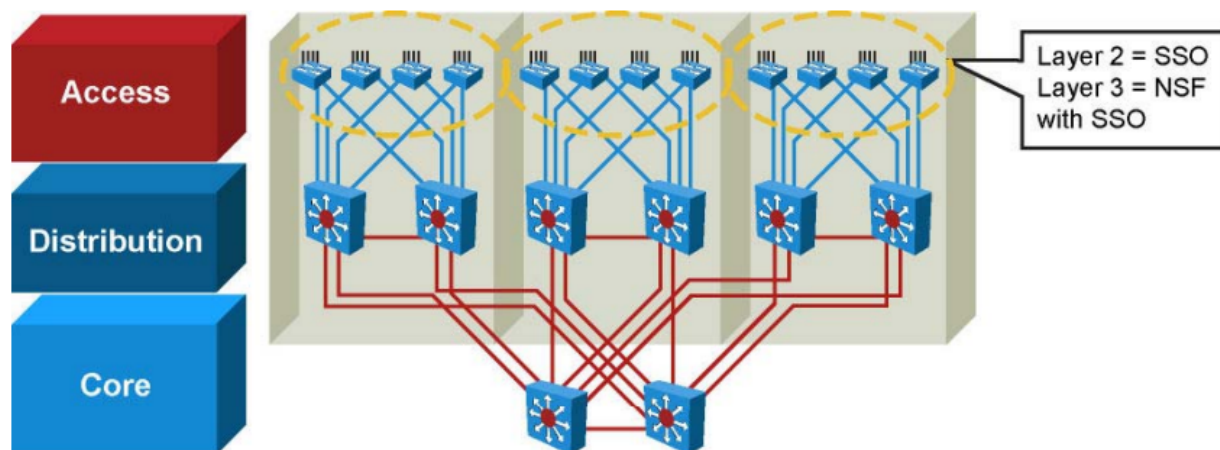


Давайте начнём строительство инфраструктуры с того элемента, который всё связывает - со свитчей. Если вы не знакомы с сетями, советую предварительно посмотреть 42 и 43 тему из курса «Основы GNU/Linux». Нажмите в левой панели GNS на иконку с двумя стрелками, выберите Ethernet Switch и перетащите на центральную часть. Официальное название свитча на русском - коммутатор, ну или сетевой коммутатор. А иконка в GNS - общепринятое обозначение. Сразу скажу, что обычно со свитчами работает сетевой администратор, но у сисадмина должно быть представление, всё таки он отвечает за подключение серверов к свитчам и должен уметь кое-что настраивать со своей стороны. Ну и в маленьких компаниях, нередко системный администратор также отвечает за настройку сети.



Свитч в реальности - это такое устройство, состоящее из кучи портов. В этом его основное предна-

значение - пересылать трафик с кабеля, воткнутого в один порт, на кабель, воткнутый в другой порт. Хотя бывают и маленькие свитчи на 5-8-12 портов, но, в средних и крупных компаниях, вы, чаще всего, будете встречать 24 или 48 портовые. В наше время большинство портов на свитчах гигабитные, при этом бывают порты на 10 гигабит, 25, 40 и даже 100. Теоретически, свитчем может быть и обычный компьютер, но когда речь идёт про большие скорости, на свитчах стоят специальные чипы, которые быстрее обрабатывают трафик.



В зависимости от задач, сетевики делят свитчи на несколько уровней - access, ditribution и core. Access свитчи обычно стоят на каждом этаже здания, у них много ethernet портов и они в основном предназначены для подключения пользователей, принтеров и прочего подобного оборудования. Витую пару нельзя делать больше 100 метров, иначе сигнал в проводах просто затухает. Поэтому, если здание большое, на каждый этаж ставят по одному или несколько access свитчей. А эти свитчи с помощью оптических кабелей на 10 гигабит подключают к другим свитчам - distribution. Порты, соединяющие одни свитчи с другими называют аплинками - т.е. эдакий выход из свитча. А distribution свитчи в свою очередь подключены к core свитчам - куда стекается весь трафик. Скорости, на самом деле, зависят от размера инфраструктуры и количества трафика - в маленькой инфраструктуре нет смысла иметь такую топологию и такие скорости. Но, когда инфраструктура растёт, всё сводится к такой схеме.





В местах, где стоят компьютеры или прочие конечные устройства - назовём их endpoint-ами - на стене или в полу делают специальные сокет, куда и вставляется медный провод.



Эти сокет ведут к небольшим рэкам, где также выводятся в виде сокетов, но уже в один или несколько рядов. Это называется патч-панелями. А из патч-панели же идут патч-корды к access свитчам. Сокет в патч-панели, соединённый с сокетом у пользовательского компьютера, маркируют одинаково. Условно,

на патч-панели всё в числовом порядке - 1, 2, 3 и т.п. И сокет, стоящий за первым портом патч-панели - тоже маркируют 1, не важно, в какой комнате он находится. А дальше нужно маркировать кабель, идущий от патч-панели к свитчу - чтобы было понятно, к какому порту какого свитча подключается 5 сокет с патч-панели. И завтра, если нужно будет что-то сделать, подойдя к компьютеру можно будет увидеть, что на соquete написано 5, а значит нужно смотреть пятый порт на патч-панели. А там уже по маркировке будет видно, на какой порт свитча идёт кабель. Потому что проводов может быть много и разобраться без маркировки становится сложно.

Правильная и актуальная маркировка - это задача IT отдела, в том числе системного администратора. И если вам попался хороший подрядчик, который провёл кабели, поставил сокеты и всё промаркировал - это здорово. Но это вы администрируете инфраструктуру, завтра вы купите новые свитчи, замените кабели, что-то поменяете - поэтому старайтесь держать маркировку актуальной.

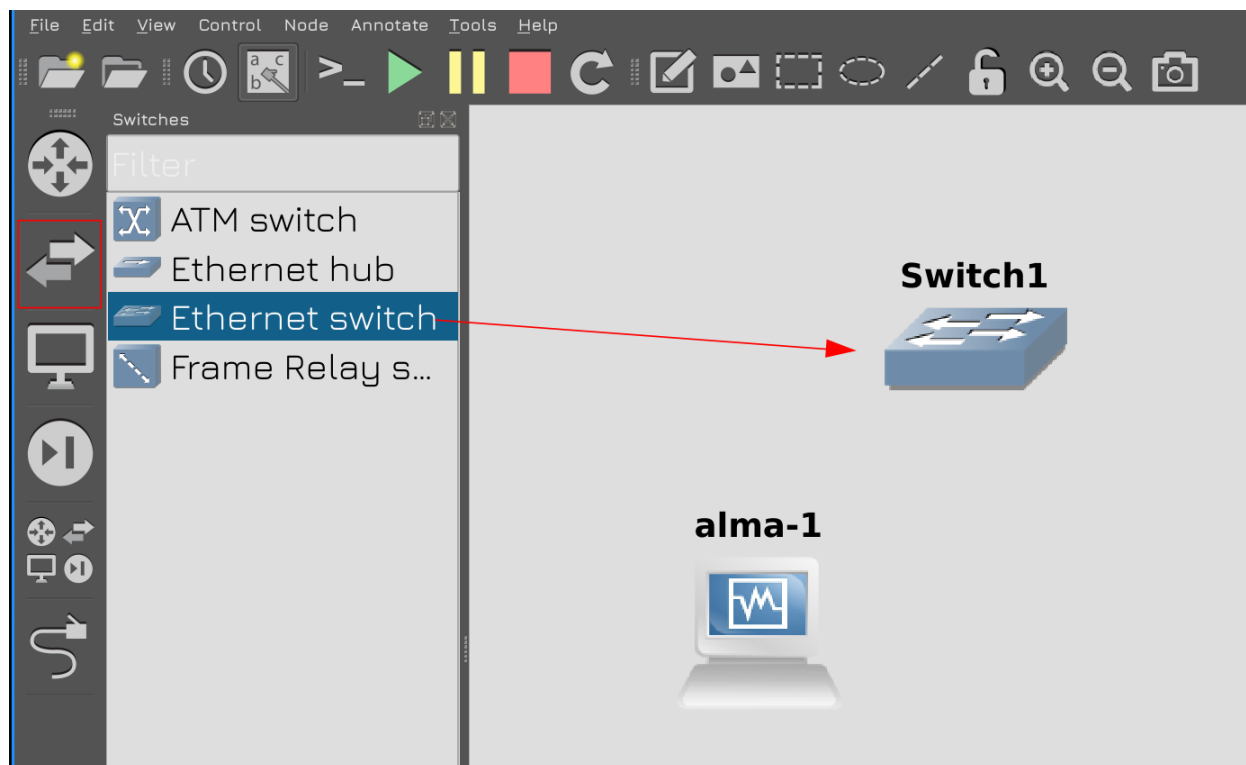


Ну и в серверной, подключая различные провода к серверам и прочему оборудованию - не важно,

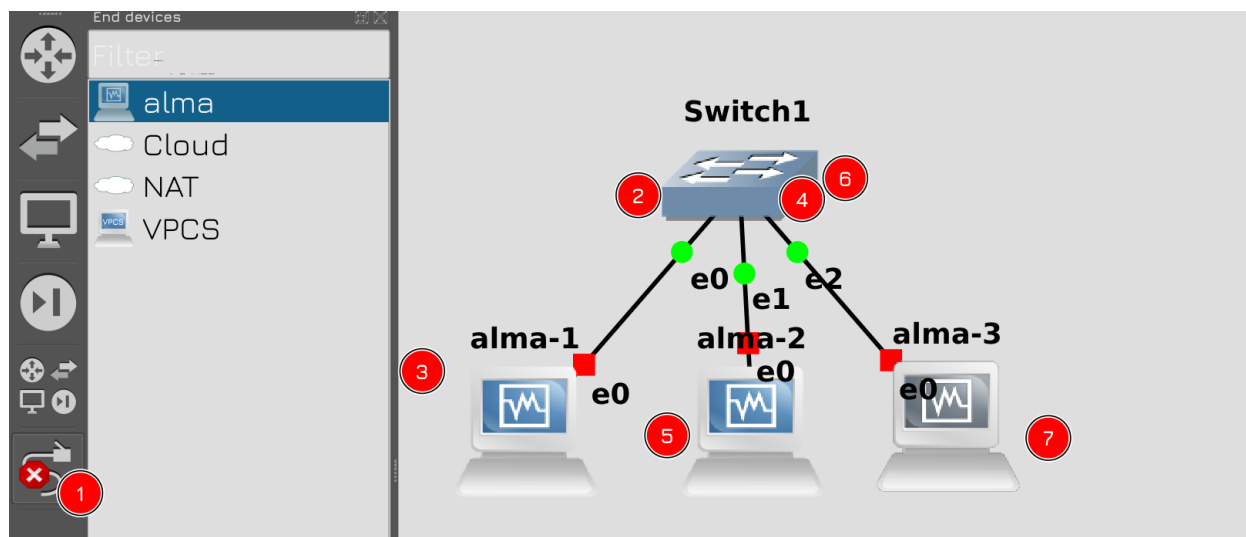
электричество, ethernet и прочее - также обязательно всё маркируйте. Для этого есть специальные устройства, которые на клейкой ленте печатают текст - они называются label-printer. И старайтесь всю маркировку держать согласно какому-то стандарту - нужно очень кратко и точно описывать, что куда подключено.



Скажем, с одной стороны электрического провода пишете, что он ведёт на 3 порт первого UPS-а, а на другом конце, который подключен к UPS-у - пишете, что этот кабель ведёт на второй блок питания третьего сервера. И всё это надо уместить в пару символов на тоненькой бумажке, причём так, чтобы просто взглянув на неё было понятно, что это и куда ведёт. Чтобы в случае проблем вы случайно не выдернули не тот провод и окончательно всё не испортили. Даже если у вас будет куча кабелей, вы всегда разберётесь - какой провод с чем связан и куда ведёт. Чуть больше про маркировку и порядок в серверной вы можете почитать по ссылкам - 1, 2.

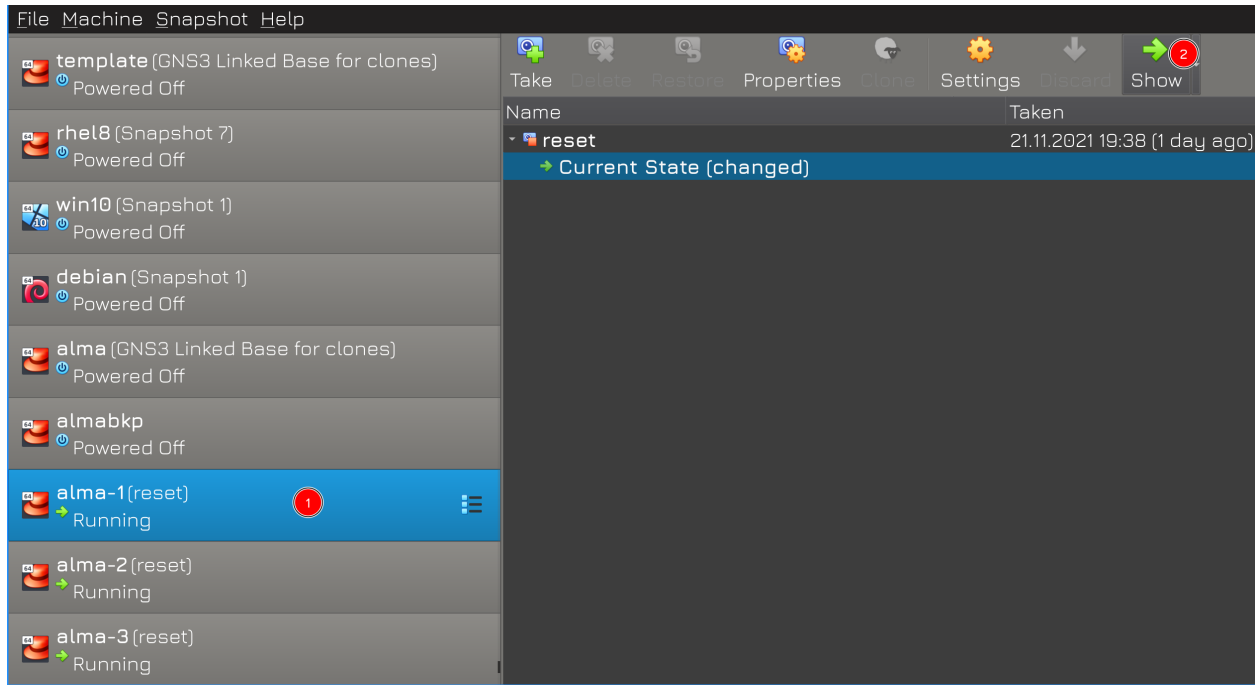


Мы же вернёмся к свитчам. Как бы не казалось, что они выполняют простую задачу, в них огромный функционал, особенно у современных свитчей. Но основная их задача - связывать тех, кто хочет связаться. В модели OSI это второй уровень - каналный, часто называемый L2, т.е. layer 2. На этом уровне устройства друг к другу обращаются по MAC адресам, а свитч запоминает, за каким портом какой MAC адрес находится. Для этого у свитча есть MAC-адресная таблица.

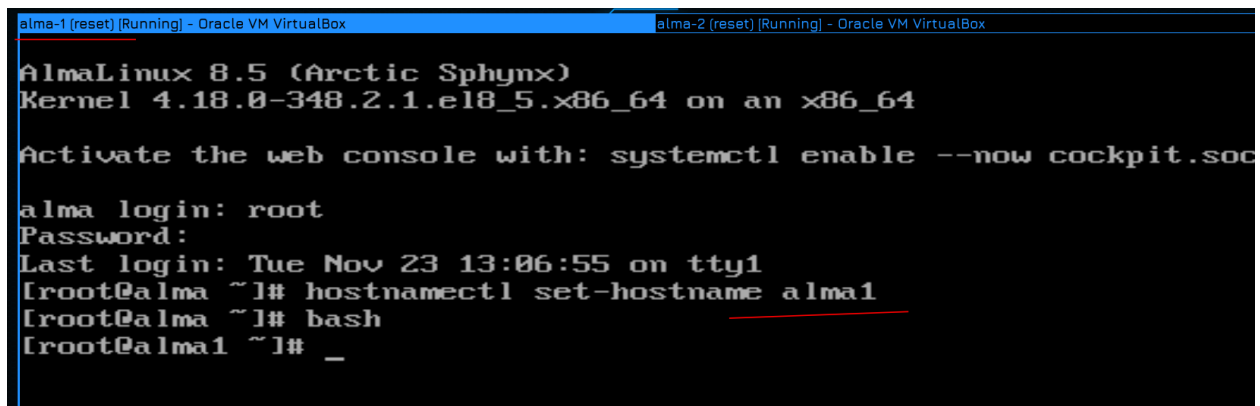


Чтобы понять и увидеть это, давайте возьмём 3 хоста. Сделайте так, чтобы у вас на схеме был 1 свитч и 3 виртуалки с alma, т.е. 3 раза перетяните alma в центр. Затем в левой панели кликните на последнюю иконку, напоминающую провод, потом нажмите на свитч, выберите Ethernet0 и затем на alma-1. Ещё раз нажмите на свитч, выберите Ethernet1 и нажмите на alma-2. Повторите тоже самое для alma-3. Затем нажмите правую кнопку мыши или на иконку провода. Так мы подключили всё в одну сеть. Обратите внимание, рядом с alma появились надписи - e0 - это говорит о том, что мы

подключили провода к нулевым адаптерам. Мы в прошлый раз для виртуалок настроили 4 адаптера, соответственно, названия будут e0, e1, e2 и e3. Со стороны свитча же подключено три провода - e0, e1 и e2.



После этого запустим все 3 виртуалки. Так как наша сеть пока не готова, по ssh я к ним подключиться не смогу. Поэтому на этот раз прибегнем к консоли виртуалок. Запустите виртуалбокс, выберите запущенные виртуалки и нажмите show.



Затем назовите каждую систему соответственно - alma1, alma2, alma3:

```
hostnamectl set-hostname alma1
hostnamectl set-hostname alma2
hostnamectl set-hostname alma3
```

Ну и перелогиньтесь, либо запустите bash, чтобы в приглашении консоли отображалось новое имя. Это позволит нам не путаться.

```

alma-1 (reset) (Running) - Oracle VM VirtualBox
root@alma1 ~]# 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: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:ef:95:1d brd ff:ff:ff:ff:ff:ff
3: enp0s8: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state DOWN group default qlen 1000
    link/ether 08:00:27:c8:18:21 brd ff:ff:ff:ff:ff:ff
4: enp0s9: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state DOWN group default qlen 1000
    link/ether 08:00:27:15:10:02 brd ff:ff:ff:ff:ff:ff
5: enp0s10: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state DOWN group default qlen 1000
    link/ether 08:00:27:48:04:49 brd ff:ff:ff:ff:ff:ff
root@alma1 ~]#

```

Теперь посмотрим список интерфейсов:

```
ip a
```

Как видите, в системе отображаются 4 интерфейса помимо loopback. И только один из них в UP-е - enp0s3. Значит он соответствует интерфейсу e0, который мы видели в GNS. На интерфейсах нет IP адресов, потому что в этой сети есть только 3 хоста и свитч - и никакого DHCP сервера, который бы раздавал IP адреса.

```

alma-1 (reset) (Running) - Oracle VM VirtualBox
root@alma1 ~]# ip address add 10.0.0.1/24 dev enp0s3
root@alma1 ~]# ip addr show enp0s3
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:ef:95:1d brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.1/24 scope global enp0s3
        valid_lft forever preferred_lft forever
root@alma1 ~]# ip ro sh
10.0.0.0/24 dev enp0s3 proto kernel scope link src 10.0.0.1
root@alma1 ~]#

```

Давайте выдадим каждой машинке по временному адресу, сеть 10.0.0.0 и адреса 1,2,3 соответственно:

```
ip address add 10.0.0.1/24 dev enp0s3
ip address add 10.0.0.2/24 dev enp0s3
ip address add 10.0.0.3/24 dev enp0s3
```

Маску выставим /24. Про IP адреса мы поговорим в другой раз, поэтому сегодня обусловимся чем-то дефолтным. Такая маска нам говорит, что в нашей сети находятся все адреса от 10.0.0.0 до 10.0.0.255. Нам этого хватает. Давайте убедимся, что IP адрес появился на интерфейсе:

```
ip addr show enp0s3
```

Как видите, адрес прописался. Ну и проверим ещё вывод команды:

```
ip ro sh
```

Здесь мы видим, что до любого хоста из сети 10.0.0.0/24 можно подключиться напрямую через интерфейс enp0s3. Пропишите адреса на всех 3 хостах.

```

alma-1 (reset) (Running) - Oracle VM VirtualBox
root@alma2 ~]# tcpdump
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v or -w for full protocol decode
listening on enp0s3, link-type EN10MB (Ethernet), capture size 262144 bytes
-

```

Теперь на втором и третьем хосте запустите команду tcpdump:

```
tcpdump
```

Она позволит увидеть информацию по всем приходящим пакетам.

```
alma-1 (reset) [Running] - Oracle VM VirtualBox  alma-2 (reset) [Running] - Oracle VM VirtualBox
[root@alma1 ~]# ping -c 1 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=2.77 ms

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 2.774/2.774/2.774/0.000 ms
[root@alma1 ~]# _
```

После чего идём на первый хост и отправляем один пинг на второй хост:

```
ping -c 1 10.0.0.2
```

Как видите, пинг сработал, значит сеть работает.

```
alma-1 (reset) [Running] - Oracle VM VirtualBox  alma-2 (reset) [Running] - Oracle VM VirtualBox  alma-3 (reset) [Running] - Oracle VM VirtualBox
[root@alma3 ~]# tcpdump
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on enp0s3, link-type EN10MB (Ethernet), capture size 262144 bytes
13:51:00.488549 ARP, Request who-has 10.0.0.2 tell 10.0.0.1, length 46
```

Теперь идём смотреть на хост 3. Видите, tcpdump выдал строчку - ARP запрос - тот у кого адрес 10.0.0.2 ответьте 10.0.0.1. И типшина. Дело в том, что alma1 пыталась достучаться до адреса 10.0.0.2. Но она знает только IP адрес, а общение в L2 сети идёт по MAC адресу. Поэтому хост послал специальный ARP запрос - всем хостам в сети рассылается вопрос - мол, у кого такой айпи адрес, скажи свой мак адрес. Этот запрос пришёл и на третий хост, но у него айпи адрес не такой, поэтому альма3 проигнорировала это сообщение.

```
alma-1 (reset) [Running] - Oracle VM VirtualBox  alma-2 (reset) [Running] - Oracle VM VirtualBox  alma-3 (reset) [Running] - Oracle VM VirtualBox
[root@alma2 ~]# tcpdump
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on enp0s3, link-type EN10MB (Ethernet), capture size 262144 bytes
13:51:00.210701 ARP, Request who-has alma2 tell 10.0.0.1, length 46
13:51:00.210750 ARP, Reply alma2 is-at 08:00:27:4a:14:9e (oui Unknown), length 28
13:51:00.212010 IP 10.0.0.1 > alma2: ICMP echo request, id 1442, seq 1, length 64
13:51:00.212109 IP alma2 > 10.0.0.1: ICMP echo reply, id 1442, seq 1, length 64
13:51:05.234708 ARP, Request who-has 10.0.0.1 tell alma2, length 28
13:51:05.235996 ARP, Reply 10.0.0.1 is-at 08:00:27:ef:95:1d (oui Unknown), length 46
```

Давайте посмотрим, что произошло на хосте 2. Как видите, тут уже сообщений побольше. Сперва мы видим тот же ARP запрос, но теперь на него есть ответ - reply. В ответе сказано, что такой-то адрес находится за таким-то мак адресом. Не обращайте внимание на слово alma2, это утилита tcpdump для удобства превратила адрес в имя. После этого прилетает ICMP запрос - это уже сам пинг. Причём, обратите внимание, что через некоторое время опять был арп запрос. Чтобы избежать проблем, мол, мало ли, этот IP адрес окажется на другом компьютере, арп таблица периодически обновляется.

```
alma-1 (reset) [Running] - Oracle VM VirtualBox  alma-2 (reset) [Running] - Oracle VM VirtualBox
[root@alma1 ~]# arp -a
? (10.0.0.2) at 08:00:27:4a:14:9e [ether] on enp0s3
[root@alma1 ~]# _
```

```

alma-1 (reset) [Running] - Oracle VM VirtualBox
alma-2 (reset) [Running] - Oracle VM VirtualBox
[roo@alma2 ~]# arp -a
? (10.0.0.1) at 08:00:27:ef:95:1d [ether] on enp0s3
[roo@alma2 ~]# _

```

И сделав это, оба хоста запомнили друг друга по мак адресу:

```
arp -a
```

```

alma-1 (reset) [Running] - Oracle VM VirtualBox
alma-2 (reset) [Running] - Oracle VM VirtualBox
alma-3 (reset) [Running] - Oracle VM VirtualBox
[roo@alma2 ~]# tcpdump
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on enp0s3, link-type EN10MB (Ethernet), capture size 262144 bytes
14:08:06.235146 IP 10.0.0.1 > alma2: ICMP echo request, id 1476, seq 1, length 64
14:08:06.235260 IP alma2 > 10.0.0.1: ICMP echo reply, id 1476, seq 1, length 64
14:08:11.282761 ARP, Request who-has 10.0.0.1 tell alma2, length 28
14:08:11.284171 ARP, Reply 10.0.0.1 is-at 08:00:27:ef:95:1d (oui Unknown), length 46
14:08:11.497419 ARP, Request who-has alma2 tell 10.0.0.1, length 46
14:08:11.497449 ARP, Reply alma2 is-at 08:00:27:4a:14:9e (oui Unknown), length 28

```

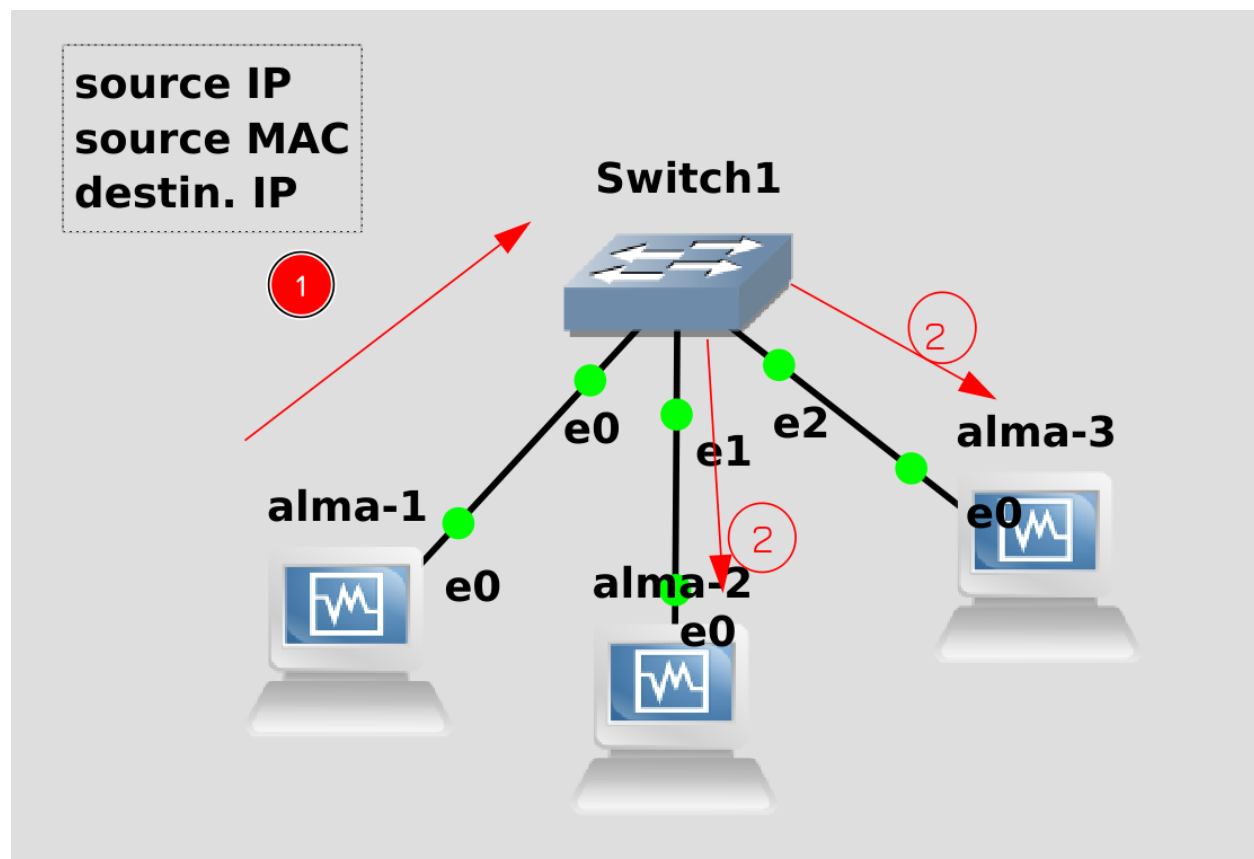
И теперь, если повторить попытку пинга, можно увидеть, что запрос сразу дошёл до хоста 2.

```

alma-1 (reset) [Running] - Oracle VM VirtualBox
alma-2 (reset) [Running] - Oracle VM VirtualBox
alma-3 (reset) [Running] - Oracle VM VirtualBox
[roo@alma3 ~]# tcpdump
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on enp0s3, link-type EN10MB (Ethernet), capture size 262144 bytes
13:51:00.488549 ARP, Request who-has 10.0.0.2 tell 10.0.0.1, length 46

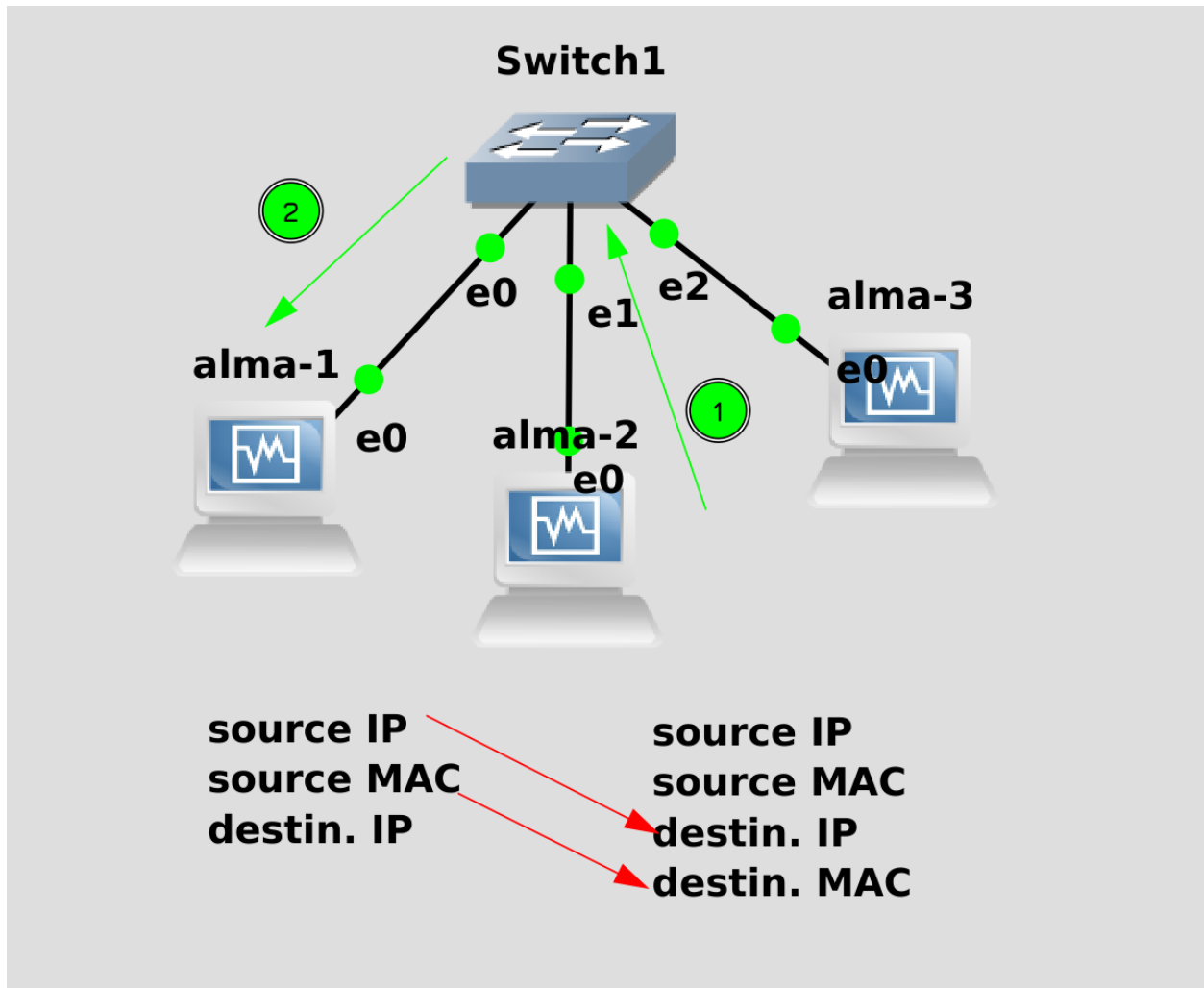
```

При этом, обратите внимание, что на хосте 3 как был один запрос, так он и остался.



Теперь попытаемся понять, какова была роль свитча. Когда альма1 попыталась найти альму2, она

отправила сетевой пакет, в котором был её IP адрес, её мак адрес и IP адрес альмы2. В тот момент ни хост 1, ни свитч не знали мак адрес хоста 2. И поэтому хост 1 решил обратиться ко всем - это называется broadcast - т.е. широковещательный запрос. Свитч этот запрос разослал по всем своим портам, кроме того, откуда пришёл запрос. Вся эта область, откуда приходит и куда доходит broadcast запрос называется broadcast доменом. Грубо говоря, представьте, что кто-то крикнул в одной комнате и все в этой комнате услышали этот крик. Крик - это broadcast запрос, а комната - broadcast домен. В итоге запрос прилетел на хост 2 и хост 3. При этом свитч записал Source MAC адрес хоста 1 к себе в таблицу, мол, за портом Ethernet0 находится такой-то MAC адрес.



Хост 2, получив запрос, взял из пакета Source IP и Source MAC, подготовил ответ и указал их в качестве Destination. Добавил к этому пакету свой IP и MAC и отправил. Свитч, увидя этот пакет, запомнил Source MAC альмы 2, мол, за портом Ethernet1 находится такой-то MAC адрес. Также свитч увидел, что в пакете указан Destination MAC адрес. А свитч то этот МАК помнит - он его на предыдущем шаге себе записал. Вот он сразу этот пакет на порт Ethernet0 и отправил.

Теперь, когда эти два мак адреса будут пытаться что-то передавать друг другу, свитч сразу будет отправлять на нужные порты, при этом никак не трогая альму3.

Вы спросите - а какой нам в этом толк? Как это относится к сисадминству? На самом деле, главное, зачем мы это рассмотрели - чтобы познакомиться с понятием broadcast domain. Понятие broadcast используется как на l2 уровне, так и на l3, мы сейчас говорим исключительно про l2. И чтобы лучше это понять, давайте кое-что изменим в нашем эксперименте.


```

[alma-1 (reset) (Running) - Oracle VM VirtualBox] [alma-2 (reset) (Running) - Oracle VM VirtualBox] [alma-3 (reset) (Running) - Oracle VM VirtualBox]
[root@alma1 ~]# ip addr del 10.0.0.1/24 dev enp0s3
[root@alma1 ~]# ip addr add 10.0.1.1/24 dev enp0s3
[root@alma1 ~]# ip addr show enp0s3
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:ef:95:1d brd ff:ff:ff:ff:ff:ff
    inet 10.0.1.1/24 scope global enp0s3
        valid_lft forever preferred_lft forever
[root@alma1 ~]#

```

Поменяем на альме 1 и 2 IP адреса, а точнее сеть, чтобы теперь они находились в сети 10.0.1.0:

```

ip addr del 10.0.0.1/24 dev enp0s3
ip addr add 10.0.1.1/24 dev enp0s3
ip addr del 10.0.0.2/24 dev enp0s3
ip addr add 10.0.1.2/24 dev enp0s3

```

А хост 3 оставим в сети 10.0.0.0. Теперь, на уровне L3, они находятся в разных сетях, и, соответственно, разные broadcast-ы на уровне L3. А вот на уровне L2.. давайте убедимся.

```

[alma-3 (reset) (Running) - Oracle VM VirtualBox]
[root@alma3 ~]# tcpdump
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on enp0s3, link-type EN10MB (Ethernet), capture size 262144 bytes
15:19:04.066218 ARP, Request who-has 10.0.1.2 tell 10.0.1.1, length 46

```

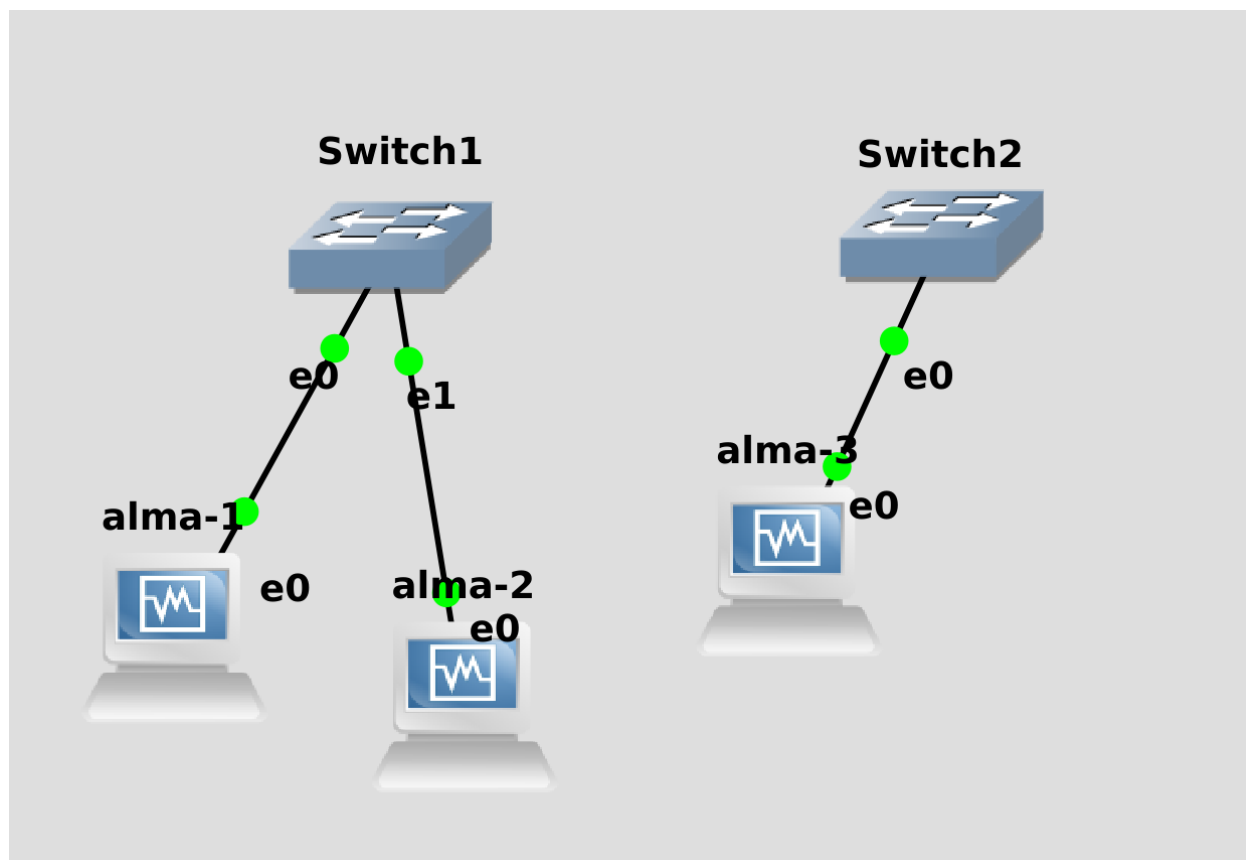
Для этого, опять же, на хосте 3 запускаем tcpdump и пытаемся с первого хоста пингануть второй:

```
ping -c 1 10.0.1.2
```

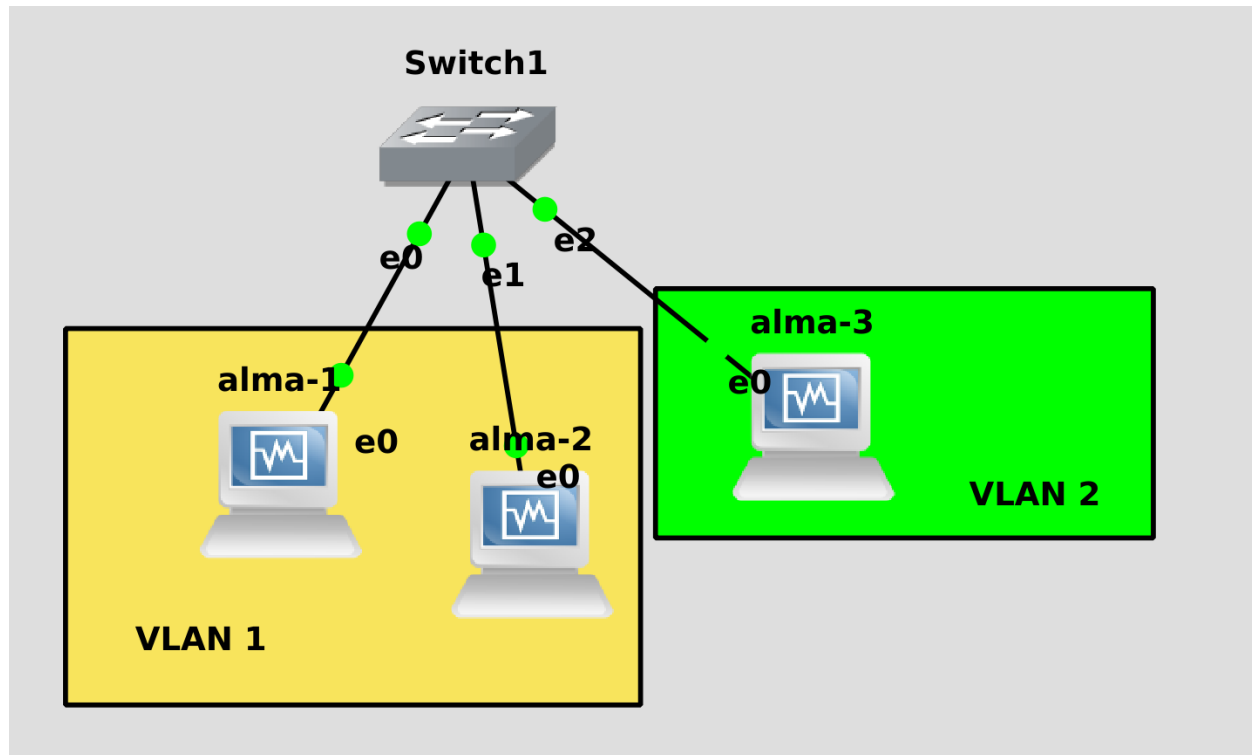
И, как видите, третий хост всё равно получил запрос, даже если у него сеть другая. broadcast на уровне L2 - общий. Т.е. компьютерам достаточно сменить IP адреса и они будут видеть друг друга.

Теперь вопрос - а зачем вообще компьютеры делить на сети, почему бы всё не держать в одной общей сети? Как и с виртуализацией и контейнерами - нужно всё изолировать. К вам в компанию приходят гости и им нужен интернет. Стоит ли их пускать в ту же сеть, что и сервера? Конечно, нет, мало ли что у них на ноутбуках и телефонах крутится. А стоит ли пускать в ту же сеть ваших юзеров? Даже если вы им доверяете больше, они запросто могут не знать базовые принципы информационной безопасности. А уж в больших сетях о доверии речи быть и не может. У одних юзеров должен быть доступ на одни ресурсы, у других пользователей на другие. Есть начальство, бухгалтерия, айти и прочие отделы - и у всех различные уровни доступа. Поэтому сеть нужно делить и изолировать. Это называется сегментацией. Чем больше вы изолируете всё друг от друга, тем сложнее становится всем управлять, но и в случае проблем ущерб будет минимальный.

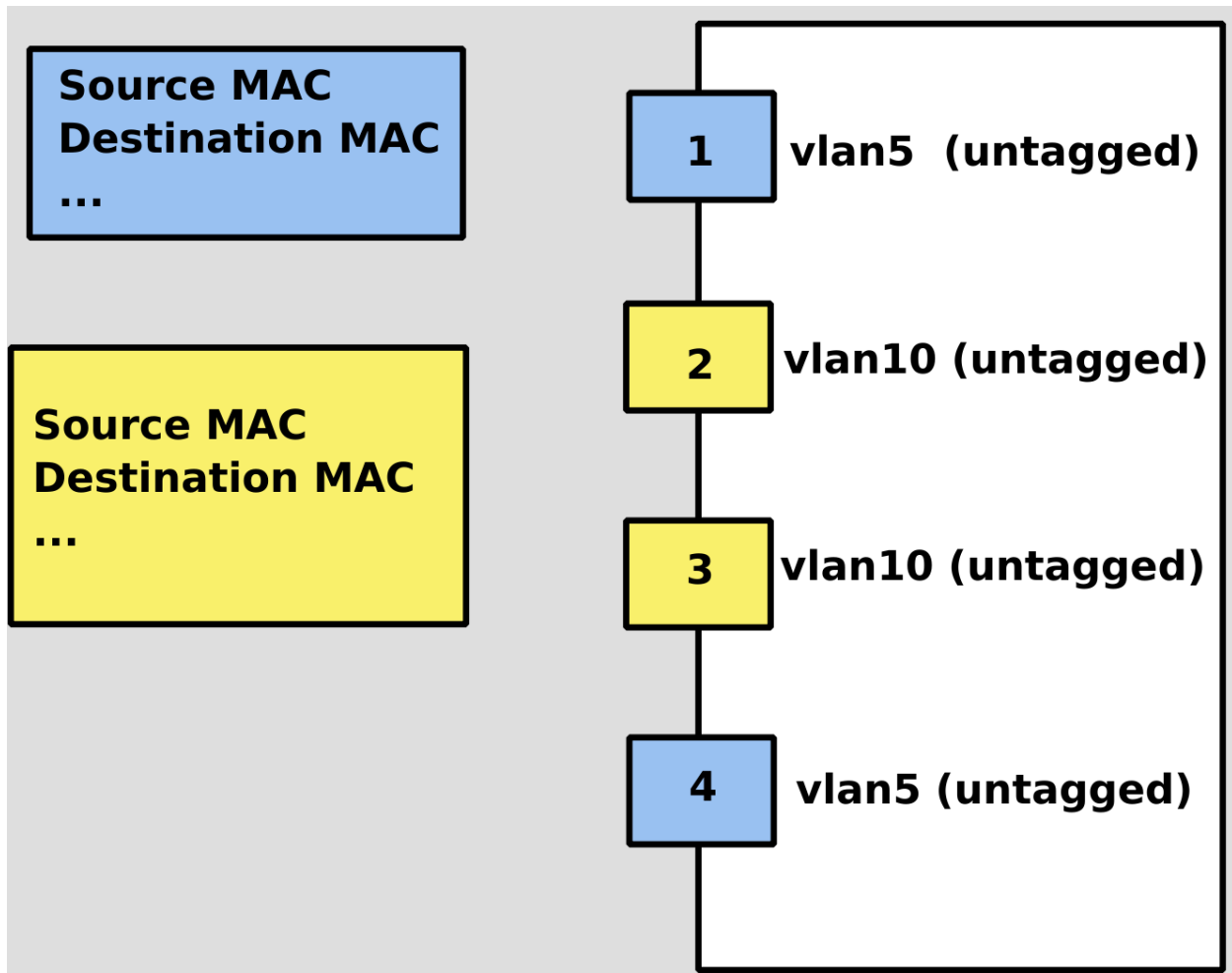
Как мы выяснили, деление просто по разным IP сетям совсем не выход, компьютеры всё равно могут друг друга видеть, потому что находятся в одном broadcast домене. Как тогда быть?



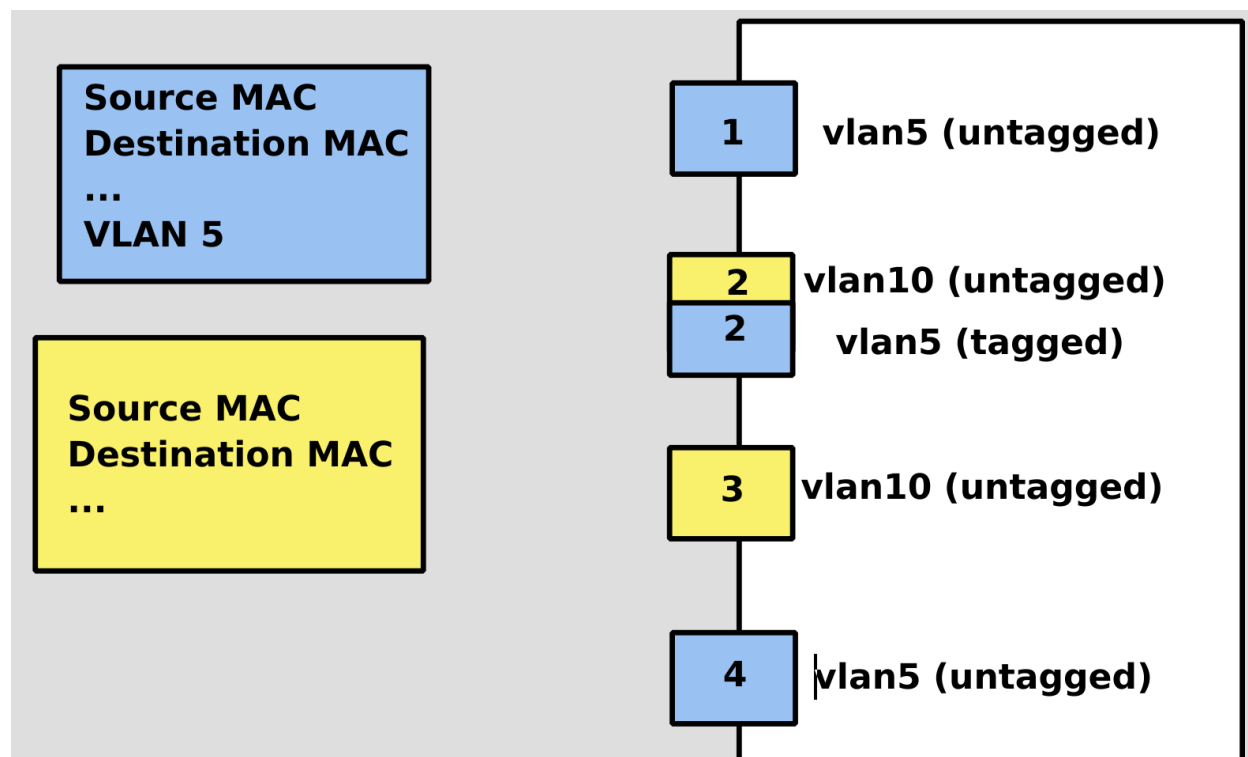
Можно, конечно, поставить второй свитч. Но это как с отдельным сервером для каждого сервиса - очень дорого и бессмысленно. У вас сегодня в отделе 5 человек, завтра 10, послезавтра 8. Вам либо для каждого отдела придётся кучу свитчей покупать, а если отдел сидит по разным этажам и зданиям - вообще беда. Конечно же, это не выход. И здесь вам приходит на помощь виртуальная сеть - Virtual LAN - VLAN.



VLAN-ы помогают создавать поверх одной физической сети разные broadcast домены. У вас устройства всё ещё будут подключены к одному свитчу одним кабелем, но они будут видеть только те хосты, которые с ними в одной виртуальной сети, в одном VLAN-е. В средних и крупных компаниях могут быть десятки, а то и сотни VLAN-ов. И, по хорошему, в одном влане живёт только одна сеть. Скажем, 10.0.0.0/24 в одном влане, а 10.0.1.0/24 в другом. У каждого VLAN-а свой идентификатор - это обычное число от 1 до 4094. Этот идентификатор также называется тегом - эдакая метка.

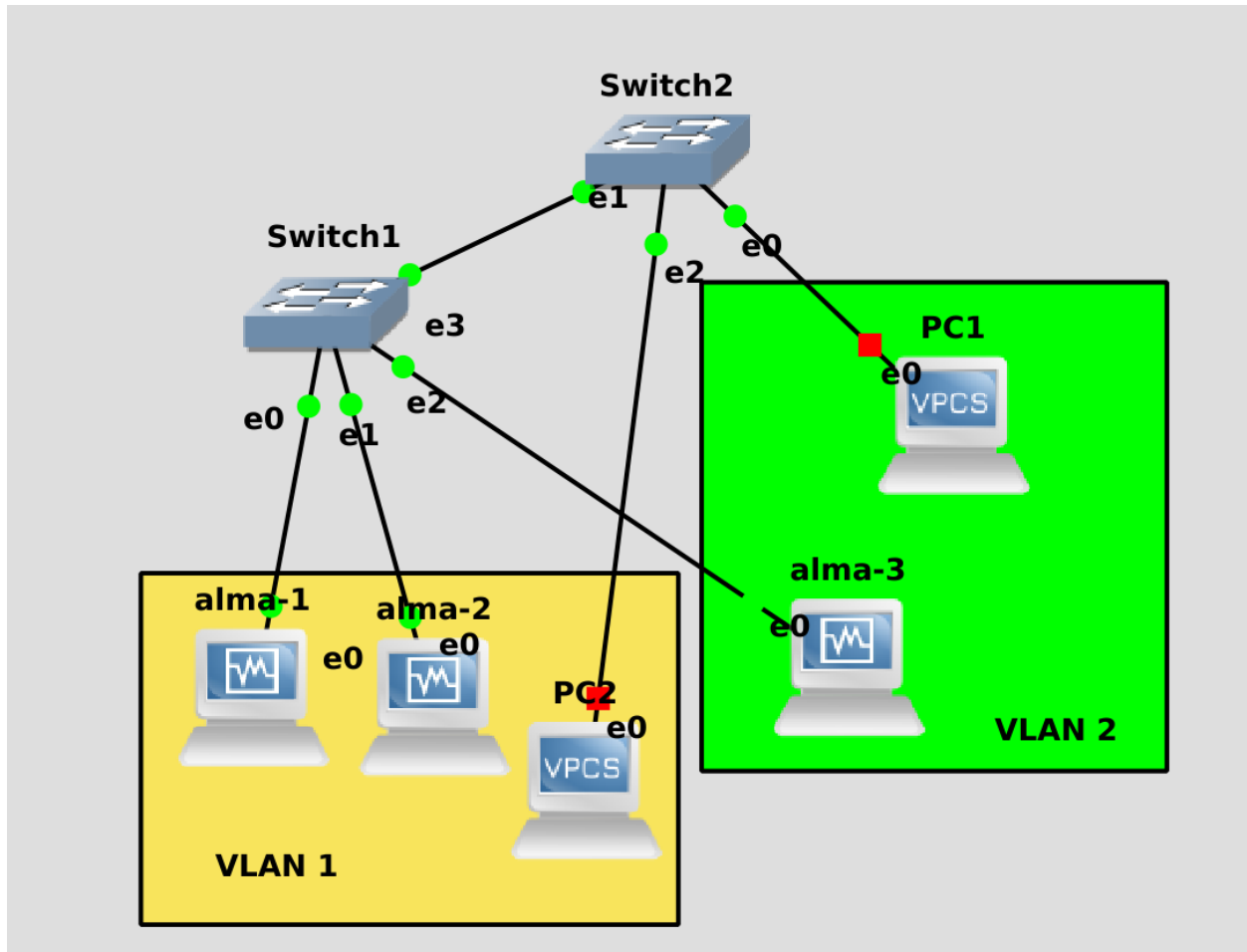


Условно, весь трафик можно поделить на нетегированный и тегированный. В тегированном трафике на всех пакетах есть идентификатор VLAN-а. Обычно на свитчах в сторону компьютеров настроены нетегированные порты - т.е. вы подключаете кабель к компьютеру и всё работает. При этом сам порт на свитче может быть помечен как принадлежащий к какому-то VLAN-у. Условно, я делю сетку на юзеров и сервера, юзеров я держу в 5 влане, а сервера в 10. И юзеры, и сервера подключены к одному свитчу, но я знаю, что за портом 1 находится юзер, а за портом 2 - сервер. И на свитче я настраиваю, что первый порт - это нетегированный порт влана 5, а второй порт - тоже нетегированный, но там уже влан 10. При этом на пользовательском и серверном компе ничего не надо настраивать. Трафик, приходящий на первый порт, будет общим с портом 4. Т.е. они в одном broadcast домене, а порты 2 и 3 в другом.



Но, предположим, я хочу, чтобы сервер, подключенный ко второму порту, был одновременно и в пользовательском VLANе, и в серверном. Я могу либо отдельным кабелем подключиться к порту 4, либо настроить, чтобы на порту 2 также был доступен 5-ый VLAN. А чтобы трафик не путался, надо ко всем пакетам, которые сервер отправляет на 5-ый VLAN, добавлять VLAN ID, а на свитче настроить, чтобы он принимал на втором порту VLAN 5 в тегированном режиме.

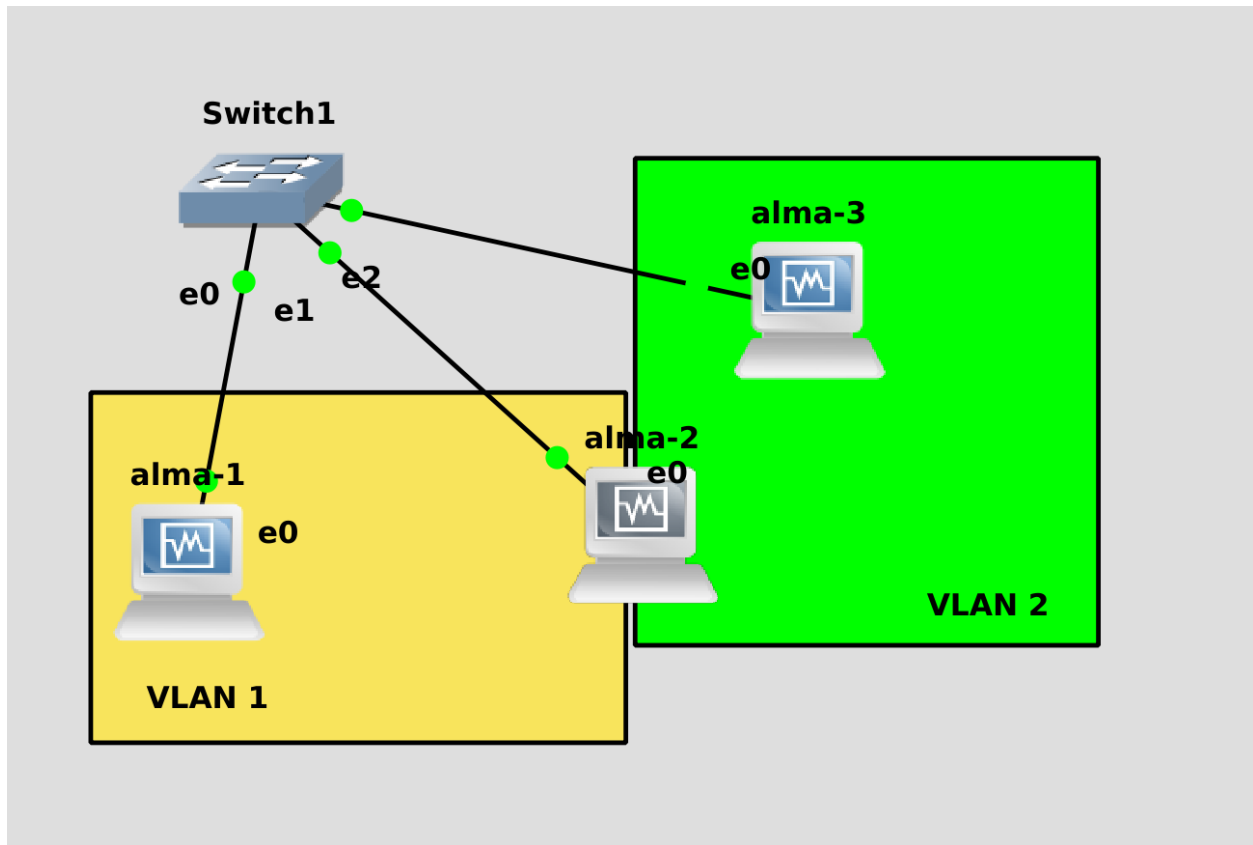
В итоге, если сервер захочет отправить что-то на VLAN 10 - то он просто отправит такой пакет как обычный. А если сервер захочет отправить что-то в 5-ый VLAN, то ему нужно будет добавить к этому пакету тег. Свитч, получив пакет с таким тегом, отправит его на любой из портов с VLANом 5. При этом, когда пакет будет выходить из нетегированного порта, свитч уберёт метку. Тоже самое в обратном направлении - если компьютер на порту 1 захочет отправить что-то на сервер по VLANу 5, то трафик в нетегированном виде придёт на свитч, затем свитч поставит на этот пакет метку и отправит его в тегированном виде на сервер. Сервер, увидя пакет с тегом, будет понимать, что это пакет с пользовательского VLANа. Порты, на которых нет тегов, часто называют аксес портами. На одном порту может быть только один нетегированный VLAN, все остальные VLANы нужно тегировать, чтобы не смешивать broadcast домены.



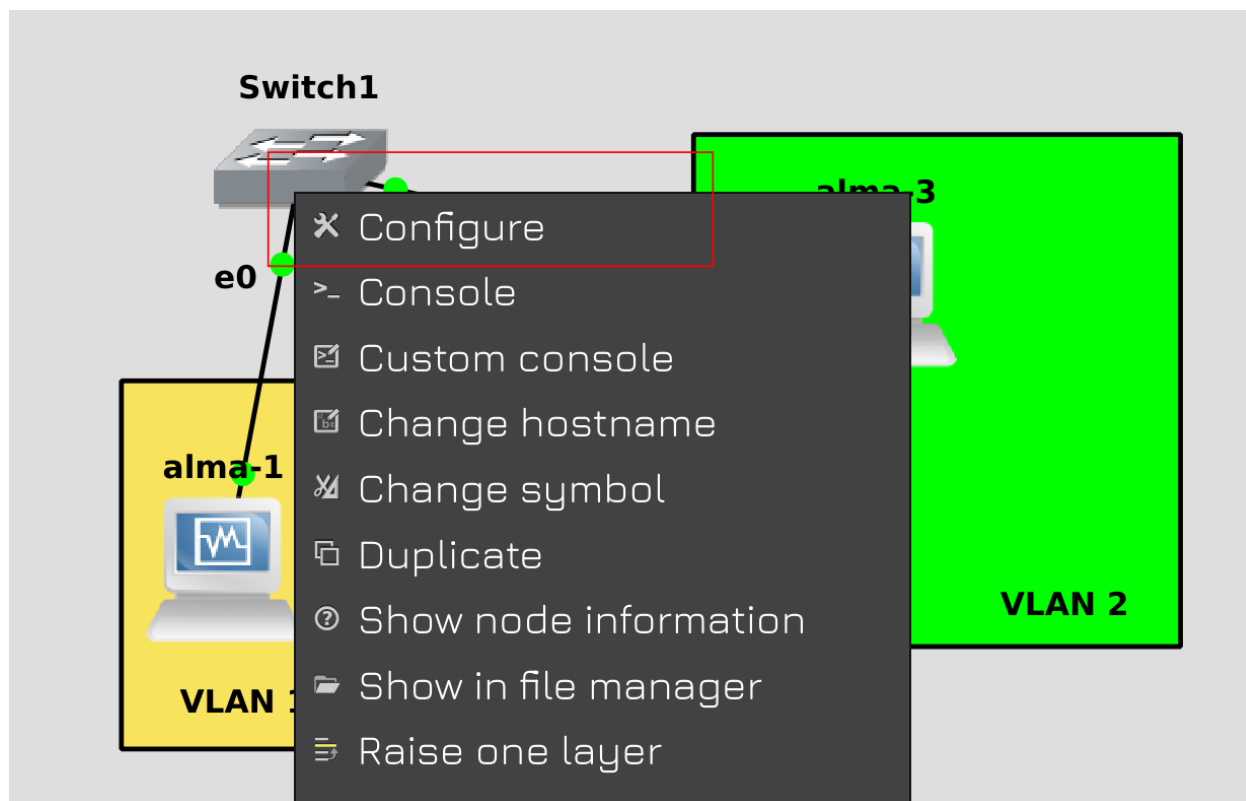
Как вы понимаете, деление зачастую происходит на уровне свитча, хотя не только и нередко на самих серверах настраиваются VLAN-ы. Они общие для сети - т.е. вы можете создать на двух разных свитчах одни и те же VLAN-ы, и тогда компьютеры, подключенные к разным свитчам, но находящиеся в одном VLAN-е будут видеть друг друга. Естественно, при условии, что свитчи связаны между собой и на обоих прописаны эти VLAN-ы.

Условно, в нарисованной схеме есть 2 VLAN-а - у одного тег 1, у второго 2. У свитча 1 на портах e0 и e1 настроен влан 1, а на порту e2 - влан 2. Есть свитч 2, у которого на порту e0 прописан влан 2, а на порту e2 - влан 1. Между первым и вторым свитчем проброшен кабель, через который должны ходить оба влана, чтобы alma3 могла видеть PC1. Alma3 подключена к свитчу 1, а чтобы достучаться до PC1, ей нужно попасть через первый свитч на второй. Порты, на которых прописаны несколько вланов, называются транк портами. Обычно их делают между двумя свитчами. Но не только.

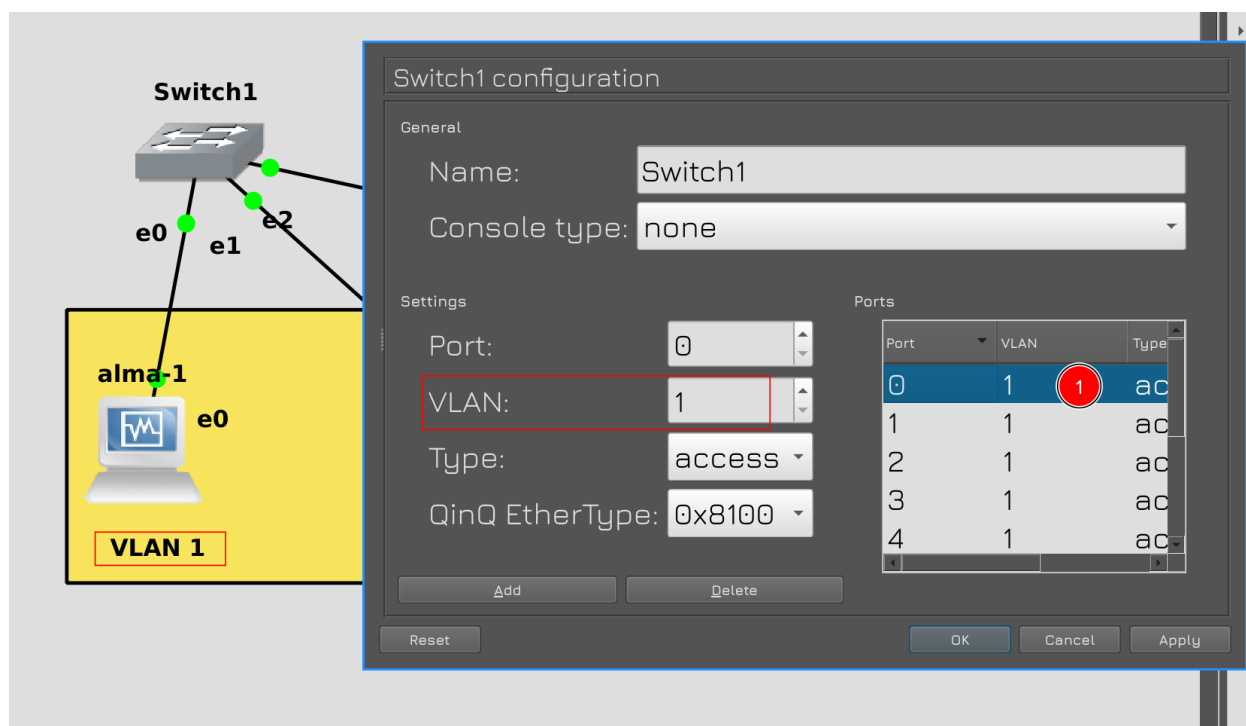
Во многих компаниях почти вся инфраструктура строится на виртуалках. Виртуалок много, они предназначены для разных задач, а значит и сеть между ними надо сегментировать. И вот часто на портах свитчей, которые идут к серверам, прописывают много вланов, а в гипервизорах в настройках сетевых адаптеров виртуалок можно указать нужный VLAN. В итоге у вас виртуалки находятся в разных VLAN-ах.



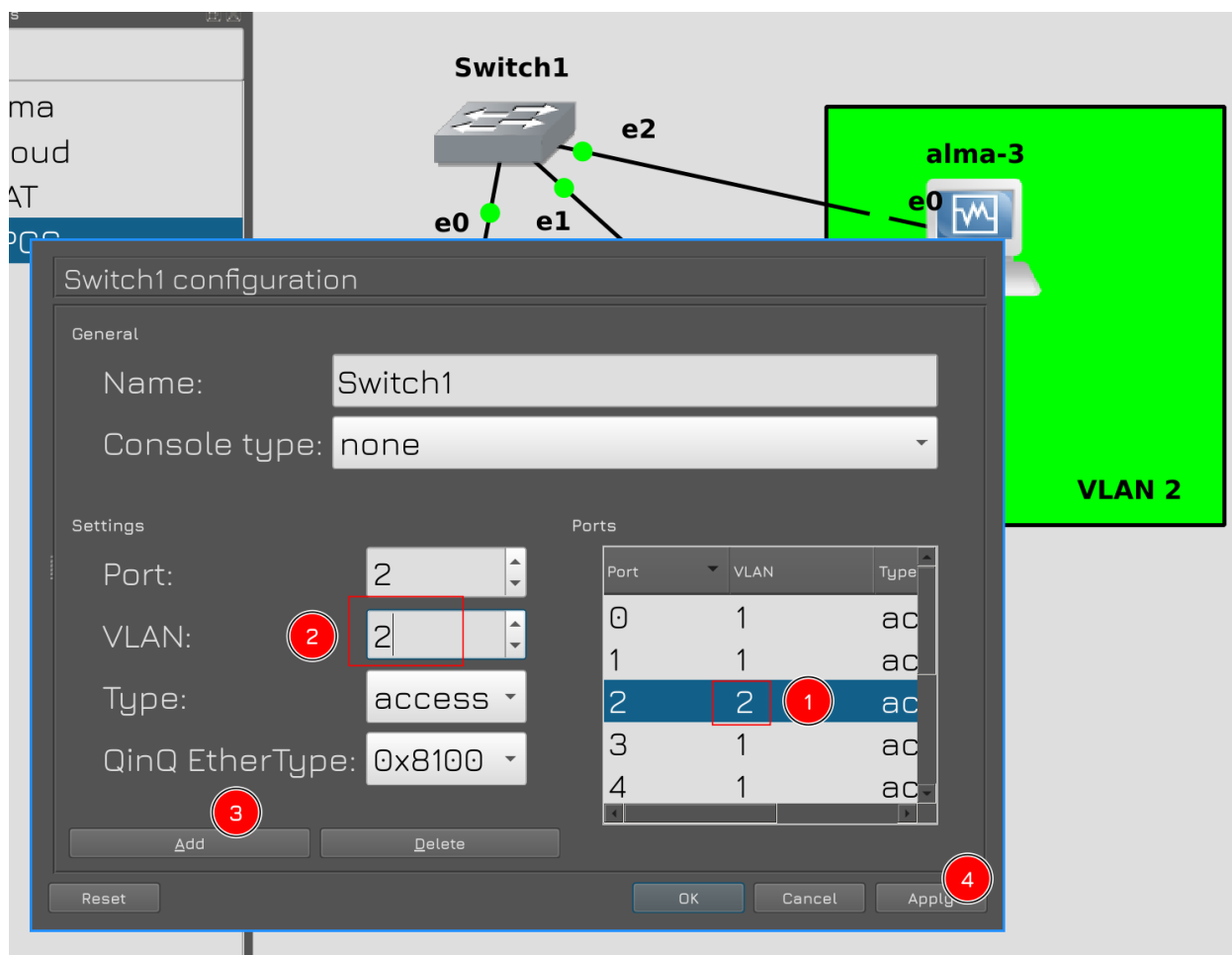
Давайте построим такую сеть, где alma1 находится в первом VLAN-е, alma3 находится во втором, а alma2 и там, и там.



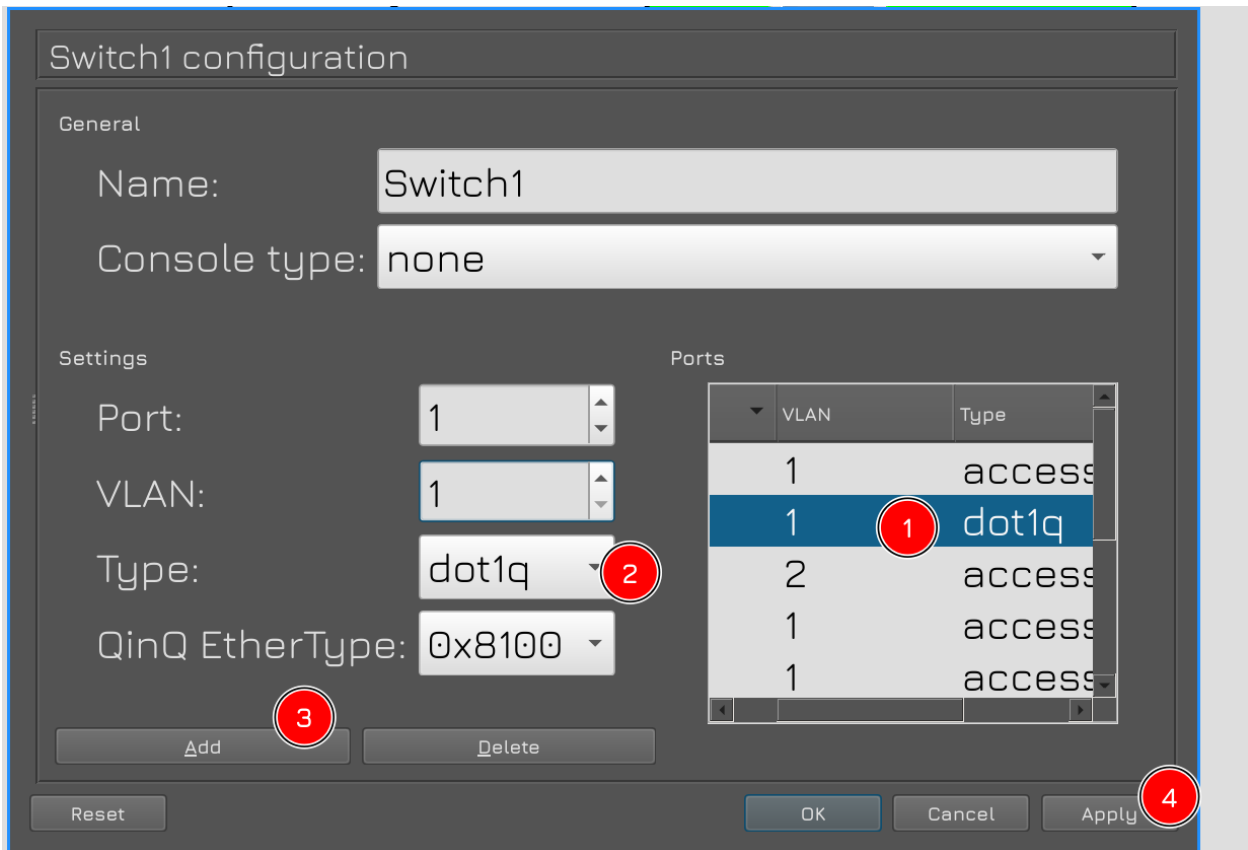
Нажмите правой кнопкой мыши на свитче и выберите Configure.



В появившемся окне в правой части будут порты, а слева можно их настраивать. Два раза ткните на порт 0, он у нас подключен к alma1. Как видите, здесь по-умолчанию прописан VLAN1 в режиме access, т.е. в нетегированном виде. Поэтому здесь мы ничего не трогаем.



Теперь касательно alma3, он подключен к порту e2. Два раза ткните на второй порт. Поменяйте там значение VLAN на 2, затем нажмите Add. Тогда в правой панели число поменяется на 2. Теперь здесь в нетегированном виде VLAN 2.



Насчёт alma2, нам нужно передавать по первому порту два влана - влан1 и влан2. Для этого дважды нажимаем на первый порт, влан не трогаем, а меняем только тип на dot1q. В стандартах VLAN-ы указаны под названием 802.1q, а dot1q - это сокращённое название. Нажимаем Add и Apply.

```

alma-1 (reset) [Running] - Oracle VM VirtualBox
[root@alma1 ~]# ping -c 1 10.0.1.3
PING 10.0.1.3 (10.0.1.3) 56(84) bytes of data.
From 10.0.1.1 icmp_seq=1 Destination Host Unreachable

--- 10.0.1.3 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms

[root@alma1 ~]# _

```

Теперь, чтобы проверить, работает ли изоляция, ещё раз с хоста 1 отправим ping, но на неизвестный IP адрес, скажем, 10.0.1.3:

```
ping -c 1 10.0.1.3
```

За этим адресом никакого хоста нет, компьютер не знает, к кому обращаться, поэтому он пошлёт ARP запрос.

```

alma-1 (reset) [Running] - Oracle VM VirtualBox
[root@alma2 ~]# tcpdump
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on enp0s3, link-type EN10MB (Ethernet), capture size 262144 bytes
22:58:46.911691 ARP, Request who-has 10.0.1.3 tell 10.0.1.1, length 46
22:58:47.932051 ARP, Request who-has 10.0.1.3 tell 10.0.1.1, length 46
22:58:48.955853 ARP, Request who-has 10.0.1.3 tell 10.0.1.1, length 46

```

Этот ARP запрос мы видим на хосте 2. Это значит, что хост 1 и хост 2 всё ещё в одном бродкаст

домене, так как оба подключены к VLANу 1.

```
alma-1 (reset) [Running] - Oracle VM VirtualBox
alma-2 (reset) [Running] - Oracle VM VirtualBox
alma-3 (reset) [Running] - Oracle VM VirtualBox
[root@alma3 ~]# tcpdump
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on enp0s3, link-type EN10MB (Ethernet), capture size 262144 bytes
```

А вот до альмы 3 запрос не дошёл, а это значит, что она не находится с хостом 1 в одном VLANе.

```
alma-1 (reset) [Running] - Oracle VM VirtualBox
alma-2 (reset) [Running] - Oracle VM VirtualBox
alma-3 (reset) [Running] - Oracle VM VirtualBox
[root@alma3 ~]# ping -c 1 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
From 10.0.0.3 icmp_seq=1 Destination Host Unreachable

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms
[root@alma3 ~]#
```

Теперь проверим работу VLANа 2. Пропингуем адрес с третьего хоста:

```
ping -c 1 10.0.0.2
```

```
alma-1 (reset) [Running] - Oracle VM VirtualBox
alma-2 (reset) [Running] - Oracle VM VirtualBox
alma-3 (reset) [Running] - Oracle VM VirtualBox
[root@alma1 ~]# tcpdump
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on enp0s3, link-type EN10MB (Ethernet), capture size 262144 bytes
_
```

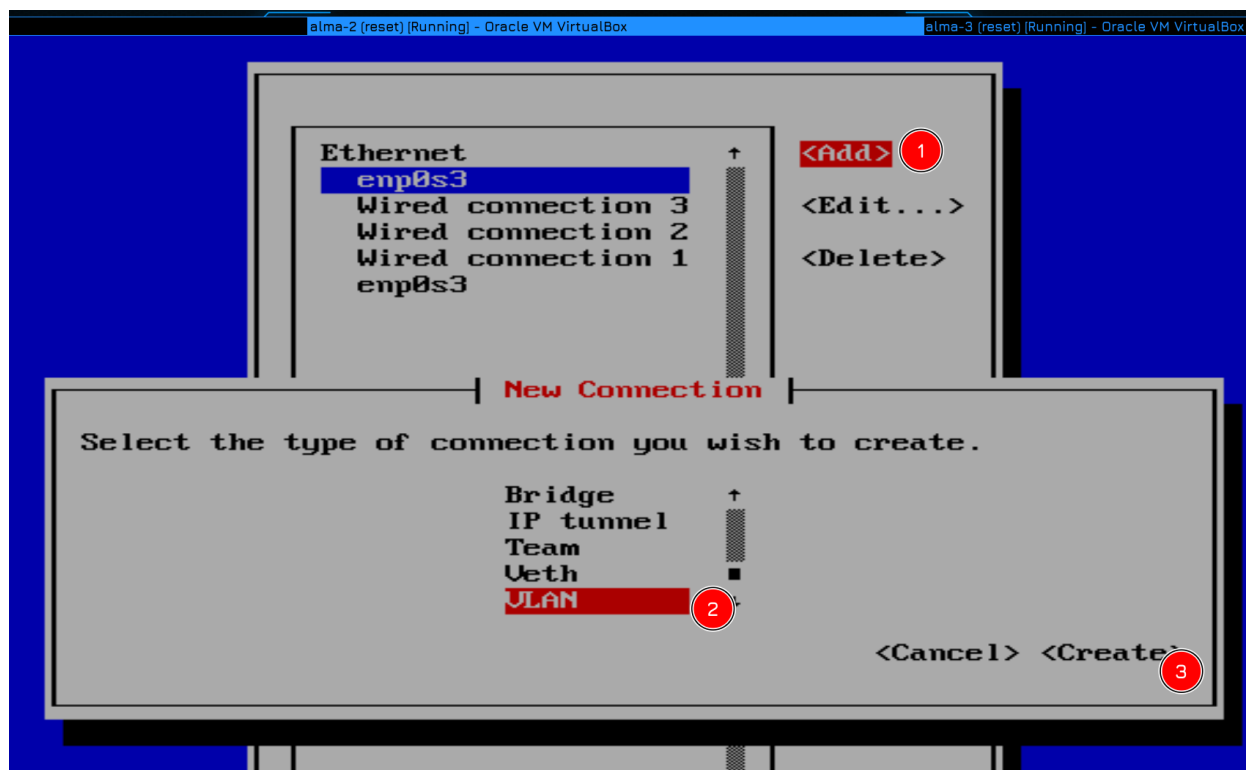
На первом хосте, как и ожидалось, tcpdump молчит - потому что он не находится с третьим хостом в одном VLANе.

```
alma-1 (reset) [Running] - Oracle VM VirtualBox
alma-2 (reset) [Running] - Oracle VM VirtualBox
alma-3 (reset) [Running] - Oracle VM VirtualBox
[root@alma2 ~]# tcpdump
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on enp0s3, link-type EN10MB (Ethernet), capture size 262144 bytes
23:08:46.787415 ARP, Request who-has 10.0.0.2 tell 10.0.0.3, length 46
23:08:47.759940 ARP, Request who-has 10.0.0.2 tell 10.0.0.3, length 46
23:08:48.784054 ARP, Request who-has 10.0.0.2 tell 10.0.0.3, length 46
```

А вот хост 2 видит и эти запросы, т.е. он в одном VLANе с хостом 3.

```
alma-1 (reset) [Running] - Oracle VM VirtualBox
alma-2 (reset) [Running] - Oracle VM VirtualBox
alma-3 (reset) [Running] - Oracle VM VirtualBox
[root@alma2 ~]# ip addr show enp0s3
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:4a:14:9e brd ff:ff:ff:ff:ff:ff
    inet 10.0.1.2/24 scope global enp0s3
        valid_lft forever preferred_lft forever
[root@alma2 ~]# _
```

Осталось настроить, чтобы хост 2 и 3 могли друг друга пинговать, т.е. чтобы сеть между ними работала и на третьем уровне. Но мы не можем просто прописать на хосте второй IP адрес - IP адрес будет не тегированным интерфейсе, а значит система будет просто отбрасывать пакет. Нам же нужно создать тегированный интерфейс.



Для этого запускаем nmtui - Edit connection - Add, в списке находим VLAN и табом переходим на Create.



Профиль можно назвать как угодно, но для ясности назовём Vlan2. В device пишем имя интерфейса, куда подключен кабель с этим вланом, затем ставим точку и пишем VLAN ID - 2, т.е. получается enp0s3.2. После нажатия tab снизу автоматом заполнятся поля Parent и Vlan id. Ну и дальше как в обычном интерфейсе - поменяем IPv4 configuration на manual, потому что у нас нет DHCP сервера, и выставим статичный IP - 10.0.0.2/24. Сохраняем и выходим из nmtui.

```

[root@alma2 ~]# 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: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default ql
0
    link/ether 08:00:27:4a:14:9e brd ff:ff:ff:ff:ff:ff
    inet 10.0.1.2/24 scope global enp0s3
        valid_lft forever preferred_lft forever
3: enp0s8: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state DOWN group default
1000
    link/ether 08:00:27:10:09:5a brd ff:ff:ff:ff:ff:ff
4: enp0s9: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state DOWN group default
1000
    link/ether 08:00:27:3f:79:a4 brd ff:ff:ff:ff:ff:ff
5: enp0s10: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state DOWN group default
n 1000
    link/ether 08:00:27:98:9a:49 brd ff:ff:ff:ff:ff:ff
6: enp0s3.2@enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group de
qlen 1000
    link/ether 08:00:27:4a:14:9e brd ff:ff:ff:ff:ff:ff
    inet 10.0.2/24 brd 10.0.0.255 scope global noprefixroute enp0s3.2
        valid_lft forever preferred_lft forever
    inet6 fe80::98a0:92e9:31ef:bfe4/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
[root@alma2 ~]# _

```

Теперь, посмотрев список интерфейсов:

```
ip a
```

можно заметить, что появился новый интерфейс - enp0s3.2 - и на нём прописан IP адрес.

```

alma-1 (reset) [Running] - Oracle VM VirtualBox
alma-2 (reset) [Running] - Oracle VM VirtualBox
[root@alma2 ~]# ping -c 1 10.0.0.3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=1.51 ms

--- 10.0.0.3 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.514/1.514/1.514/0.000 ms
[root@alma2 ~]# ping -c 1 10.0.1.1
PING 10.0.1.1 (10.0.1.1) 56(84) bytes of data.
64 bytes from 10.0.1.1: icmp_seq=1 ttl=64 time=1.63 ms

--- 10.0.1.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.634/1.634/1.634/0.000 ms
[root@alma2 ~]# _

```

Попробуем пингануть hosts 1 и 3:

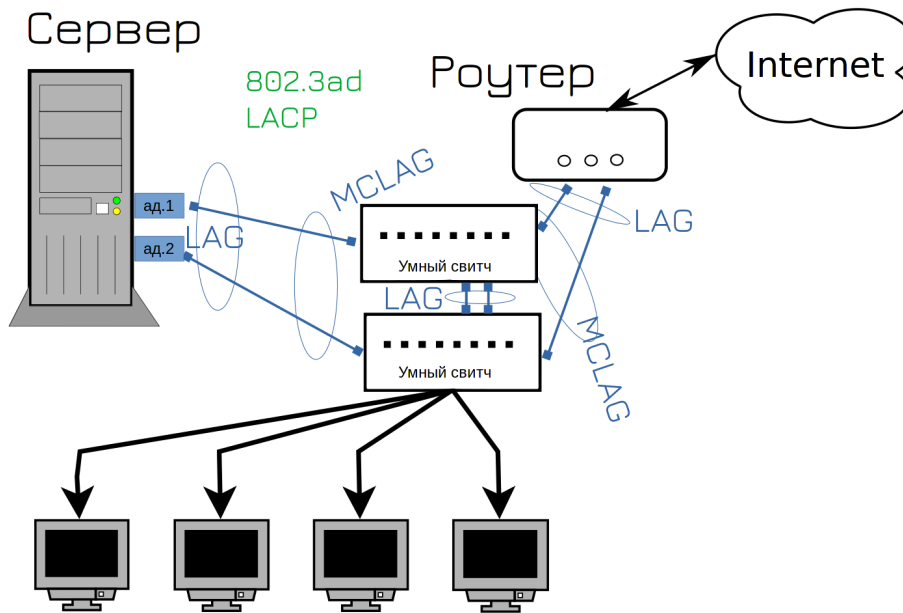
```
ping -c 1 10.0.0.3
ping -c 1 10.0.1.1
```

alma2 видит оба хоста, т.е. она находится в двух разных бродкаст доменах, как мы этого и хотели. Таким же макаром мы могли бы добавить ещё несколько вланов, но этим уже займитесь вы.

Давайте подведём итоги. Сегодня мы с вами немного разобрали L2, поговорили о свитчах, которые там обитают и познакомились с такими понятиями, как broadcast, arp, vlan-ы и даже немного разобрали маркировку. Также мы научились прописывать VLAN-ы на сервере - а это было главным смыслом данного урока - понимать, зачем это нужно и как это делать. Мы это сделали через одну утилиту nmtui, на деле же есть и другие способы, но это лишь инструменты.

2.4 04. Объединение портов

2.4.1 04. Объединение портов



Ещё одна задача, с которой администратор сталкивается с точки зрения взаимодействия со свитчем - настройка агрегации каналов, т.е. объединение нескольких портов. Мы об этом говорили в первой теме, мол, нужно использовать два сетевых адаптера и два свитча. На самом деле можно больше портов и проводов, хоть 4, хоть 8. Но, обычно, это либо 2, либо 4.

Bonding:- Bonding refers to create a bond or aggregation of two or more links into one in order to provide failover and redundancy to the network.

Teaming:- Is a new way implementation of bonding and provides a separate driver. We have both drivers available as of now, and can implement either of them as per requirement.

Advantages of Teaming

1. Support for IPv6 (NA/NS) link monitoring
2. Teaming has a *small kernel module* which provides the same implementation with more features.
3. Less performance overhead
4. By default, Teamed interface is capable of working with D-Bus and Unix Domain Sockets.
5. *Full userspace runtime control*
6. Modular design allows function extensible option in future
7. **Load balancing feature** added for LACP support
8. Use of NetworkManager(nmcli) to manage our network connections

Teaming provides more features and a way towards automation as *network manager can handle the interfaces with modular design* and hence considered a modern way of handling the

Агрегацию каналов называют по разному - к серверам обычно относят понятия NIC teaming, bonding, а к свитчам etherchannel, portchannel, lag и т.п. Но обычно все под этим подразумевают одно и тоже. Правда в линуксах есть два модуля, раньше был только bonding, но потом добавили новый и более продвинутый вариант - тиминг.

Table 8.1. A Comparison of Features in Bonding and Team

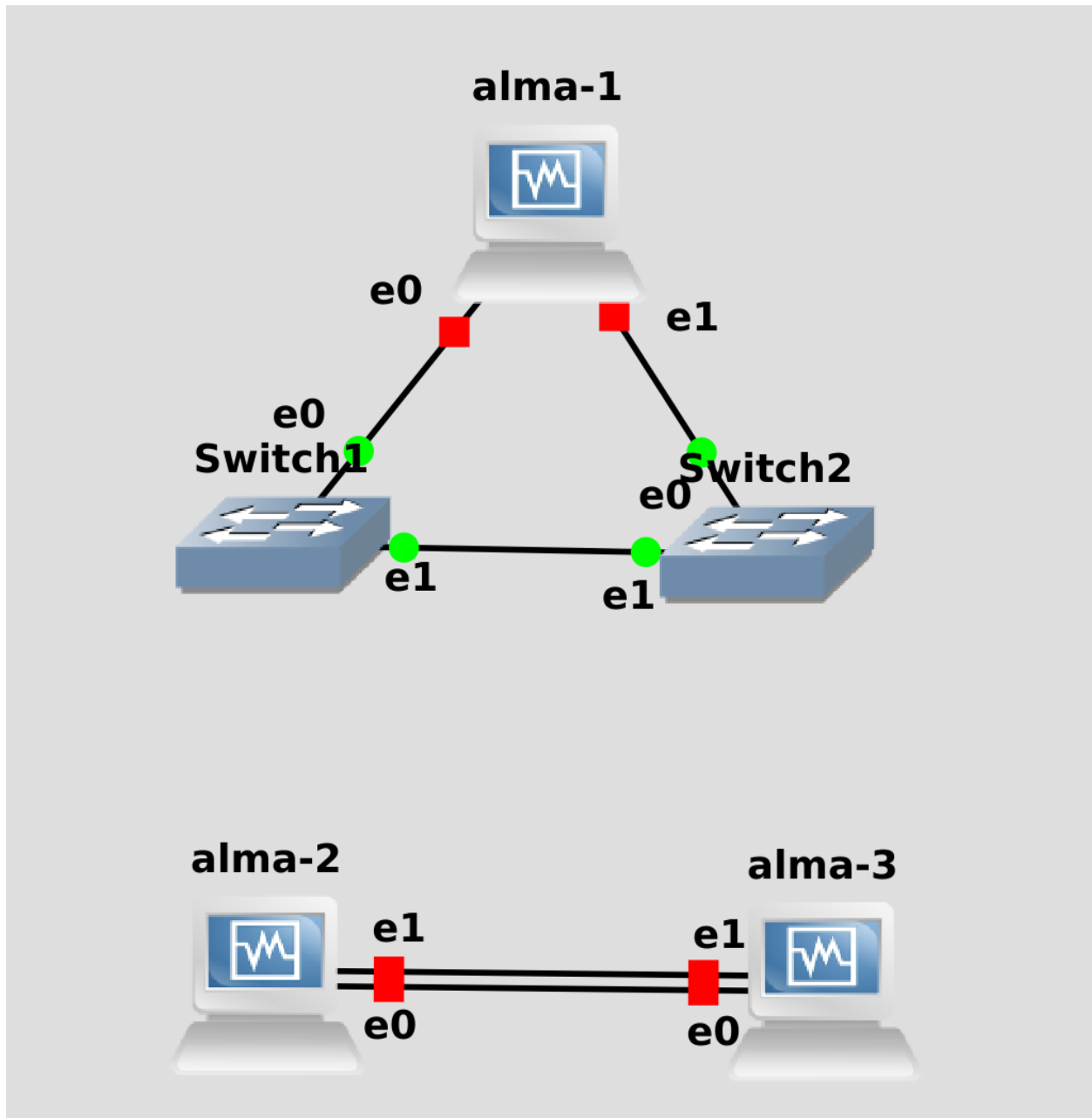
Feature	Bonding	Team
broadcast Tx policy	Yes	Yes
round-robin Tx policy	Yes	Yes
active-backup Tx policy	Yes	Yes
LACP (802.3ad) support	Yes (active only)	Yes
Hash-based Tx policy	Yes	Yes
User can set hash function	No	Yes
Tx load-balancing support (TLB)	Yes	Yes
LACP hash port select	Yes	Yes
load-balancing for LACP support	No	Yes
Ethtool link monitoring	Yes	Yes
ARP link monitoring	Yes	Yes
NS/NA (IPv6) link monitoring	No	Yes
ports up/down delays	Yes	Yes
port priorities and stickiness (" primary " option enhancement)	No	Yes
separate per-port link monitoring setup	No	Yes
multiple link monitoring setup	Limited	Yes
lockless Tx/Rx path	No (rwlock)	Yes (RCU)
VLAN support	Yes	Yes
user-space runtime control	Limited	Full
Logic in user-space	No	Yes
Extensibility	Hard	Easy
Modular design	No	Yes
Performance overhead	Low	Very Low
D-Bus interface	No	Yes
multiple device stacking	Yes	Yes
zero config using LLDP	No	(in planning)
NetworkManager support	Yes	Yes

В итоге при настройке вы будете наткаться и на bonding, и на teaming, оба делают одно и то же, просто teaming немного лучше.

LAG даёт сразу два плюса - во-первых, повышает отказоустойчивость, во-вторых - пропускную способность. Насчёт второго, кстати, не любая агрегация портов это позволяет, а только конкретная реализация - LACP. При этом не путайте скорость с пропускной способностью. Для одного соединения максимальная скорость будет всё равно скоростью одного линка. Т.е. условно, если будете качать один файл - то скорость будет максимум гигабит, а если несколько файлов - то они распределятся по портам и суммарно выйдет примерно два гигабита. Ну и нельзя делать LACP между портами с разными скоростями.

LACP нужно настраивать с двух сторон - т.е. и на сервере, и на свитче. А на тупых свитчах этого не настроить - соответственно, для LACP нужны управляемые свитчи. Но это обычно не проблема - сервера к тупым свитчам цепляют разве что в совсем грустных компаниях.

Но именно в такой ситуации мы с вами и оказались - свитчи, которые даёт GNS - почти что тупые. С одной стороны в них есть настройка VLAN-ов - а это нельзя сделать на неуправляемых свитчах. С другой - это всё, что они могут. Может быть в будущем мы с вами в GNS поднимем другие свитчи, на линуксах, но пока будем довольствоваться этими.



Поэтому в нашей схеме вместо LACP мы будем использовать «activebackup» - в таком случае из двух соединений будет работать только одно. Но для примера мы настроим LACP между двумя хостами, просто чтобы было понимание.

```
[root@alma1 ~]# 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: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:a4:41:4c brd ff:ff:ff:ff:ff:ff
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:2b:fe:47 brd ff:ff:ff:ff:ff:ff
4: enp0s9: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state DOWN group default qlen 1000
    link/ether 08:00:27:7a:ff:12 brd ff:ff:ff:ff:ff:ff
5: enp0s10: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state DOWN group default qlen 1000
    link/ether 08:00:27:52:c3:8e brd ff:ff:ff:ff:ff:ff
[root@alma1 ~]#
```

Для начала запустим alma1. Нам опять придётся пользоваться консолью, но скоро мы пересядем на удобный ssh. Посмотрим список активных интерфейсов:

```
ip a
```

Так как мы подключили 0 и 1 адаптеры в GNS, то первые два интерфейса в системе будут в UP-е - enp0s3 и enp0s8. Используя их нам нужно будет создать один логический интерфейс.

```
[root@alma1 ~]# nmcli connection show
NAME                                UUID                                TYPE    DEVICE
enp0s3                             40eee77d-a591-4beb-bca3-91fd110170d5 ethernet enp0s3
Wired connection 2                  26bfb896-3496-3826-9456-ee1270868d13 ethernet --
Wired connection 1                  a93f9786-88c0-35c5-93f7-1f6222f89771 ethernet --
Wired connection 3                  359fe43c-d472-3353-807a-e4554b4c8a61 ethernet --
[root@alma1 ~]#
```

Сегодня, вместо nmtui мы будем пользоваться nmcli - командной версией этой утилиты. Это будет немного нагляднее. Помните, заходя в nmtui мы в edit connection видели список профилей сетей? В командной строке их можно увидеть с помощью команды:

```
nmcli connection show
```

Network manager по-умолчанию создал для активных интерфейсов профили - и по столбику Device их можно распознать. Нам нужно будет создать новый профиль, в котором мы будем использовать эти устройства. И чтобы профили друг с другом не конфликтовали, эти надо удалить. Можно даже все 4 удалить.

```
[root@alma1 ~]# nmcli connection delete enp0s3 Wired\ connection\ 1 Wired\ connection\ 2 Wired\ connection\ 3
Connection 'enp0s3' (4bfcfe30-2fe4-4aa5-a242-815672e9fb2e) successfully deleted.
Connection 'Wired connection 1' (68599276-0b11-3904-b848-c5d5f995f05a) successfully deleted.
Connection 'Wired connection 2' (4fdf1d09-05ed-499b-bf22-e5ec3aa101ad) successfully deleted.
Connection 'Wired connection 3' (c838a3da-8372-426c-85a9-1a9474584a4f) successfully deleted.
[root@alma1 ~]# nmcli connection show
[root@alma1 ~]#
```

Для этого используем опцию delete и указываем все профили через пробел:

```
nmcli connection delete enp0s3 Wired\ connection\ 1 Wired\ connection\ 2 Wired\
connection\ 3
nmcli con sh
```

Как видите, теперь профилей не осталось.

```
[root@alma1 ~]# nmcli connection add type team con-name teams3s8 ifname myteam
Connection 'teams3s8' (12e0feb1-24bc-4ca2-81be-5ec4ec8c36f2) successfully added.
[root@alma1 ~]# nmcli connection show
NAME          UUID                                  TYPE    DEVICE
teams3s8      12e0feb1-24bc-4ca2-81be-5ec4ec8c36f2 team    myteam
[root@alma1 ~]# _
```

Теперь надо создать новый профиль. Для этого используем опцию add:

```
nmcli connection add type team con-name teams3s8 ifname myteam
```

При создании мы должны указать тип интерфейса. Помните, мы при создании VLAN-а в списке выбирали VLAN? А тут мы выбираем team. Дальше даём имя профилю с помощью опции con-name. Пусть будет teams3s8. Ну и помните при создании VLAN-а мы указывали device? Мы создаём логический интерфейс и можем назвать его как угодно. Назовём myteam. Ну и посмотрим, что получилось:

```
nmcli con sh
```

```
[root@alma1 ~]# nmcli connection add type team-slave ifname enp0s3 master myteam
Connection 'team-slave-enp0s3' (0c3b4fe4-68ea-4e8c-9004-426884dd4477) successfully added.
[root@alma1 ~]# nmcli connection add type team-slave ifname enp0s8 master myteam
Connection 'team-slave-enp0s8' (1659f0c9-3d67-4f06-a8a8-e5c21fe69ffe) successfully added.
[root@alma1 ~]# nmcli connection show
NAME          UUID                                  TYPE    DEVICE
teams3s8      12e0feb1-24bc-4ca2-81be-5ec4ec8c36f2 team    myteam
team-slave-enp0s3 0c3b4fe4-68ea-4e8c-9004-426884dd4477 ethernet enp0s3
team-slave-enp0s8 1659f0c9-3d67-4f06-a8a8-e5c21fe69ffe ethernet enp0s8
[root@alma1 ~]#
```

Теперь нужно привязать enp0s3 и enp0s8 к этому тимиинг интерфейсу. Для этого надо создать два новых профиля с типом team-slave:

```
nmcli con add type team-slave ifname enp0s3 master myteam
nmcli con add type team-slave ifname enp0s8 master myteam
```

В них указываем имена интерфейсов, которые будет входить в группу портов - ifname enp0s3 и enp0s8. Также после master указываем имя общего интерфейса - myteam.

Ещё раз проверим, что получилось:

```
nmcli con sh
```

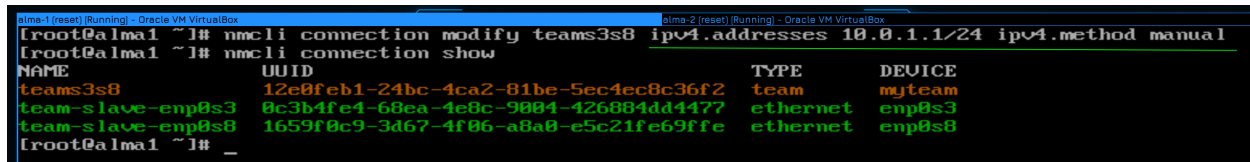
Как видите, теперь тут 3 профиля.

Теперь всё что касается IP адресов и прочего нужно настраивать на мастер интерфейсе, т.е. на профиле teams3s8.

```
[root@alma1 ~]# nmcli connection modify teams3s8 team.runner
activebackup broadcast lacp loadbalance random roundrobin
[root@alma1 ~]# nmcli connection modify teams3s8 team.runner activebackup
[root@alma1 ~]#
```

Начнём с раннера. Это, собственно, механизм, как несколько интерфейсов будут работать вместе. Тот же самый lacp или activebackup. Есть и другие раннеры, но они специфичны и используются не так часто. Можете почитать о них по [ссылке](#). В нашей ситуации нужно выбрать activebackup. И так, в профиле teams3s8 меняем runner на нужный:

```
nmcli con modify teams3s8 team.runner activebackup
```



```
alma-1 (reset) [Running] - Oracle VM VirtualBox
[root@alma1 ~]# nmcli connection modify teams3s8 ipv4.addresses 10.0.1.1/24 ipv4.method manual
[root@alma1 ~]# nmcli connection show
NAME                UUID                                  TYPE      DEVICE
teams3s8            12e8feb1-24bc-4ca2-81bc-5ec4ec8c36f2  team      myteam
team-slave-enp8s3    8c3b4fe4-68ea-4e8c-9884-426884dd4477  ethernet  enp8s3
team-slave-enp8s8    1659f8c9-3d67-4f86-a8a8-e5c21fe69ffe  ethernet  enp8s8
[root@alma1 ~]# _

alma-2 (reset) [Running] - Oracle VM VirtualBox
[root@alma1 ~]# nmcli connection modify teams3s8 ipv4.addresses 10.0.1.1/24 ipv4.method manual
```

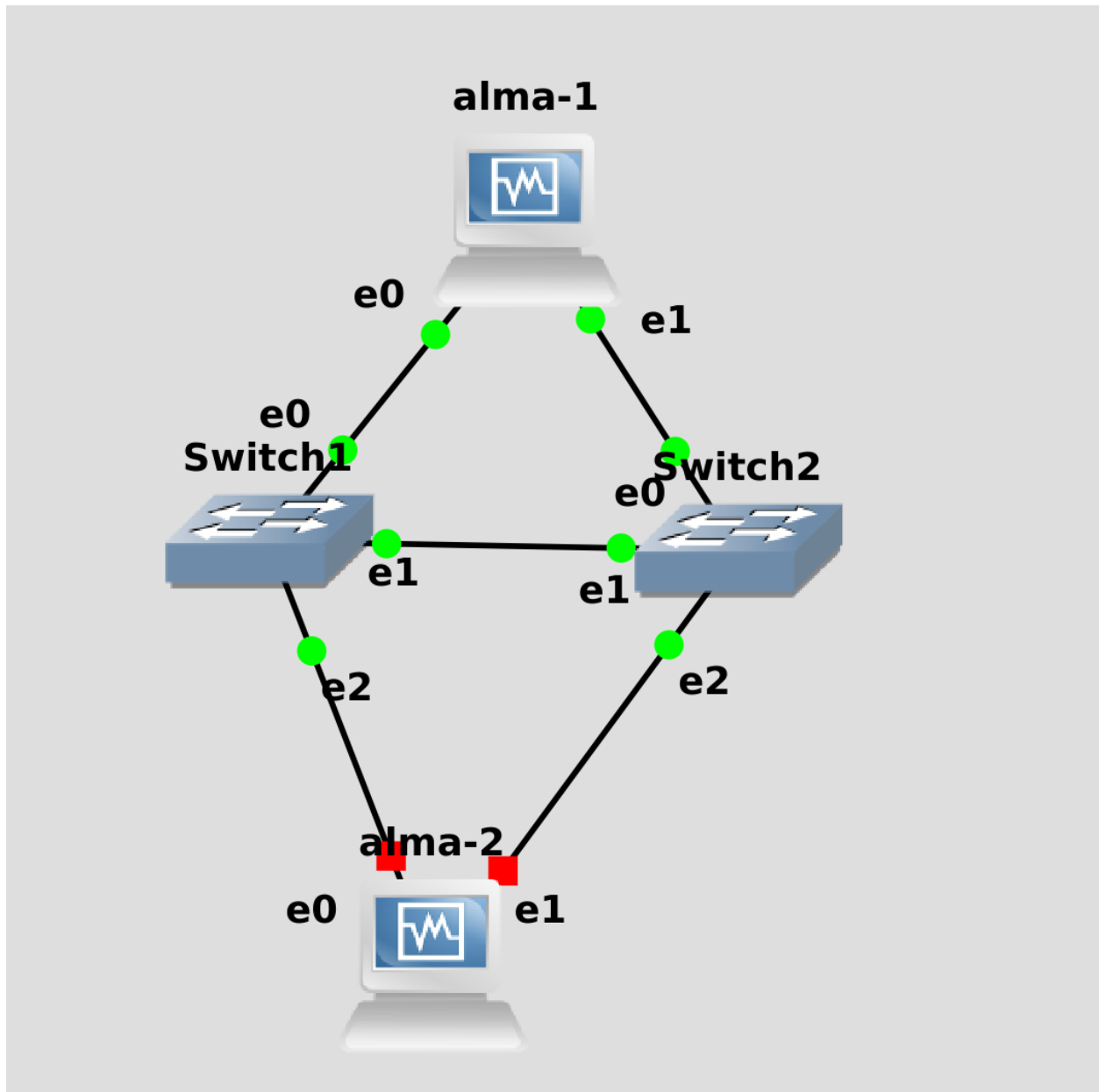
Ну и напоследок пропишем IP адрес в этом профиле:

```
nmcli con mod teams3s8 ipv4.addresses 10.0.1.1/24 ipv4.method manual
```

Также мы должны указать метод получения IP - manual. Помните, мы в nmtui меняли Automatic на manual? То же самое.

Чтобы увидеть всевозможные настройки интерфейса, можно посмотреть его с помощью опции show:

```
nmcli con show teams3s8
```



Мы вроде всё настроили, но для проверки нужен второй хост. Давайте подключим к этой же сети alma2 и на ней настроим такую же сеть.

```

[root@alma2 ~]# nmcli connection show
NAME                UUID                                  TYPE      DEVICE
enp0s3              40eee77d-a591-4beb-bca3-91fd110170d5 ethernet  enp0s3
Wired connection 2  b97afe6c-2f4e-3268-b2d2-184eed21ae1c ethernet  enp0s8
ULAN2               d276f4c9-e3da-4d86-abcd-e06b601123f7 vlan      enp0s3.2
Wired connection 1  105c423a-169a-3f11-816d-e49614fecc0 ethernet  --
Wired connection 3  dea3d258-9fe2-3ff0-b97f-a9948a23e9b3 ethernet  --
[root@alma2 ~]# nmcli connection delete enp0s3 Wired\ connection\ 2 ULAN2 Wired\ connection\ 1 Wired\ connection\ 3
Connection 'enp0s3' (40eee77d-a591-4beb-bca3-91fd110170d5) successfully deleted.
Connection 'Wired connection 2' (b97afe6c-2f4e-3268-b2d2-184eed21ae1c) successfully deleted.
Connection 'ULAN2' (d276f4c9-e3da-4d86-abcd-e06b601123f7) successfully deleted.
Connection 'Wired connection 1' (105c423a-169a-3f11-816d-e49614fecc0) successfully deleted.
Connection 'Wired connection 3' (dea3d258-9fe2-3ff0-b97f-a9948a23e9b3) successfully deleted.
[root@alma2 ~]# nmcli connection show
[root@alma2 ~]#
  
```

Для начала посмотрим список всех профилей и удалим их:

```
nmcli con show
nmcli con del enp0s3 Wired\ connection\ 2 VLAN2 Wired\ connection\ 1 Wired\ connection\ 3
nmcli con sh
```

```

[alma-1(reset)(Running) - Oracle VM VirtualBox]
[alma-2(reset)(Running) - Oracle VM VirtualBox]
[root@alma2 ~]# nmcli connection add type team con-name teams3s8 ifname myteam team.runner active
kup ipv4.addresses 10.0.1.2/24 ipv4.method manual
Connection 'teams3s8' (9118ffea-3b26-4dd3-ab69-e4f7470e0862) successfully added.
[root@alma2 ~]# nmcli connection add type team-slave ifname enp0s3 master myteam
Connection 'team-slave-enp0s3' (0c27a620-b20a-483d-8f3f-6099560750ad) successfully added.
[root@alma2 ~]# nmcli connection add type team-slave ifname enp0s8 master myteam
Connection 'team-slave-enp0s8' (25d60ce9-fffa-4669-bb46-05508d73ba2e) successfully added.
[root@alma2 ~]# nmcli connection sh
NAME                                UUID                                TYPE                                DEVICE
teams3s8                           9118ffea-3b26-4dd3-ab69-e4f7470e0862 team                                myteam
team-slave-enp0s3                   0c27a620-b20a-483d-8f3f-6099560750ad ethernet                          enp0s3
team-slave-enp0s8                   25d60ce9-fffa-4669-bb46-05508d73ba2e ethernet                          enp0s8
[root@alma2 ~]#

```

Затем можно в одну строчку создать тими́нг интерфейс со всеми нужными параметрами:

```
nmcli con add type team con-name teams3s8 ifname myteam team.runner activebackup ipv4.  
addresses 10.0.1.2/24 ipv4.method manual
```

Ну и добавим к этому мастеру слейв интерфейсы:

```
nmcli con add type team-slave ifname enp0s3 master myteam
nmcli con add type team-slave ifname enp0s8 master myteam
```

И посмотрим, что получилось:

```
nmcli con show
```

```

alma-1 (reset) (running) - Oracle VM VirtualBox
[root@alma1 ~]# nmcli connection up teams3s8
Connection successfully activated (master waiting for slaves) (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/162)
[root@alma1 ~]# ip a show myteam
57: myteam: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 08:00:27:a4:41:4c brd ff:ff:ff:ff:ff:ff
    inet 10.0.1.1/24 brd 10.0.1.255 scope global noprefixroute myteam
        valid_lft forever preferred_lft forever
    inet6 fe80::3ed7:b414:2927:51a1/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
[root@alma1 ~]#

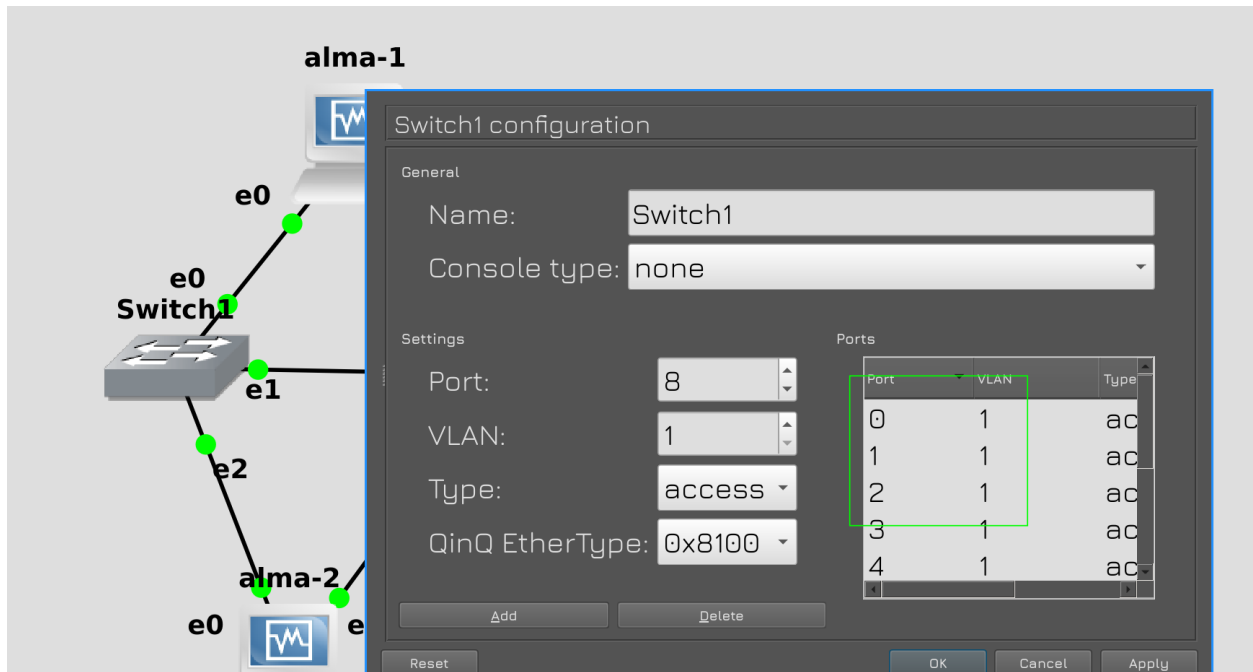
```

Осталось на обоих хостах поднять профиль, чтобы настройки применились:

```
nmcli con up teams3s8
```

Убедиться, что появился интерфейс и на нём есть IP адрес:

```
ip a show myteam
```



Ну и вернём настройки свитча в исходное положение, т.е. уберём VLAN-ы и транк интерфейсы.

```

alma-1 (reset) [Running] - Oracle VM VirtualBox
[root@alma1 ~]# ping 10.0.1.2
PING 10.0.1.2 (10.0.1.2) 56(84) bytes of data.
64 bytes from 10.0.1.2: icmp_seq=1 ttl=64 time=1.52 ms
64 bytes from 10.0.1.2: icmp_seq=2 ttl=64 time=1.46 ms
^C
--- 10.0.1.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 1.463/1.489/1.515/0.026 ms
[root@alma1 ~]#

```

Теперь попробуем пингануть альму 2 с первого хоста:

ping 10.0.1.2

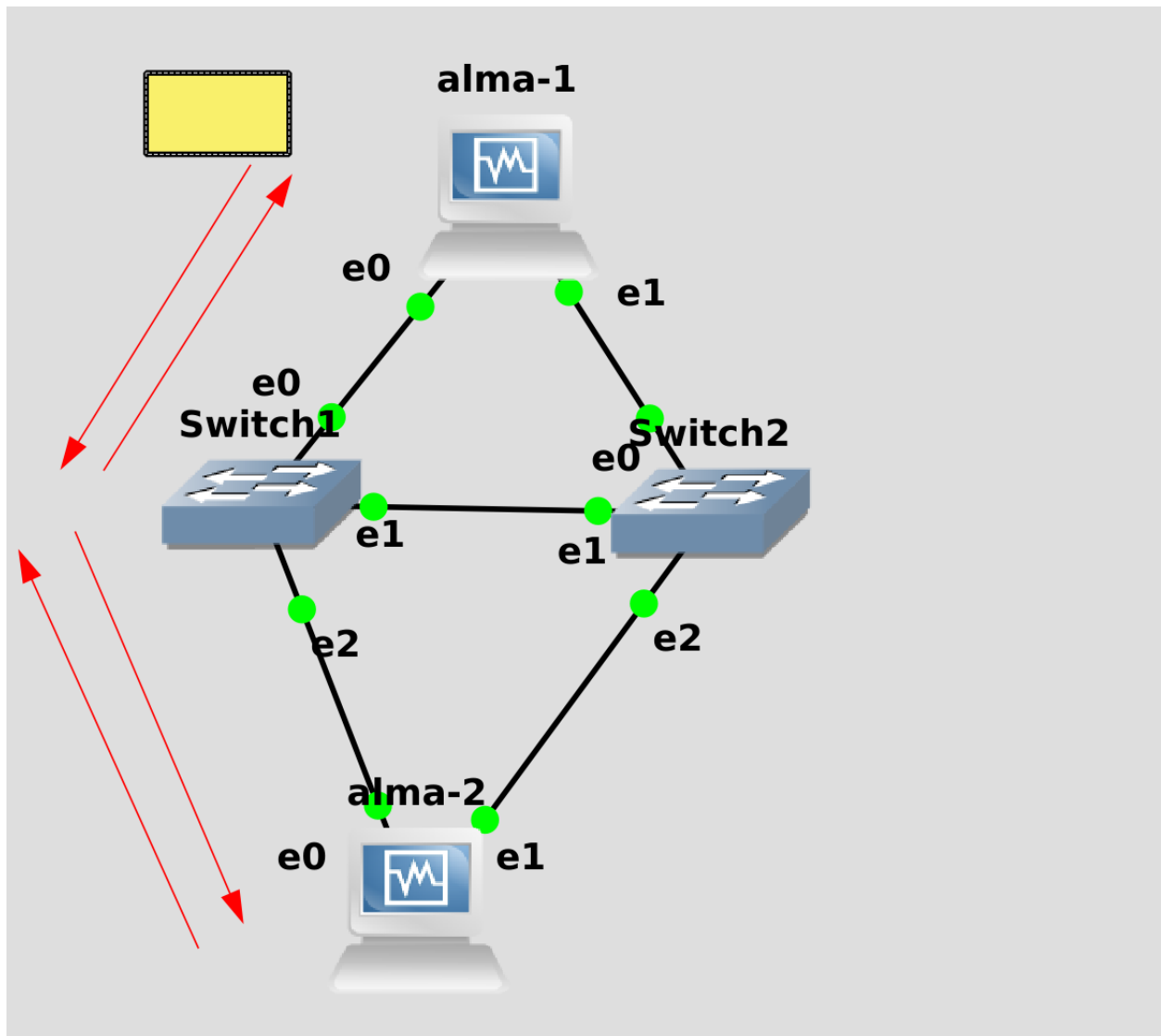
Всё пингуется, значит связь есть.

```
alma-1 (reset) [Running] - Oracle VM VirtualBox
[root@alma1 ~]# teamdctl myteam state
setup:
  runner: activebackup
ports:
  enp0s3
    link watches:
      link summary: up
      instance[link_watch_0]:
        name: ethtool
        link: up
        down count: 1
  enp0s8
    link watches:
      link summary: up
      instance[link_watch_0]:
        name: ethtool
        link: up
        down count: 0
runner:
  active port: enp0s3
[root@alma1 ~]# _
```

Мы говорили, что при activebackup работает только один интерфейс. С помощью утилиты teamdctl мы можем посмотреть информацию о тиминг интерфейсе:

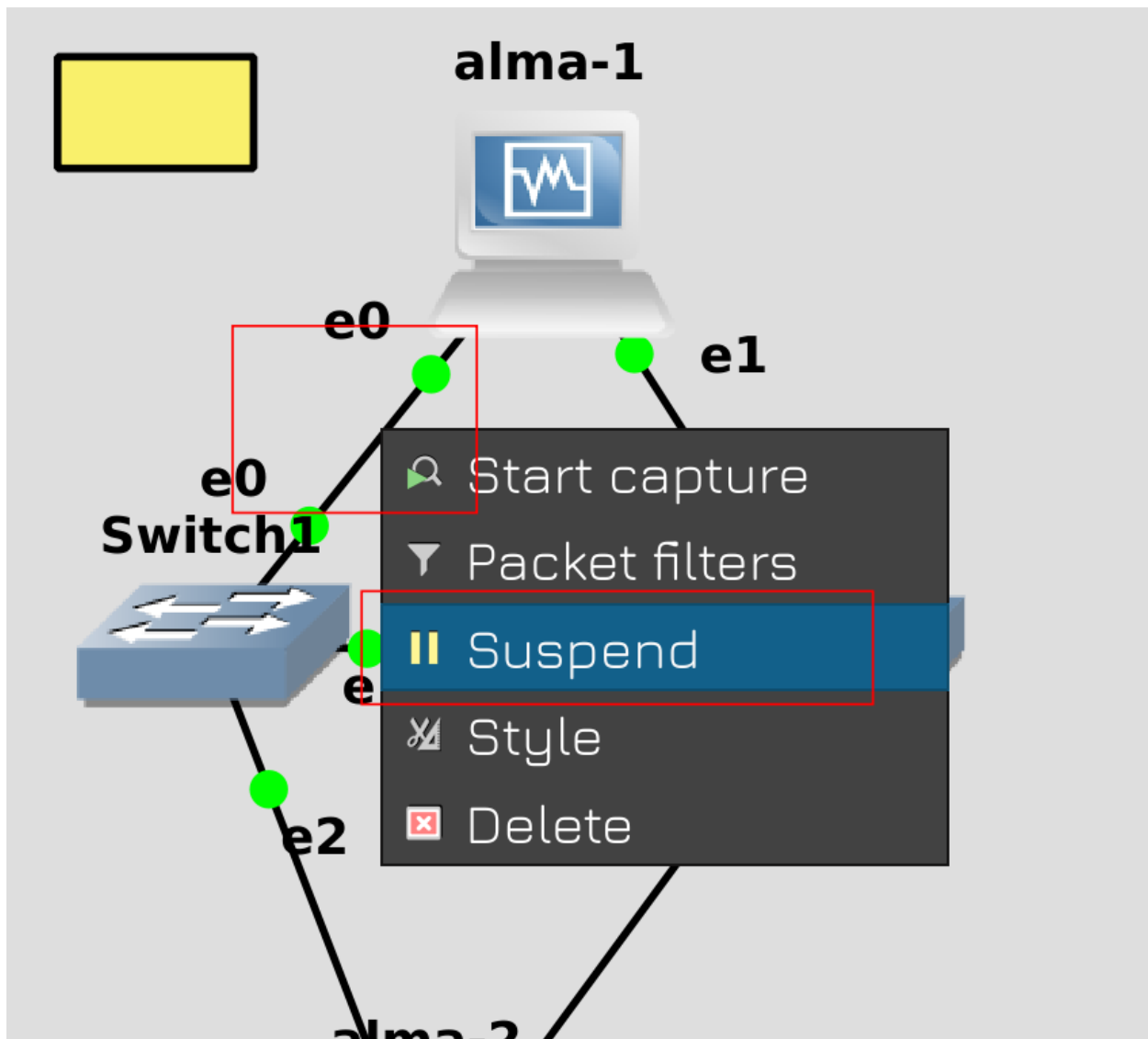
```
teamdctl myteam state
```

Обратите внимание, что утилите нужно указывать имя интерфейса, а не имя профиля. Здесь мы видим, что для этого интерфейса в качестве раннера используется activebackup, что оба интерфейса в апе, и что активным является enp0s3. На второй альме ситуация та же, т.е. активный интерфейс - enp0s3.



Мы помним, что интерфейс enp0s3 - это e0 в GNS. Оба хоста этими интерфейсами идут к свитчу 1, т.е. сейчас всё проходит через свитч 1.

Для наглядности, давайте в GNS «испортим» провод, ведущий от альмы 1 к первому свитчу. Но сперва запустим пинг к хосту 2, чтобы увидеть, как себя ведёт сеть.



Затем нажмите правой кнопкой мыши на кабеле и нажмите Suspend.

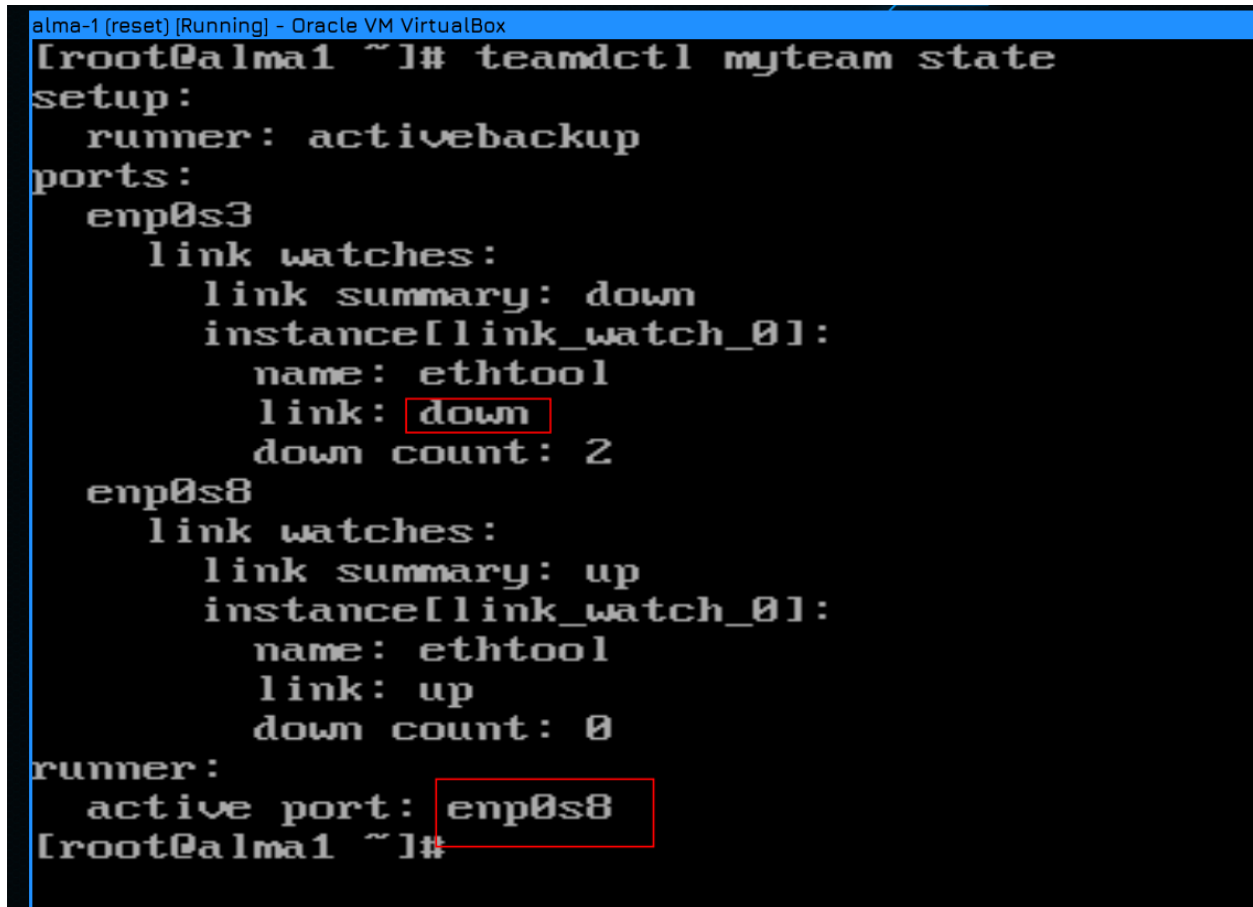
```

alma-1 (reset) [Running] - Oracle VM VirtualBox
[ root@alma1 ~ ]# ping 10.0.1.2
PING 10.0.1.2 (10.0.1.2) 56(84) bytes of data.
64 bytes from 10.0.1.2: icmp_seq=1 ttl=64 time=1.45 ms
64 bytes from 10.0.1.2: icmp_seq=2 ttl=64 time=1.53 ms
64 bytes from 10.0.1.2: icmp_seq=3 ttl=64 time=1.43 ms
64 bytes from 10.0.1.2: icmp_seq=4 ttl=64 time=1.59 ms
[ 9584.223665] e1000 0000:00:03:00 enp0s3: Reset adapter
64 bytes from 10.0.1.2: icmp_seq=6 ttl=64 time=1.75 ms
64 bytes from 10.0.1.2: icmp_seq=7 ttl=64 time=1.71 ms
64 bytes from 10.0.1.2: icmp_seq=8 ttl=64 time=1.70 ms
64 bytes from 10.0.1.2: icmp_seq=9 ttl=64 time=1.51 ms
^C
--- 10.0.1.2 ping statistics ---
 9 packets transmitted, 8 received, 11.111% packet loss, time 8060ms
rtt min/avg/max/mdev = 1.425/1.582/1.748/0.122 ms
[ root@alma1 ~ ]# _

```

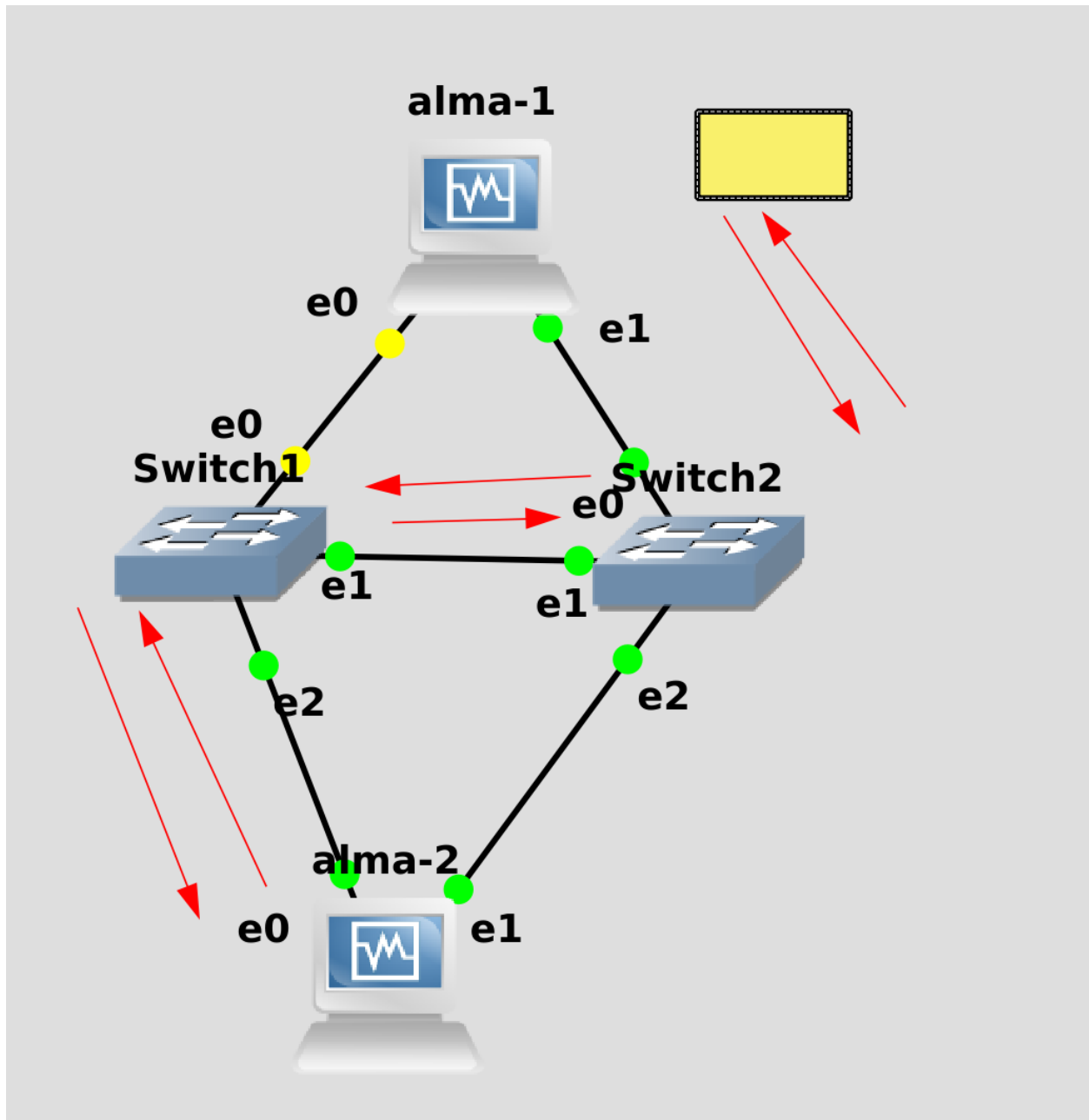
Вернёмся к альме 1. Как видите, в момент пинга мы увидели ошибку с адаптером enp0s3, а затем пинг продолжился. После остановки пинга в результатах видно, что была потеря пакетов, но небольшая -

всего 1 пинг. Для большинства случаев потерять пару пакетов не критично, сеть в целом сделана так, чтобы потерянные пакеты потом в любом случае дошли.

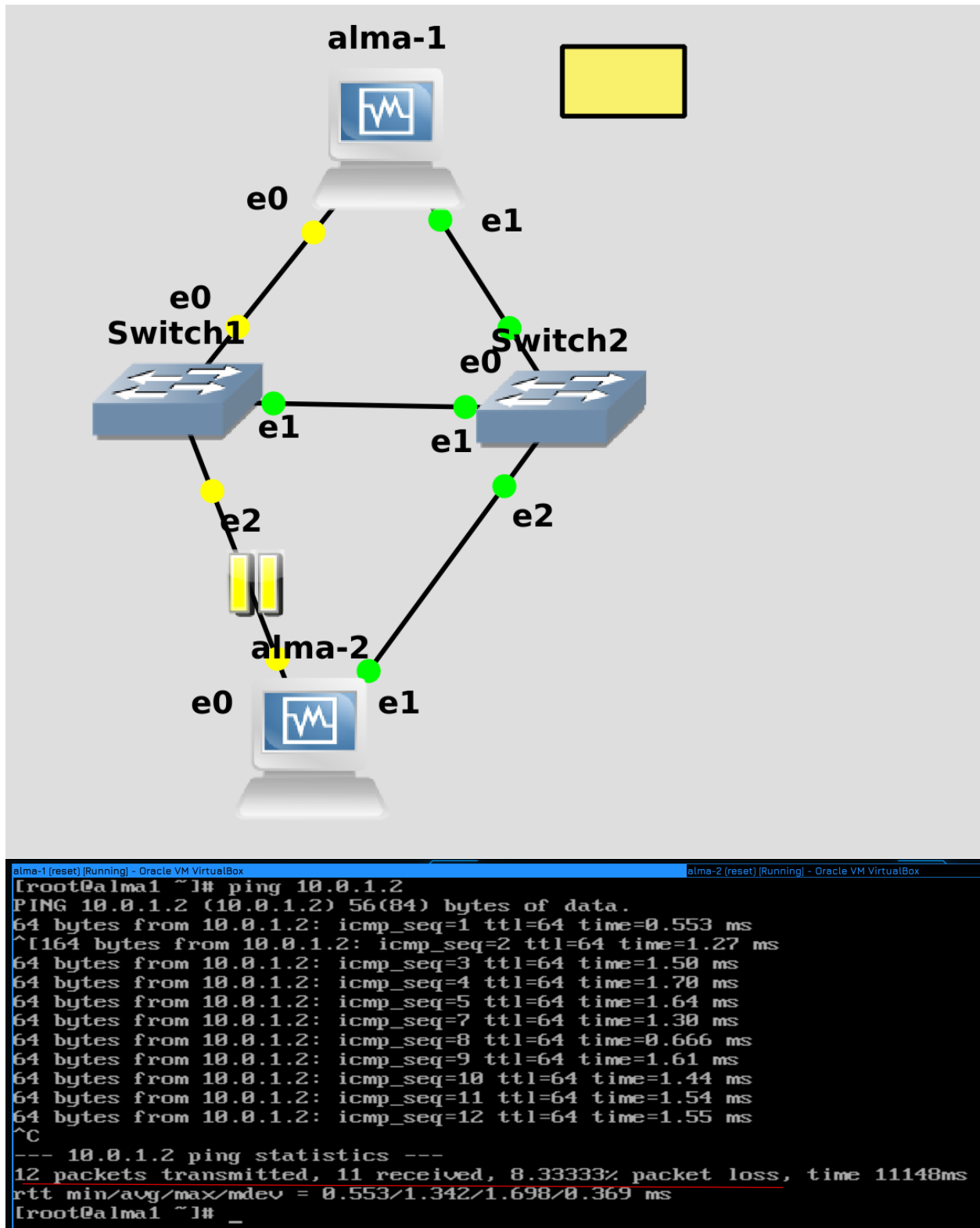


```
alma-1 [reset] [Running] - Oracle VM VirtualBox
[root@alma1 ~]# teamctl myteam state
setup:
  runner: activebackup
ports:
  enp0s3
    link watches:
      link summary: down
      instance[link_watch_0]:
        name: ethtool
        link: down
        down count: 2
  enp0s8
    link watches:
      link summary: up
      instance[link_watch_0]:
        name: ethtool
        link: up
        down count: 0
runner:
  active port: enp0s8
[root@alma1 ~]#
```

Если ещё раз глянуть вывод команды `teamctl`, то можно увидеть, что `enp0s3` теперь недоступен и вместо него активным стал `enp0s8`.



Теперь альма 1 видит только свитч2. Но на альма2 по прежнему активен интерфейс e0, поэтому он обращается только к свитчу 1. В итоге пакет из альмы 1 идёт сначала на свитч 2, потом на свитч 1 и только потом на альму 2.



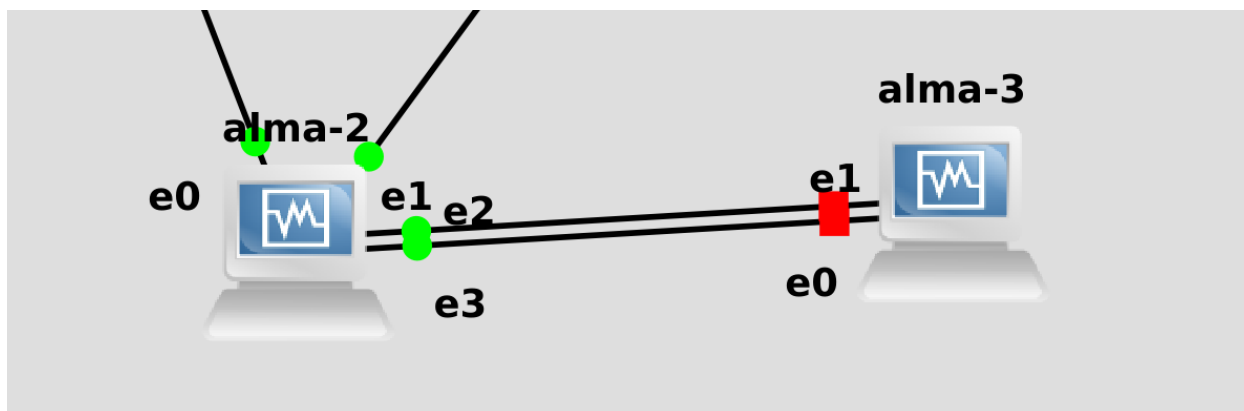
Давайте ещё раз запустим пинг, но теперь вырубим кабель, идущий от второй альмы к первому свитчу. И опять мы потеряли один пакет. Теперь для общения между хостами остался только второй свитч.

```

alma-1 (reset) [Running] - Oracle VM VirtualBox
[root@alma1 ~]# teamdctl myteam state
setup:
  runner: activebackup
ports:
  enp0s3
    link watches:
      link summary: up
      instance[link_watch_0]:
        name: ethtool
        link: up
        down count: 2
  enp0s8
    link watches:
      link summary: up
      instance[link_watch_0]:
        name: ethtool
        link: up
        down count: 0
runner:
  active port: enp0s8
[root@alma1 ~]# _

```

Ну и после восстановления обоих проводов teamdctl показывает, что оба линка доступны.



Теперь настроим LACP. Как вы, возможно, догадываетесь, отличий в настройке вообще не будет, кроме опции team.runner. Поэтому для разнообразия альму 2 подключим сразу двумя тимиинг интерфейсами - activebackup будет смотреть на свитчи, а lacp будет смотреть на третью альму.

```

[alma-1 (reset) (Running) - Oracle VM VirtualBox] [alma-2 (reset) (Running) - Oracle VM VirtualBox] [alma-3 (reset) (Running) - Oracle VM VirtualBox]
[root@alma2 ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen
    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: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel master myteam state
    fault qlen 1000
    link/ether 08:00:27:5c:fe:dc brd ff:ff:ff:ff:ff:ff
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel master myteam state
    fault qlen 1000
    link/ether 08:00:27:5c:fe:dc brd ff:ff:ff:ff:ff:ff permaddr 08:00:27:1c:ba:0d
4: enp0s9: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group de
    link/ether 08:00:27:3e:d6:35 brd ff:ff:ff:ff:ff:ff
5: enp0s10: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group d
    link/ether 08:00:27:47:d0:ad brd ff:ff:ff:ff:ff:ff
19: myteam: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group de
    link/ether 08:00:27:5c:fe:dc brd ff:ff:ff:ff:ff:ff
    inet 10.0.1.2/24 brd 10.0.1.255 scope global noprefixroute myteam
        valid_lft forever preferred_lft forever
    inet6 fe80::453a:886a:e69d:827b/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
[root@alma2 ~]#

```

После подключения на alma2 мы видим:

```
ip a
```

что интерфейсы enp0s9 и enp0s10 поднялись.

```

[alma-1 (reset) (Running) - Oracle VM VirtualBox] [alma-2 (reset) (Running) - Oracle VM VirtualBox] [alma-3 (reset) (Running) - Oracle VM VirtualBox]
[root@alma2 ~]# nmcli connection add type team con-name teams9s10 ifname myteam2 team.runner lacp ip
v4.addresses 10.0.2.2/24 ipv4.method manual
Connection 'teams9s10' (13af5dbc-0ee7-464f-9d95-a28d030d9f4b) successfully added.
[root@alma2 ~]# nmcli connection add type team-slave ifname enp0s9 master myteam2
Connection 'team-slave-enp0s9' (d53c3543-dac9-4a5e-8330-7733f8c1fd62) successfully added.
[root@alma2 ~]# nmcli connection add type team-slave ifname enp0s10 master myteam2
Connection 'team-slave-enp0s10' (6780caab-b456-4631-bd97-dc70cd958893) successfully added.
[root@alma2 ~]# nmcli connection show
NAME                                UUID                                TYPE    DEVICE
teams3s8                           9118ffea-3b26-4dd3-ab69-e4f7470e0862    team    myteam
teams9s10                           13af5dbc-0ee7-464f-9d95-a28d030d9f4b    team    myteam2
team-slave-enp0s10                  6780caab-b456-4631-bd97-dc70cd958893    ethernet enp0s10
team-slave-enp0s3                   0c27a620-b20a-483d-8f3f-6099560750ad    ethernet enp0s3
team-slave-enp0s8                   25d60ce9-fffa-4669-bb46-05500d73ba2e    ethernet enp0s8
team-slave-enp0s9                   d53c3543-dac9-4a5e-8330-7733f8c1fd62    ethernet enp0s9
[root@alma2 ~]#

```

А дальше те же команды создания тиминг интерфейса и слейвов:

```

nmcli con add type team con-name teams9s10 ifname myteam2 team.runner lacp ipv4.
→addresses 10.0.2.2/24 ipv4.method manual
nmcli con add type team-slave ifname enp0s9 master myteam2
nmcli con add type team-slave ifname enp0s10 master myteam2
nmcli con show

```

Разве что для имени интерфейса придумываем другое название - myteam2, указываем раннер lacp, ну и другой айпи адрес. И на слейвах мастером указываем myteam2. Мои названия myteam и myteam2, конечно, не лучшие. В реальной среде обычно называют team0, team1 и т.д., но для наглядности я использовал такие названия, чтобы вы отличали имя профиля в network-manager-е от имени интерфейса.

```

[alma-1 (reset) (Running)] - Oracle VM VirtualBox
[root@alma3 ~]# nmcli connection show
NAME                                UUID                                TYPE    DEVICE
enp0s3                             40eee77d-a591-4beb-bca3-91fd110170d5 ethernet --
Wired connection 1                  ea08c5ec-e5f7-3e7a-a2f3-4e4ee9c77371 ethernet --
Wired connection 2                  c00802d4-f19c-36a7-8cbf-74ee4425972e ethernet --
Wired connection 3                  a68fba29-de96-3c27-955d-b5926a25eeb0 ethernet --
[root@alma3 ~]# nmcli connection delete enp0s3 Wired\ connection\ 1 Wired\ connection\ 2 Wired\ connection\ 3
Connection 'enp0s3' (40eee77d-a591-4beb-bca3-91fd110170d5) successfully deleted.
Connection 'Wired connection 1' (ea08c5ec-e5f7-3e7a-a2f3-4e4ee9c77371) successfully deleted.
Connection 'Wired connection 2' (c00802d4-f19c-36a7-8cbf-74ee4425972e) successfully deleted.
Connection 'Wired connection 3' (a68fba29-de96-3c27-955d-b5926a25eeb0) successfully deleted.
[root@alma3 ~]# nmcli connection add type team con-name teams3s8 ifname myteam team.runner lacp ipv4.addresses 10.0.2.3/24 ipv4.method manual
Connection 'teams3s8' (e0085054-facd-4349-884f-f4e05f7760b5) successfully added.
[root@alma3 ~]# nmcli connection add type team-slave ifname enp0s3 master myteam
Connection 'team-slave-enp0s3' (c0e467e7-a60b-44c0-9eea-6f2f5025c8fa) successfully added.
[root@alma3 ~]# nmcli connection add type team-slave ifname enp0s8 master myteam
Connection 'team-slave-enp0s8' (a0c4da8f-8f8b-4959-a1e4-06ebdca22251) successfully added.
[root@alma3 ~]# nmcli connection show
NAME                                UUID                                TYPE    DEVICE
teams3s8                           e0085054-facd-4349-884f-f4e05f7760b5 team     myteam
team-slave-enp0s3                   c0e467e7-a60b-44c0-9eea-6f2f5025c8fa ethernet enp0s3
team-slave-enp0s8                   a0c4da8f-8f8b-4959-a1e4-06ebdca22251 ethernet enp0s8
[root@alma3 ~]# _

```

Ну и на alma3 всё по той же схеме - очищаем профили, создаём профиль с тиминг интерфейсом с раннером ласр и создаём слейвы:

```

nmcli con show
nmcli con del enp0s3 Wired\ connection\ 2 VLAN2 Wired\ connection\ 1 Wired\ connection\ 3
nmcli con add type team con-name teams3s8 ifname myteam2 team.runner lacp ipv4.addresses 10.0.2.3/24 ipv4.method manual
nmcli con add type team-slave ifname enp0s3 master myteam2
nmcli con add type team-slave ifname enp0s8 master myteam2
nmcli con show

```

Понимаю, в консоли со шрифтами беда и неудобно читать, но для настройки ssh нужно было бы настроить сеть и NAT, а это будущие темы, поэтому не хотелось прыгать вперёд.

```

[alma-1 (reset) (Running)] - Oracle VM VirtualBox
[alma-2 (reset) (Running)] - Oracle VM VirtualBox
[root@alma3 ~]# ping 10.0.2.2
PING 10.0.2.2 (10.0.2.2) 56(84) bytes of data.
64 bytes from 10.0.2.2: icmp_seq=1 ttl=64 time=1.93 ms
64 bytes from 10.0.2.2: icmp_seq=2 ttl=64 time=1.39 ms
^C
--- 10.0.2.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 1.386/1.658/1.930/0.272 ms
[root@alma3 ~]# _

```

После настройки сети попробуем с альмы3 пингануть второй хост:

```
ping 10.0.2.2
```

Всё пингуется - значит сеть работает.


```

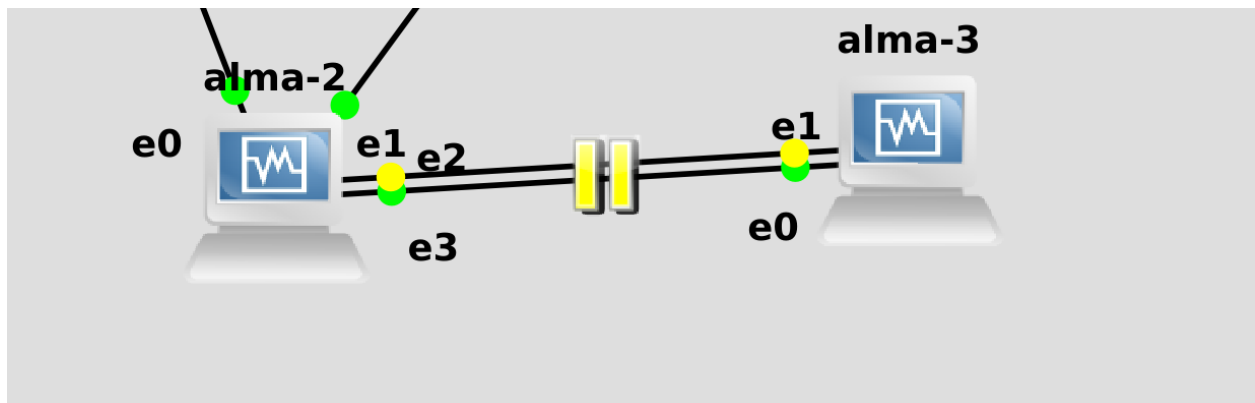
alma-1 (reset) [Running] - Oracle VM VirtualBox
alma-2 (reset) [Running] - Oracle VM VirtualBox
[root@alma2 ~]# teamdctl myteam2 state
setup:
runner: lacp
ports:
  enp0s10
    link watches:
      link summary: up
      instance[link_watch_0]:
        name: ethtool
        link: up
        down count: 1
    runner:
      aggregator ID: 4, Selected
      selected: yes
      state: current
  enp0s9
    link watches:
      link summary: up
      instance[link_watch_0]:
        name: ethtool
        link: up
        down count: 1
    runner:
      aggregator ID: 4, Selected
      selected: yes
      state: current
runner:
  active: yes
  fast rate: no
[root@alma2 ~]#

```

Давайте ещё на альме 2 посмотрим teamdctl на второй интерфейс:

```
teamdctl myteam2 state
```

Как тут видно, в качестве раннера выбран lacp и оба интерфейса в апе.



Ну и давайте тестировать. Я запускаю пинг:

```
ping 10.0.2.3
```

потом вырубаю один из линков в GNS, жду пару секунд, врубаю его, потом жду пару секунд, чтобы первый интерфейс пришёл в себя, затем выключаю второй линк, опять жду и включаю.

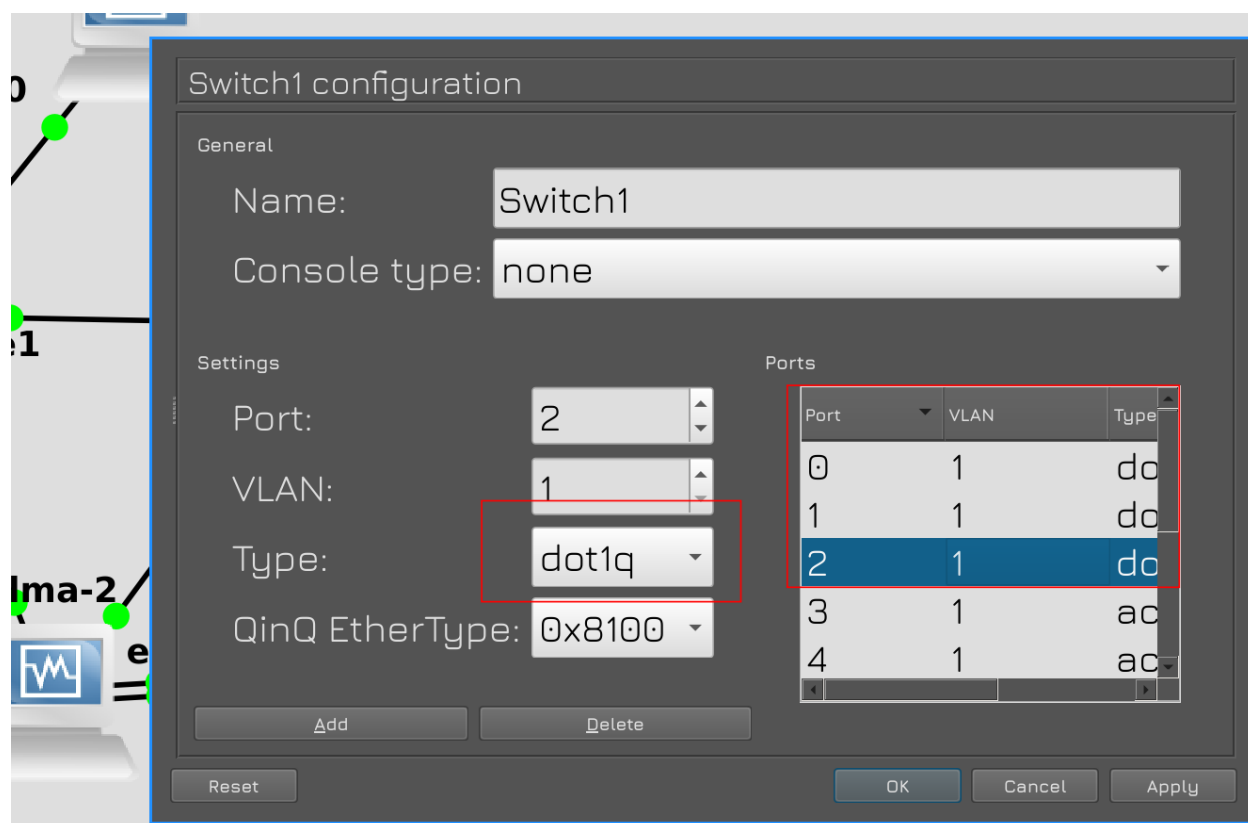
```

64 bytes from 10.0.2.3: icmp_seq=30 ttl=64 time=1.30 ms
64 bytes from 10.0.2.3: icmp_seq=31 ttl=64 time=1.11 ms
64 bytes from 10.0.2.3: icmp_seq=32 ttl=64 time=1.30 ms
64 bytes from 10.0.2.3: icmp_seq=33 ttl=64 time=1.36 ms
^C
--- 10.0.2.3 ping statistics ---
33 packets transmitted, 33 received, 0% packet loss, time 32066ms
rtt min/avg/max/mdev = 0.538/1.302/1.472/0.179 ms
[root@alma2 ~]#

```

И, как видите, по результатам пинга ни одного потерянного пакета, т.е. всё работает идеально. Поэтому LACP более предпочтительнее в рабочей среде, но, в крайнем случае, даже с тупыми свитчами можно настроить отказоустойчивость через activebackup.

Что касается одновременной настройки никтиминга и вланов, то сначала следует создать тиминг интерфейс, а уже потом использовать его в качестве интерфейса для вланов.



Для примера создадим второй влан и пустим его между альма 1 и альма 2. Сначала надо на обоих свитчах на портах 0, 1 и 2 выставить тип dot1q. Эти свитчи не умеют пропускать только два влана, поэтому они пускают либо 1, как стоит по дефолту, либо все. Ну вот мы сейчас все и разрешаем. При этом первый влан остаётся нетегированным.

```

[root@alma1 ~]# nmcli connection add type vln con-name vln2 dev myteam id 2 ipv4.addresses 10.0.3.1/24 ipv4.method manual
Connection 'vln2' (5e4a5fa5-f1dd-4a6b-8355-bf5e57fcb709) successfully added.
[root@alma1 ~]# nmcli connection show
NAME                UUID                                  TYPE      DEVICE
teams3s8            12e0feb1-24bc-4ca2-81be-5ec4ec8c36f2 team       myteam
vln2                5e4a5fa5-f1dd-4a6b-8355-bf5e57fcb709 vln        myteam.2
team-slave-enp0s3    0c3b4fe4-60ea-4e8c-9004-426084dd4477 ethernet  enp0s3
team-slave-enp0s8    1659f0c9-3d67-4f06-a0a0-e5c21fe69ffe ethernet  enp0s8
[root@alma1 ~]#

```

```
[root@alma2 ~]# nmcli connection add type vlan con-name vlan2 dev myteam id 2 ip4.addresses 10.0.3.2/24 ip4.method manual
Connection 'vlan2' (fe24f654-7e60-4cc1-9fd2-f0c72b68b3d5) successfully added.
[root@alma2 ~]# nmcli connection show
```

NAME	UUID	TYPE	DEVICE
teams3s8	9118ffea-3b26-4dd3-ab69-e4f7470e0862	team	myteam
teams9s10	13af5dbdc-0ee7-464f-9d95-a28d030d9f4b	team	myteam
vlan2	fe24f654-7e60-4cc1-9fd2-f0c72b68b3d5	vlan	myteam.2
team-slave-enp0s10	6780caab-b456-4631-bd97-dc70cd950893	ethernet	enp0s10
team-slave-enp0s3	0c27a620-b20a-483d-8f3f-6099560750ad	ethernet	enp0s3
team-slave-enp0s8	25d60ce9-fffa-4669-bb46-05508d73ba2e	ethernet	enp0s8
team-slave-enp0s9	d53c3543-dac9-4a5e-8330-7733f8c1fd62	ethernet	enp0s9

```
[root@alma2 ~]#
```

Дальше на альме 1 и 2 создаём через nmcli профили с типом vlan:

```
nmcli con add type vlan con-name vlan2 dev myteam id 2 ipv4.addresses 10.0.3.1/24 ipv4.  
↪method manual
```

```
nmcli con add type vlan con-name vlan2 dev myteam id 2 ipv4.addresses 10.0.3.2/24 ipv4.  
↪method manual
```

Обратите внимание, что при создании я указываю интерфейс, поверх которого будет влан - `dev myteam`, и `id` этого влана - `id 2`. А имя профиля и адреса - всё по стандарту.

Ну и посмотрим список полученных интерфейсов:

```
nmcli con show
```

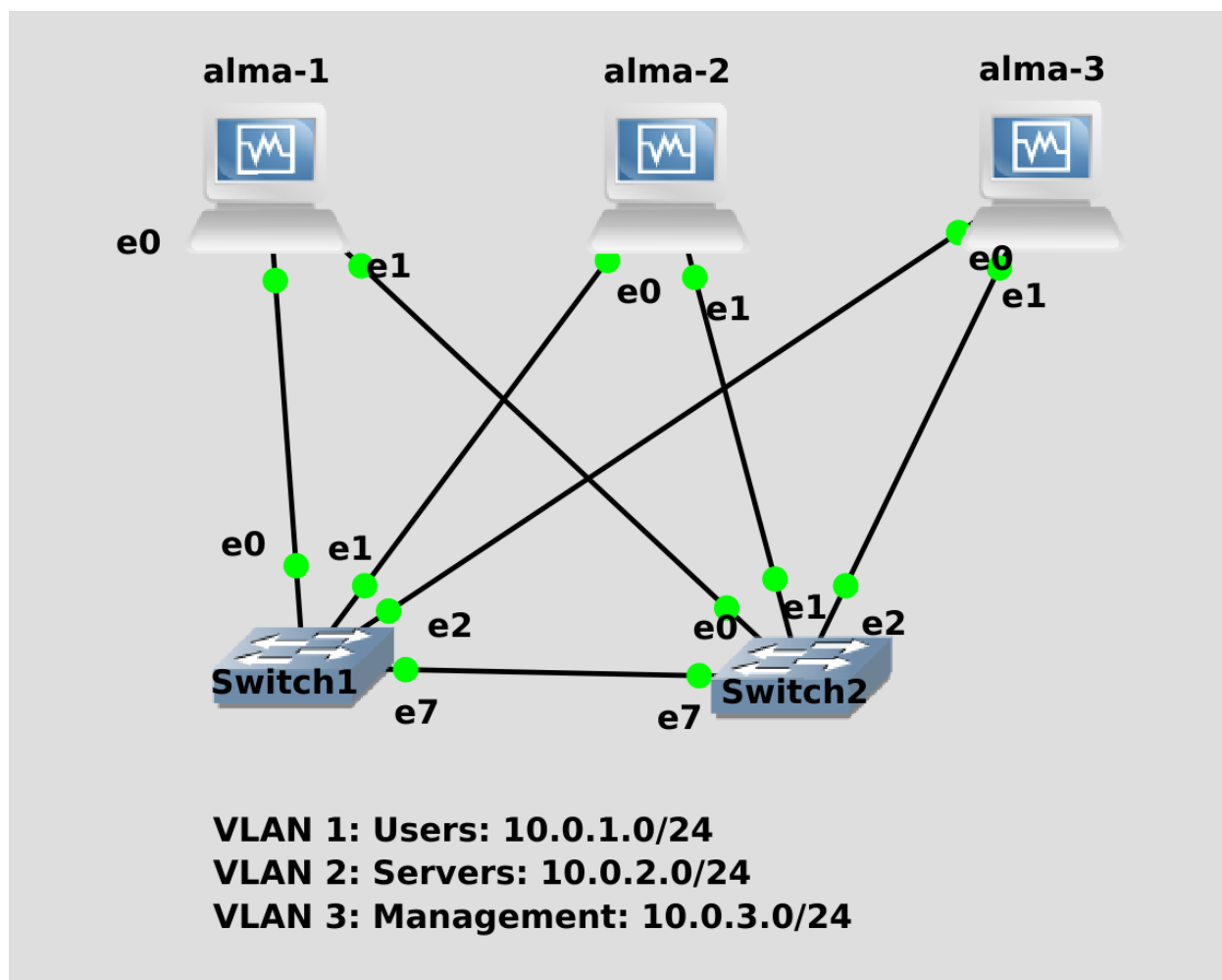
Как видите, профили создались.

```

[alma-1 (reset) [Running] - Oracle VM VirtualBox] [alma-2 (reset) [Running] - Oracle VM VirtualBox]
[root@alma1 ~]# ping 10.0.3.2
PING 10.0.3.2 (10.0.3.2) 56(84) bytes of data.
64 bytes from 10.0.3.2: icmp_seq=1 ttl=64 time=0.802 ms
64 bytes from 10.0.3.2: icmp_seq=2 ttl=64 time=1.61 ms
^C
--- 10.0.3.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1053ms
rtt min/avg/max/mdev = 0.802/1.208/1.614/0.406 ms
[root@alma1 ~]# _

```

Ну и пинги тоже работают, т.е. всё как мы и хотели.

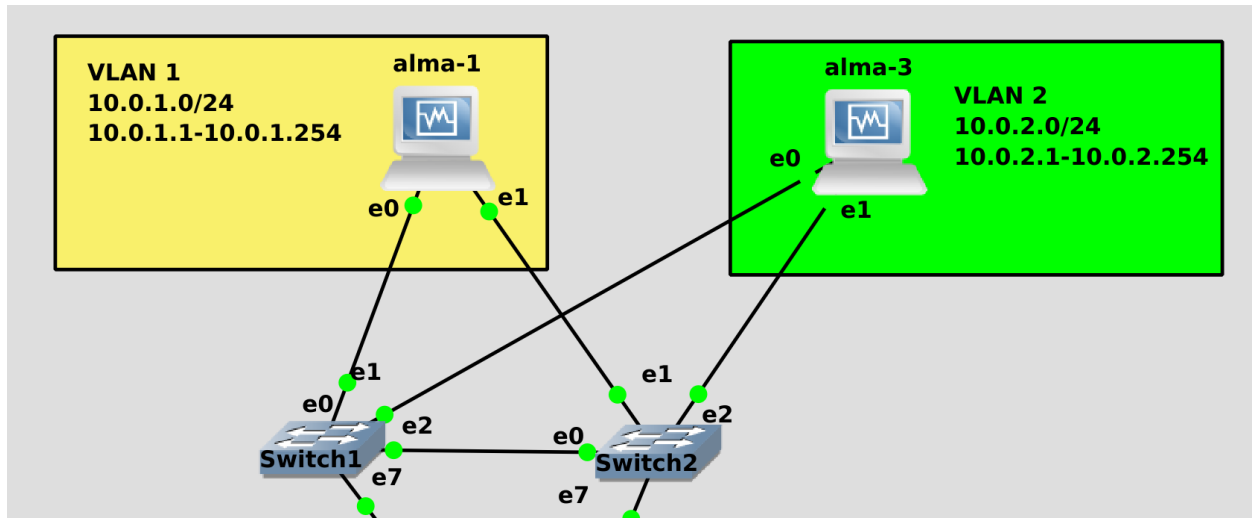


Суммируя всё что мы сегодня сказали, вам нужно будет создать показанную схему. Т.е. подключите 3 хоста к двум свитчам используя ник тими́нг в режиме activebackup и настройте на этих интерфейсах 3 влана с тремя сетями. Ну и конечно же убедитесь, что всё работает - что все друг друга пингуют по всем трём сетям, потом поотключайте провода и проверьте на отказоустойчивость.

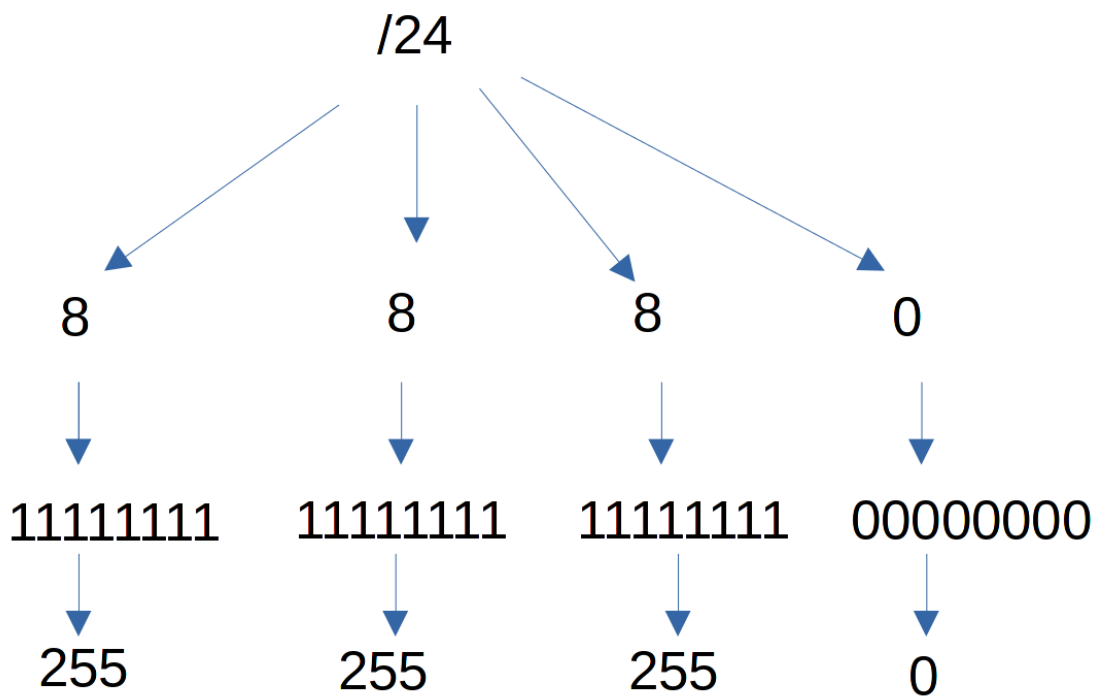
Подводя итоги, сегодня мы с вами разобрали, что такое никтими́нг, научились его настраивать, разобрали разницу между activebackup и lаср, и даже научились пускать вланы поверх агрегированного интерфейса. И пусть вас не путают команды, они хоть и длинные, но довольно простые.

2.5 05. Деление на подсети

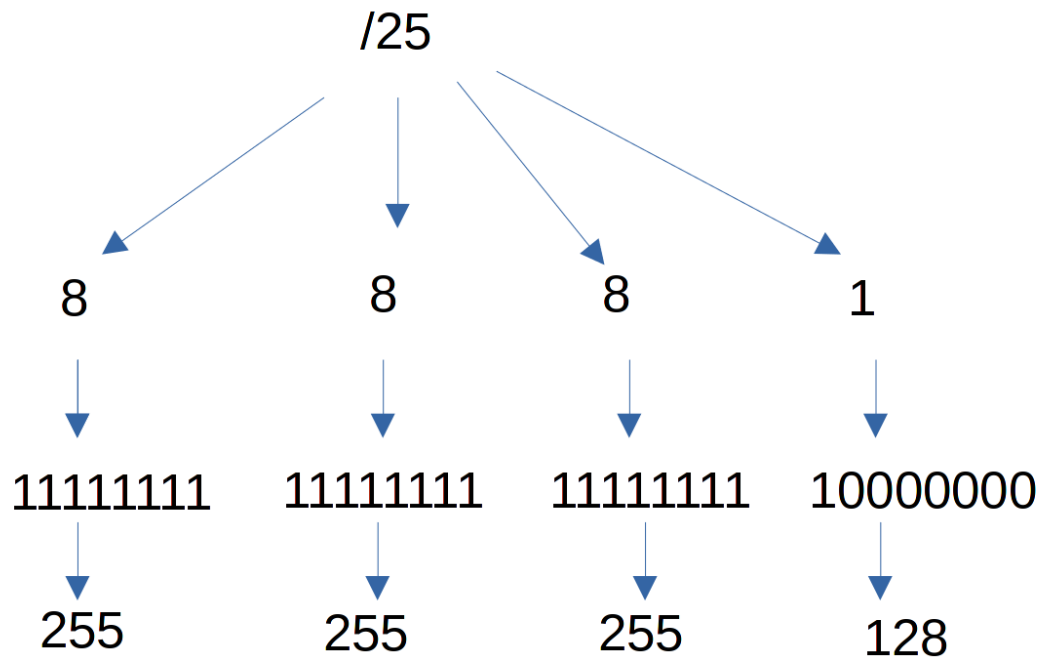
2.5.1 05. Деление на подсети



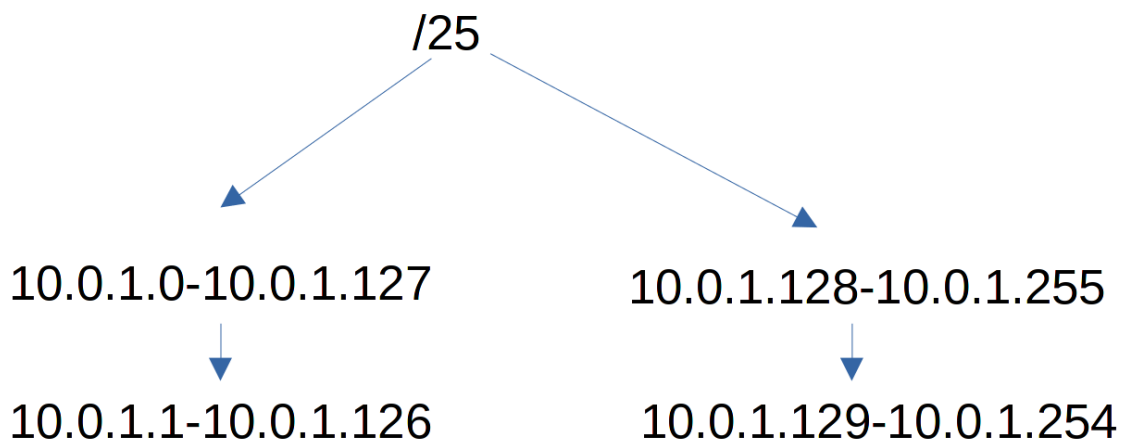
Представим, что у нас в сети есть 2 VLAN-а - VLAN 1 для пользователей и VLAN 2 для серверов. Устройства в одном влане могут видеть друг друга напрямую, используя лишь свитчи для соединения. Можно даже без свитчей, просто соединив два компьютера патч-кордом. Но мало видеть друг друга - нужно, чтобы и адреса были в одной подсети - скажем, 10.0.1.0/24. У пользователей IP адреса находятся в подсети 10.0.1.0 и могут варьироваться от 10.0.1.1 до 10.0.1.254. У серверов всё тоже самое, но только с приставкой 10.0.2. Считайте, что VLAN-ы - это физические границы, а подсети - логические. Но чем вообще обусловлены логические границы, почему именно от 10.0.1.1 до 10.0.1.254?



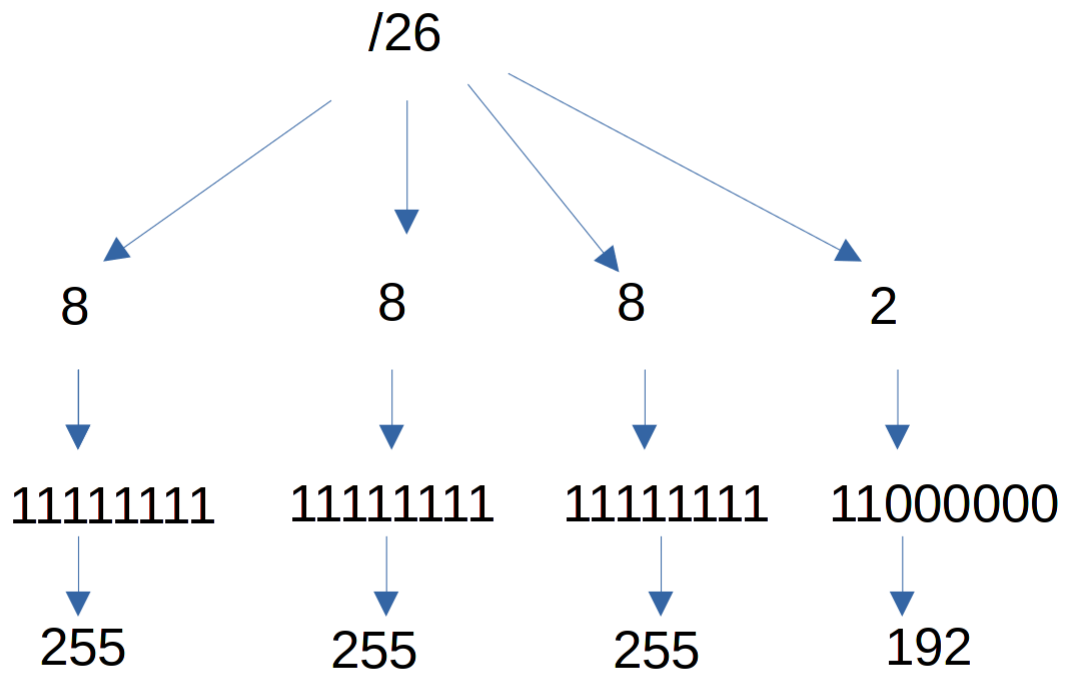
Всё дело в маске подсети, в том самом /24, который мы указываем после адреса. Я не буду сильно вдаваться в подробности, но вкратце: в ipv4 всё сводится к 4 байтам, для IP адреса есть 4 байта и для маски есть 4 байта. Каждый байт состоит из 8 бит в которых можно уместить 256 значений - от 0 до 255. Скажем, в маске /24 - который по другому будет 255.255.255.0 - полностью забиты первые 3 октета, т.е. первые 3 байта, но последний байт полностью свободен. А значит он может принять 256 значений. Это и говорит нам, какие адреса могут быть использованы в этой подсети. Правда адрес 0 выделяется как адрес сети, а адрес 255 - как бродкаст адрес. Скажем, если послать какой-то пакет на адрес 10.0.1.255 - то этот пакет придёт всем хостам, у которых адрес в подсети 10.0.1.0/24.



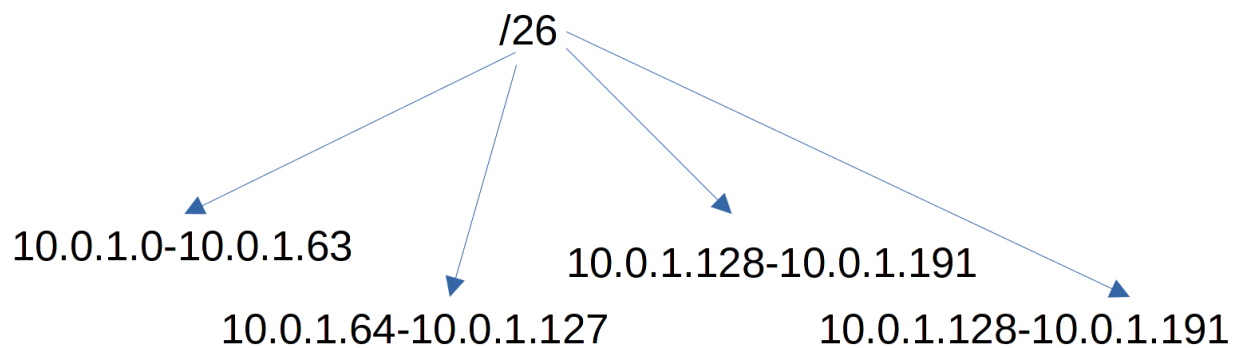
Для примера возьмём другую маску - /25. В таком случае в последнем октете забивается первый бит, поэтому количество вариаций сокращается вдвое - остаётся только 128 адресов. В итоге адреса будут от 0 до 127, при этом 0 останется адресом сети, а broadcast адресом станет 127. Устройствам же можно будет назначить адреса от 1 до 126.



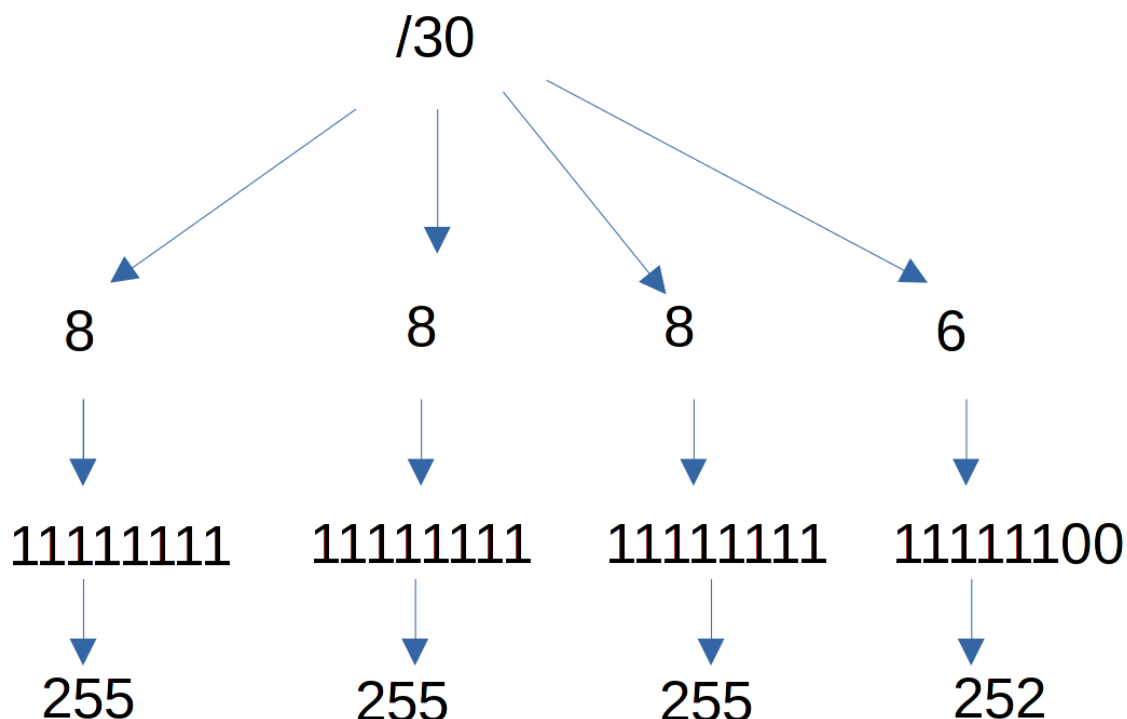
При этом, из-за того, что мы поделили сеть пополам, появляется вторая сеть - она начинается от 10.0.1.128 до 10.0.1.255. Ну и IP адреса будут варьироваться от 129 до 254.



По той же логике /26 - теперь у нас свободными останутся 64 адреса.



И таким образом появятся 4 сети по 64 адреса, ну и естественно не забывайте про адрес сети и бродкаст, из-за чего -2 адреса.



Если вы арендуете у провайдера один публичный адрес, очень часто провайдер выделяет вам вместе с адресом маску **/30** - в такую сетку входит 4 адреса. Отнимаем -2 - получается два адреса. Один адрес будет адресом вашего роутера, а второй - адресом роутера провайдера. Просто провайдер не хочет сажать всех клиентов в одну сеть, так как это не очень безопасно. Но при этом не может ради вас одного выделить целых 254 публичных адреса. Поэтому и нужна такая сегментация - когда адресов мало, но вам нужно как-то делить их.

Если у вас в компании не десятки тысяч устройств, то зачастую можно не заморачиваться с масками - адресов для локальных сетей огромное количество, поэтому на каждую сеть можно вешать маску **/24**. Хотя иногда приходится делить подсеть, т.е. делать из одной большой подсети несколько небольших.

State of IT Management Survey Report 2020-21 - Access Report →

Site24x7 Products ▾ Plans and Pricing Features Resources ▾ Free Tools Support Free Uptime Monitoring

Subnet Calculator for IPV4

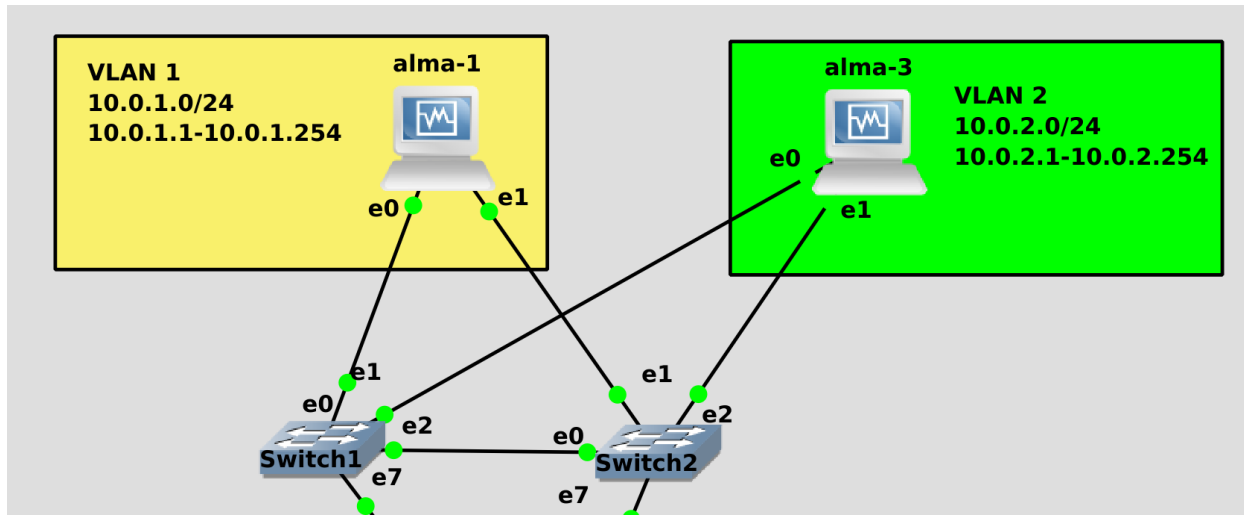
The IP Subnet Calculator performs subnet calculations for the given network address block, subnet mask, maximum required hosts per subnet and determines the resulting broadcast address, subnet, Cisco wildcard mask and host range.

Network Address Block	10.0.1.54/22	Host Address Range	10.0.0.1 - 10.0.3.254
Subnet Mask	255.255.252.0/22	Broadcast Address	10.0.3.255
No. of Hosts/Subnet	1024	Wildcard Mask	0.0.3.255
Number of Subnets	1	CIDR Notation	10.0.0.0/22

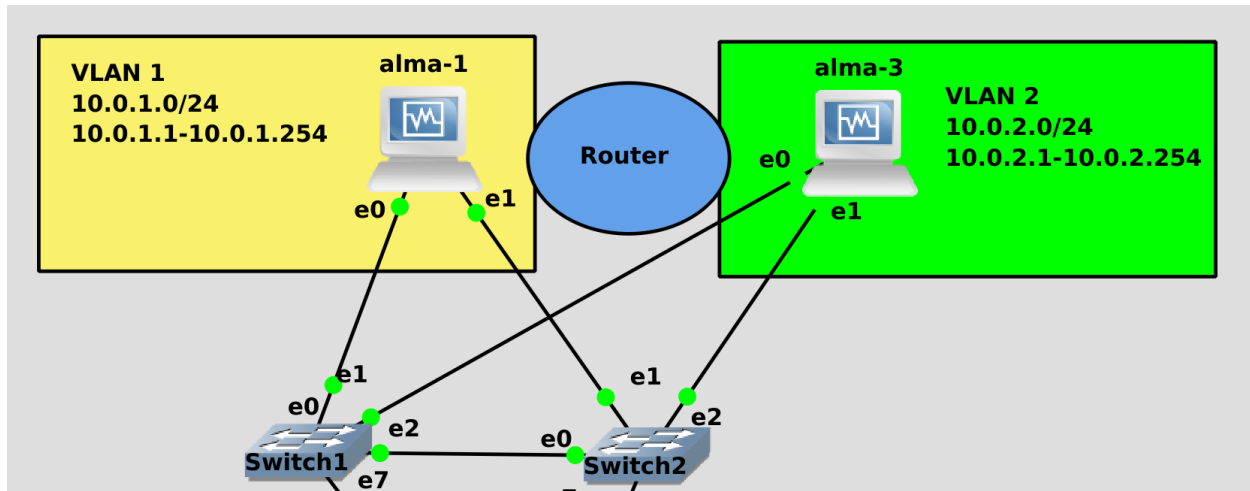
Subnet Details

Subnet ID	Subnet Address	Host Address Range	Broadcast Address
1	10.0.0.0	10.0.0.1 - 10.0.3.254	10.0.3.255

Но обычно подсчёты подсетей, масок и адресов это задача сетевого администратора, однако системный администратор должен иметь об этом представление. Чтобы хотя бы понимать, что выставляя такую-то маску, такие-то хосты будут с ним в одной подсети, а такие-то в другой. В целом неплохо бы уметь это делать просто увидев маску, но это приходит с опытом. В любом случае, в интернете есть огромное количество калькуляторов подсетей - просто погуглите subnet calculator. Они позволяют вам понять, какую маску нужно выставлять, ну или какие хосты входят в вашу подсеть.



А видя кто входит в вашу подсеть, вы понимаете, кто в неё не входит. И хотя мы отделяем, изолируем подсети - мы всё же хотим, чтобы хосты могли общаться. Отделяя пользователей от серверов, нам всё равно нужно, чтобы пользователи могли обращаться к серверам.



И для этого нам нужно устройство, которое будет одновременно и в первой сети, и во второй. Если каждая подсеть это комната, то нужна дверь, которая есть одновременно в обеих комнатах. И этой дверью выступает маршрутизатор, ну или по другому его называют роутер. Основная задача роутера - перенаправлять пакеты из одной подсети в другую. Т.е. роутер обитает на 3 уровне модели OSI - сетевом.

Скажем, alma1 хочет обратиться к alma3. Для этого alma1 должна понимать, что путь к alma3 ведёт через router. А роутер при этом должен иметь адрес как в сети 10.0.1.0, так и в сети 10.0.2.0. Т.е. свитчи соединяют устройства в рамках одной подсети, а роутеры соединяют разные подсети, формируя целую сеть. А соединяя одни роутеры с другими мы связываем сети и в итоге это всё формирует целый интернет.



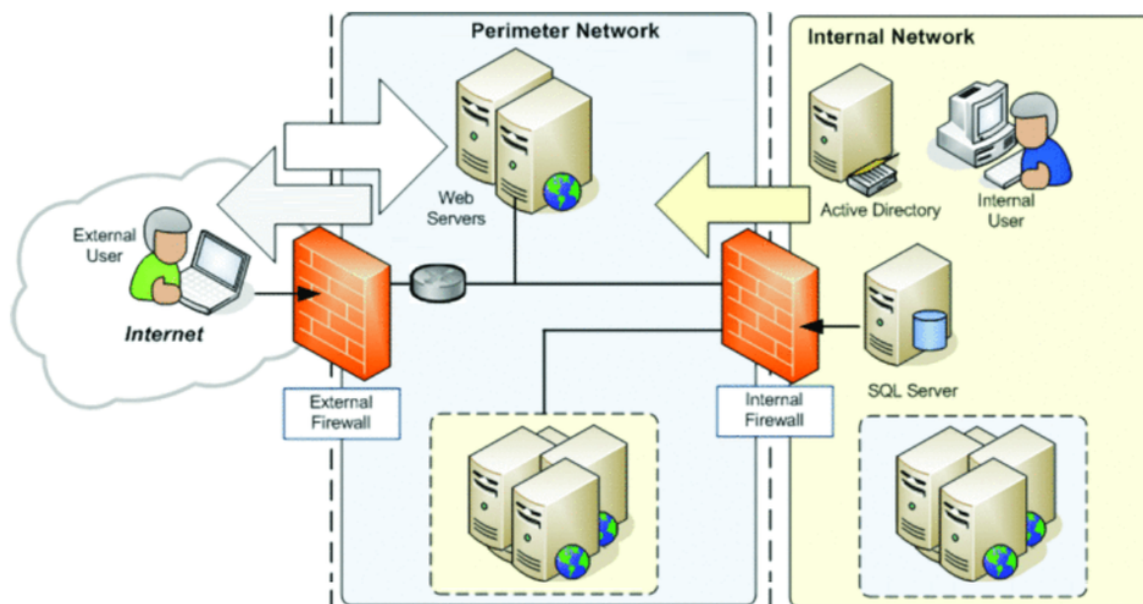
С домашними роутерами вы все знакомы - небольшая коробочка, в которой как минимум 2 порта - один идёт к провайдеру, а второй к вашей локальной сети.



Естественно, в средних и крупных компаниях много трафика и такой коробочки недостаточно, поэтому используются роутеры посерьёзнее, которые могут передавать десятки и сотни гигабит в секунду, при этом их можно монтировать в рэки.

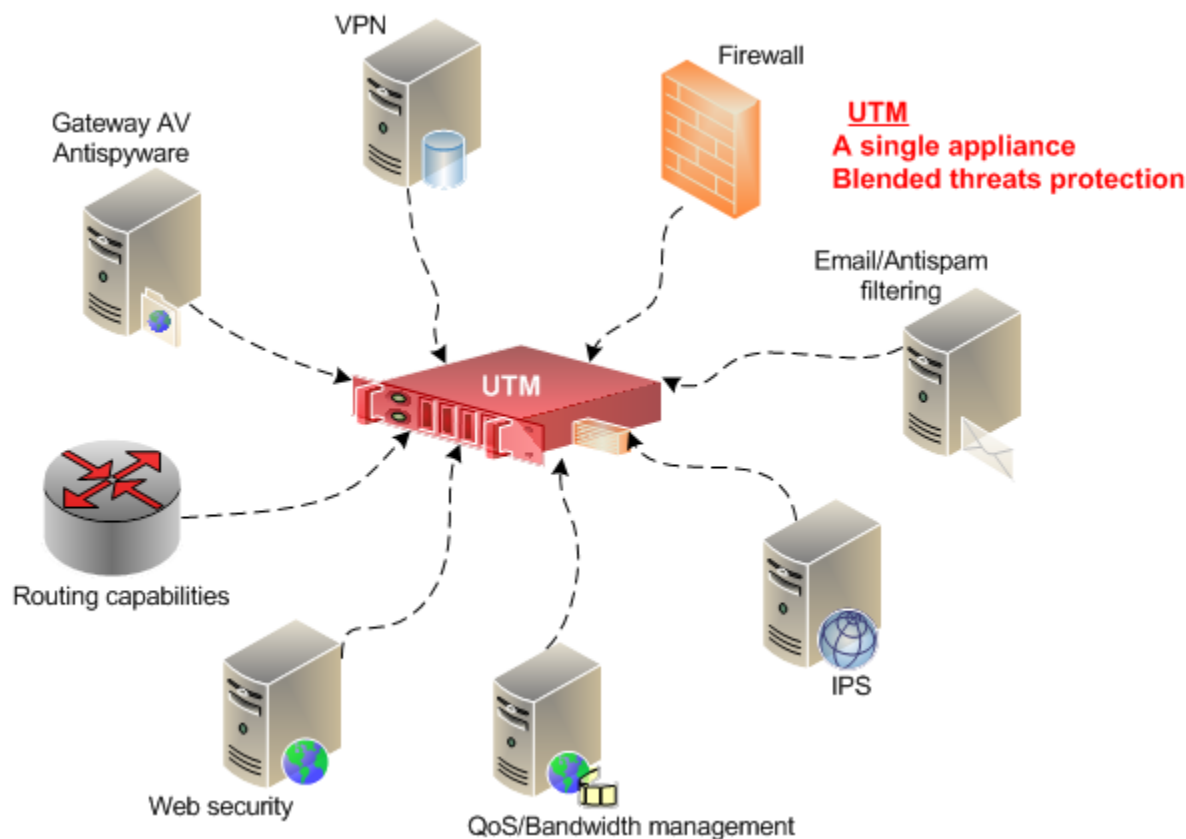
Возможно, вы задались вопросом - а зачем нам нужно отделять сети, если в итоге мы их соединяем через роутеры? В чём смысл?

Если бы роутер был только роутером, то смысла действительно было бы мало. Не то чтобы совсем не было, как минимум через роутер не ходит L2 broadcast трафик, и поэтому он не ходил бы от миллиардов устройств и не перегружал бы все свитчи в мире. Но я не об этом. В современном мире смысла от чистого роутера нет - недостаточно просто пропускать трафик, надо его ещё проверять. Да, мы хотим, чтобы юзеры могли обращаться к серверам, но не ко всем и не по любому протоколу. Не каждому юзеру нужен доступ на сервер по ssh, обычно хватает доступа по HTTPS. Ну и мало кому из юзеров нужно на напрямую на сервер с базой данных.

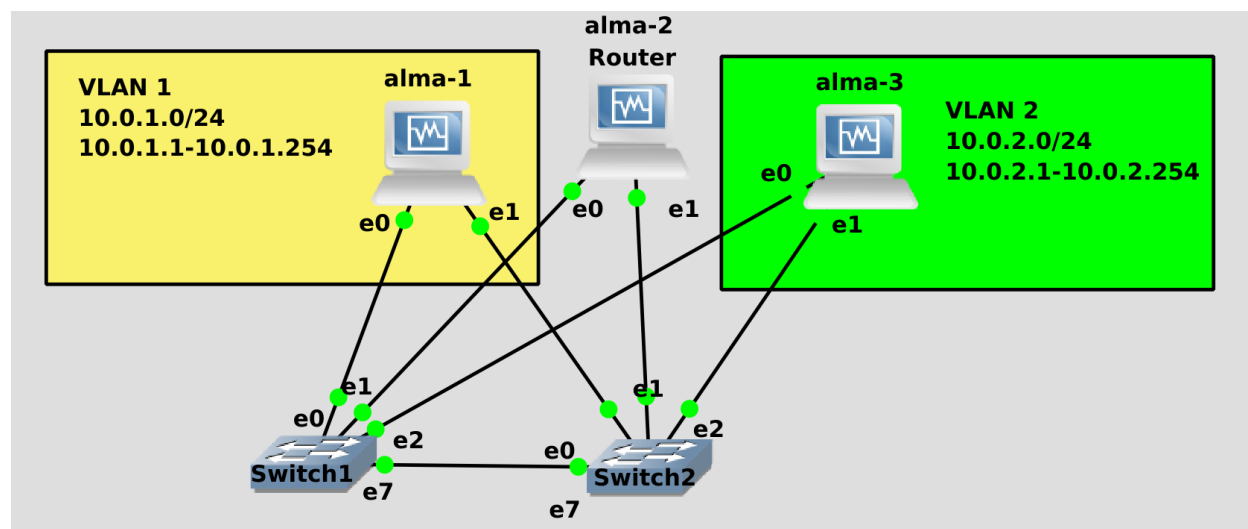


Поэтому на роутерах почти всегда есть сетевой firewall. Он не просто пропускает весь трафик, но и проверяет - а можно ли компьютеру 1 доступ к серверу 1? Если нет - то этот пакет блокируется. Но недостаточно просто проверить доступ от одного IP к другому - может мы хотим разрешить доступ по 80, но заблокировать по 22 порту? Значит из-за файрвола роутер должен выйти за рамки l3 - и, как минимум, мыслить на 4 уровне - транспортном, который отвечает за порты.

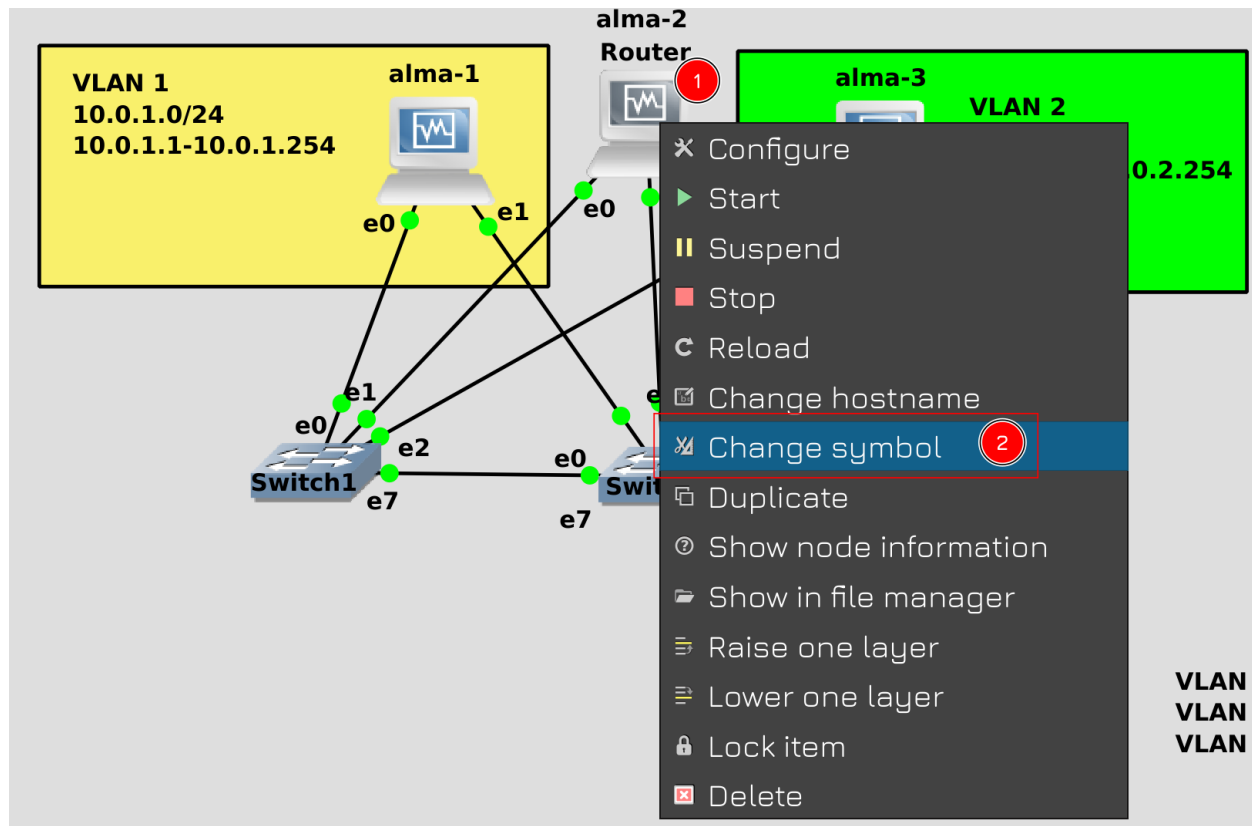
Но для современных роутеров даже этого недостаточно. Да, мы хотим, чтобы пользователь мог заходить на сервер по 80 порту, но мало ли что там пользователь делает? А вдруг он пытается использовать какую-то уязвимость вебсервера? Или, скажем, мы разрешаем нашему пользователю доступ в интернет, но не хотим, чтобы он заходил на какие-то опасные сайты? Просто дав ему доступ по 80 и 443 порту мы никак не проверим, на какой именно сайт он заходит.



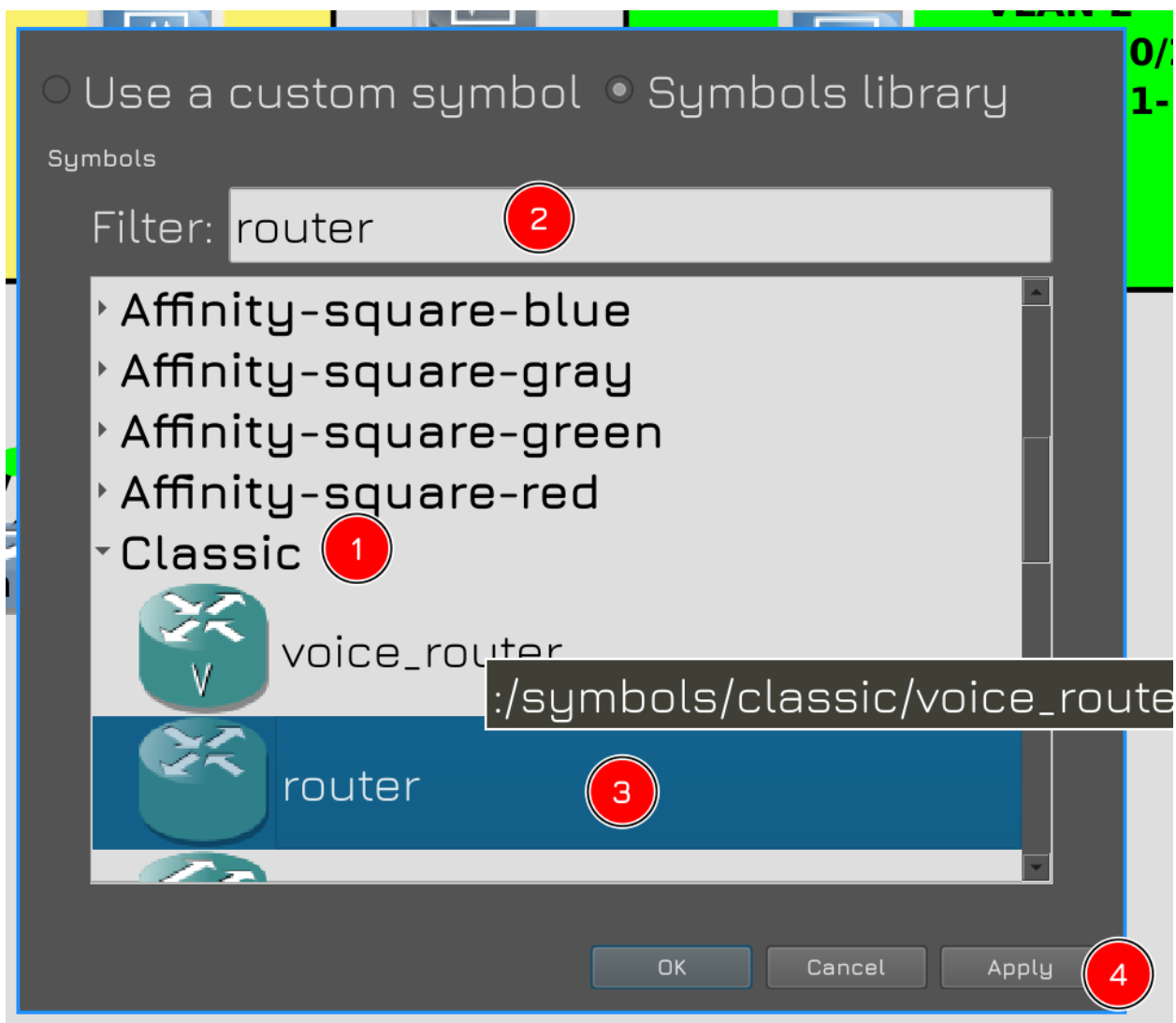
Поэтому простые сетевые фаерволы также давно утратили смысл, вместо них используются так называемые UTM - это одновременно и роутер, и сетевой фаервол, и VPN сервер, и антивирус, и у него есть куча других примочек касательно безопасности. Это устройство уже действует на всех уровнях сети для комплексной защиты инфраструктуры. Естественно, такие решения стоят гораздо дороже обычных роутеров. Но если уж мы взялись связывать то, что изолировали - то и эта связь должна быть безопасной. Однако какие бы многофункциональные не были современные решения, основы маршрутизации остались те же, всё остальное это примочки.



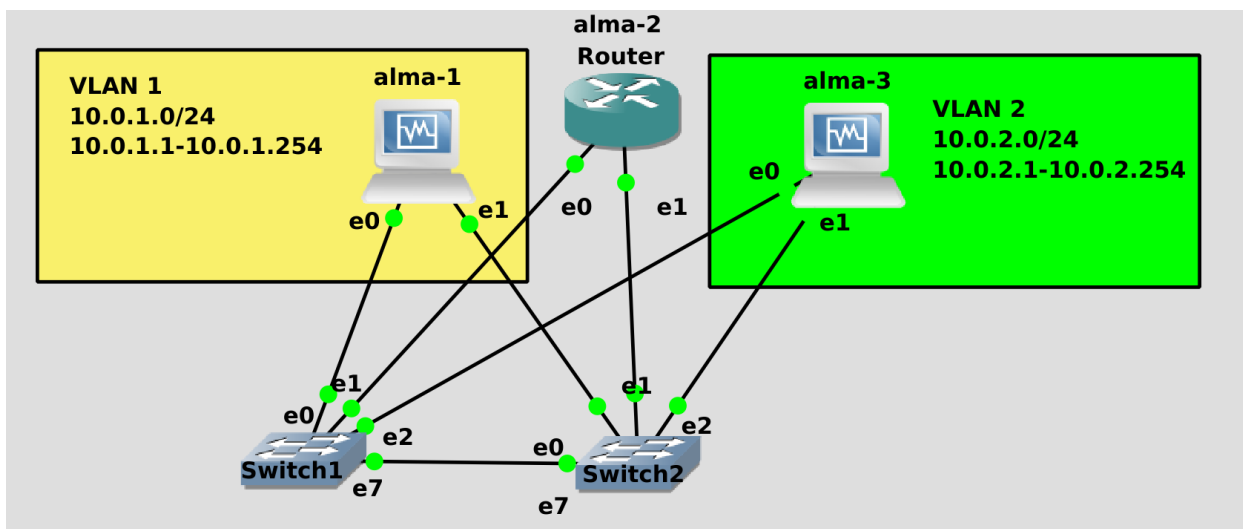
Поэтому для начала надо разобраться с роутингом. Не мало современных роутеров основаны на Linux, ну и мы сделаем так же. Пусть alma2 будет нашим роутером.



При этом для аутентичности поменяем его иконку на символ роутера. Для этого правой кнопкой мыши нажмем на **alma2** и выберем «Change symbol».



Раскройте Classic и в поле Filter напишите router. Выберите указанный символ и нажмите Apply.



Теперь у alma2 символ роутера. Осталось его таковым сделать.

```

[alma-1 (reset) (Running)] - Oracle VM VirtualBox
[alma-2 (reset) (Running)] - Oracle VM VirtualBox
[alma-3 (reset) (Running)] - Oracle VM VirtualBox

[root@alma1 ~]# nmcli connection show
NAME                UUID                                  TYPE      DEVICE
teams3s8            614944fe-4f0e-4dbb-ba7b-a621c1cad1ff team       myteam
vlan2               016a5454-e619-42d9-9b49-d89078a73e8c vlan       myteam.2
vlan3               febe863d-a4de-465d-a583-938db2322840 vlan       myteam.3
team-slave-enp0s3    9b622613-1ad4-4c67-a8d9-7cb6bfa5797e ethernet  enp0s3
team-slave-enp0s8    f02a197e-a321-4110-b0b7-c807ffa02853 ethernet  enp0s8

[root@alma1 ~]# nmcli connection delete vlan2
Connection 'vlan2' (016a5454-e619-42d9-9b49-d89078a73e8c) successfully deleted.

[root@alma1 ~]# nmcli connection show
NAME                UUID                                  TYPE      DEVICE
teams3s8            614944fe-4f0e-4dbb-ba7b-a621c1cad1ff team       myteam
vlan3               febe863d-a4de-465d-a583-938db2322840 vlan       myteam.3
team-slave-enp0s3    9b622613-1ad4-4c67-a8d9-7cb6bfa5797e ethernet  enp0s3
team-slave-enp0s8    f02a197e-a321-4110-b0b7-c807ffa02853 ethernet  enp0s8

[root@alma1 ~]#

```

Но для начала надо подготовить сеть на хостах. В прошлый раз на всех 3 хостах мы добавили все 3 влана, поэтому хосты будут видеть друг друга напрямую:

```
nmcli con sh
```

Нам же надо, чтобы хост 1 был только в первом влане - для этого удаляем влан 2 с первого хоста:

```
nmcli con del vlan2
nmcli con sh
```

```

[alma-1 (reset) (Running)] - Oracle VM VirtualBox
[alma-2 (reset) (Running)] - Oracle VM VirtualBox
[alma-3 (reset) (Running)] - Oracle VM VirtualBox

[root@alma3 ~]# nmcli connection sh
NAME                UUID                                  TYPE      DEVICE
vlan2               2412908c-dd35-46b8-a012-6f36e68b44c6 vlan       myteam.2
vlan3               5a36fb8b-99f6-4a6b-b475-17f8b8fa969f vlan       myteam.3
teams3s8            f011cd4-7336-4df1-ab98-c903537ad3c2 team       myteam
team-slave-enp0s3    8b99d804-8fa6-44d5-ba3a-4e4c66025815 ethernet  enp0s3
team-slave-enp0s8    f1adf27e-c6af-40bf-abec-9e7b00af5f26 ethernet  enp0s8

[root@alma3 ~]# nmcli con mod teams3s8 ipv4.method disabled -ipv4.addresses 10.0.1.3/24 ipv6.method disabled
[root@alma3 ~]# nmcli connection up teams3s8
Connection successfully activated (master waiting for slaves) (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/84)
[root@alma3 ~]# ip a show myteam
48: myteam: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 08:00:27:30:46:4d brd ff:ff:ff:ff:ff:ff
[root@alma3 ~]#

```

На alma3 не всё так просто:

```
nmcli con sh
```

Мы не можем просто удалить тиминг интерфейс, от этого второй влан перестанет работать. Поэтому надо удалить адрес на тиминг интерфейсе:

```
nmcli con mod teams3s8 ipv4.method disabled -ipv4.address 10.0.1.3/24 ipv6.method disabled
```

При этом отключив получение IP адреса и на IPv4, и на IPv6, иначе NetworkManager будет перегружать интерфейс. После чего нужно применить настройки на интерфейсе и убедиться, что всё сработало:

```
nmcli con up teams3s8
ip a show myteam
```

Да, теперь alma1 не сможет обратиться к alma3 напрямую, так как на alma1 нет сети 10.0.2.0, а alma3 тоже не сможет ответить напрямую, потому что у неё нет сети 10.0.1.0. Им придётся общаться через роутер.

```

alma-1 (reset) [Running] - Oracle VM VirtualBox
alma-2 (reset) [Running] - Oracle VM VirtualBox
[root@alma2 ~]# ping 10.0.1.1
PING 10.0.1.1 (10.0.1.1) 56(84) bytes of data.
64 bytes from 10.0.1.1: icmp_seq=1 ttl=64 time=3.05 ms
^C
--- 10.0.1.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 3.051/3.051/3.051/0.000 ms
[root@alma2 ~]# ping 10.0.2.3
PING 10.0.2.3 (10.0.2.3) 56(84) bytes of data.
64 bytes from 10.0.2.3: icmp_seq=1 ttl=64 time=2.79 ms
64 bytes from 10.0.2.3: icmp_seq=2 ttl=64 time=1.44 ms
^C
--- 10.0.2.3 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 1.441/2.115/2.790/0.676 ms
[root@alma2 ~]# _

```

Для начала убедимся, что alma2 видит первый хост по сети 1.0:

```
ping 10.0.1.1
```

и третий по сети 2.0:

```
ping 10.0.2.3
```

Всё пингуется, значит сеть готова.

```

alma-1 (reset) [Running] - Oracle VM VirtualBox
alma-2 (reset) [Running] - Oracle VM VirtualBox
[root@alma2 ~]# echo "net.ipv4.ip_forward=1" >> /etc/sysctl.conf
[root@alma2 ~]# sysctl -p
net.ipv4.ip_forward = 1
[root@alma2 ~]# sysctl net.ipv4.ip_forward
net.ipv4.ip_forward = 1
[root@alma2 ~]#

```

Дальше мы должны на уровне ядра включить маршрутизацию. Это можно сделать временно, но давайте лучше сразу сохраним это в настройках. Для этого надо в файле `/etc/sysctl.conf` добавить строчку `net.ipv4.ip_forward=1`:

```
echo "net.ipv4.ip_forward=1" >> /etc/sysctl.conf
```

После чего для применения настроек используем `sysctl` с опцией `-p`:

```
sysctl -p
```

Если всё нормально - вы увидите параметр с новым значением. А чтобы в любой момент проверить, включена ли эта опция, можно просто указать её с `sysctl`:

```
sysctl net.ipv4.ip_forward
```

И теперь alma2 будет перенаправлять пакеты с одного интерфейса на другой.

```

alma-1 (reset) [Running] - Oracle VM VirtualBox
alma-2 (reset) [Running] - Oracle VM VirtualBox
alma-3 (reset) [Running] - Oracle VM VirtualBox
[root@alma1 ~]# ping 10.0.2.3
connect: Network is unreachable
[root@alma1 ~]# ip ro sh
10.0.1.0/24 dev myteam proto kernel scope link src 10.0.1.1 metric 350
10.0.3.0/24 dev myteam.3 proto kernel scope link src 10.0.3.1 metric 400
[root@alma1 ~]#

```

Но этого недостаточно. Если мы попытаемся с alma1 пингануть 10.0.2.3:

```
ping 10.0.2.3
```

мы увидим ошибку - network is unreachable. Просто alma1 вообще без понятия, как достучаться до этого адреса. Чтобы это понять, посмотрим таблицу маршрутизации:

```
ip ro sh
```

Как видите, alma1 знает только о сетях 10.0.1.0 и 10.0.3.0.

```

[alma-1 (reset) [Running] - Oracle VM VirtualBox]
[root@alma1 ~]# ip route add 10.0.2.3 via 10.0.1.2
[root@alma1 ~]# ip ro sh
10.0.1.0/24 dev myteam proto kernel scope link src 10.0.1.1 metric 350
10.0.2.3 via 10.0.1.2 dev myteam
10.0.3.0/24 dev myteam.3 proto kernel scope link src 10.0.3.1 metric 400
[root@alma1 ~]# ping 10.0.2.3
PING 10.0.2.3 (10.0.2.3) 56(84) bytes of data.
^C
--- 10.0.2.3 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2054ms
[root@alma1 ~]#

```

Т.е. мы ей должны указать, что путь к адресу 10.0.2.3 лежит через альму2, причём через тот адрес, который нам доступен:

```
ip route add 10.0.2.3 via 10.0.1.2
ip ro sh
```

Как видите, у нас появился ещё один маршрут. Попробуем пингануть?

```
ping 10.0.2.3
```

И ничего, нет ответа. Есть идеи, почему так? Попробуем разобраться.

```

[alma-1 (reset) [Running] - Oracle VM VirtualBox] [alma-2 (reset) [Running] - Oracle VM VirtualBox] [alma-3 (reset) [Running] - Oracle VM VirtualBox]
[root@alma2 ~]# tcpdump
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on myteam, link-type EN10MB (Ethernet), capture size 262144 bytes
12:28:44.418953 IP 10.0.1.1 > 10.0.2.3: ICMP echo request, id 10429, seq 1,
12:28:44.419047 IP 10.0.1.1 > 10.0.2.3: ICMP echo request, id 10429, seq 1,
12:28:45.426886 IP 10.0.1.1 > 10.0.2.3: ICMP echo request, id 10429, seq 2,
12:28:45.426953 IP 10.0.1.1 > 10.0.2.3: ICMP echo request, id 10429, seq 2,
12:28:46.450881 IP 10.0.1.1 > 10.0.2.3: ICMP echo request, id 10429, seq 3,
12:28:46.450948 IP 10.0.1.1 > 10.0.2.3: ICMP echo request, id 10429, seq 3,
^C

```

Для этого запускаем пинг и идём на alma2, где запускаем tcpdump:

```
tcpdump
```

И так, в выводе мы видим, что пакеты от alma1 всё таки доходят до alma2. Более того, они перенаправляются на alma3. Но мы не видим, чтобы alma3 присылала ответ - все пакеты исходят исключительно от alma1. Если ответов нет - значит либо alma3 не хочет отвечать, либо не знает куда.

```

[alma-1 (reset) (Running) - Oracle VM VirtualBox] [alma-2 (reset) (Running) - Oracle VM VirtualBox] [alma-3 (reset) (Running) - Oracle VM VirtualBox]
[root@alma3 ~]# ip ro sh
10.0.2.0/24 dev myteam.2 proto kernel scope link src 10.0.2.3 metric 401
10.0.3.0/24 dev myteam.3 proto kernel scope link src 10.0.3.3 metric 400
[root@alma3 ~]# ip ro add 10.0.1.0/24 via 10.0.2.2
[root@alma3 ~]# ip ro sh
10.0.1.0/24 via 10.0.2.2 dev myteam.2
10.0.2.0/24 dev myteam.2 proto kernel scope link src 10.0.2.3 metric 401
10.0.3.0/24 dev myteam.3 proto kernel scope link src 10.0.3.3 metric 400
[root@alma3 ~]# _

```

Смотрим таблицу маршрутизации на alma3:

```
ip ro sh
```

И действительно, alma3 без понятия, как ответить alma1. Давайте добавим маршрут, но, на этот раз, не на один конкретный хост, а на целую сеть:

```
ip ro add 10.0.1.0/24 via 10.0.2.2
ip ro sh
```

Теперь alma3 на любой из адресов сети 10.0.1.0 будет направлять пакет через 10.0.2.2.

```

[alma-1 (reset) (Running) - Oracle VM VirtualBox] [alma-2 (reset) (Running) - Oracle VM VirtualBox]
[root@alma3 ~]# ping 10.0.1.1
PING 10.0.1.1 (10.0.1.1) 56(84) bytes of data.
64 bytes from 10.0.1.1: icmp_seq=1 ttl=63 time=3.12 ms
^C
--- 10.0.1.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 3.116/3.116/3.116/0.000 ms
[root@alma3 ~]# _

```

И теперь при попытке пинга:

```
ping 10.0.1.1
```

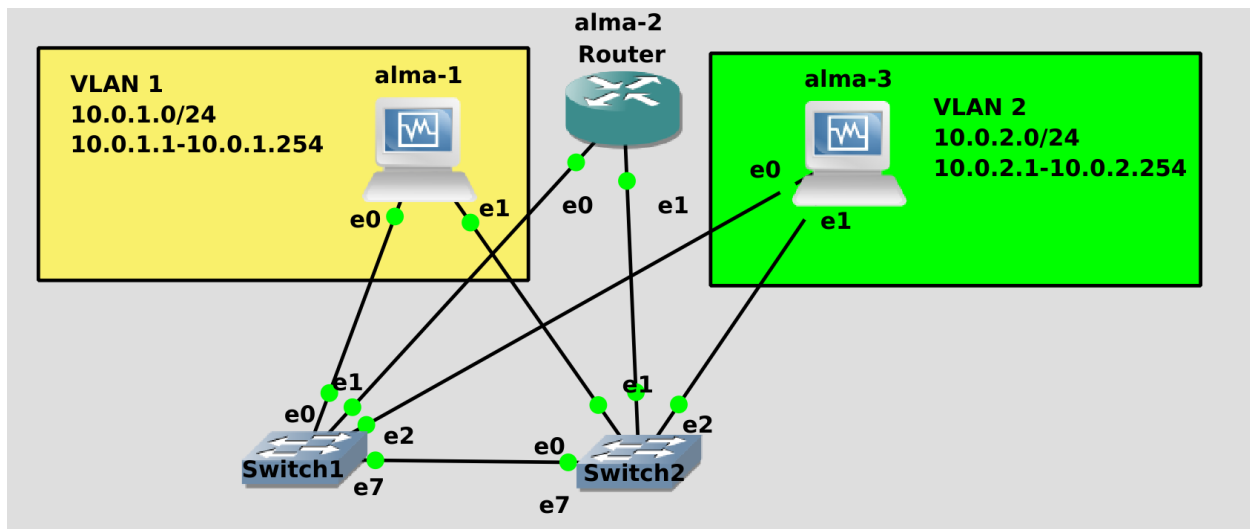
всё работает.

```

[alma-1 (reset) (Running) - Oracle VM VirtualBox] [alma-2 (reset) (Running) - Oracle VM VirtualBox] [alma-3 (reset) (Running) - Oracle VM VirtualBox]
[root@alma2 ~]# tcpdump
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on myteam, link-type EN10MB (Ethernet), capture size 262144 bytes
12:50:46.972182 IP 10.0.2.3 > 10.0.1.1: ICMP echo request, id 3901, seq 3, length 64
12:50:46.972207 IP 10.0.2.3 > 10.0.1.1: ICMP echo request, id 3901, seq 3, length 64
12:50:46.972774 IP 10.0.1.1 > 10.0.2.3: ICMP echo reply, id 3901, seq 3, length 64
12:50:46.972798 IP 10.0.1.1 > 10.0.2.3: ICMP echo reply, id 3901, seq 3, length 64
^C
4 packets captured
4 packets received by filter
0 packets dropped by kernel

```

А в tcpdump видно, что пакеты идут в обоих направлениях.



Но это у нас 2 подсети. А если их 5? 10? 100? Сколько подсетей в интернете? Разве мы можем прописать каждую подсеть в мире? Нет конечно! Вместо этого мы используем специальный маршрут, называемый шлюзом или gateway, или default gateway. Когда компьютеру надо обратиться к сети, куда он не знает маршрута, он всё посылает в гейтвей. Мы можем на хостах вместо всех маршрутов указать, что гейтвеем является alma2. И не нужно будет указывать, в какие сети этот гейтвей ведёт - просто все неизвестные сети по-умолчанию будут посылаться на alma2.

```

alma-1 (reset) [Running] - Oracle VM VirtualBox
[root@alma1 ~]# ip ro sh
10.0.1.0/24 dev myteam proto kernel scope link src 10.0.1.1 metric 350
10.0.2.3 via 10.0.1.2 dev myteam
10.0.3.0/24 dev myteam.3 proto kernel scope link src 10.0.3.1 metric 400
[root@alma1 ~]# ip ro del 10.0.2.3 via 10.0.1.2
[root@alma1 ~]# ip ro add default via 10.0.1.2
[root@alma1 ~]# ip ro sh
default via 10.0.1.2 dev myteam
10.0.1.0/24 dev myteam proto kernel scope link src 10.0.1.1 metric 350
10.0.3.0/24 dev myteam.3 proto kernel scope link src 10.0.3.1 metric 400
[root@alma1 ~]# _

```

Для этого на первой альме удалим маршрут, ведущий к третьему хосту:

```
ip ro sh
ip ro del 10.0.2.3 via 10.0.1.2
```

После чего добавим дефолтный маршрут:

```
ip ro add default via 10.0.1.2
ip ro sh
```

Как видите, появилась надпись default.

```

alma-1 (reset) [Running] - Oracle VM VirtualBox
alma-2 (reset) [Running] - Oracle VM VirtualBox
alma-3 (reset) [Running] - Oracle VM VirtualBox
[root@alma3 ~]# ip ro sh
10.0.1.0/24 via 10.0.2.2 dev myteam.2
10.0.2.0/24 dev myteam.2 proto kernel scope link src 10.0.2.3 metric 400
10.0.3.0/24 dev myteam.3 proto kernel scope link src 10.0.3.3 metric 401
[root@alma3 ~]# ip ro del 10.0.1.0/24 via 10.0.2.2
[root@alma3 ~]# ip ro add 0.0.0.0/0 via 10.0.2.2
[root@alma3 ~]# ip ro sh
default via 10.0.2.2 dev myteam.2
10.0.2.0/24 dev myteam.2 proto kernel scope link src 10.0.2.3 metric 400
10.0.3.0/24 dev myteam.3 proto kernel scope link src 10.0.3.3 metric 401
[root@alma3 ~]#

```

На альме 3 тоже удаляем добавленный маршрут:

```
ip ro sh
ip ro del 10.0.1.0/24 via 10.0.2.2
```

После чего добавляем gateway. Гейтвей можно указывать и в виде специального IP адреса:

```
ip ro add 0.0.0.0/0 via 10.0.2.2
ip ro sh
```

Это означает любой адрес в любой подсети.

```

Alma-1 (reset) [Running] - Oracle VM VirtualBox
[root@alma3 ~]# ping 10.0.1.1
PING 10.0.1.1 (10.0.1.1) 56(84) bytes of data.
64 bytes from 10.0.1.1: icmp_seq=1 ttl=63 time=3.16 ms
^C
--- 10.0.1.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 3.160/3.160/3.160/0.000 ms
[root@alma3 ~]#

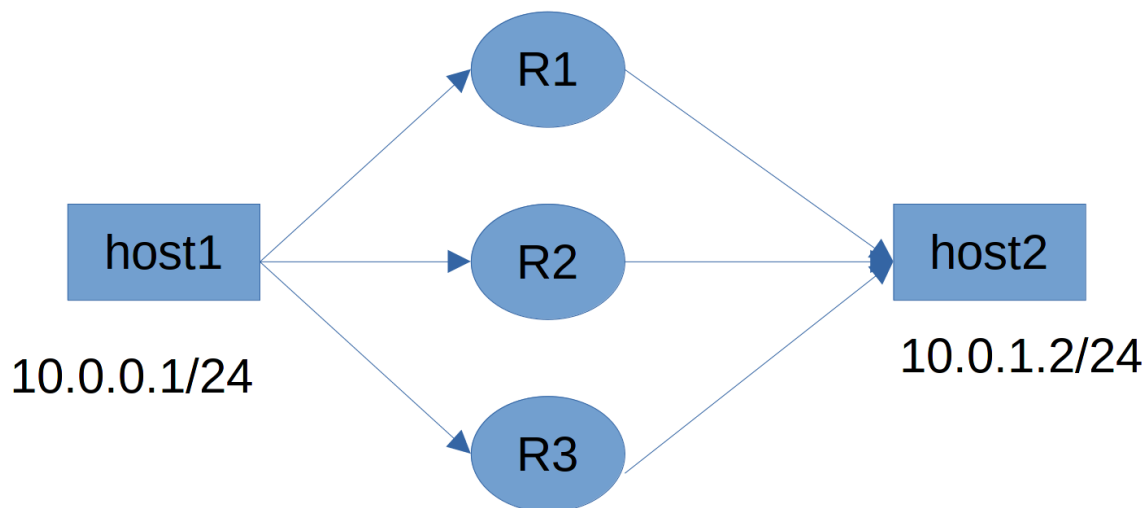
Alma-2 (reset) [Running] - Oracle VM VirtualBox

```

Ну и для проверки запустим пинг:

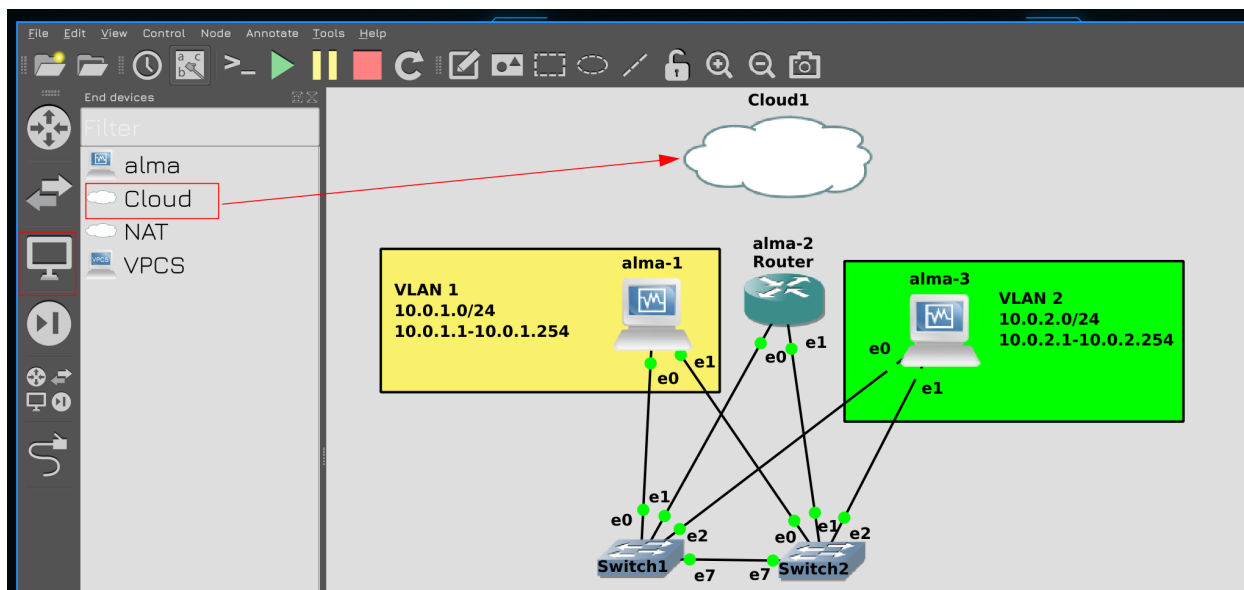
```
ping 10.0.1.1
```

Как видно, всё работает.

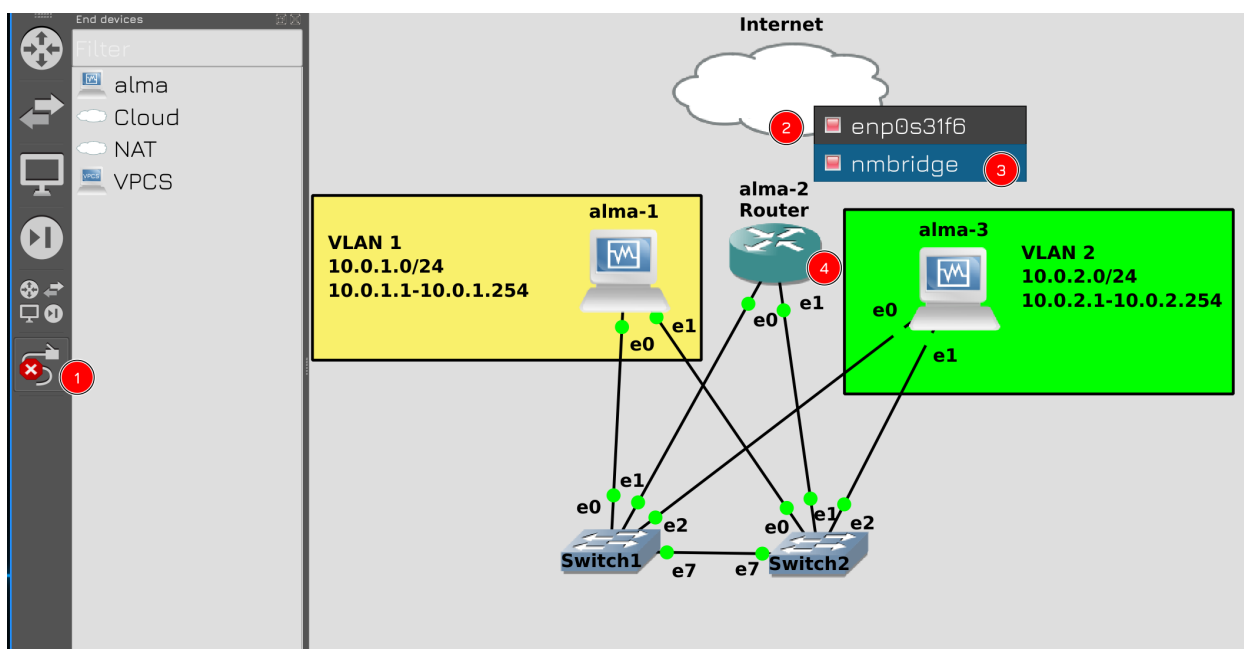


- 1) 10.0.1.2 via R1
- 2) 10.0.1.0/24 via R2
- 3) 0.0.0.0/0 via R3

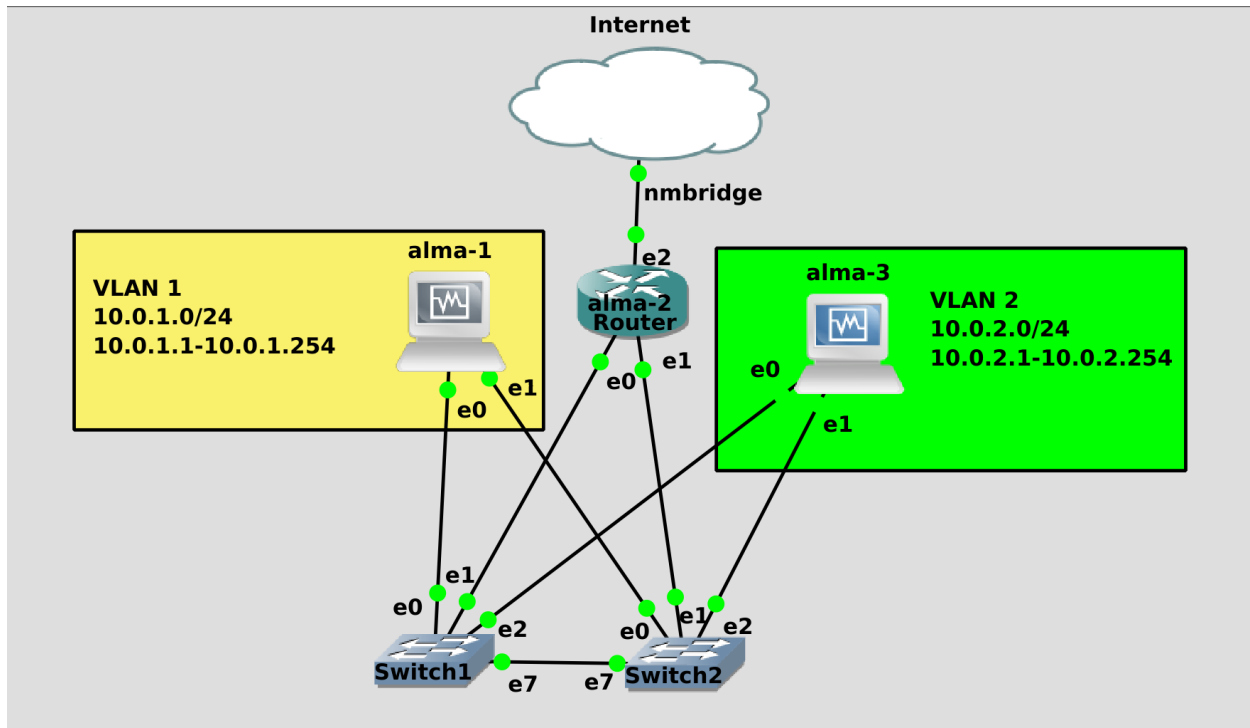
Маршрутов и роутеров, ведущих к одним и тем же хостам может быть несколько. Но приоритет у тех маршрутов, которые точнее. Т.е. условно, если я укажу маршрут через роутер 1 к конкретному хосту, через роутер 2 к целой подсети, а роутер 3 укажу в качестве гейтвея, то обращаясь к хосту будет использоваться маршрут через роутер 2.



Теперь давайте подключим нашу схему к интернету. Для этого нажмите в GNS на иконку монитора в левой панели, выберите Cloud и перетяните в центр.



Затем подключите облако к роутеру с помощью кабеля. Облако представляет ваш компьютер, и как мы в VirtualBox выбирали адаптер для сетевого моста, также и тут нужно выбрать адаптер компьютера, подключенный к сети. Со стороны alma2 выберите 3 порт.



В итоге у вас получится такая схема. Здесь у нас локальная сеть, состоящая из 3 виртуалок. Одна из виртуалок выступает роутером и будет подключена как к локальной сети внутри GNS, так и к домашней сети.

```

[alma-1 (reset) (Running) - Oracle VM VirtualBox] [alma-2 (reset) (Running) - Oracle VM VirtualBox] [alma-3 (reset) (Running) - Oracle VM VirtualBox]
[root@alma2 ~]# ip a show enp0s9
4: enp0s9: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:3e:d6:35 brd ff:ff:ff:ff:ff:ff
[root@alma2 ~]# nmcli connection add type ethernet con-name internet ifname enp0s9
Connection 'internet' (7c413fbd-27ee-4de1-b6bf-f6f6211bc96c) successfully added.
[root@alma2 ~]# nmcli connection show
NAME                UUID                                  TYPE      DEVICE
internet            7c413fbd-27ee-4de1-b6bf-f6f6211bc96c ethernet  enp0s9
teams3s8            bd17cb75-d9f6-472b-afbc-8611648e8d52 team      myteam
vlan2               0491440e-6a4b-4cb8-9824-2cfb7e36e643 vlan      myteam.2
vlan3               1c81ba71-08c9-4573-8903-57f33470b3c1 vlan      myteam.3
team-slave-enp0s3   0f3832d5-b2e4-458f-a456-b3b4a452db2b ethernet  enp0s3
team-slave-enp0s8   26ce0b17-52bb-4fc7-a9b8-61aacb06fa90 ethernet  enp0s8
[root@alma2 ~]# ip a show enp0s9
4: enp0s9: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:3e:d6:35 brd ff:ff:ff:ff:ff:ff
    inet 192.168.31.252/24 brd 192.168.31.255 scope global dynamic noprefixroute enp0s9
        valid_lft 43193sec preferred_lft 43193sec
    inet6 fe80::cad3:7dcf:1832:3532/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
[root@alma2 ~]#

```

Так как на alma2 мы подключили провод к третьему порту, то поднимется интерфейс enp0s9:

```
ip a show enp0s9
```

Нам в NetworkManager надо будет создать новый профиль:

```
nmcli con add type ethernet con-name internet ifname enp0s9
nmcli con sh
```

типом выбираем ethernet, так как это не vlan и не туннельный интерфейс, а в качестве интерфейса указываем enp0s9. IP адрес указывать не нужно, так как наш домашний роутер по DHCP даст ему адрес:

```
ip a show enp0s9
```

```

[alma-1 (reset) (Running)] - Oracle VM VirtualBox
[alma-2 (reset) (Running)] - Oracle VM VirtualBox
[alma-3 (reset) (Running)] - Oracle VM VirtualBox

[root@alma2 ~]# ip ro sh
default via 192.168.31.1 dev enp0s9 proto dhcp metric 102
10.0.1.0/24 dev myteam proto kernel scope link src 10.0.1.2 metric 350
10.0.2.0/24 dev myteam.2 proto kernel scope link src 10.0.2.2 metric 400
10.0.3.0/24 dev myteam.3 proto kernel scope link src 10.0.3.2 metric 401
192.168.31.0/24 dev enp0s9 proto kernel scope link src 192.168.31.252 metric 102
[root@alma2 ~]# ping 1.1.1.1
PING 1.1.1.1 (1.1.1.1): 56(84) bytes of data.
64 bytes from 1.1.1.1: icmp_seq=1 ttl=55 time=88.6 ms
^C
--- 1.1.1.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 88.636/88.636/88.636/0.000 ms
[root@alma2 ~]# ping ya.ru
PING ya.ru (87.250.250.242): 56(84) bytes of data.
64 bytes from ya.ru (87.250.250.242): icmp_seq=1 ttl=245 time=46.6 ms
^C
--- ya.ru ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 46.643/46.643/46.643/0.000 ms
[root@alma2 ~]# _

```

При этом по DHCP мы получаем также default gateway:

```
ip ro sh
```

Им выступает наш домашний роутер. Ну и давайте убедимся, что интернет доступен, пинганём 1.1.1.1 и ya.ru:

```
ping 1.1.1.1
ping ya.ru
```

Всё пингуется.

```

[alma-1 (reset) (Running)] - Oracle VM VirtualBox
[alma-2 (reset) (Running)] - Oracle VM VirtualBox
[alma-3 (reset) (Running)] - Oracle VM VirtualBox

[root@alma1 ~]# nmcli connection modify teams3s8 ipv4.gateway 10.0.1.2
[root@alma1 ~]# nmcli connection up teams3s8
Connection successfully activated (master waiting for slaves) (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/8)
[root@alma1 ~]# ip ro sh
default via 10.0.1.2 dev myteam proto static metric 350 linkdown
10.0.1.0/24 dev myteam proto kernel scope link src 10.0.1.1 metric 350 linkdown
10.0.2.0/24 dev myteam.2 proto kernel scope link src 10.0.2.1 metric 400 linkdown
10.0.3.0/24 dev myteam.3 proto kernel scope link src 10.0.3.1 metric 400 linkdown
[root@alma1 ~]#

```

Теперь давайте поправим маршруты на alma1 и alma3. Мы их прописывали с помощью команды ip, а значит эти настройки слетят. Нам же нужно указать gateway в профиле NetworkManager-a:

```
nmcli con mod teams3s8 ipv4.gateway 10.0.1.2
```

После чего поднять заново профиль и проверить:

```
nmcli con up teams3s8
ip ro sh
```

```

[alma-1 (reset) (Running)] - Oracle VM VirtualBox
[alma-2 (reset) (Running)] - Oracle VM VirtualBox
[alma-3 (reset) (Running)] - Oracle VM VirtualBox

[root@alma3 ~]# nmcli con mod vlan2 ipv4.gateway 10.0.2.2
[root@alma3 ~]# nmcli connection up vlan2
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/87)
[root@alma3 ~]# ip ro sh
default via 10.0.2.2 dev myteam.2 proto static metric 402
10.0.2.0/24 dev myteam.2 proto kernel scope link src 10.0.2.3 metric 402
10.0.3.0/24 dev myteam.3 proto kernel scope link src 10.0.3.3 metric 401
[root@alma3 ~]# _

```

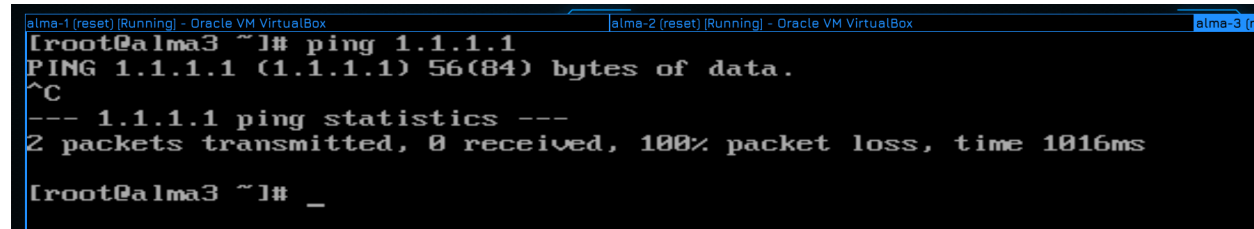
Тоже самое проделываем на alma3, изменяем профиль:

```
nmcli con mod vlan2 ipv4.gateway 10.0.2.2
```

Затем поднимаем интерфейс и проверяем маршруты:

```
nmcli con up vlan2
ip ro sh
```

Всё как мы и хотели.



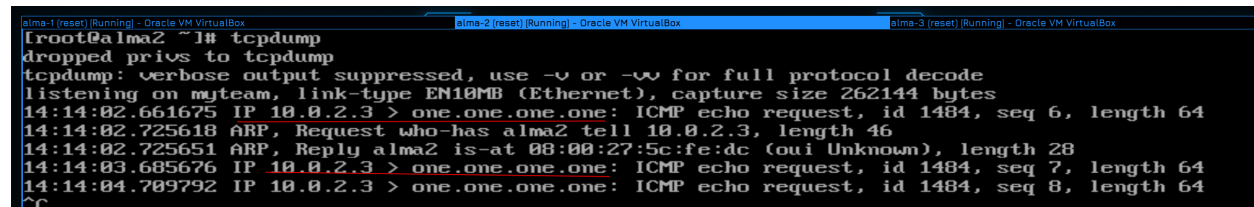
```
alma-1 (reset) [Running] - Oracle VM VirtualBox
alma-2 (reset) [Running] - Oracle VM VirtualBox
alma-3 (reset) [Running] - Oracle VM VirtualBox
[root@alma3 ~]# ping 1.1.1.1
PING 1.1.1.1 (1.1.1.1) 56(84) bytes of data.
^C
--- 1.1.1.1 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1016ms

[root@alma3 ~]# _
```

Как думаете, интернет заработает на этих хостах? Давайте проверим:

```
ping 1.1.1.1
```

Как видите, не пингуется. При этом нет ошибки network unreachable - у нас есть gateway, а значит мы знаем куда обращаться, за кем искать этот адрес. Но почему тогда не пингуется?



```
alma-1 (reset) [Running] - Oracle VM VirtualBox
alma-2 (reset) [Running] - Oracle VM VirtualBox
alma-3 (reset) [Running] - Oracle VM VirtualBox
[root@alma2 ~]# tcpdump
dropped prius to tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on myteam, link-type EN10MB (Ethernet), capture size 262144 bytes
14:14:02.661675 IP 10.0.2.3 > one.one.one.one: ICMP echo request, id 1484, seq 6, length 64
14:14:02.725618 ARP, Request who-has alma2 tell 10.0.2.3, length 46
14:14:03.685676 IP 10.0.2.3 > one.one.one.one: ICMP echo request, id 1484, seq 7, length 64
14:14:04.709792 IP 10.0.2.3 > one.one.one.one: ICMP echo request, id 1484, seq 8, length 64
^C
```

Помните проблему, когда мы в начале пытались с alma1 пингануть alma3? Давайте также запустим пинг и посмотрим tcpdump на alma2:

```
tcpdump
```

Ошибка ничего не напоминает? Пакет идёт от хоста 3 к 1.1.1.1, но ответов то нет. С чем это было связано в прошлый раз? Правильно, другой хост не знал обратный маршрут, нам нужно было на alma3 прописать маршрут к alma1.

Но можем ли мы на сервере 1.1.1.1 прописать маршрут к нашей виртуалке? Конечно нет, между нашей виртуалкой и этим сервером десятков разных роутеров, стоящих у разных провайдеров. У каждого из этих провайдеров может быть своя сеть с адресом 10.0.2.3, потому что это адрес, выделенный для приватного использования. А как тогда наш домашний комп видит интернет? Правильно, наш роутер на исходящие пакеты вешает свой адрес, заменяя адрес нашего компьютера. И это называется NAT.

И мы по тому же принципу, должны на alma2 на исходящем интерфейсе применить NAT, т.е. заменять адреса всех исходящих пакетов на адрес интерфейса enp0s9 на alma2.

```

[alma-1 (reset) [Running] - Oracle VM VirtualBox] [alma-2 (reset) [Running] - Oracle VM VirtualBox]
[root@alma2 ~]# firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp0s3 enp0s8 enp0s9 myteam myteam.2 myteam.3
  sources:
  services: cockpit dhcpv6-client ssh
  ports:
  protocols:
  forward: no
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
[root@alma2 ~]# firewall-cmd --list-all --zone=external
external
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: ssh
  ports:
  protocols:
  forward: no
  masquerade: yes
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
[root@alma2 ~]# _

```

Обычно за NAT отвечает firewall:

```
firewall-cmd --list-all
```

Наш интерфейс enp0s9 сейчас находится в зоне public. Нам в идеале нужна отдельная зона для внешней сети и чтобы исходящие пакеты из этой зоны натились. И такая зона есть - external:

```
firewall-cmd --list-all --zone=external
```

Как видите, в этой зоне опция masquerade имеет значение yes, а это именно то, что нам нужно для NAT-а.

```

[alma-1 (reset) [Running] - Oracle VM VirtualBox] [alma-2 (reset) [Running] - Oracle VM VirtualBox] [alma-3 (reset) [Running] - Oracle VM VirtualBox]
[root@alma2 ~]# firewall-cmd --change-interface=enp0s9 --zone=external --permanent
The interface is under control of NetworkManager, setting zone to 'external'.
success
[root@alma2 ~]# firewall-cmd --reload
success
[root@alma2 ~]# firewall-cmd --list-all --zone=external
external (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp0s9
  sources:
  services: ssh
  ports:
  protocols:
  forward: no
  masquerade: yes
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
[root@alma2 ~]#

```

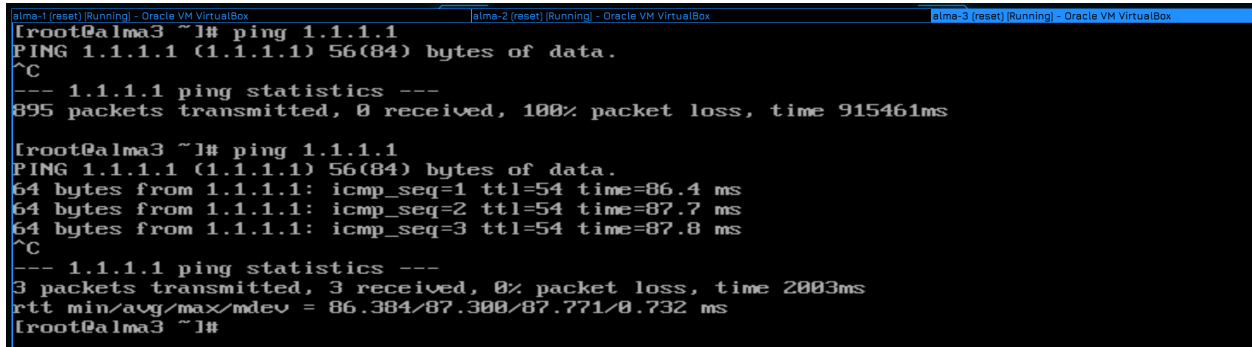
Поэтому всё что нам остаётся - сменить зону для интерфейса enp0s9:

```
firewall-cmd --change-interface=enp0s9 --zone=external --permanent
```

После чего не забудьте перечитать настройки фаервола и убедиться, что всё применилось:

```
firewall-cmd --reload
firewall-cmd --list-all --zone=external
```

Как видите, теперь enp0s9 находится в зоне external.



```

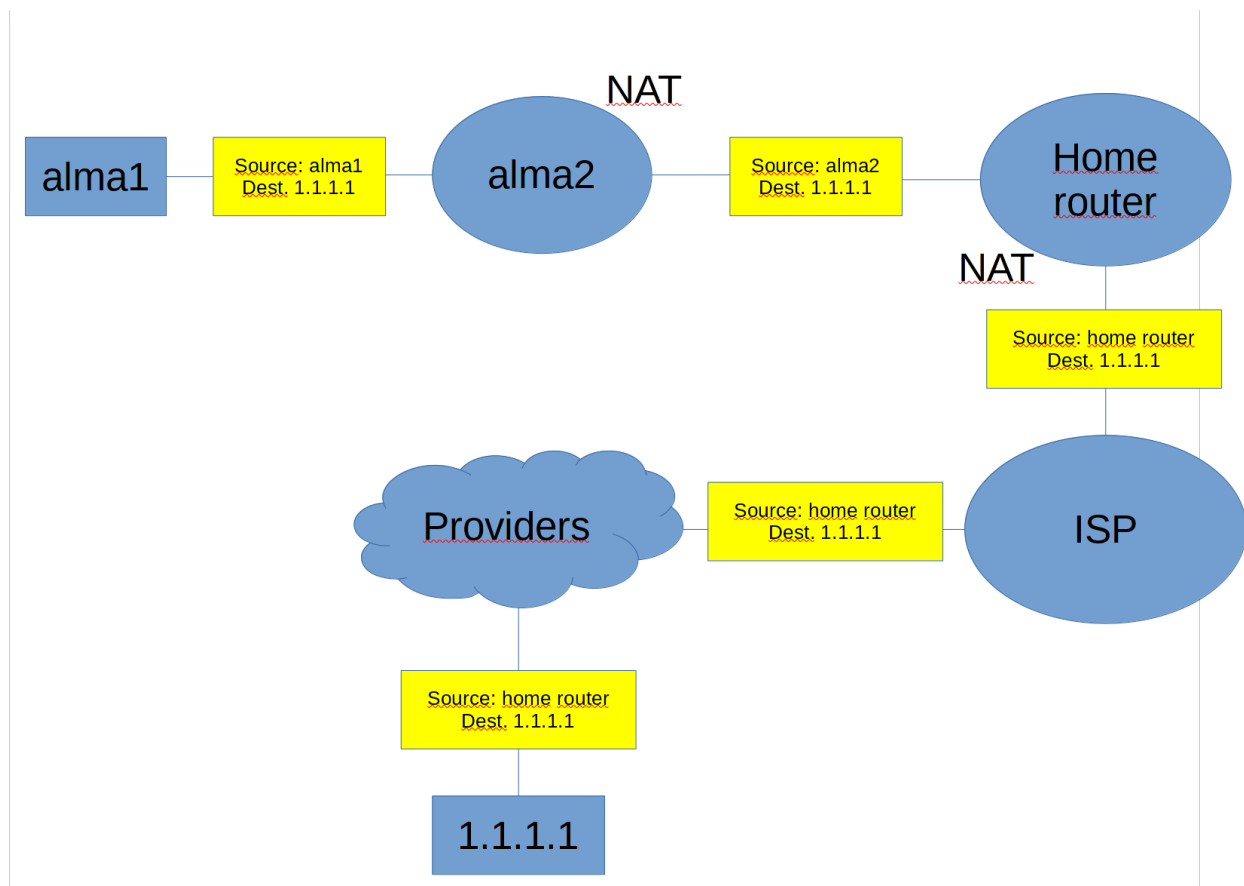
[alma-1 (reset) (Running) - Oracle VM VirtualBox] [alma-2 (reset) (Running) - Oracle VM VirtualBox] [alma-3 (reset) (Running) - Oracle VM VirtualBox]
[root@alma3 ~]# ping 1.1.1.1
PING 1.1.1.1 (1.1.1.1) 56(84) bytes of data.
^C
--- 1.1.1.1 ping statistics ---
895 packets transmitted, 0 received, 100% packet loss, time 915461ms

[root@alma3 ~]# ping 1.1.1.1
PING 1.1.1.1 (1.1.1.1) 56(84) bytes of data:
64 bytes from 1.1.1.1: icmp_seq=1 ttl=54 time=86.4 ms
64 bytes from 1.1.1.1: icmp_seq=2 ttl=54 time=87.7 ms
64 bytes from 1.1.1.1: icmp_seq=3 ttl=54 time=87.8 ms
^C
--- 1.1.1.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 86.384/87.300/87.771/0.732 ms
[root@alma3 ~]#
  
```

Проверим ping:

```
ping 1.1.1.1
```

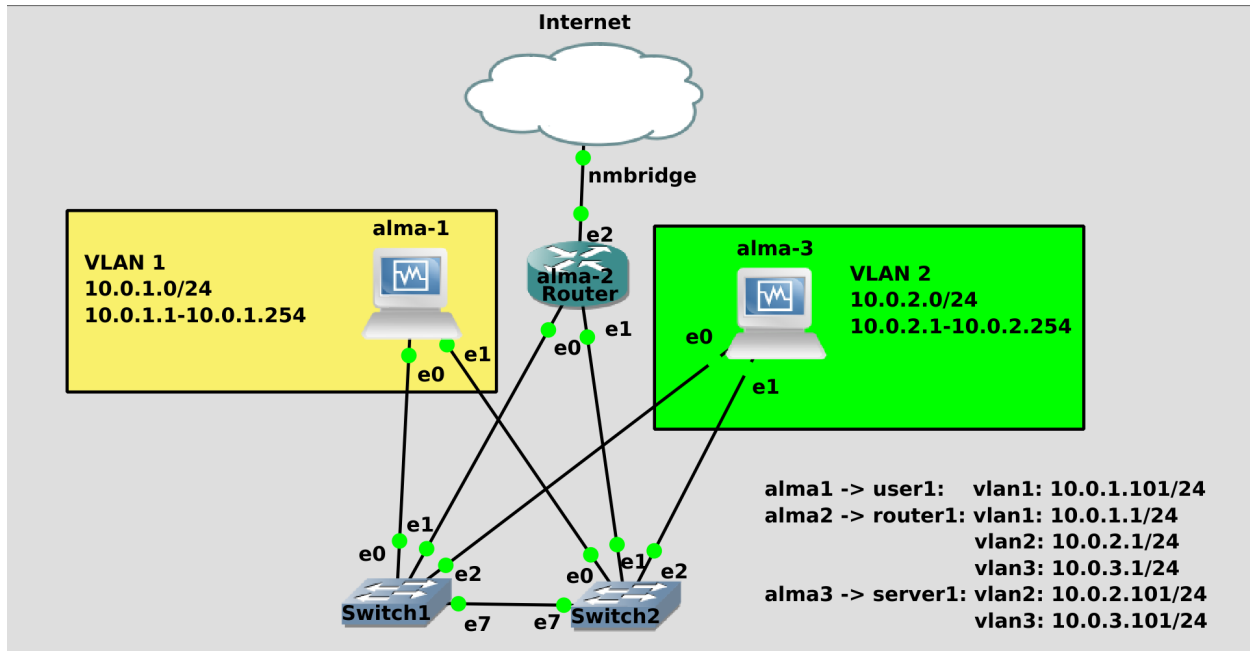
Пинги идут.



В итоге получается такая схема. Когда **alma1** пытается обратиться к серверу **1.1.1.1**, то сначала она посылает пакет на **alma2**, которая натит пакет и отправляет его на мой домашний роутер. Дальше мой домашний роутер натит пакет и отправляет провайдеру. Если на моём роутере внутренний адрес провайдера, т.е. серый IP, то провайдер сам в свою очередь натит, но если провайдер даёт мне публичный адрес - то он просто перенаправляет пакет на следующего провайдера, скажем, государственного. И дальше по цепочке от одного провайдера к другому пересылается пакет, пока он не доходит до сервера **1.1.1.1**. А дальше сервер сможет ответить, потому что знает куда.

На самом деле я мог бы избежать двойного использования NAT-а, если бы на домашнем роутере прописал маршрут к внутренней сети моего GNS через **alma2**. И тогда мой компьютер смог бы видеть все виртуалки, потому что он бы посылал все пакеты на мой домашний роутер, тот бы перенаправлял их на **alma2**, а она бы в свою очередь перенаправляла на **alma1** и 3.

Но я хочу, чтобы наша сетка в GNS оставалась в изолированном виде и единственным способом попасть в неё был бы один внешний адрес **alma2**.



Суммируя всё что мы сегодня прошли, я хочу, чтобы вы немного перестроили сеть:

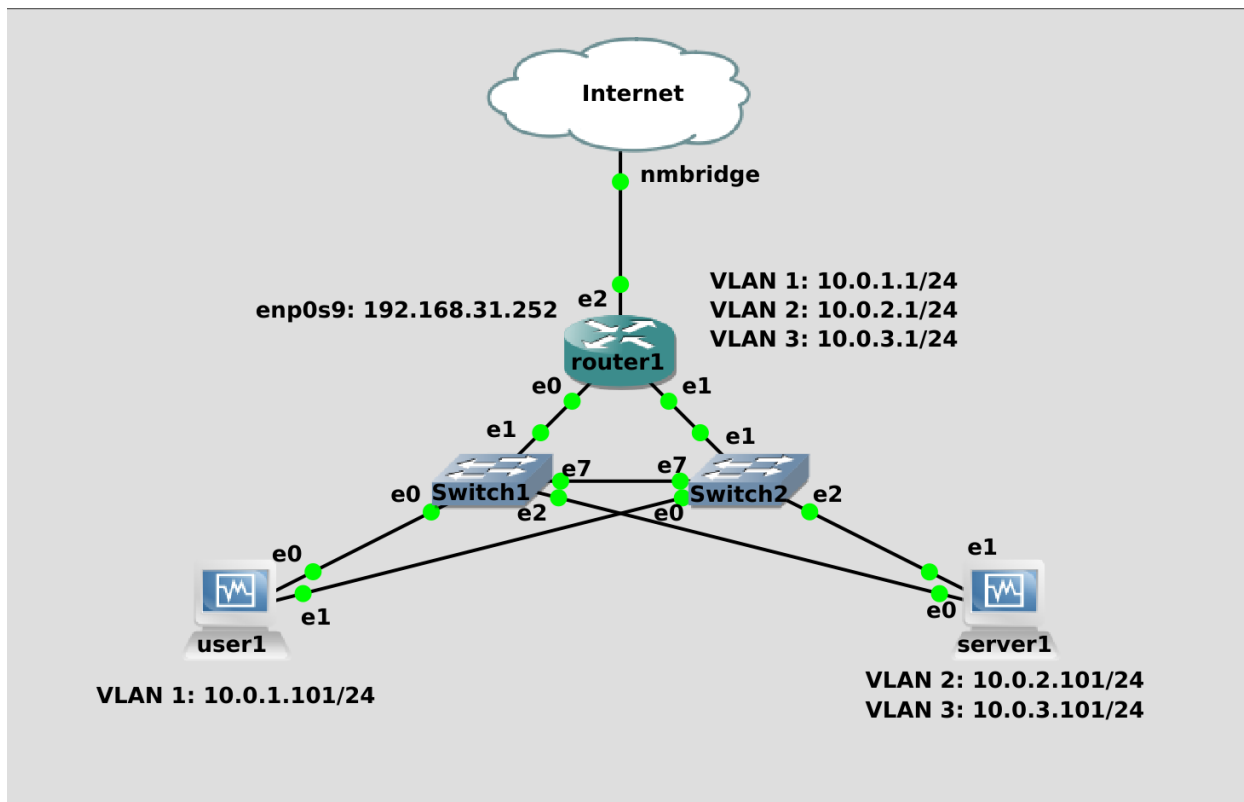
- переименуйте alma1 в user1 и оставьте на ней только влан1 с адресом 1.101.
- переименуйте alma2 в router1 и поменяйте IP адреса всех 3 вланов на 1. Обычно в организациях гейтвеям дают либо первый, либо последний адрес подсети.
- переименуйте alma3 в server1 и оставьте на ней только 2 и 3 вланы. Адреса поменяйте на 2.101 и 3.101 соответственно.

И, естественно, убедитесь, что всё работает - что все друг друга пингуют по разным сетям, и что со всех хостов пингуются 1.1.1.1.

Давайте подведём итоги. Сегодня мы с вами разобрали, что такое подсети, что такое маска подсети, для чего нужна маршрутизация, зачем нужен gateway и в чём смысл NAT-а. Это, конечно, совсем основы маршрутизации, но для системного администратора этого хватает для решения большинства задач.

2.6 06. Как попасть в сеть через SSH

2.6.1 06. Как попасть в сеть через SSH



В прошлый раз мы наконец настроили связь между сетью в GNS и нашей домашней сетью. Связывает их router1, у которого на одном из интерфейсов есть адрес из домашней сети. Точно также в реальной инфраструктуре у вас есть роутер, у которого на одном из интерфейсов есть публичный адрес, выданный провайдером.

```
[doctor@tardis]~$ ping 10.0.2.101
PING 10.0.2.101 (10.0.2.101) 56(84) bytes of data.
^C
--- 10.0.2.101 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1023ms

[doctor@tardis]~$ ip ro sh
default via 192.168.31.1 dev nmbridge proto dhcp metric 425
192.168.31.0/24 dev nmbridge proto kernel scope link src 192.168.31.227 metric 425
[doctor@tardis]~$
```

В реальной сети вы не можете так просто из дома подключиться в рабочую сеть:

```
ping 10.0.2.101
ip ro sh
```

Как правило, для удалённой работы сотрудников организуют VPN сервер, с помощью которого созда-

ётся безопасный туннель из одной сети в другую. Почти все современные роутеры и UTM решения имеют такой функционал, поэтому мы тоже будем поднимать свой vpn сервер, но не сегодня. Даже без VPN-а мы можем организовать безопасный туннель и для этого достаточно одного ssh. Для начала представим, что нам надо подключиться по ssh к серверам.

```
[doctor@tardis]-[~]
$ ssh root@192.168.31.252
The authenticity of host '192.168.31.252 (192.168.31.252)' can't be established.
ED25519 key fingerprint is SHA256:aHIqhQPsr0cnbISJGZz7KugA7a7/0IvpMa0E8suhn7I.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.31.252' (ED25519) to the list of known hosts.
root@192.168.31.252's password:
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Sat Dec 11 08:49:26 2021
[root@router1 ~]# ssh 10.0.2.101
The authenticity of host '10.0.2.101 (10.0.2.101)' can't be established.
ECDSA key fingerprint is SHA256:jub6CYigICQITanYXEKMOFOW09ChdC5Pf4SnrY50B8U.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.2.101' (ECDSA) to the list of known hosts.
root@10.0.2.101's password:
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Wed Dec  8 19:45:20 2021
[root@server1 ~]#
```

Самый примитивный способ - подключиться по ssh к роутеру:

```
ssh root@192.168.31.252
```

а дальше из него подключаться к другим машинкам, например, к серверу1:

```
ssh root@10.0.2.101
```

В принципе, и так сойдёт. Но если у нас множество машин и нам постоянно нужно заходить туда-сюда, это становится немного неудобно.

Инсталляционный сервер

Материал из Википедии — свободной энциклопедии

[[править](#) | [править код](#)]

[Править пре](#)



Значимость предмета статьи поставлена под сомнение.

Пожалуйста, покажите в статье значимость её предмета, добавив в неё доказательства значимости по [частным критериям значимости](#) или, в случае если частные критерии значимости для предмета статьи отсутствуют, по [общему критерию значимости](#). Подробности могут быть [на странице обсуждения](#).

- Дата постановки шаблона: 19 сентября 2019

Инсталляционный сервер (install server, jump server, jump host или jumpbox) — так называют специально отведенный компьютер в сети, предназначенный для доступа к устройствам в демилитаризованной зоне ([ДМЗ](#)) организации. Наиболее часто используется для управления узлами в [ДМЗ](#) из доверенных сетей или компьютеров.

Инсталляционный сервер — это защищённый и [мониторящийся](#) компьютер находящийся в двух разных зонах безопасности, он служит контролируемым средством для доступа между ними. [Права доступа](#) пользователей к нему должны быть строго [разграничены](#) и [контролироваться](#).

Мы роутер используем как промежуточный хост для подключения. Такие промежуточные хосты, через которые можно попасть из одной сети в другую, при этом без маршрутизации - называются jump host-ами. Не обращайтесь внимания, что мы используем роутер как джамп хост, на деле это может быть любая машинка, у которой есть доступ в обе сети.

```
[doctor@tardis]~$ ssh -J root@192.168.31.252 root@10.0.2.101
root@192.168.31.252's password:
The authenticity of host '10.0.2.101 (<no hostip for proxy command>)' can't be established.
ED25519 key fingerprint is SHA256:pneyKLqUSNb1MALbU0Th97JKZ296yui7WD6lCim4sgc.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.2.101' (ED25519) to the list of known hosts.
root@10.0.2.101's password:
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Sat Dec 11 09:17:04 2021 from 10.0.2.1
[root@server1 ~]#
```

Так вот, ssh клиент может упростить подключения через джамп хосты. Достаточно сразу указать с помощью ключа -J, что такой-то сервер является джамп хостом, а мы хотим подключиться туда-то:

```
ssh -J root@192.168.31.252 root@10.0.2.101
```

На фоне происходит тот же процесс, что и в прошлый раз - сначала мы заходим на роутер, а уже потом на сервер1. Мы всё равно должны указывать пароли от обоих хостов. Но теперь для нас это выглядит не как два шага, а как один. Да, команда получилась подлиннее, но это дело поправимое.

```
GNU nano 5.9 /home/doctor/.ssh/config
Host router1
  Hostname 192.168.31.252
  User root

Host server1
  Hostname 10.0.2.101
  User root
  ProxyJump router1
```

Мы можем в конфигах своего ssh клиента ~/.ssh/config указать оба хоста. И сказать, что для подключения к server1 нужно использовать router1 как ProxyJump:

```
Host router1
  Hostname 192.168.31.252
  User root

Host server1
  Hostname 10.0.2.101
  User root
  ProxyJump router1
```

```
[doctor@tardis]~$ nano ~/.ssh/config
[doctor@tardis]~$ ssh server1
root@192.168.31.252's password:
root@10.0.2.101's password:
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Sat Dec 11 09:33:33 2021 from 10.0.2.1
[root@server1 ~]#
```

И теперь для подключения достаточно указать имя целевого хоста, которое мы прописали в конфиге:

```
ssh server1
```

Мы по-прежнему должны вводить пароли как от джамп хоста, так и целевого. Но даже это дело поправимое. Для этого нам нужны ssh ключи.

```
[doctor@tardis]~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/doctor/.ssh/id_rsa):
```

Для начала сгенерируем ключи.

```
ssh-keygen
```

Раз уж взялись строить инфраструктуру, давайте делать это правильно.

1 пара ключей на все сервера
отдельная пара ключей для каждого сервера
1 пара ключей на все сервера с паролем
отдельная пара ключей для каждого сервера с одним паролем
отдельная пара ключей для каждого сервера с разными паролями

Для начала вопрос - должны ли мы делать для каждого сервера отдельный ключ? Или использовать один ключ на всех серверах? Не смотря на всю очевидность, вопрос довольно сложный. С одной стороны, иметь для каждого сервера отдельный ключ кажется безопаснее - потеряв один приватный ключ мы не подвергнем опасности остальные. С другой стороны - какая вероятность, что мы потеряем только один ключ, а не все сразу? Они же лежат в одной директории на одном компе админа. Плюс ещё куда-то бэкапятся. Из-за того что подобрать приватный ключ практически нереально, взломщикам гораздо легче будет украсть ключи с компьютера админа. От этого мы можем защититься поставив пароль на ключи. Правда взломщик может генерацией паролей подобрать нужный. И тут, если у нас один пароль на все ключи - то опять же он получит доступ на все сервера. А вот если мы поставим разные пароли на разные ключи - то взлом всех ключей займёт гораздо больше времени. Но, опять же, и это теоретически возможно. Поэтому, в идеальном случае, у нас должны быть разные ключи для разных хостов с разными паролями и мы их должны периодически менять. Когда инфраструктура не маленькая, так заморачиваться с ключами отнимает слишком много времени. Возможно, если у нас очень критичные данные и нас постоянно кто-то пытается взломать - этого того стоит.

отдельная пара ключей для группы серверов с разными паролями

Но в более спокойных условиях можно пойти на компромисс. Скажем, делить хосты по группам и на каждую группу выдавать одну пару ключей с отдельным паролем. Ну и естественно, периодически менять ключи.

```

[doctor@tardis]~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/doctor/.ssh/id_rsa): .ssh/jumphost
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in .ssh/jumphost
Your public key has been saved in .ssh/jumphost.pub
The key fingerprint is:
SHA256:7VkHpQM+iHvYXJPa6083n0LX/oigewWC6aWVwJ0YWm8 doctor@tardis
The key's randomart image is:
+---[RSA 3072]-----+
|
| .oo o o
| .+.*.+
| .+=Eo o
| o S=O.. .
| .O.=+00 ..
| . ++0 +.
| oo.. + +0
| 00000 o.+
+---[SHA256]-----+
[doctor@tardis]~$

```

И следуя нашему выводу, сгенерируем отдельный ключ для джамп хоста:

```
ssh-keygen
```

Назовём ключи `.ssh/jumphost` и зададим пароль для приватного ключа.

```

[doctor@tardis]~$ ssh-copy-id -i .ssh/jumphost.pub router1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: ".ssh/jumphost.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are
already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to
install the new keys
root@192.168.31.252's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'router1'"
and check to make sure that only the key(s) you wanted were added.

[doctor@tardis]~$ ssh router1
root@192.168.31.252's password:

❗ [doctor@tardis]~$ ssh -i .ssh/jumphost router1
Enter passphrase for key '.ssh/jumphost':
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Sat Dec 11 09:16:53 2021 from 192.168.31.227
[root@router1 ~]#

```

После генерации закинем ключ на джамп хост с помощью команды `ssh-copy-id`:

```
ssh-copy-id -i .ssh/jumphost.pub router1
```

И после ввода пароля ключ добавился на сервер. Теперь, при подключении к серверу, надо указывать ключ, с помощью которого мы хотим зайти на роутер1:

```
ssh -i .ssh/jumphost router1
```

При подключении нас просят ввести пароль от ключа, который мы создали. А после ввода пароля от ключа, на сам сервер мы попадаем без пароля.

```
[doctor@tardis]~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/doctor/.ssh/id_rsa): .ssh/servers
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in .ssh/servers
Your public key has been saved in .ssh/servers.pub
The key fingerprint is:
SHA256:ji/qEkGsFiL0hKGpye9rcZwCUgnFwyYtLtvR4ULpRL0 doctor@tardis
The key's randomart image is:
+---[RSA 3072]-----+
|o@==
|*+^ o
|*@.= o
|*== E
|*+.. . S
|. +o + o
|  o+ . .
|  o. . .
|. =+ . .
+-----[SHA256]-----+
[doctor@tardis]~$
```

Теперь сгенерируем ключи для серверов:

```
ssh-keygen
```

Указываем для них отдельный файл и пароль.

```
GNU nano 5.9 .ssh/config
Host router1
  Hostname 192.168.31.252
  User root
  IdentityFile ~/.ssh/jumphost
Host server1
  Hostname 10.0.2.101
  User root
  ProxyJump router1
```

Прежде чем закинуть ключ на сервер1, укажем в конфиг файле ~/.ssh/config, что для джампхоста требуется отдельный ключ:

```
...
IdentityFile ~/.ssh/jumphost
...
```

```
[doctor@tardis]~$ ssh-copy-id -i .ssh/servers.pub server1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: ".ssh/servers.pub"
Enter passphrase for key '/home/doctor/.ssh/jumphost':
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are
already installed
Enter passphrase for key '/home/doctor/.ssh/jumphost':
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to
install the new keys
Enter passphrase for key '/home/doctor/.ssh/jumphost':
root@10.0.2.101's password:

Number of key(s) added: 1

Now try logging into the machine, with:  "ssh 'server1'"
and check to make sure that only the key(s) you wanted were added.

[doctor@tardis]~$ ssh -i .ssh/servers server1
Enter passphrase for key '/home/doctor/.ssh/jumphost':
Enter passphrase for key '.ssh/servers':
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Sat Dec 11 09:51:33 2021 from 10.0.2.1
[root@server1 ~]#
```

Теперь попытаемся закинуть ключ на сервер1:

```
ssh-copy-id -i .ssh/servers.pub server1
```

ssh у нас попросит несколько раз ввести пароль от ключа для джампхоста - один раз он попытается зайти с помощью ключа, мало ли, вдруг он уже установлен на сервере, затем он попытается добавить ключ, для этого он дополнительно попросит пароль от самого сервера. Затем же мы увидим информацию о том, что ключ успешно добавлен. Теперь можем попытаться зайти с помощью ключа:

```
ssh -i .ssh/servers server1
```

Для этого надо будет ввести пароли от ключей джампхоста и серверов. После чего на сам сервер мы зайдём без пароля.

```
GNU nano 5.9 .ssh/config
Host router1
  Hostname 192.168.31.252
  User root
  IdentityFile ~/.ssh/jumphost

Host server1
  Hostname 10.0.2.101
  User root
  ProxyJump router1
  IdentityFile ~/.ssh/servers
```

Также укажем ключ в конфиг файле ~/.ssh/config, чтобы не указывать ключи при подключении:

```
...
IdentityFile ~/.ssh/servers
...
```

```
[doctor@tardis]~$ ssh server1
Enter passphrase for key '/home/doctor/.ssh/jumphost':
Enter passphrase for key '/home/doctor/.ssh/servers':
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Sat Dec 11 12:11:36 2021 from 10.0.2.1
[root@server1 ~]#
```

На этот раз при подключении мы опять же не указываем никакие опции, а просто вводим пароли от ключей. Но вводить эти пароли при каждом подключении может быть неудобно, особенно, когда у нас множество серверов и мы постоянно подключаемся туда-сюда. И вроде не охота постоянно вводить пароли, но они должны быть на ключах. Для этого есть простое решение - ssh-agent. Вкратце - вы запускаете процесс ssh-agent-a и добавляете в него ключи. Пока процесс работает - вам не нужно будет вводить пароли от ключей. Условно, начинаете работать - запускаете ssh-agent и добавляете в него ключи, вводя пароли от ключей. Дальше спокойно работаете без паролей. Прекращаете работать - идёте на обед или перерыв - убиваете процесс.

```
[doctor@tardis]~$ ssh-agent
SSH_AUTH_SOCK=/tmp/ssh-XXXXXXuo0Hmy/agent.159457; export SSH_AUTH_SOCK;
SSH_AGENT_PID=159458; export SSH_AGENT_PID;
echo Agent pid 159458;
[doctor@tardis]~$ env | grep SSH
[doctor@tardis]~$ SSH_AUTH_SOCK=/tmp/ssh-XXXXXXuo0Hmy/agent.159457; export SSH_AUTH_SOCK;
SSH_AGENT_PID=159458; export SSH_AGENT_PID;
[doctor@tardis]~$ env | grep SSH
SSH_AUTH_SOCK=/tmp/ssh-XXXXXXuo0Hmy/agent.159457
SSH_AGENT_PID=159458
[doctor@tardis]~$
```

Для запуска агента используете команду:

```
ssh-agent
```

При этом команда показывает вам пару команд, с помощью которых вы должны создать переменные окружения. Без этих команд переменные не пропишутся:

```
env | grep SSH
```

И тогда вы не сможете работать с агентом. ssh-агент сам не может прописать эти переменные. Поэтому нужно вручную скопировать эти команды и выполнить:

```
SSH_AUTH_SOCK=/tmp/ssh-XXXXXXuo0Hmy/agent.159457; export SSH_AUTH_SOCK;
SSH_AGENT_PID=159458; export SSH_AGENT_PID;
env | grep SSH
```



```
[doctor@tardis]~$ pkill ssh-agent
[doctor@tardis]~$ eval $(ssh-agent)
Agent pid 163874
[doctor@tardis]~$ env | grep SSH
SSH_AUTH_SOCK=/tmp/ssh-XXXXXXZemAnE/agent.163873
SSH_AGENT_PID=163874
[doctor@tardis]~$
```

Но это может быть не совсем удобно. Есть и другой способ, но для этого давайте завершим процесс агента:

```
pkill ssh-agent
```

А затем запустим новый, но через команду eval:

```
eval $(ssh-agent)
```

Она сразу добавит переменные:

```
env | grep SSH
```

```
[doctor@tardis]~$ ssh-add .ssh/jumphost
Enter passphrase for .ssh/jumphost:
Identity added: .ssh/jumphost (doctor@tardis)
[doctor@tardis]~$ ssh-add .ssh/servers
Enter passphrase for .ssh/servers:
Identity added: .ssh/servers (doctor@tardis)
[doctor@tardis]~$ ssh server1
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Sat Dec 11 12:19:51 2021 from 10.0.2.1
[root@server1 ~]# logout
Connection to 10.0.2.101 closed.
[doctor@tardis]~$ ssh router1
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Sat Dec 11 12:13:24 2021 from 192.168.31.227
[root@router1 ~]# logout
Connection to 192.168.31.252 closed.
[doctor@tardis]~$
```

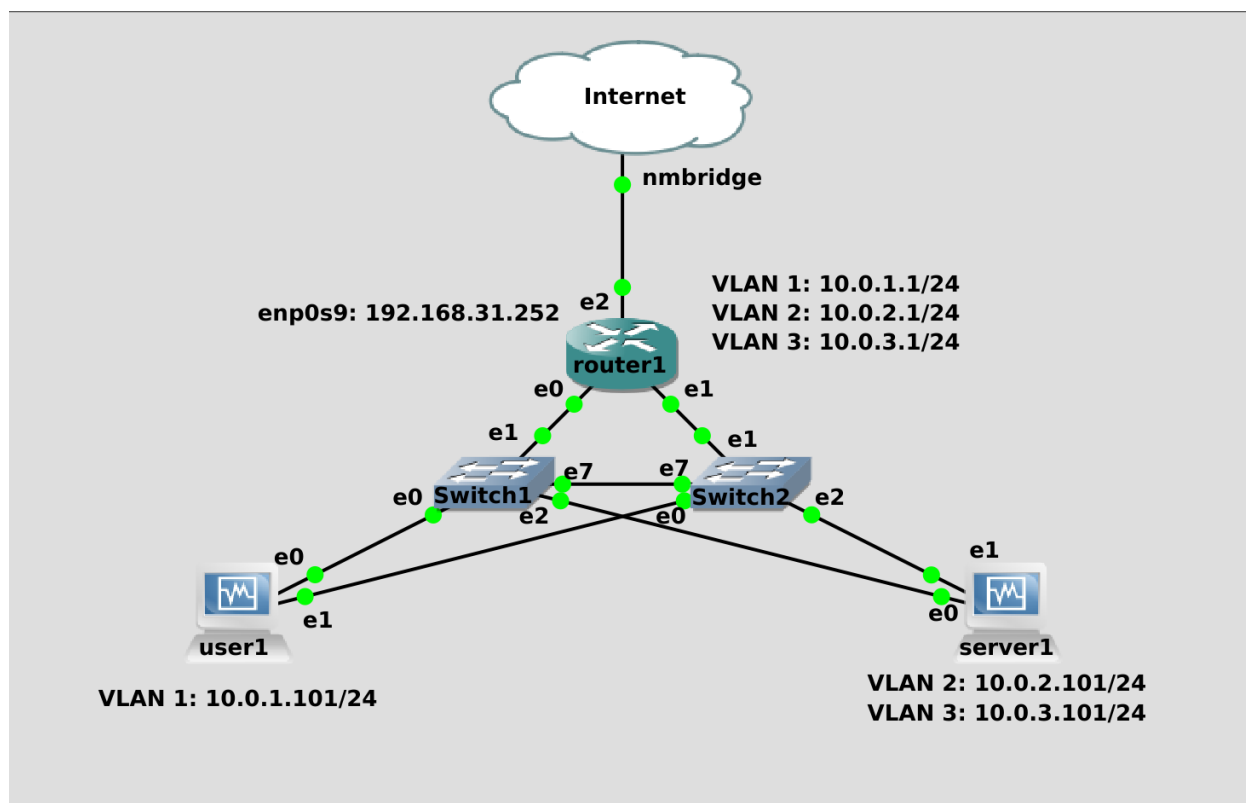
Теперь мы можем добавить ключи в агент с помощью команды `ssh-add`:

```
ssh-add .ssh/jumphost
ssh-add .ssh/servers
```

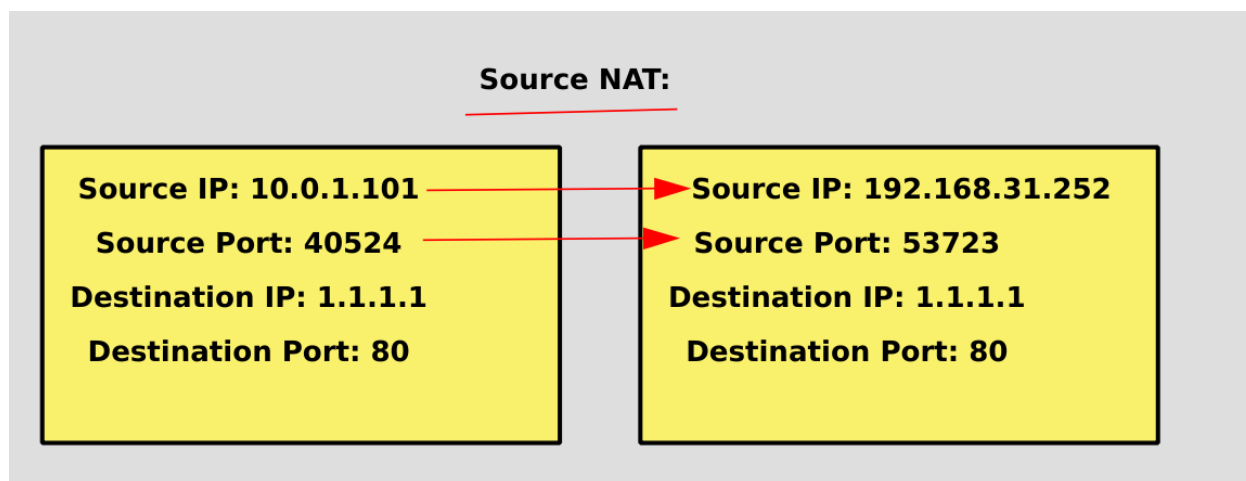
После чего, при попытке логина на сервер или джампхост, никакого пароля не потребуется:

```
ssh server1
ssh router1
```

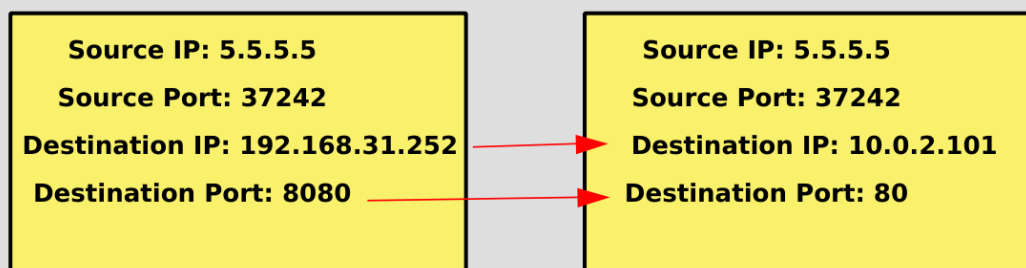
И, с одной стороны, у нас ключи защищены паролем, а с другой - теперь это также удобно, как без пароля, разве что перед работой надо прописать пару команд.



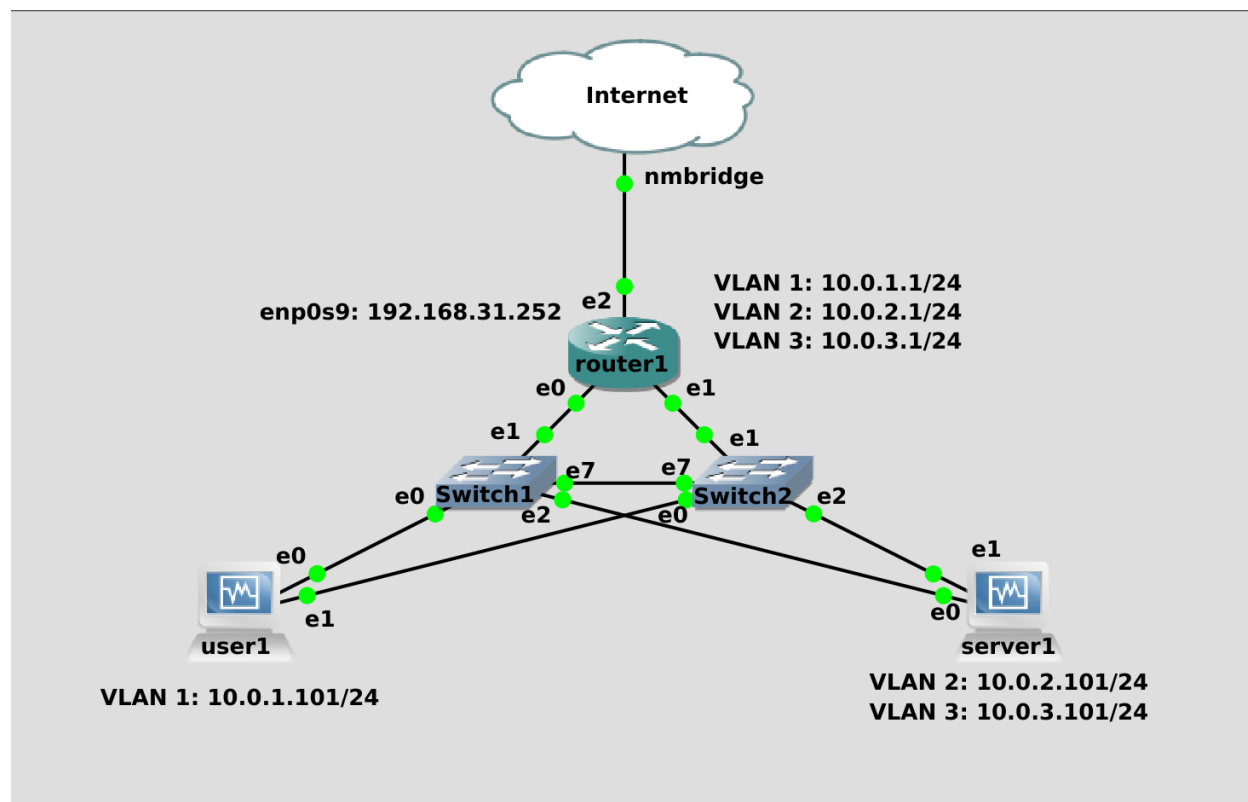
Давайте постараемся сделать нашу схему чуть реальнее. Обычно в компаниях на роутере не стоит линукс, либо там такой урезанный линукс, что мало что сделаешь. Поэтому сам роутер использовать в качестве джамп хоста не получится. Пусть лучше им будет сервер1. Но как, спросите вы, он же недоступен из интернета? А это дело поправимое. Когда у нас какой-то хост находится в изолированной сети и у нас нет к нему прямого доступа, мы всё же можем к нему попасть используя NAT.



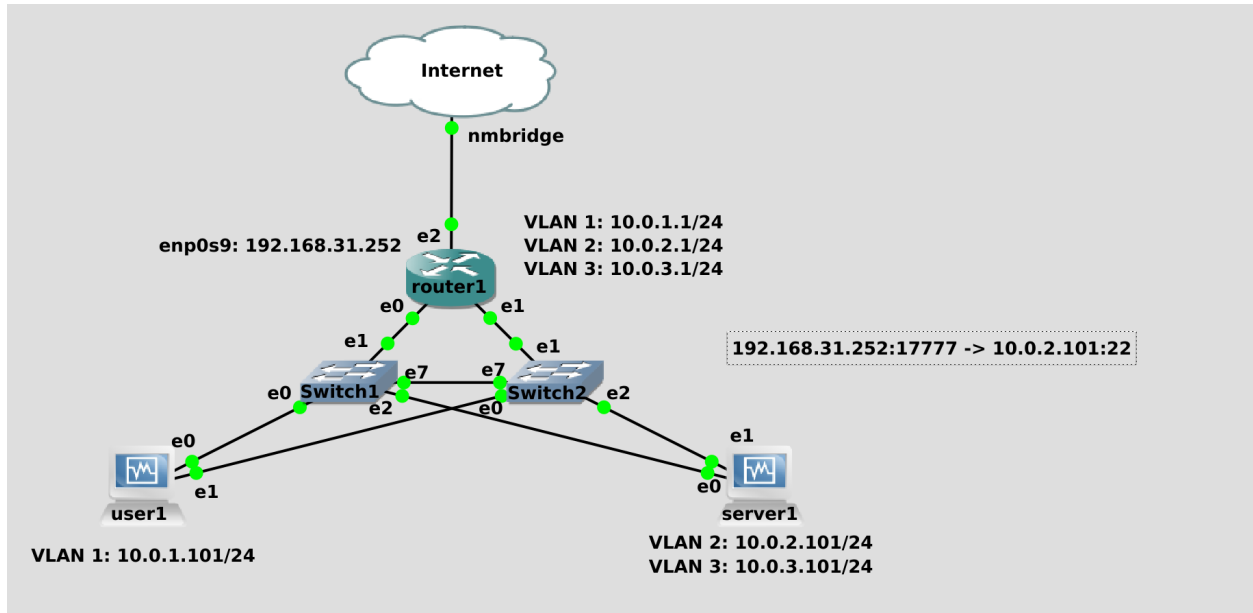
В прошлый раз мы разбирали source nat, когда у наших пакетов на выходе из сети подменялся Source IP и Source Port, чтобы пакеты могли вернуться обратно в сеть с приватными адресами.

Destination NAT:

Сейчас же нам нужно, чтобы подменялись destination адреса, так как тот, кто хочет подключиться, не знает или не может подключиться к внутренним адресам, а нам нужно перенаправлять пакеты на внутренний адрес. Поэтому с помощью NAT-а мы можем заменить destination адрес и порт. Это называется dnat или проброс портов.



И так, мы хотим, чтобы указывая адрес роутера1, мы подключались по ssh к server1. При этом, из-за того, что наш роутер, условно, находится в интернете, 22 порт использовать не стоит - в интернете куча ботов, которые будут пытаться взломать наш сервер. Поэтому выберем какой-нибудь нестандартный порт для ssh.



Давайте перенаправлять всё, что приходит на внешний адрес роутера на порт 17777 на адрес сервера 1 по порту 22.

```
[doctor@tardis]~$ ssh router1
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Sat Dec 11 12:46:52 2021 from 192.168.31.227
[root@router1 ~]# firewall-cmd --list-all --zone=external
external (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp0s9
  sources:
  services: ssh
  ports:
  protocols:
  forward: no
  masquerade: yes
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
[root@router1 ~]#
```

Для этого заходим на роутер:

```
ssh router1
```

Интерфейс с внешним адресом у нас находится в зоне external на файрволе:

```
firewall-cmd --list-all --zone=external
```

Здесь мы и должны пробросить порт.

```
[root@router1 ~]# firewall-cmd --zone=external --add-forward-port=port=17777:proto=tcp:toaddr=10.0.2.101:toport=22 --permanent
success
[root@router1 ~]# firewall-cmd --reload
success
[root@router1 ~]# firewall-cmd --list-all --zone=external
external (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp0s9
  sources:
  services: ssh
  ports:
  protocols:
  forward: no
  masquerade: yes
  forward-ports:
    port=17777:proto=tcp:toport=22:toaddr=10.0.2.101
  source-ports:
  icmp-blocks:
  rich rules:
[root@router1 ~]#
```

Команда получится чуть длинная, но ничего сложного в ней нет:

```
firewall-cmd --zone=external --add-forward-port=port=17777:proto=tcp:toaddr=10.0.2.101:toport=22 --permanent
```

Для начала мы указываем зону, в которой хотим добавить правило. Так как интерфейс находится в зоне external, её и указываем. Дальше используем ключ `--add-forward-port` чтобы добавить правило для проброса порта. После равно пишем, какой порт мы хотим пробросить - `port=17777`, какой протокол - `proto=tcp`, на какой адрес - `toaddr=10.0.2.101` и на какой порт - `toport=22`. Все эти значения разделяются двоеточиями. Ну и в конце не забываем указать `--permanent`, чтобы наши изменения сохранились в настройках.

После чего перегружаем правила:

```
firewall-cmd --reload
```

И проверяем результат:

```
firewall-cmd --list-all --zone=external
```

Как видите, в зоне external появилось правило под `forward-ports`.

```

GNU nano 5.9 /home/doctor/.ssh/config
Host router1
  Hostname 192.168.31.252
  User root
  IdentityFile ~/.ssh/jumphost

Host server1
# Hostname 10.0.2.101
  User root
# ProxyJump router1
  IdentityFile ~/.ssh/servers
  Hostname 192.168.31.252
  Port 17777

```

Так как теперь наш сервер1 доступен напрямую из интернета, то в настройках ssh - ~/.ssh/config - убираем строчку ProxyJump, а вместо неё прописываем внешний адрес роутера и порт:

```

...
Hostname 192.168.31.252
Port 17777
...

```

```

❗ [doctor@tardis]~$ nano ~/.ssh/config
[doctor@tardis]~$ ssh server1
The authenticity of host '[192.168.31.252]:17777 ([192.168.31.252]:17777)' can't be established.
ED25519 key fingerprint is SHA256:pneyKLqUSNb1MALbU0Th97JKZ296yui7WD6lCim4sgc.
This host key is known by the following other names/addresses:
  ~/.ssh/known_hosts:201: 10.0.2.101
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[192.168.31.252]:17777' (ED25519) to the list of known hosts.
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Sat Dec 11 13:37:04 2021
[root@server1 ~]#

```

При первом подключении:

```
ssh server1
```

мы увидим сообщение, что наш компьютер узнал этот хост и раньше у него был адрес 10.0.2.101, сейчас же мы подключаемся к нему по другому адресу. Пишем yes и подключаемся без проблем.

```
[root@router1 ~]# firewall-cmd --list-all --zone=external
external (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp0s9
  sources:
  services: ssh
  ports:
  protocols:
  forward: no
  masquerade: yes
  forward-ports:
    port=17777:proto=tcp:toport=22:toaddr=10.0.2.101
  source-ports:
  icmp-blocks:
  rich rules:
[root@router1 ~]# firewall-cmd --remove-service=ssh --zone=external --permanent
success
[root@router1 ~]# firewall-cmd --reload
success
[root@router1 ~]#
```

Теперь поднастроим роутер. Сейчас, условно, любой желающий может к нему подключиться по ssh:

```
firewall-cmd --list-all --zone=external
```

Для большей безопасности убираем эту возможность:

```
firewall-cmd --remove-service=ssh --zone=external --permanent
firewall-cmd --reload
```

```
GNU nano 5.9 /home/doctor/.ssh/config
Host router1
# Hostname 192.168.31.252
User root
IdentityFile ~/.ssh/jumphost
ProxyJump server1
Hostname 10.0.2.1
Host server1
# Hostname 10.0.2.101
User root
# ProxyJump router1
IdentityFile ~/.ssh/servers
Hostname 192.168.31.252
Port 17777
```

Теперь у нас сложилась такая ситуация, что напрямую на роутер мы попасть не сможем, зато можем через server1. Т.е. теперь в `~/.ssh/config` указываем, что для подключения к роутеру нужно использовать server1 в качестве JumpHost-а. Ну и пишем внутренний адрес роутера:


```
...
ProxyJump server1
Hostname 10.0.2.1
...
```

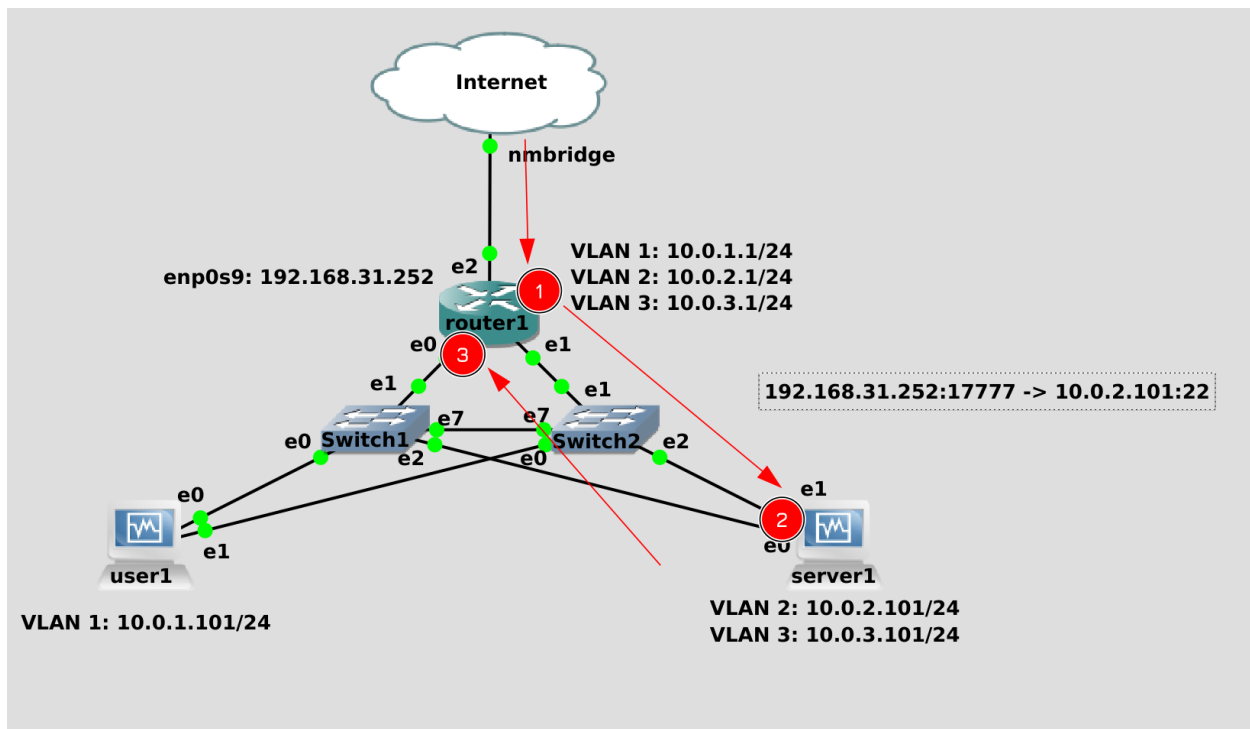
```
[doctor@tardis]~$ nano ~/.ssh/config
[doctor@tardis]~$ ssh router1
The authenticity of host '10.0.2.1 (<no hostip for proxy command>)' can't be established.
ED25519 key fingerprint is SHA256:aHIqhQPsr0cnbISJGZz7KugA7a7/0IvpMa0E8suhn7I.
This host key is known by the following other names/addresses:
  ~/.ssh/known_hosts:198: 192.168.31.252
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.2.1' (ED25519) to the list of known hosts.
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Sat Dec 11 13:42:42 2021 from 192.168.31.227
[root@router1 ~]#
```

Проверяем подключение:

```
ssh router1
```

всё работает.



И наше соединение к роутеру теперь выглядит так. Сначала мы подключаемся к внешнему адресу роутера по порту 17777, откуда файрвол перекидывает наши пакеты на сервер1 на порт 22. Там мы логинимся по ssh и сразу же подключаемся оттуда по ssh к внутреннему адресу роутера. И всё это одной командой:

```
ssh router1
```

У вас может возникнуть вопрос - почему бы просто не пробрасывать порты на все сервера? Зачем нужен jump host? Тут несколько причин:

- во-первых, безопасность. Когда у вас jump host, то снаружи по ssh доступен только один хост. А это значит, что все попытки взлома будут направлены только на него. Пока его не взломают, все остальные хосты будут невидимы. По сути, jump host это ваша дверь в сеть. Легче защитить одну дверь, чем множество.
- во-вторых, удобство. Когда у вас 3 сервера - можно и пробросом портов обойтись. А что, если у вас 300 серверов? Что, если они постоянно добавляются и удаляются? Что, сидеть и постоянно пробрасывать порты? А так, с одного jump-хоста можно попасть на все другие машинки.

И так, как сделать наше подключение ещё безопаснее?

```
[doctor@tardis]~$ ssh-copy-id -i .ssh/jumphost.pub user@server1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: ".ssh/jumphost.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are
already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to
install the new keys
user@192.168.31.252's password:

Number of key(s) added: 1

Now try logging into the machine, with:  "ssh 'user@server1'"
and check to make sure that only the key(s) you wanted were added.

[doctor@tardis]~$ ssh-copy-id -i .ssh/servers.pub user@router1
```

Ну, для начала, перестать использовать рута. Давайте вместо рута логиниться с помощью юзера. А для этого ключи нужно закинуть на пользователя:

```
ssh-copy-id -i .ssh/jumphost.pub user@server1
ssh-copy-id -i .ssh/servers.pub user@router1
```

Так как теперь у нас джамп хостом является server1, то и ключ кидаем соответствующий. А для роутера кидаем второй ключ. Потом нужно будет подтереть старые ключи у рута.

```
GNU nano 5.9 /home/doctor/.ssh/config
Host router1
  User user
  IdentityFile ~/.ssh/servers
  ProxyJump server1
  Hostname 10.0.2.1

Host server1
  User user
  IdentityFile ~/.ssh/jumphost
  Hostname 192.168.31.252
  Port 1777
```

Подправляем ssh config файл - ~/.ssh/config. Во-первых, меняем пользователя на user-а, чтобы не приходилось при каждом подключении указывать его. Во-вторых - меняем ключи.

```
...
User user
...
```

```
[doctor@tardis]-[~]
$ ssh server1
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Sat Dec 11 14:09:52 2021 from 192.168.31.227
[user@server1 ~]$ sudo nano /etc/ssh/sshd_config
```

После того, как убедимся, что подключение работает:

```
ssh server1
```

лучше всего кое-что поправить на ssh-сервере, который доступен из интернета:

```
sudo nano /etc/ssh/sshd_config
```

```
GNU nano 2.9.8 /etc/ssh/sshd_config

# Logging
#SyslogFacility AUTH
SyslogFacility AUTHPRIV
#LogLevel INFO

# Authentication:

#LoginGraceTime 2m
PermitRootLogin no
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10
```

Находим строчку PermitRootLogin и меняем значение на no:

```
...
PermitRootLogin no
...
```

Эта опция позволяет подключаться по ssh пользователем root. А так как все знают, что на любой линукс системе есть пользователь рут, то и большинство ботов будут пытаться ломиться этим пользователем. Поэтому запретив логиниться пользователем root - мы немного усложняем работу ботам, так как теперь им придётся угадывать логин пользователя, а не только пароль.

```

GNU nano 2.9.8 /etc/ssh/sshd_config

#AuthorizedKeysCommand none
#AuthorizedKeysCommandUser nobody

# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
#PasswordAuthentication yes
#PermitEmptyPasswords no
PasswordAuthentication no

# Change to no to disable s/key passwords
#ChallengeResponseAuthentication yes
ChallengeResponseAuthentication no

```

А дальше находим строчку PasswordAuthentication и меняем на no:

```

...
PasswordAuthentication no
...

```

Тем самым запрещая логин по паролю. Теперь единственным способом залогиниться будет использование ssh ключей, что отсеивает всех ботов, потому что вряд ли они могут достать наши ssh ключи. Хотя, теоретически, какие-то боты могут использовать базу из каких-то слитых приватных ключей, но это прям редкость. Этих настроек вполне хватает для большинства случаев, поэтому сохраняем конфиг и выходим.

```

[user@server1 ~]$ sudo systemctl restart sshd
[user@server1 ~]$ logout
Connection to 192.168.31.252 closed.
[doctor@tardis]~$
[doctor@tardis]~$ ssh root@server1
root@192.168.31.252: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
[doctor@tardis]~$ ssh server1
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Sat Dec 11 14:18:16 2021 from 192.168.31.227
[user@server1 ~]$

```

Чтобы применить настройки перезапускаем sshd:

```
sudo systemctl restart sshd
```

Для теста попытаемся залогиниться рутом:

```
ssh root@server1
```

Как видите, сразу получаем ошибку - permission denied.

А если попытаться зайти обычным user-ом:

```
ssh server1
```

всё получится.

Хотя использование нестандартного порта, запрет логина руту и запрет логина по паролям почти решают проблему с ботами, я не могу не рассказать про ещё один механизм - fail2ban. В нашей ситуации он не особо нужен, но, если вы не можете по каким-то причинам поменять стандартный порт, не можете запретить логиниться по паролям - то fail2ban вам пригодится.

По сути, большинство ботов просто сканирует публичные адреса по 22 и по другим частоиспользуемым портам и пытаются подобрать пароль. Зачастую пароль рута, но иногда и какие-то другие стандартные логины. Этот тип атаки называется brute-force. Лёгкие пароли могут подобрать за пару минут, ну а сложные - как повезёт. Ну и один из вариантов хоть как-то сдерживать это - блокировать адреса, которые часто вводят неправильный пароль. Для этого собственно и нужен fail2ban. Он работает не только с ssh, но и с другими сервисами.

```
[user@server1 ~]$ sudo dnf install epel-release
Last metadata expiration check: 0:00:08 ago on Sat 11 Dec 2021 02:48:28 PM +04.
Dependencies resolved.
=====
Package                        Architecture      Version           Repository        Size
=====
Installing:
epel-release                   noarch            8-10.el8          extras            22 k
=====
Transaction Summary
=====
Install 1 Package

Total download size: 22 k
Installed size: 32 k
Is this ok [y/N]: y
```

Чтобы установить fail2ban, для начала надо подключить репозиторий epel:

```
sudo dnf install epel-release
```

```
[user@server1 ~]$ sudo dnf install fail2ban
Last metadata expiration check: 0:00:10 ago on Sat 11 Dec 2021 02:54:27 PM +04.
Dependencies resolved.
=====
Package                        Arch             Version           Repository        Size
=====
Installing:
fail2ban                       noarch           0.11.2-1.el8      epel              19 k
Installing dependencies:
esmtp                          x86_64           1.2-15.el8        epel              57 k
fail2ban-firewalld             noarch           0.11.2-1.el8      epel              19 k
fail2ban-sendmail              noarch           0.11.2-1.el8      epel              22 k
fail2ban-server                noarch           0.11.2-1.el8      epel              459 k
libesmtp                       x86_64           1.0.6-18.el8      epel              70 k
liblockfile                    x86_64           1.14-1.el8         appstream         31 k
python3-pip                    noarch           9.0.3-20.el8       appstream         19 k
python36                       x86_64           3.6.8-38.module_el8.5.0+2569+5c5719bc appstream         18 k
Enabling module streams:
python36                       3.6
=====
Transaction Summary
=====
Install 9 Packages
```

А затем установить пакет с названием fail2ban:

```
sudo dnf install fail2ban -y
```

```
[user@server1 ~]$ sudo systemctl enable --now fail2ban
Created symlink /etc/systemd/system/multi-user.target.wants/fail2ban.service → /usr/lib/systemd/system/fail2ban.service.
[user@server1 ~]$ sudo fail2ban-client status
Status
|- Number of jail:      0
`- Jail list:
[user@server1 ~]$
```

После установки давайте стартанём сервис и добавим его в автозагрузку:

```
sudo systemctl enable --now fail2ban
```

fail2ban очень гибкая штука, которая позволяет защитить от брутфорса почти всё что угодно. Для некоторых сервисов, таких как ssh, есть готовые шаблоны, а для каких-то самописных программ можно настроить самостоятельно. Дефолтные настройки fail2ban могут отличаться в зависимости от дистрибутива. Давайте проверим, защищает ли fail2ban сейчас что-то? Многие вещи делаются через утилиту fail2ban-client:

```
sudo fail2ban-client status
```

Сейчас мы видим, что fail2ban ничего не защищает.

```
[user@server1 ~]$ sudo nano /etc/fail2ban/
action.d/      fail2ban.d/      jail.conf      paths-common.conf
fail2ban.conf  filter.d/        jail.d/        paths-fedora.conf
[user@server1 ~]$ sudo nano /etc/fail2ban/jail.local
```

А значит нам надо создать файл /etc/fail2ban/jail.local.

```
sudo nano /etc/fail2ban/jail.local
```

Настройки различных сервисов есть в соседнем файле jail.conf. В jail.local мы лишь перезаписываем эти настройки, а также включаем их.

```
GNU nano 2.9.8 /etc/fail2ban/jail.local

[sshd]

enabled = true
```

Например, чтобы включить настройки sshd, добавим здесь пару строчек:

```
[sshd]

enabled = true
```

```
[user@server1 ~]$ sudo nano /etc/fail2ban/
action.d/          fail2ban.d/          jail.conf          paths-common.conf
fail2ban.conf      filter.d/            jail.d/            paths-fedora.conf
[user@server1 ~]$ sudo nano /etc/fail2ban/jail.local
[user@server1 ~]$ sudo systemctl restart fail2ban
[user@server1 ~]$ sudo fail2ban-client status
Status
|- Number of jail:      1
`- Jail list:  sshd
[user@server1 ~]$ sudo fail2ban-client status sshd
Status for the jail: sshd
|- Filter
|  |- Currently failed: 0
|  |- Total failed:     0
|  `-- Journal matches: _SYSTEMD_UNIT=sshd.service + _COMM=sshd
`- Actions
   |- Currently banned: 0
   |- Total banned:     0
   `-- Banned IP list:
[user@server1 ~]$
```

После чего перезапустим сервис fail2ban:

```
sudo systemctl restart fail2ban
```

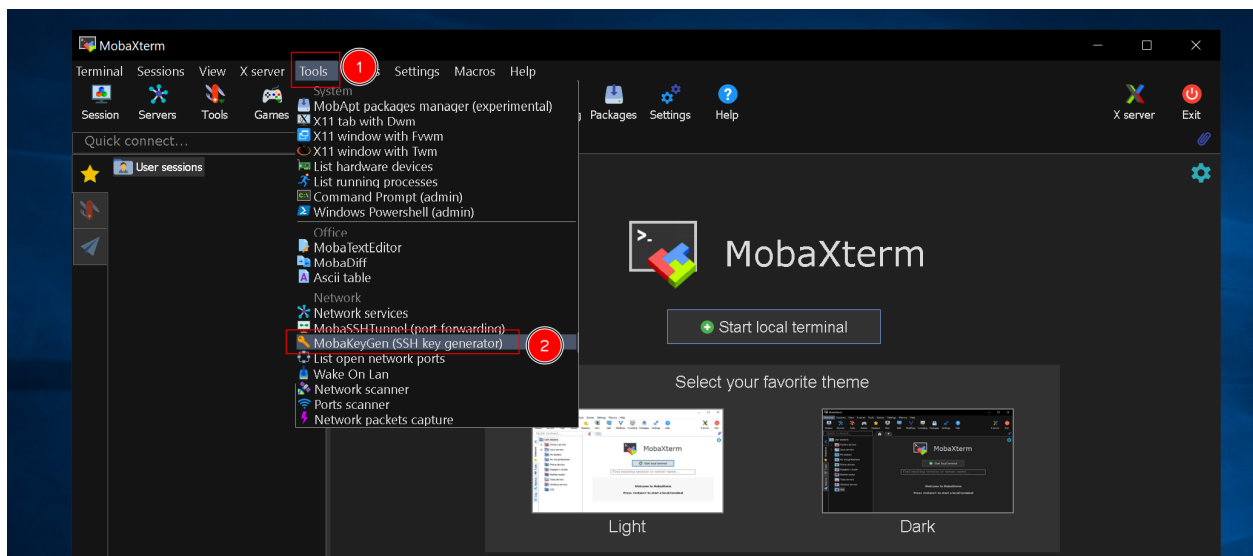
И ещё раз проверим статус через fail2ban-client:

```
sudo fail2ban-client status
sudo fail2ban-client status sshd
```

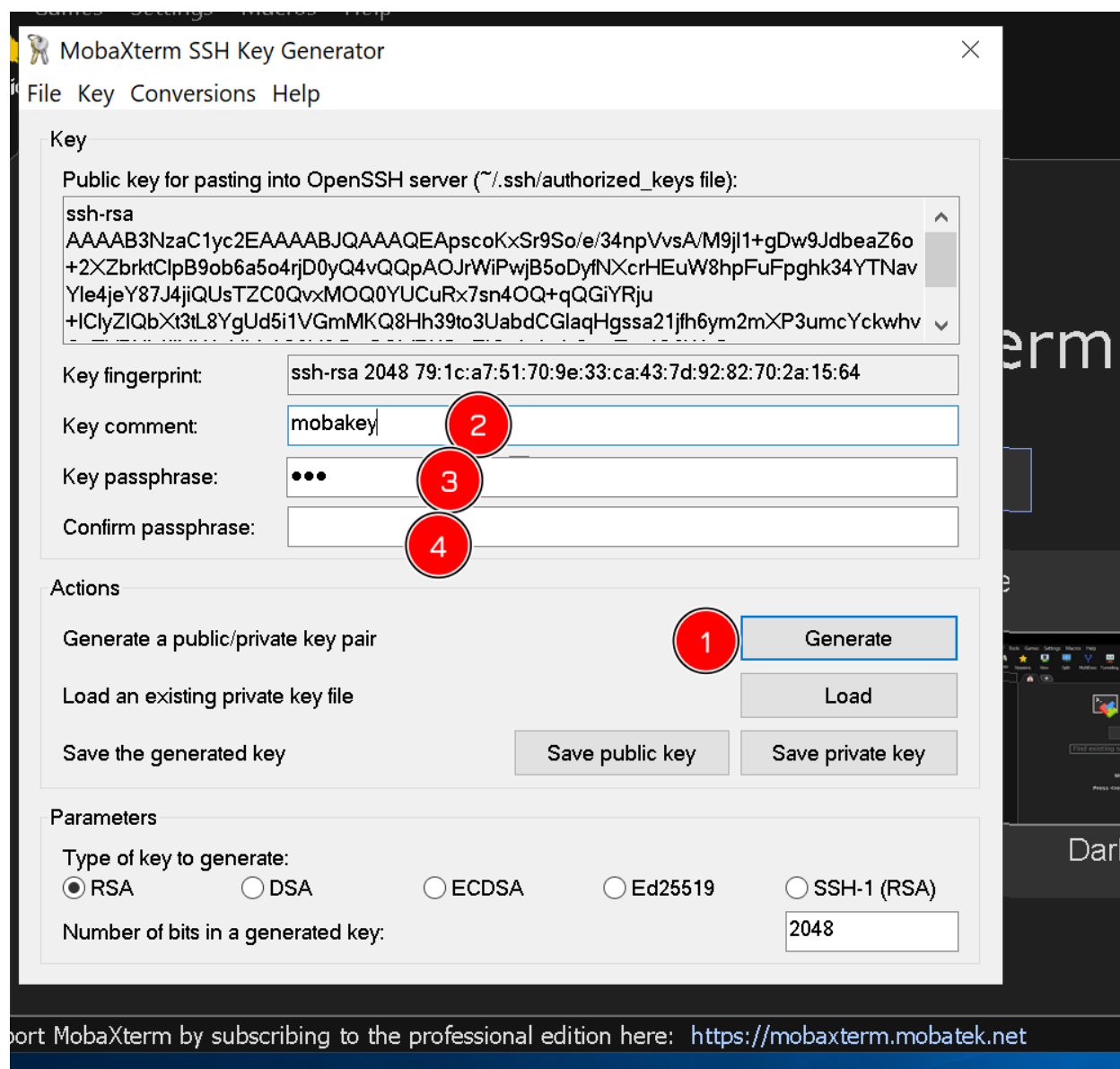
Как видите, теперь у нас есть 1 jail - sshd. И пока что ни одного забаненного IP.

На пока этого хватит. Настроек много, но детальнее с fail2ban поработаем как-нибудь в будущем. Если вам интересно - в интернете множество статей, да и в системе есть документация.

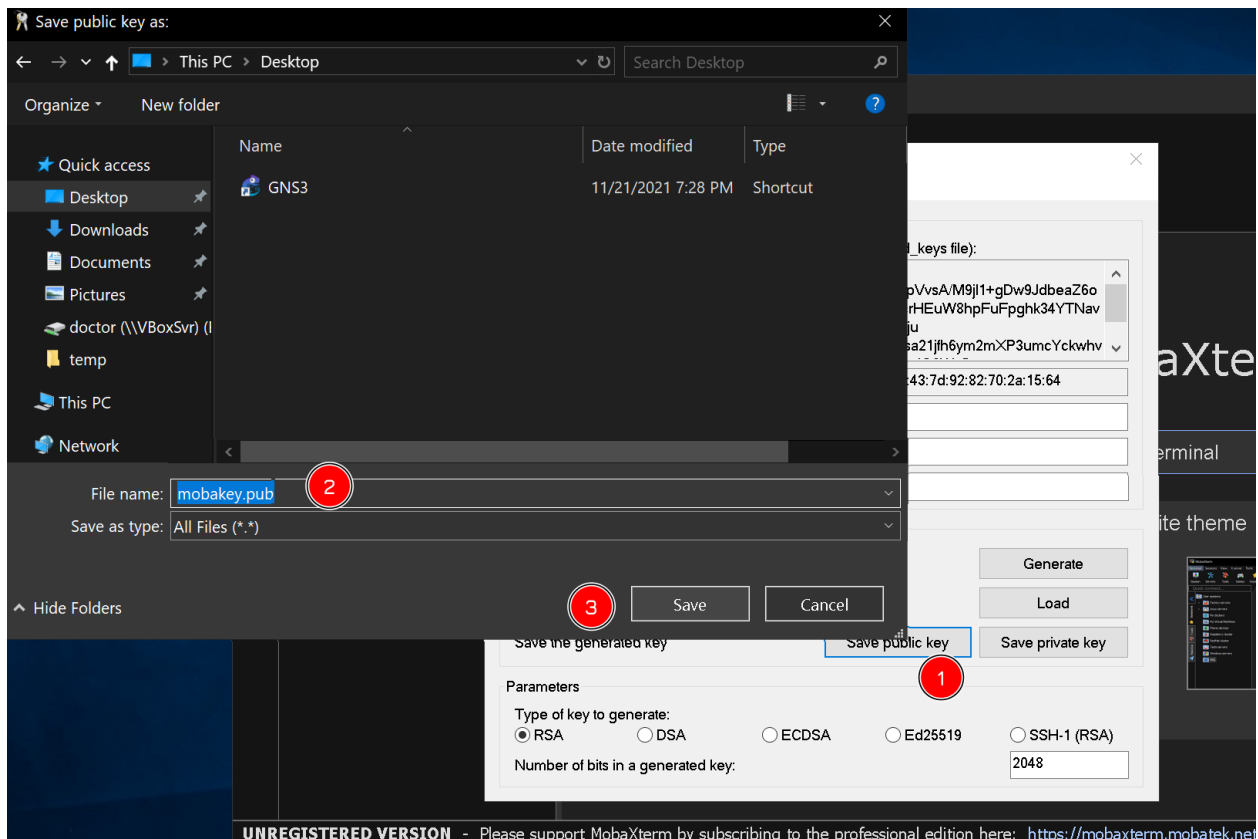
Напоследок покажу, как с Windows подключаться. Любой ssh клиент поддерживает работу с jump хостами и ключами, и mobaXterm - не исключение.



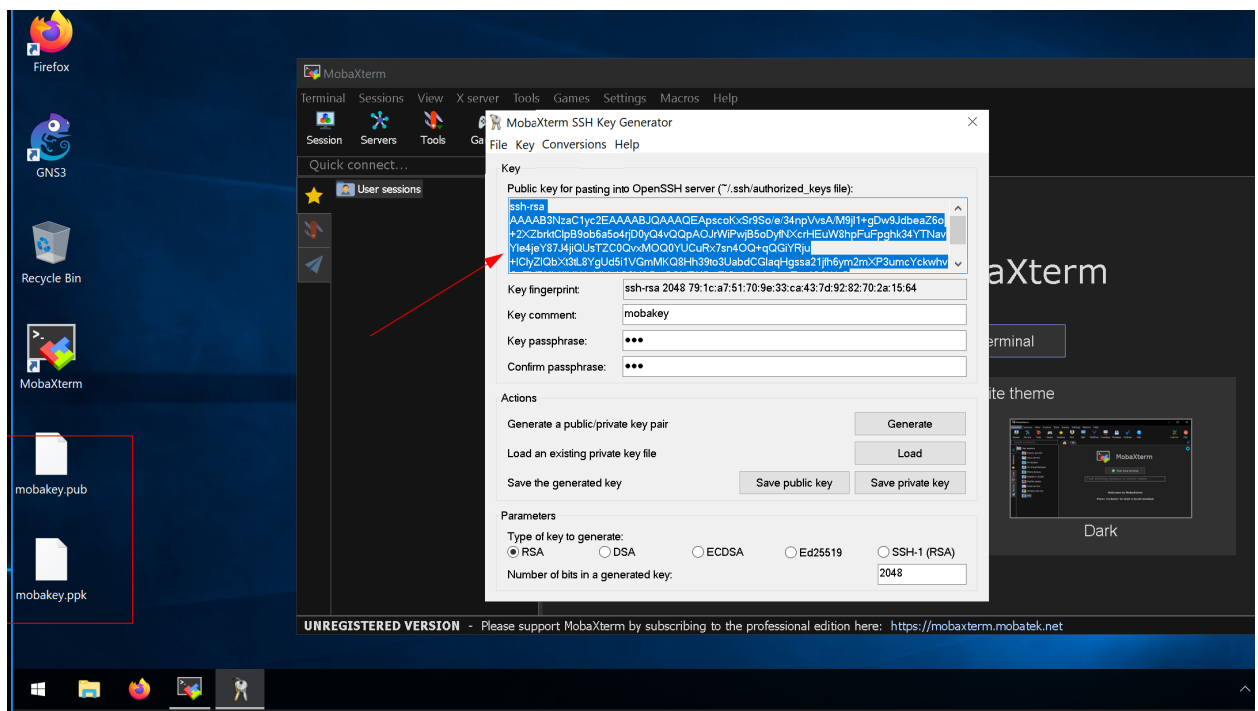
Для начала сгенерируем ключи для ssh. В интерфейсе выбираем Tools - MobaKeyGen.



В появившемся окне нажимаем Generate, немного водим мышкой, чтобы сгенерировать ключ, затем, опционально, пишем комментарий к ключу, задаём пароль и подтверждаем его.



Затем нажимаем Save public key, выбираем папку для сохранения и название файла. Также проделываем для приватного ключа.



Я сохранил на рабочем столе, вам же рекомендую держать ключи так, чтобы их не могли легко найти и украсть. Затем выделите и скопируйте публичный ключ из окна генератора.

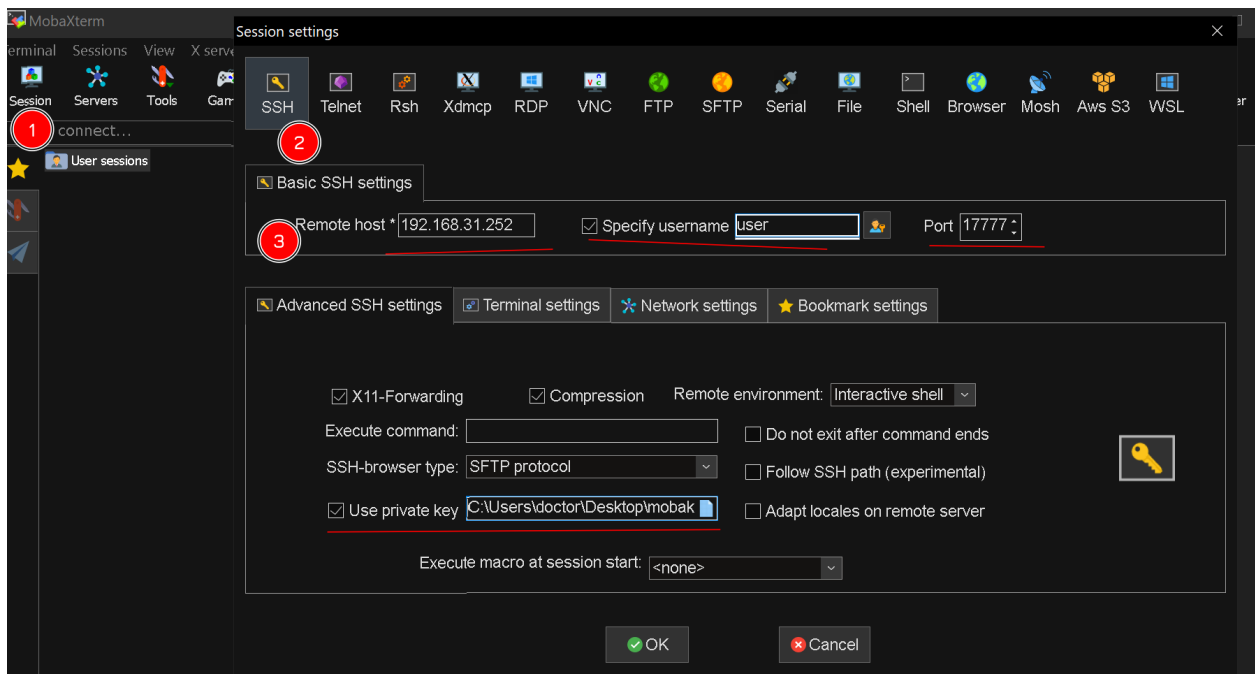
```
GNU nano 2.9.8 /home/user/.ssh/authorized_keys Mo
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGC+cT12A0Mn6kdQMwzRgGTouGBRA9vi+pCg9mW/SKQuolGdk81Mp
$LMnm4c/Xn8T/yJJUVFDCI+lqQ== mobakey
```

После чего вставьте ключ на сервере в домашней директории пользователя в файле `~/.ssh/authorized_keys` в новой строке. Причём убедитесь, чтобы один ключ был в одну строку. На первой строке у меня ключ, сгенерированный на Linux, а на второй - то что в mobaxterm. После чего сохраните файл и выйдите.

```
[user@server1 ~]$ nano ~/.ssh/authorized_keys
[user@server1 ~]$ ssh 10.0.2.1
The authenticity of host '10.0.2.1 (10.0.2.1)' can't be established.
ECDSA key fingerprint is SHA256:XpAz+BIujRqwgWDqlashv7wY6Dw+x4IDwzSmw4QWDKI.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.2.1' (ECDSA) to the list of known hosts.
user@10.0.2.1's password:
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Sat Dec 11 15:15:32 2021 from 10.0.2.101
[user@router1 ~]$ nano ~/.ssh/authorized_keys
[user@router1 ~]$
```

Таким же образом скопируйте ключ в файл `authorized_keys` на роутере. В идеале надо было сделать второй ключ, но я думаю вы с этим справитесь.

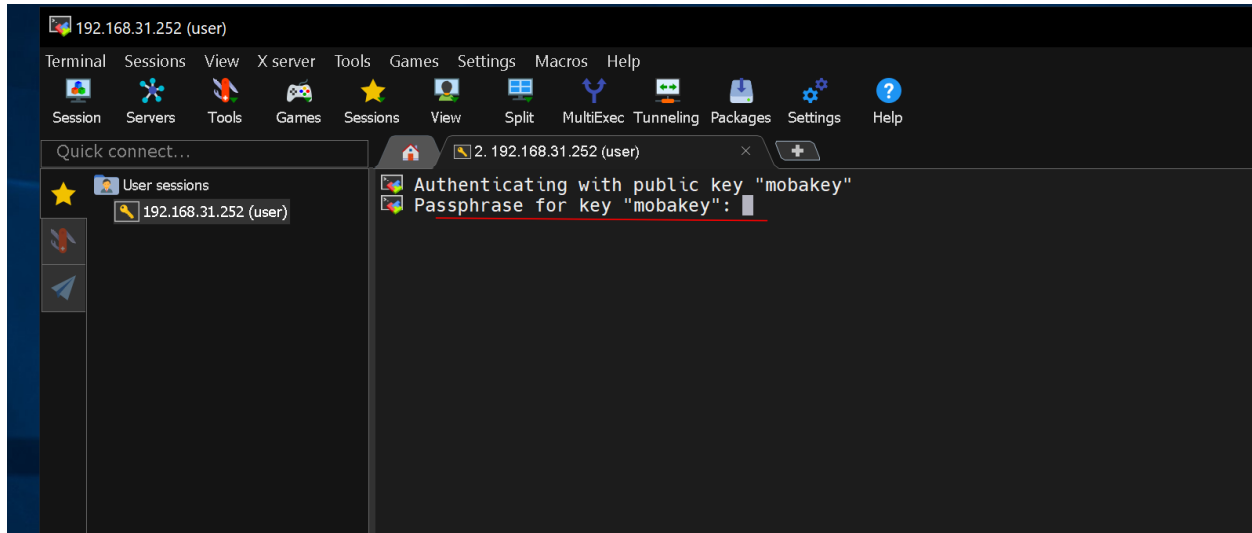


Дальше закрываем окно генератора, в mobaxterm выбираем Session - SSH и заполняем поля:

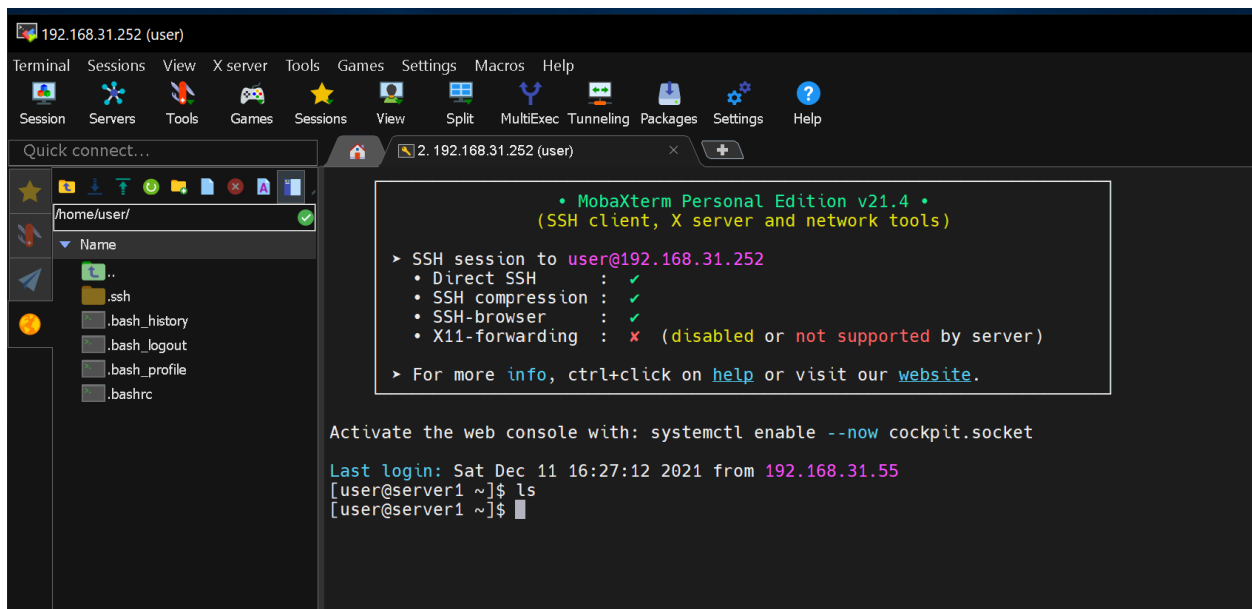
- remote host: внешний адрес роутера

- username: user
- port: 17777

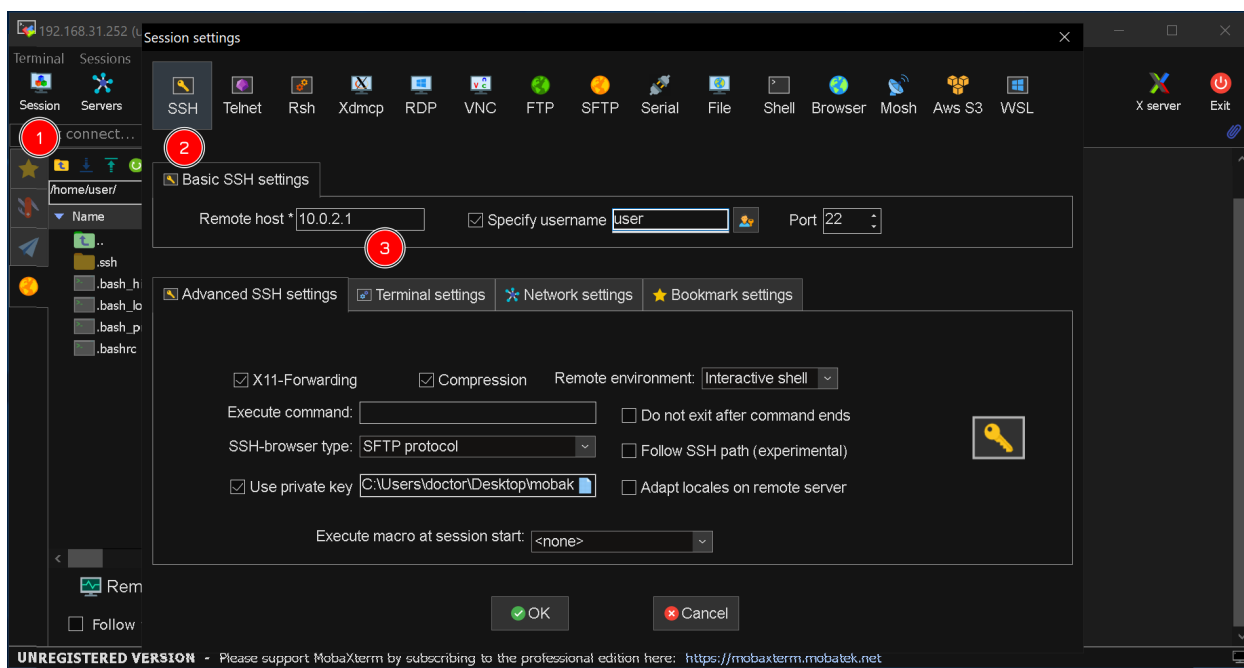
И внизу, во вкладке Advanced SSH settings ставим галочку на «Use private key» и выбираем приватный ключ, который мы сохранили до этого. После чего нажимаем ОК.



При подключении mobaxterm попросит пароль для ключа. Вводим его.

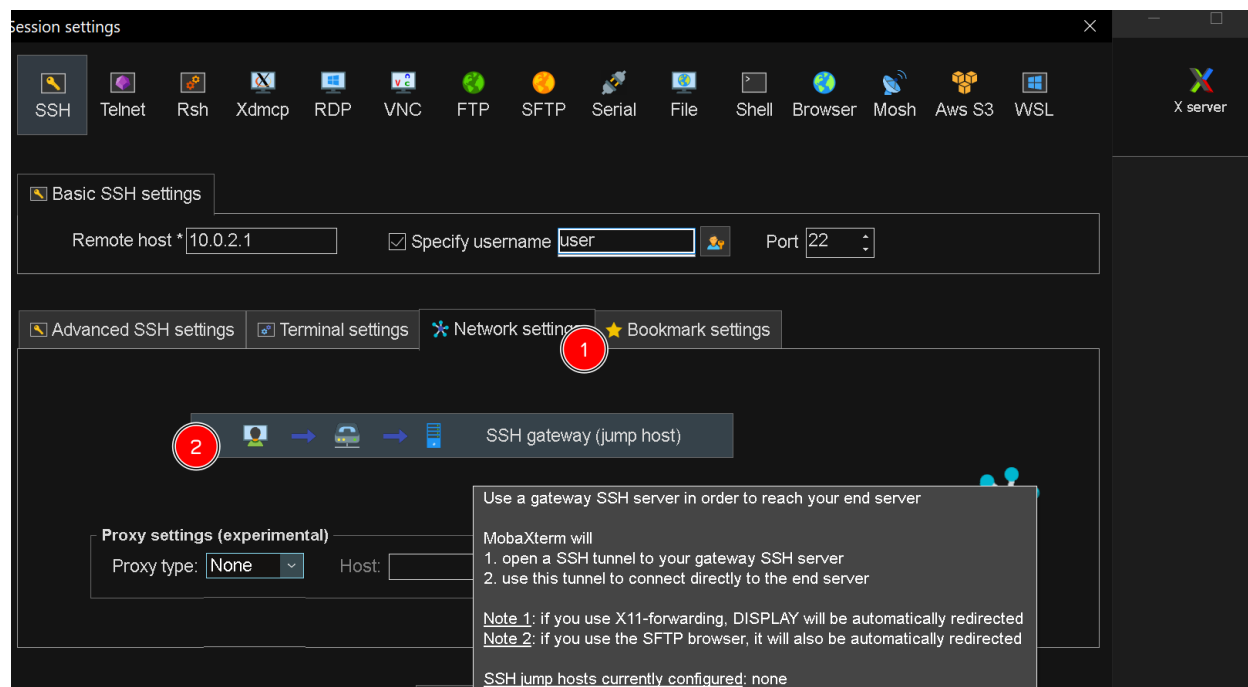


После чего вы попадёте на сервер.

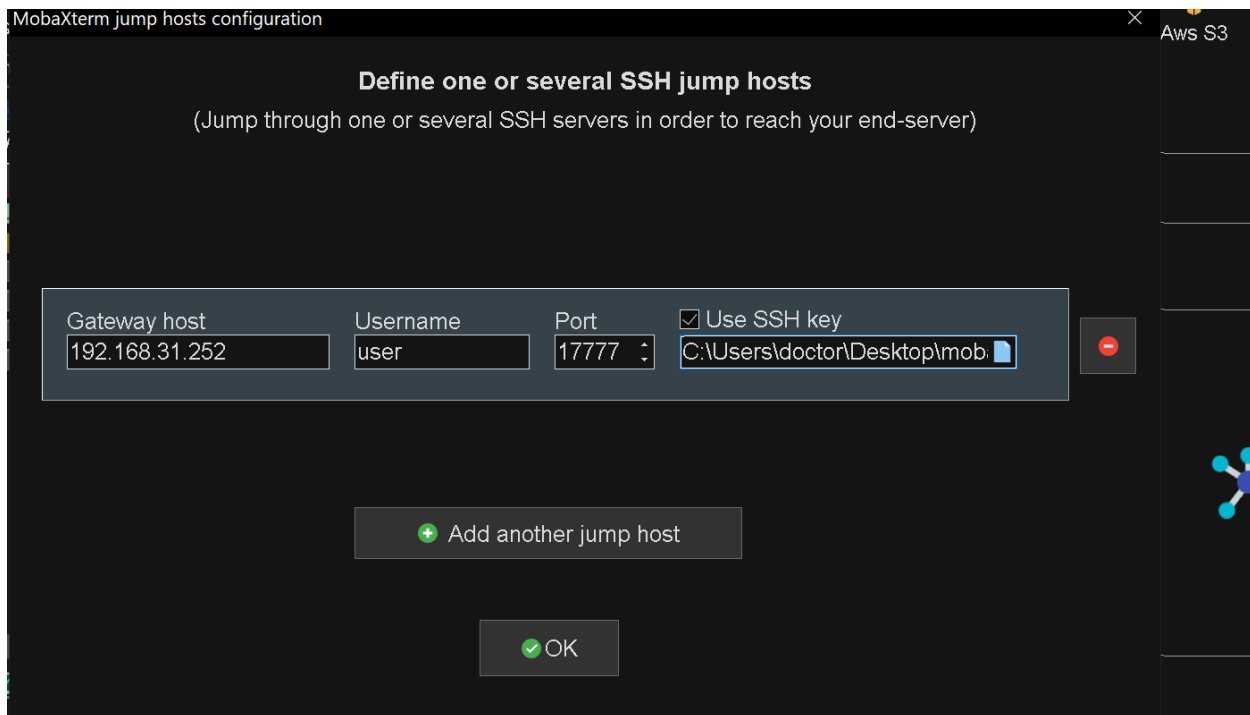


Теперь попробуем подключиться к роутеру, используя сервер1 как джампхост. Начинаем также - Session - SSH и заполняем поля:

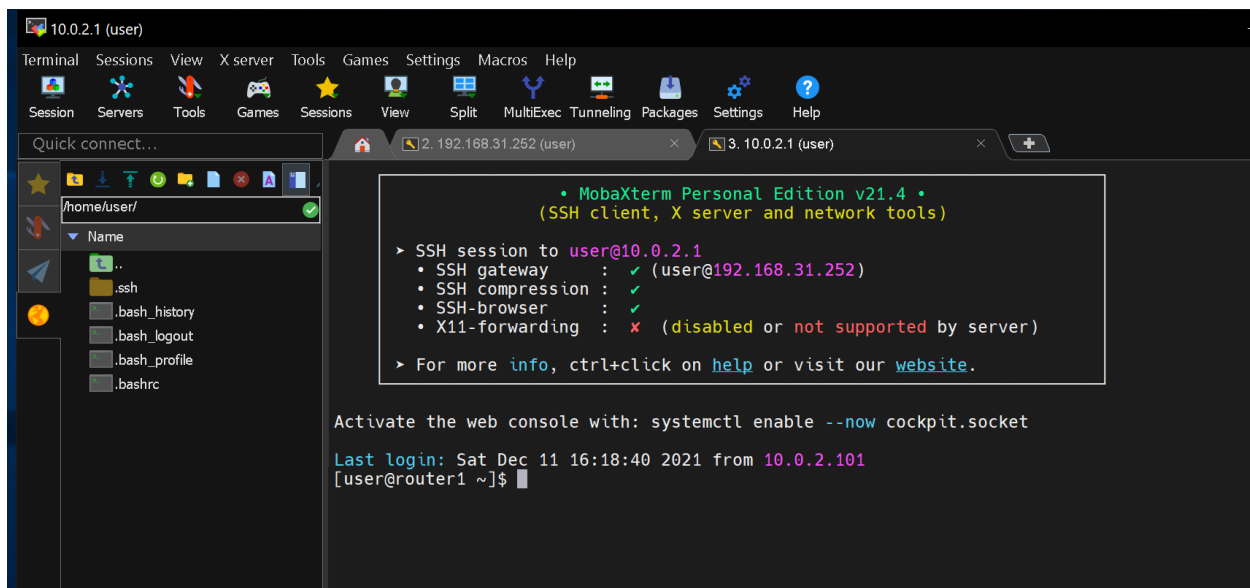
- remote host: внутренний адрес роутера
- username: user
- port: 22
- Advanced ssh settings - private key - выбираете приватный ключ.



Дальше открываете Network Settings и нажимаете на SSH Gateway.



В появившемся окне заполняете точно как заполняли сервер1, т.е. внешний адрес, порт 17777 и приватный ключ от сервера. Кстати, тут есть кнопка - add another jump host. Теоретически, ваши целевые сервера могут находиться за несколькими джамп хостами. Но суть та же. Нажимаем ОК, а потом ещё раз ОК.



И вы сразу попадёте на роутер. MobaXterm сохранил пароль от ключа и пока вы не закроете его, вам не нужно будет вводить пароли от ключей, как мы это делали с ssh агентом.

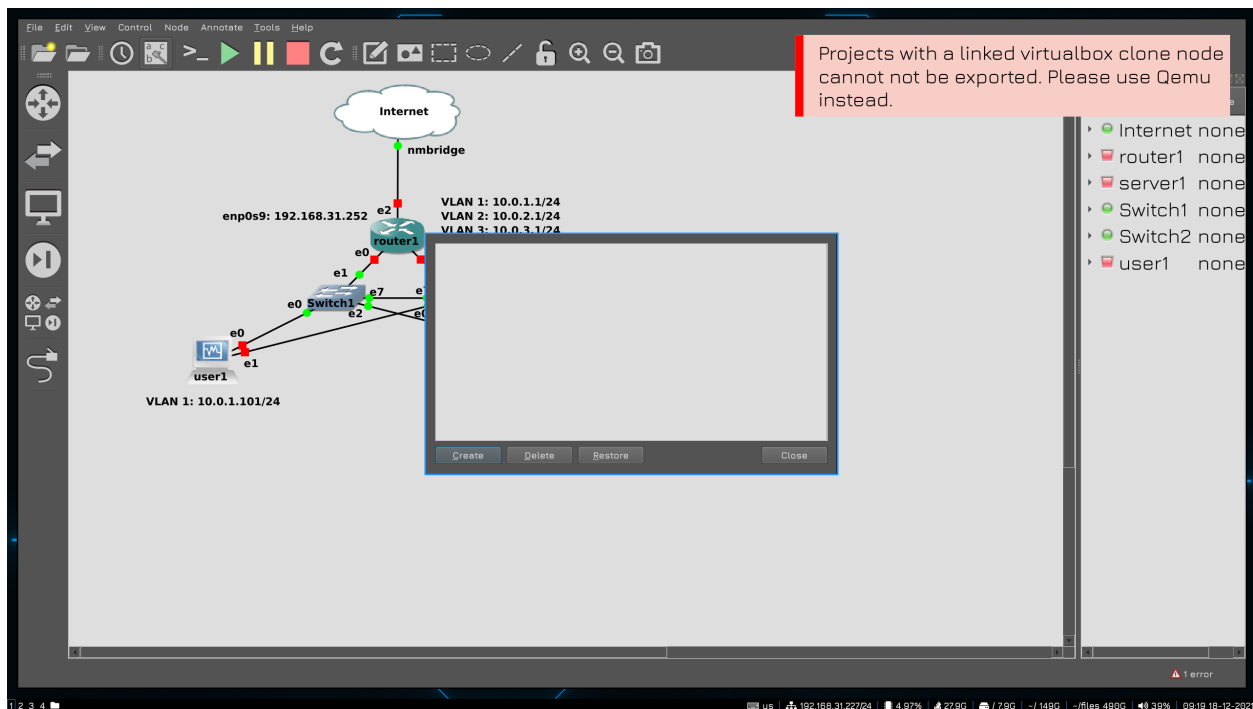
Небольшая задачка для вас - переименуйте server1 в jumphost, организуйте отдельные ключи для хоста user1, поднимите ещё один сервер, закиньте на него соответствующие ключи и подключитесь на все хосты через jumphost.

Давайте подведём итоги. Теперь у нас есть доступ в нашу инфраструктуру извне по ssh с помощью

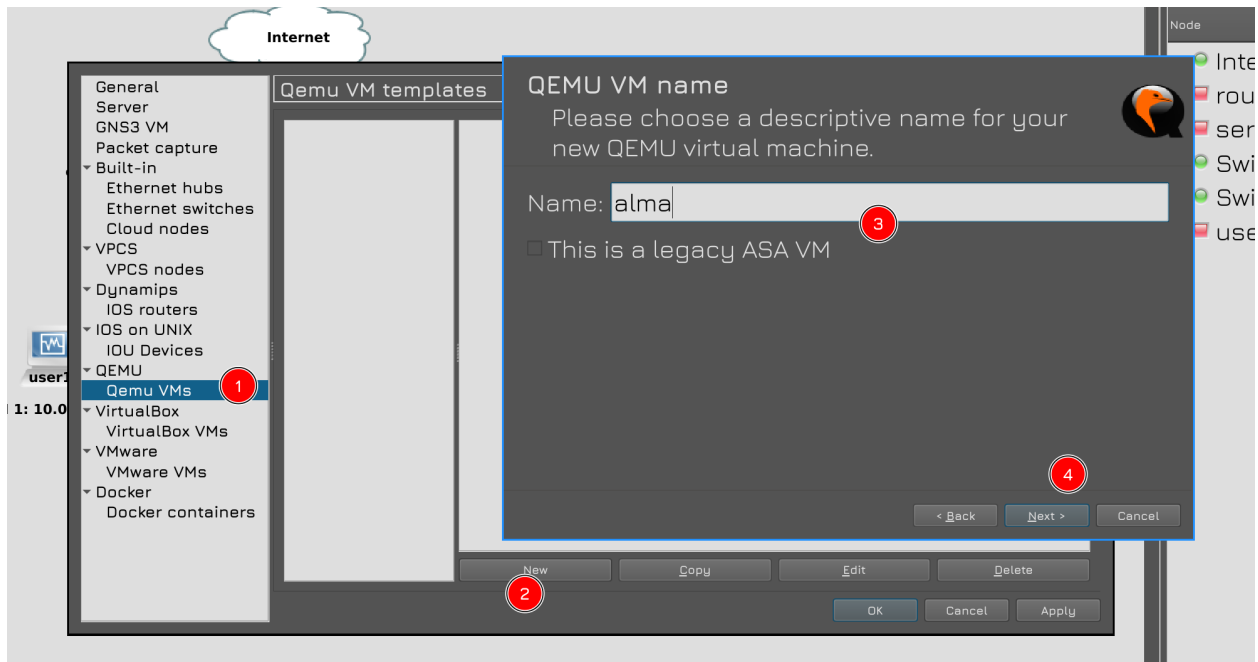
джамп хоста и проброса портов. Мы также разобрали как немного обезопасить наше подключение запретив логин по руту и без ключей.

2.7 07. Деление на зоны

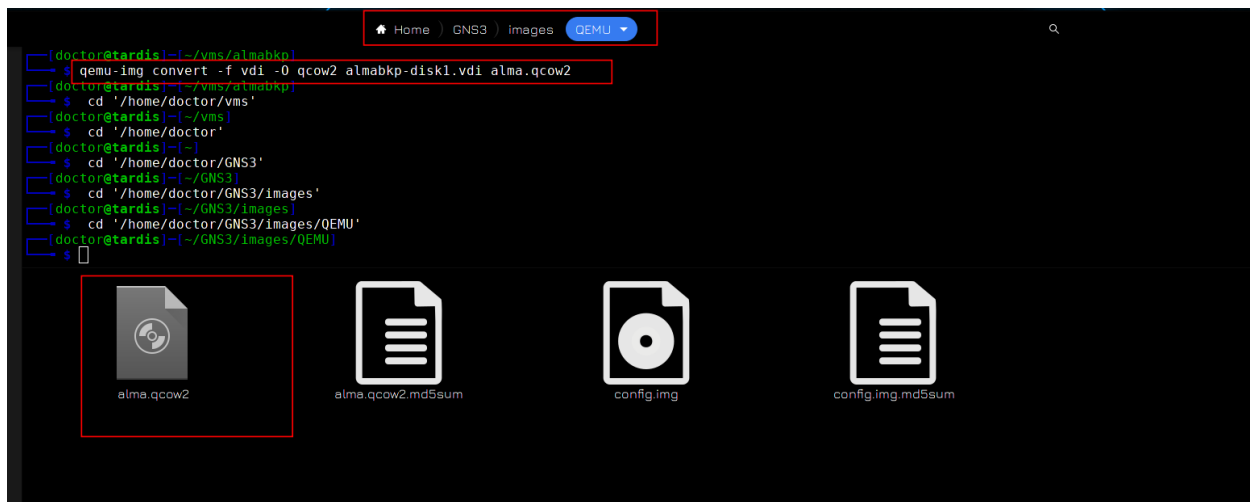
2.7.1 07. Деление на зоны



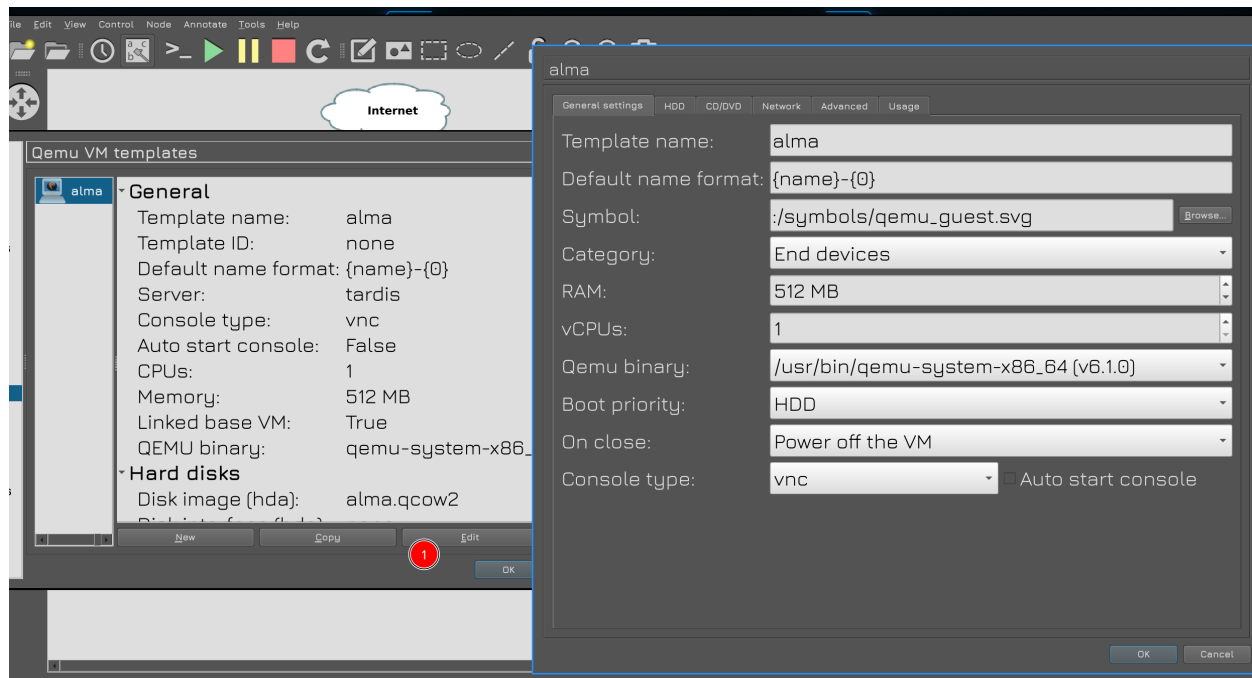
Прежде чем начнём, стоит упомянуть, что я решил вместо VirtualBox-а использовать Qemu. Не потому что Virtualbox чем-то плох, просто у GNS-а есть функционал снапшотов, когда он может сохранить состояние всей топологии, в том числе виртуалок. Но это работает только с Qemu. Мне этот функционал нужен для записи курса, а вам он в принципе не обязателен, поэтому можете продолжать использовать виртуалбокс. Установку Qemu я не буду разбирать, сами справитесь.



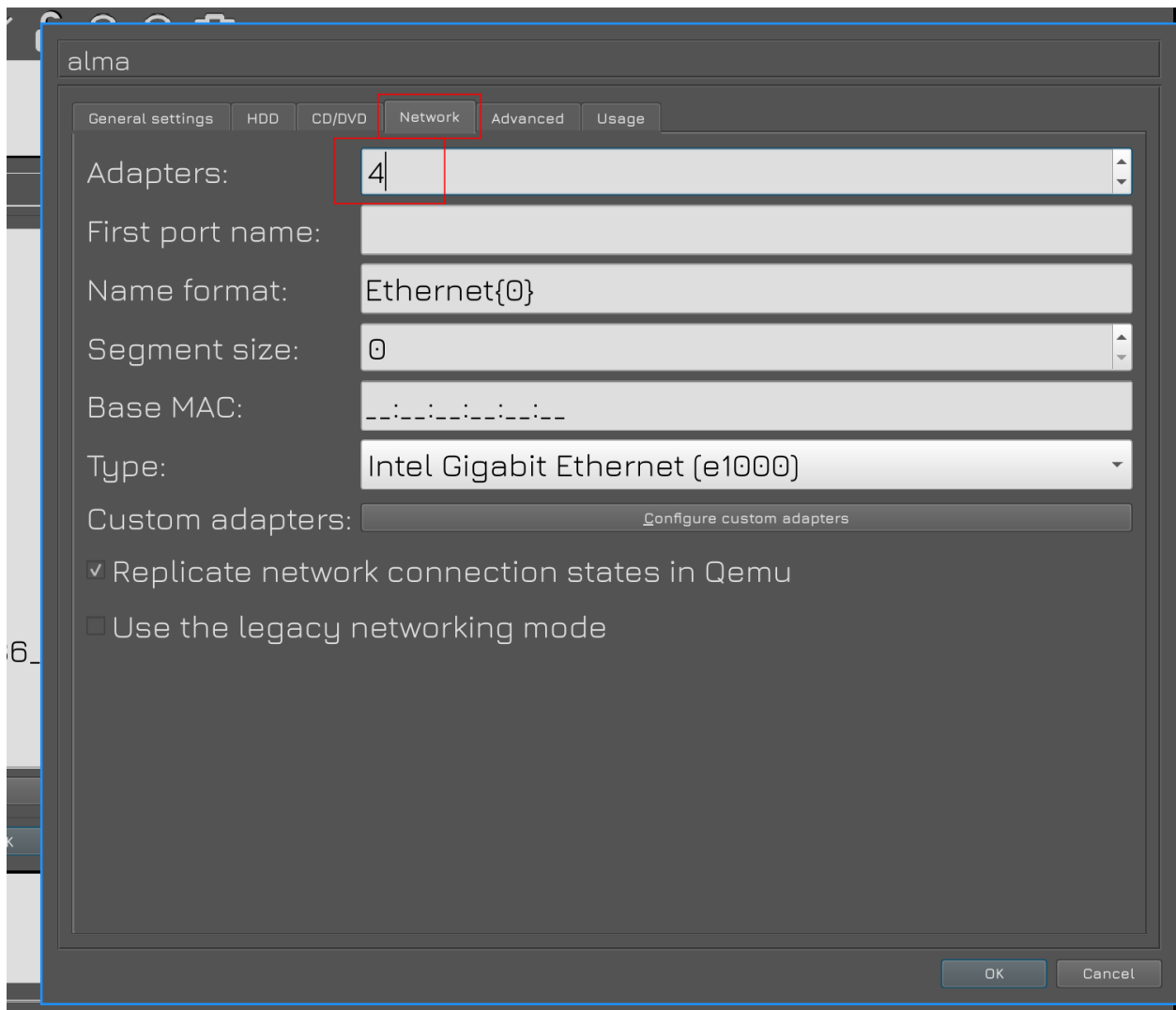
Сами виртуалки нужно создавать в GNS. Это делается примерно там же, где мы связывали VirtualBox и GNS: Edit - Preferences - Qemu VMs.



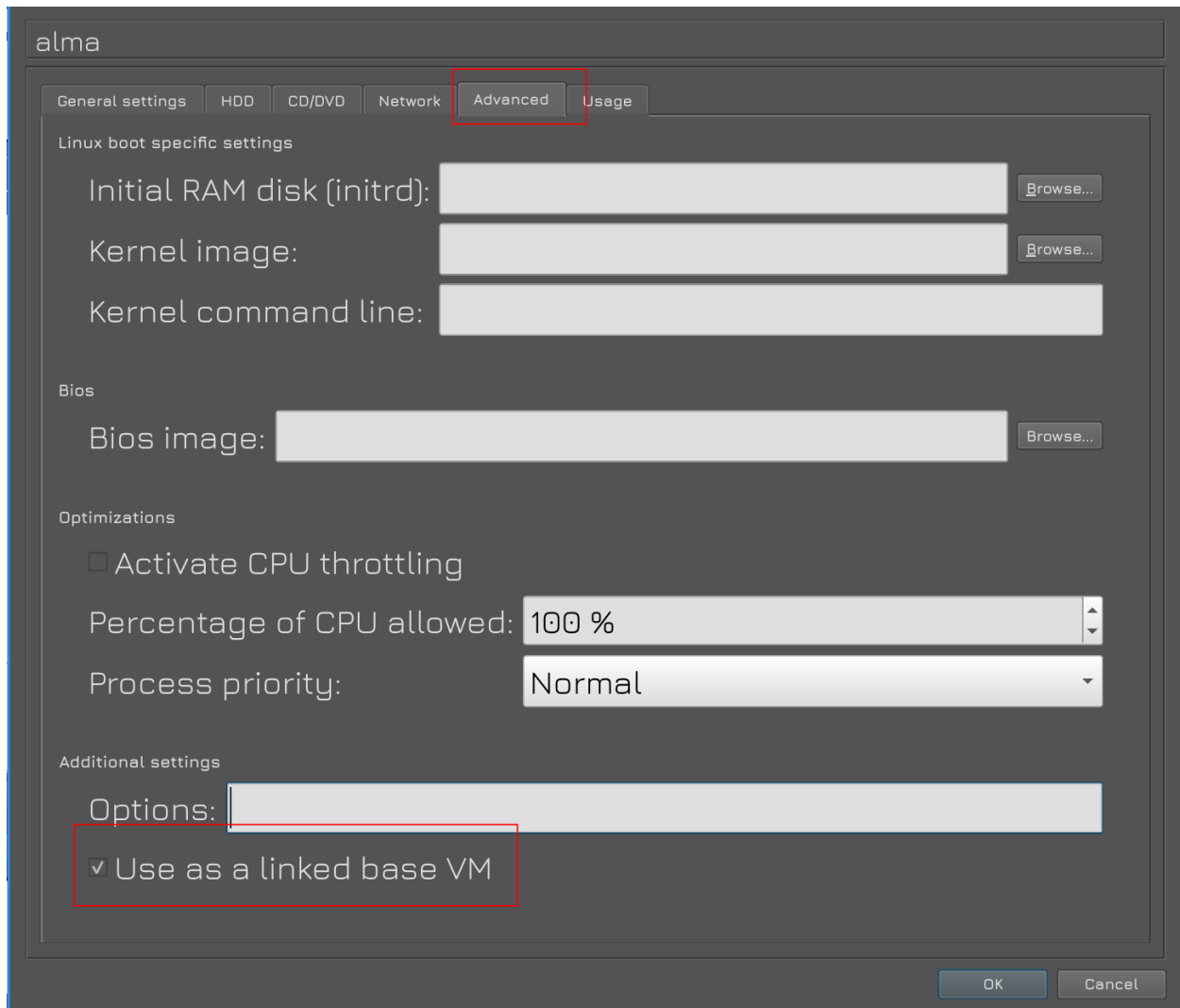
Я предварительно преобразовал образ дисков VDI от VirtualBox в qcow2, который нужен для Qemu и положил в нужную директорию.



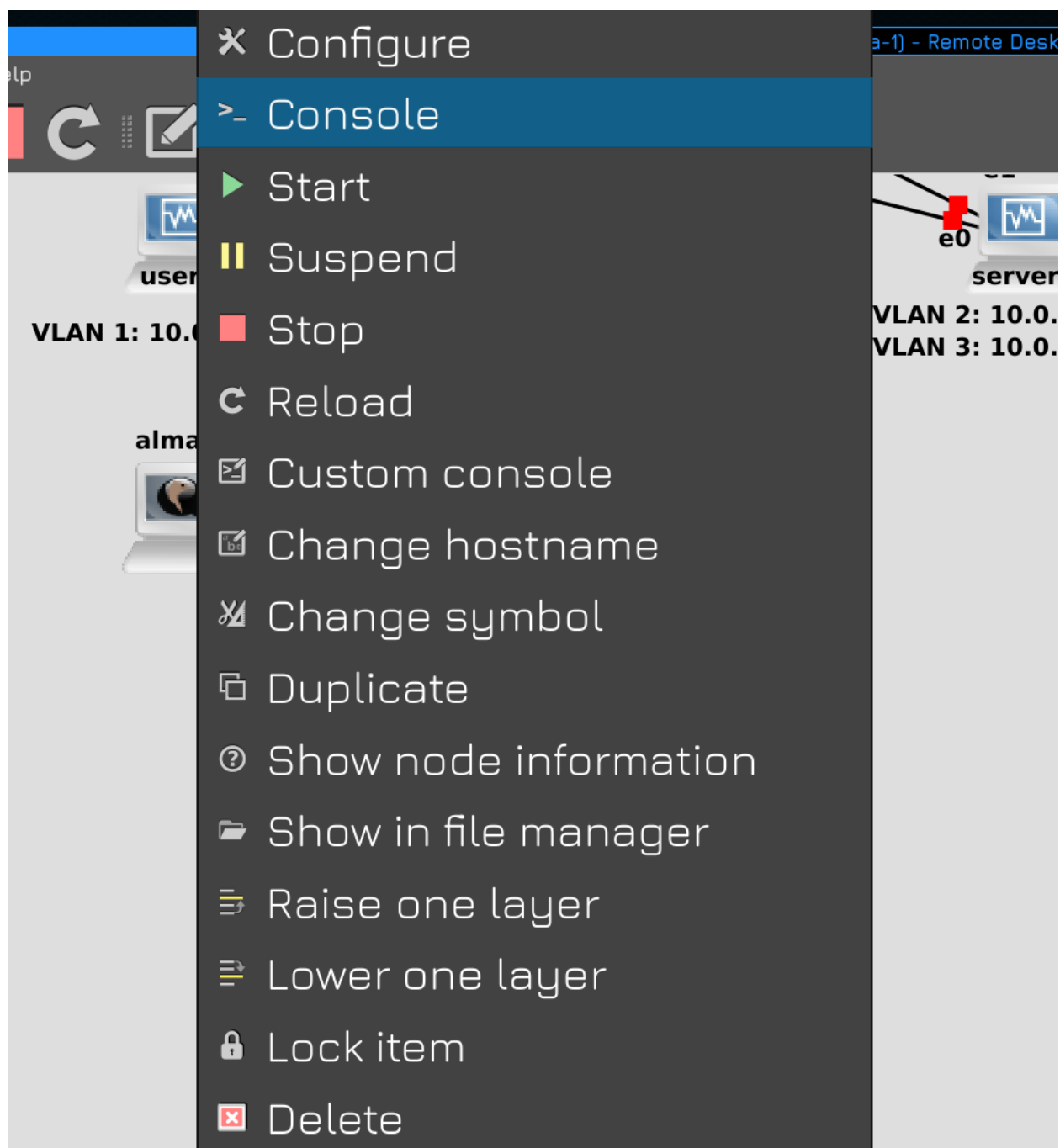
По настройкам виртуалки, я опять же ей выделил всего 512 мегабайт оперативки. Для подключения к консоли буду использовать vnc, это всё делается через GNS, без необходимости что-то менять в самой виртуалке.



Во вкладке Network всё также выделил виртуалке 4 сетевых адаптера.



Ну и во вкладке Advanced нужно убедиться, что стоит галочка «Use as a linked base VM», чтобы мы простым перетягиванием создавали новые виртуалки.



Для подключения к консоли виртуалки надо нажать на ней правой кнопкой мыши и выбрать Console. Для этого в системе должен быть какой-нибудь VNC клиент.

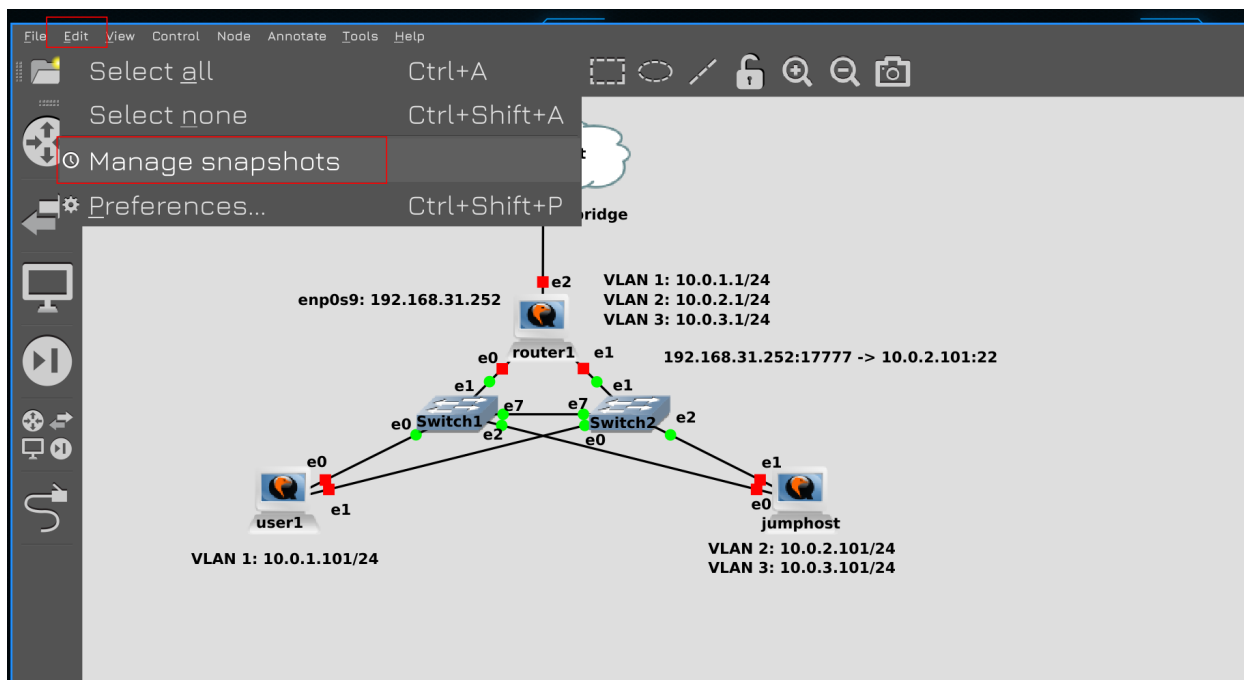
```

[root@user1 ~]# nmcli connection show
NAME                UUID                                  TYPE      DEVICE
team                 2d19ced8-814f-4e4b-a9d8-4d7efaa658f9 team       team0
team-slave-ens3      42744b20-bdb2-42b1-b3b8-163de115de04 ethernet  ens3
team-slave-ens4      8534bf2e-f470-42f5-9c37-56ba08cc2ccf ethernet  ens4
[root@user1 ~]# 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: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel master team0 state UP group default qlen 1000
    link/ether 0c:63:e9:b3:00:00 brd ff:ff:ff:ff:ff:ff
3: ens4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel master team0 state UP group default qlen 1000
    link/ether 0c:63:e9:b3:00:00 brd ff:ff:ff:ff:ff:ff permaddr 0c:63:e9:b3:00:01
4: ens5: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state DOWN group default qlen 1000
    link/ether 0c:63:e9:b3:00:02 brd ff:ff:ff:ff:ff:ff
5: ens6: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state DOWN group default qlen 1000
    link/ether 0c:63:e9:b3:00:03 brd ff:ff:ff:ff:ff:ff
7: team0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 0c:63:e9:b3:00:00 brd ff:ff:ff:ff:ff:ff
    inet 10.0.1.101/24 brd 10.0.1.255 scope global noprefixroute team0
        valid_lft forever preferred_lft forever
    inet6 fe80::acb5:7199:fc35:af37/64 scope link noprefixroute
        valid_lft forever preferred_lft forever

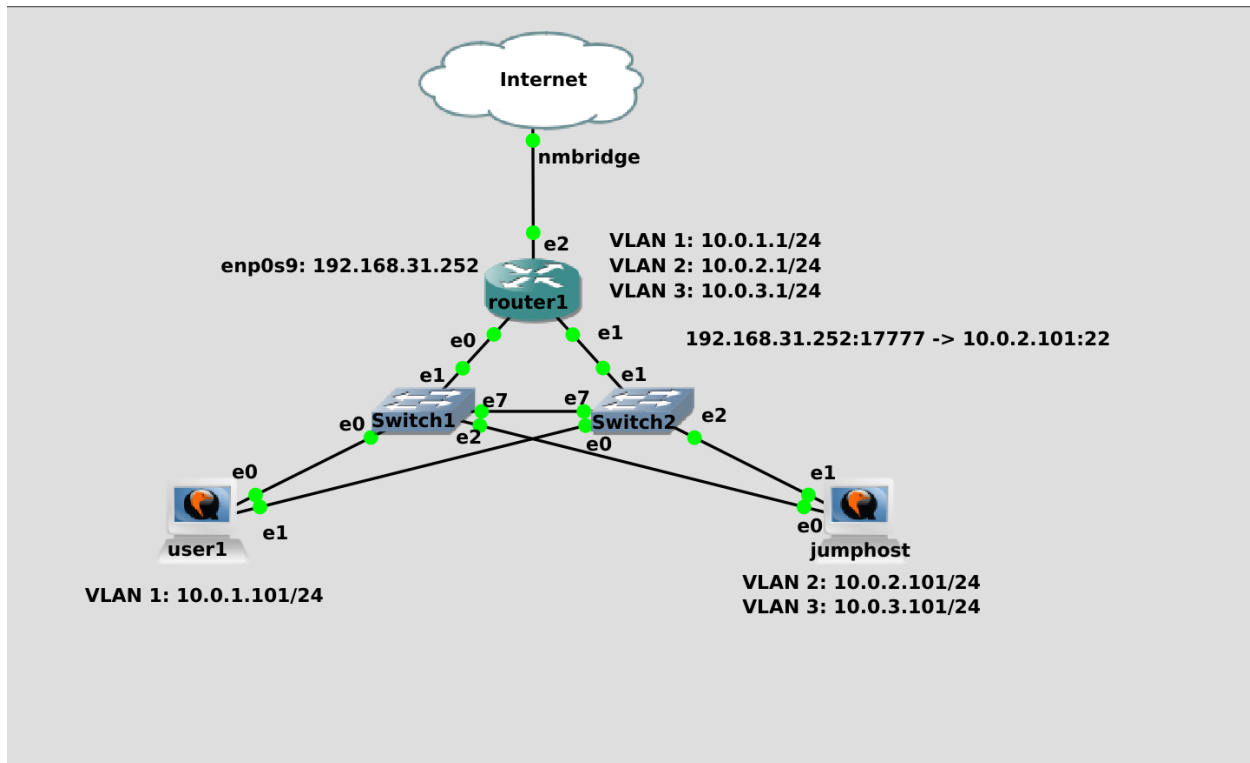
```

Ну и учтите, что тут имена сетевых интерфейсов будут отличаться, но вы всегда можете их узнать по выводу команды:

```
ip a
```



В общем, я настроил всё как было. И при выключенных виртуалках теперь можно делать снапшоты всей топологии. Если вы по какой-то причине тоже решили перейти на Qemu и у вас что-то не получилось, напишите, разберёмся. Это было небольшое вступление, теперь перейдём к самой теме.



И так, в прошлый раз мы создали jumphost, через который мы можем логиниться на внутренние машинки из интернета. На сам jumphost мы можем попасть напрямую из интернета, так как на роутере мы пробросили порт. Из-за того что это не очень безопасно, мы приняли какие-то меры безопасности - использовали нестандартный порт, отключили аутентификацию по паролям и запретили руту логиниться. Но представьте, что у SSH есть какая-то уязвимость, о которой пока никто не знает - уязвимость нулевого дня. Условно, она позволяет зайти на сервер даже не зная логина и пароля. И вот какие-то взломщики нашли её и попытались взломать наш jumphost, через который они смогут попасть на другие наши сервера. При определённых условиях нас мог бы спасти selinux, но вот по каким-то причинам это не произошло. И вот взломщики получили доступ на джамп хост, просканировали сеть и увидели все наши сервера, которые они дальше будут ломать.

И вот из этой идеи - «нас уже взломали» - должна строиться система информационной безопасности. Невозможно построить систему, которая будет безопасной на все 100% - всё можно взломать, вопрос только в том, сколько времени, сил и денег на этой уйдёт. Системы информационной безопасности стоят не малых денег и всё упирается в принцип - система защиты не должна стоить дороже того, что она охраняет. Тоже самое касается взлома. Т.е. грубо говоря нет смысла покупать сейф за миллион долларов и хранить в нём 100 рублей. И нет смысла нанимать команду из профессионалов, у которых супер инструменты для взлома такого сейфа, чтобы в итоге достать 100 рублей.

Поэтому, строя систему защиты, с одной стороны нужно стараться сделать её неуязвимой, с другой - понимать, что это невозможно. А значит думать о следующем шаге - «нас уже взломали». Что дальше? Нужно смягчить урон. В инфобезе это называется «mitigation». Если кто-то попал на один сервер - сделайте так, чтоб он с этого сервера не мог попасть на другие. Если взломают одну систему - сделайте так, чтобы это не распространилось на другие. Попал в вашу сеть шифровальщик, зашифровал какие-то сервера - нужно сделать так, чтобы это не затронуло другие, чтобы не перекинулось в другие сети.

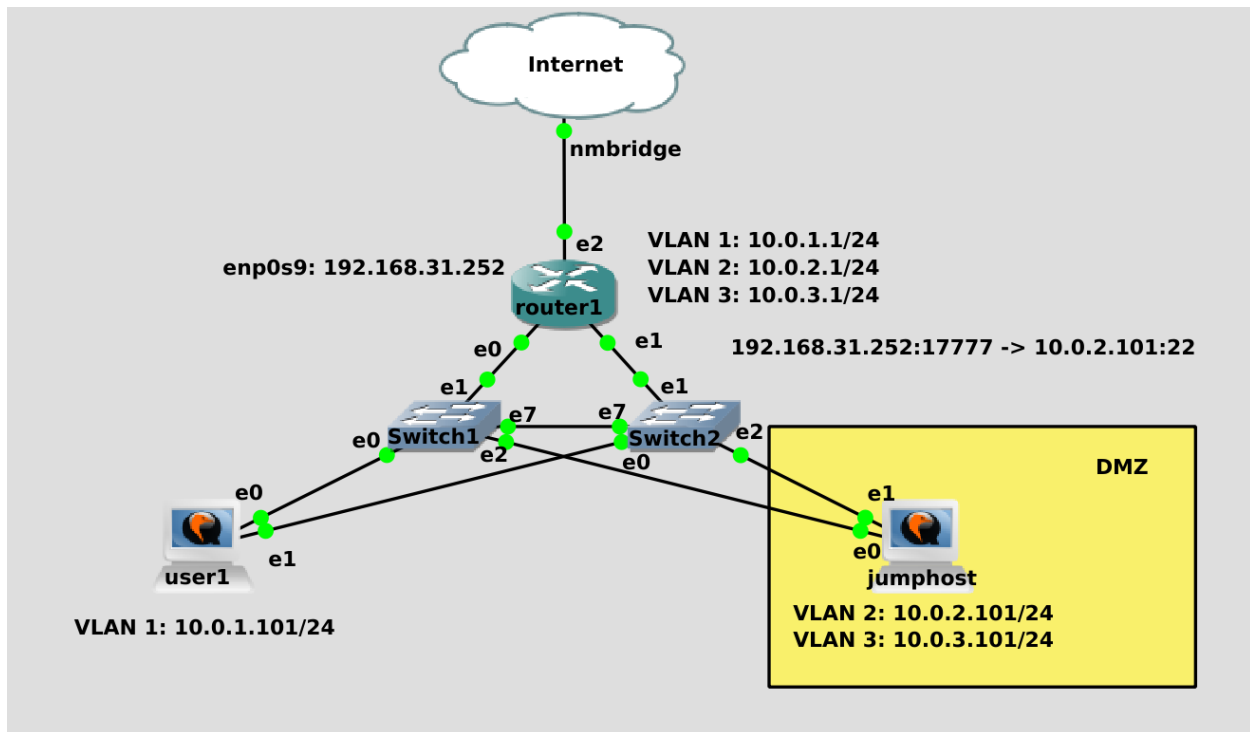
Чем сильнее вы усложните хакерам работу, чем больше всяких препятствий вы им поставите - тем сложнее им будет добраться до важных данных. А пока они будут со всем этим возиться, возможно, вы успеете что-то заметить и среагировать.

В общем, возвращаясь к джамп хосту. Он у нас доступен из интернета и один из первых претендентов

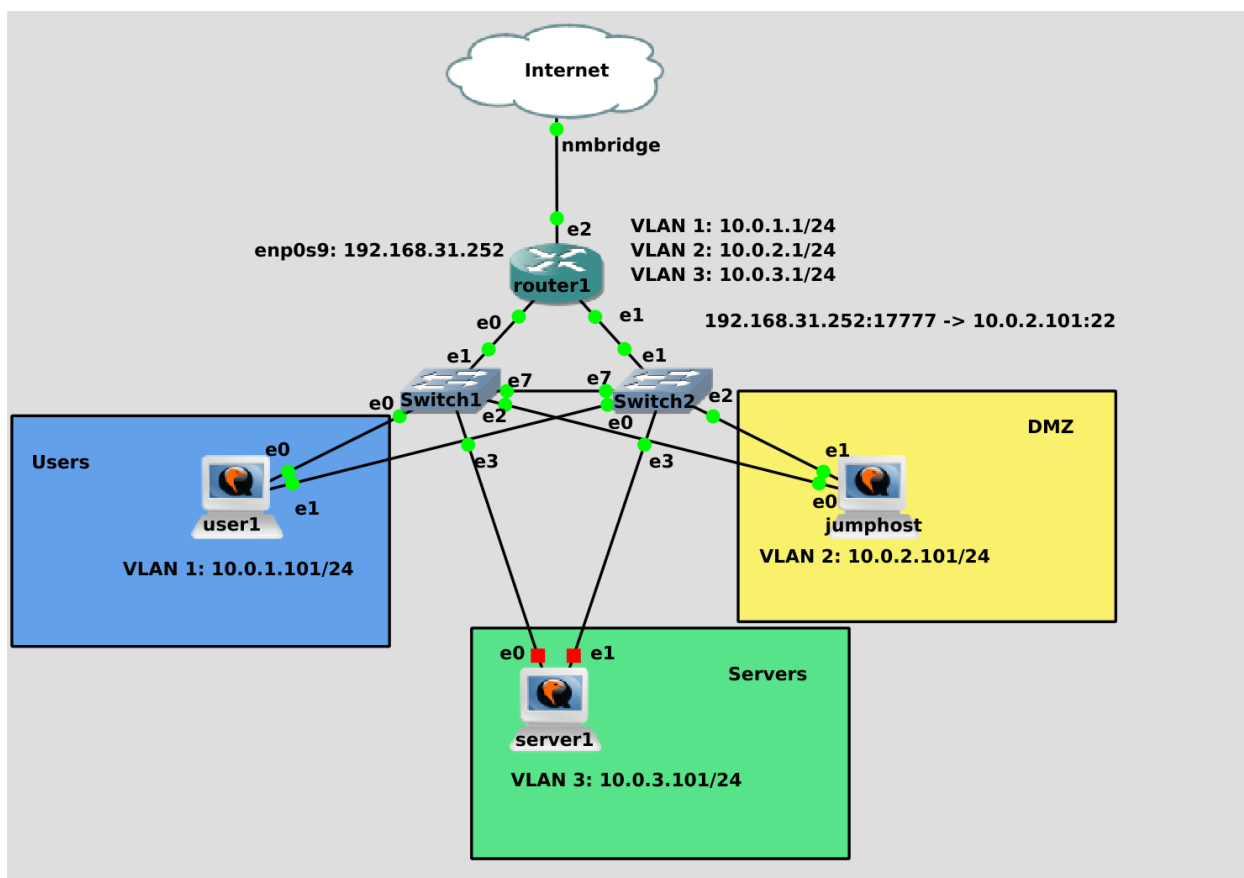
на взлом. А значит его надо как-то отделить от других систем. Мы говорили об изоляции на уровне VLANов и подсетей, а с точки зрения информационной безопасности вводится ещё одно понятие - зоны. Мы уже знакомы с этим понятием благодаря фаерволу - firewalld у нас zone-based firewall. Так вот, зона - это больше логическое деление, одна зона может состоять как из одного VLANа с одной подсетью, так и из множества VLANов и подсетей.



Например, мы можем для каждой аудитории в универе выделить отдельный VLAN с отдельной подсетью. Но с точки зрения зон - все аудитории находятся в одной зоне, так как по уровню доступа студенты одной зоны аудитории ничем не отличаются от студентов другой аудитории. Т.е. деление по зонам - это в основном про доступы. Скажем, студентам можно на такие-то сервера - значит все студенты в одной зоне, сервера в другой - и мы пишем правила, что зоне «студенты» разрешён доступ к зоне «сервера для студентов» по порту 80. На самом деле обычно настраивается более гранулярный доступ, мол, на эти сервера по 80 порту, а на эти по 443, на эти вообще нельзя и т.п. В общем-то всё начинается с зон, но ими не ограничивается.

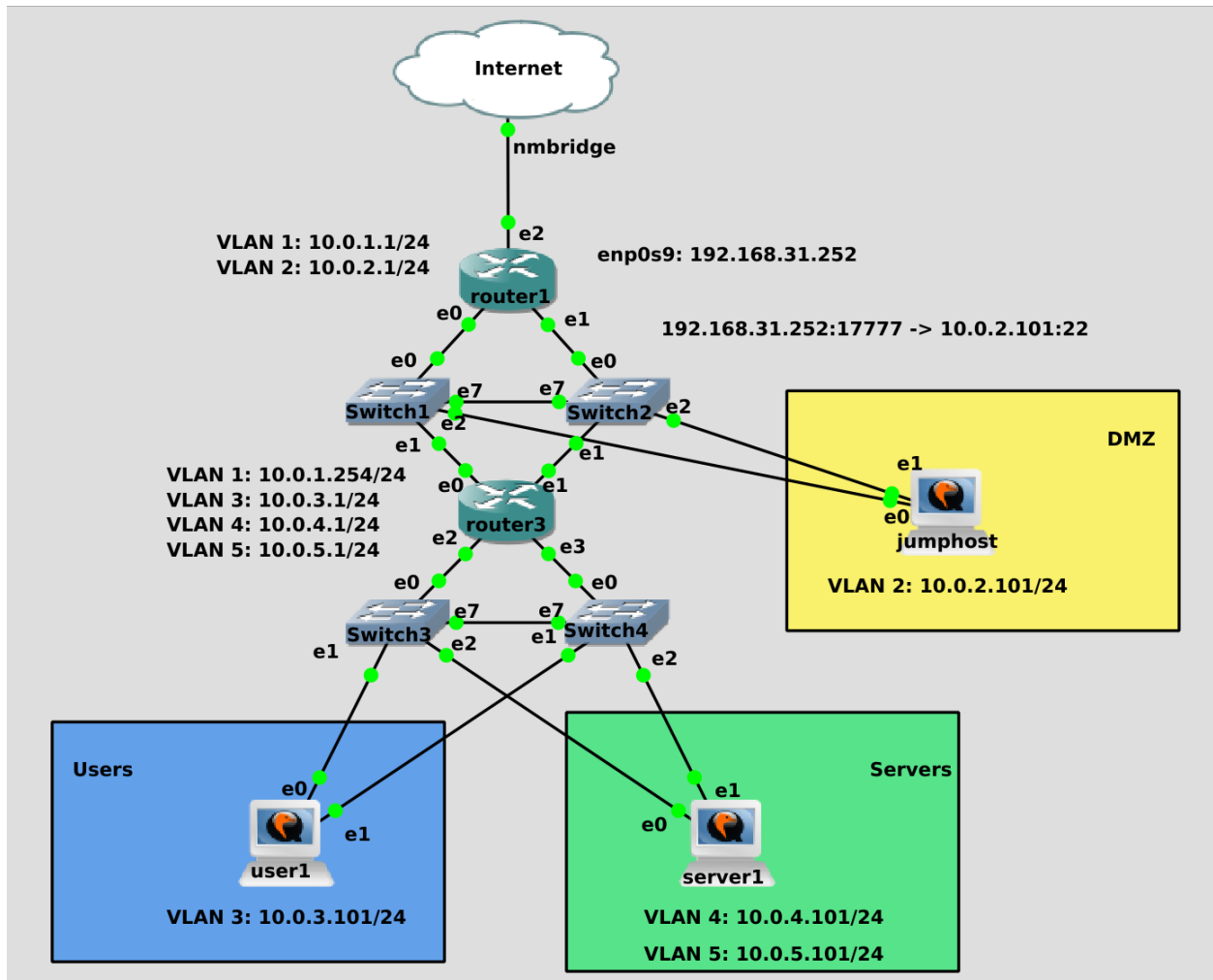


Так вот, системы, которые доступны из интернета, с точки зрения безопасности отделяют в одну зону, называемую DMZ - демилитаризованная зона. Обычно здесь находятся вебсервера, джамп хосты и прочие системы, к которым меньше всего доверия. Чем сильнее мы изолируем эту зону, чем меньше доступов мы дадим из этой зоны - тем лучше. Понятное дело, что джамп хост на то и джамп хост, что мы с него должны иметь возможность попасть на другие сервера.

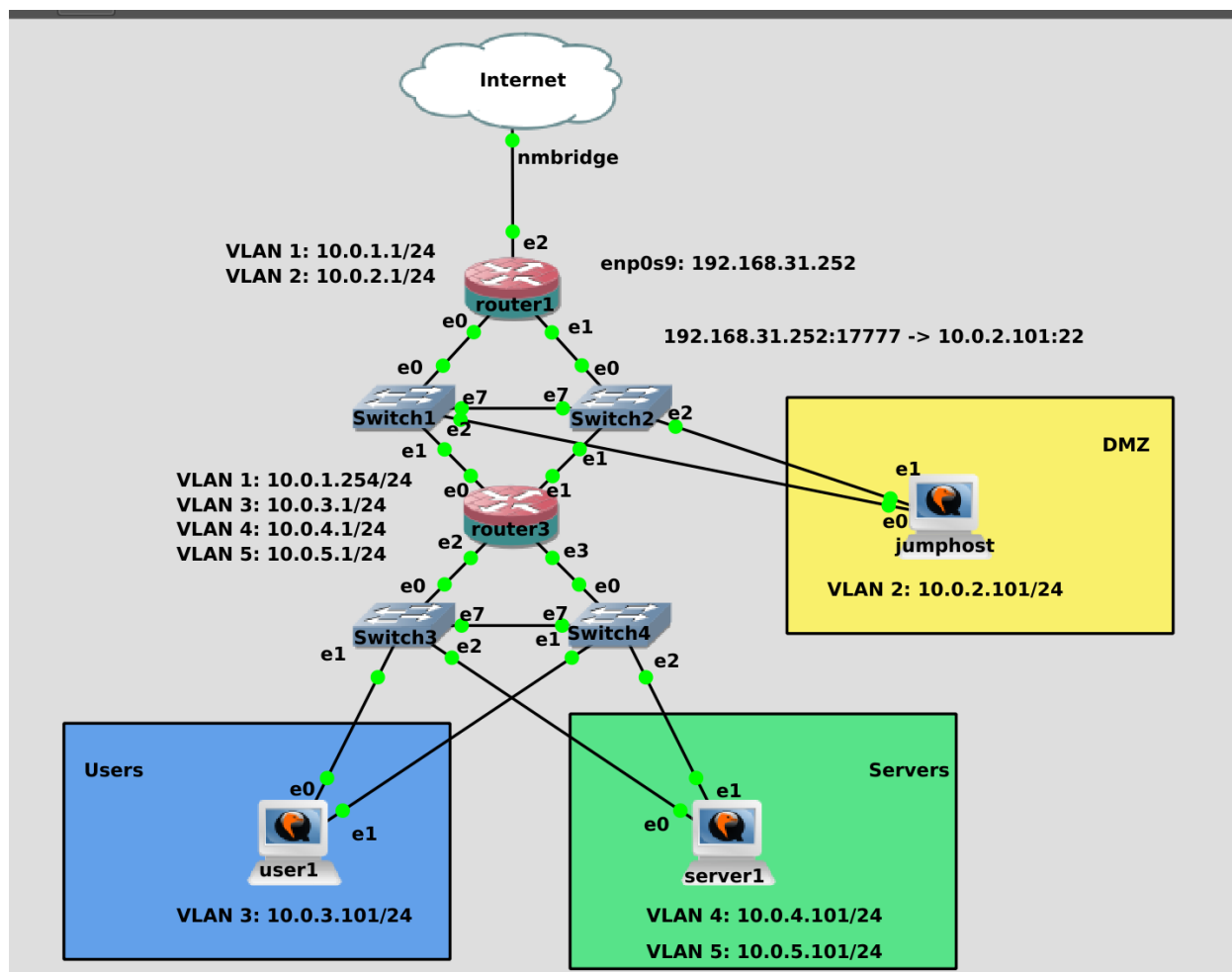


Но, допустим, мы можем разрешить джампхосту подключаться только к одному серверу во внутренней зоне. И использовать этот сервер в качестве второго джампхоста, у которого уже будет доступ к другим системам. Это сократит площадь атак.

В нашей схеме помимо джампхоста в интернет смотрит ещё одно устройство - роутер1. Да, мы его обезопасили, туда нельзя попасть из интернета, мы даже можем запретить доступ с джампхоста и запретить пинги. Но тем не менее, он подключен в интернет, как минимум, его могут вывести из строя DDOS атакой. Да и шанс взлома, хоть и мизерный, никогда не стоит исключать. И если наш роутер выйдет из строя, либо начнёт тормозить - ляжет вся наша сеть, юзеры не смогут заходить на сервера, а сервера не смогут обмениваться данными для работы. Пофиг на интернет, но нарушение внутренней работы сети - в средней или крупной компании большая проблема.



Поэтому внутренняя сеть располагается на другом роутере. Давайте немного преобразим нашу сеть. router1 у нас будет внешним роутером, который смотрит в интернет. С него будет проброс портов на различные сервисы, которые должны быть доступны в интернете, например, на джампхост. Пока что для DMZ отделим второй влан. Роутеру1 также нужен другой влан, например, влан1, чтобы он мог общаться с внутренним роутером. Его мы назовём router3. Он уже будет связывать внутреннюю сеть - пользователей, внутренние сервера и прочее. На нём нам нужен первый влан, чтобы общаться с роутером1, а также 3 и 4 вланы для юзеров и серверов соответственно. В дальнейшем возможно будут дополнительные зоны, но на пока этого достаточно.



Так как теперь наши роутеры будут не только связывать подсети, но и разделять сеть на зоны, т.е. выступать сетевыми фаерволами, для аутентичности поменяем иконку у роутеров. Так как теперь на джампхост и юзер1 приходит только один влан, на свитчах в их сторону настроим нетегированные вланы, и только к роутерам и серверу1 будем использовать транки. Я пропущу этап настройки адресов и вланов, так как мы это делали раньше, а для вас это будет небольшим заданием на повторение. Однако с маршрутизацией начинающим может быть сложно, поэтому это покажу.

```

└─ $ cat .ssh/config
Host jumphost
  User user
  IdentityFile ~/.ssh/jumphost
  Hostname 192.168.31.252
  Port 17777

Host router1
  User user
  IdentityFile ~/.ssh/servers
  ProxyJump jumphost
  Hostname 10.0.2.1
└─ [doctor@tardis]~]
└─ $ eval $(ssh-agent)
Agent pid 175781
└─ [doctor@tardis]~]
└─ $ ssh-add .ssh/jumphost .ssh/servers
Enter passphrase for .ssh/jumphost:
Identity added: .ssh/jumphost (doctor@tardis)
Identity added: .ssh/servers (doctor@tardis)
└─ [doctor@tardis]~]
└─ $ ssh router1
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Sat Dec 18 13:38:05 2021 from 10.0.2.101
[user@router1 ~]$

```

Для начала подключимся во внутреннюю сеть. Запускаем ssh agent:

```
eval $(ssh-agent)
```

И добавляем ключи:

```
ssh-add .ssh/jumphost .ssh/servers
```

Дальше подключаемся к роутеру:

```
ssh router1
```

```
[user@router1 ~]$ sudo firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: ens3 ens4 team0 team0.2
  sources:
  services: cockpit dhcpv6-client ssh
  ports:
  protocols:
  forward: no
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
[user@router1 ~]$
```

Для начала посмотрим дефолтную зону на файрволе:

```
sudo firewall-cmd --list-all
```

Сейчас у нас интерфейсы, которые смотрят в разные вланы - team0 и team0.2 - находятся в одной зоне.

```
[user@router1 ~]$ sudo firewall-cmd --change-interface=team0.2 --zone=dmz --permanent
The interface is under control of NetworkManager, setting zone to 'dmz'.
success
[user@router1 ~]$ sudo firewall-cmd --reload
success
[user@router1 ~]$ sudo firewall-cmd --list-all --zone=dmz
dmz (active)
  target: default
  icmp-block-inversion: no
  interfaces: team0.2
  sources:
  services: ssh
  ports:
  protocols:
  forward: no
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
[user@router1 ~]$
```

team0.2 - это интерфейс, который смотрит во второй влан - а он у нас для DMZ. Поэтому стоит перенести этот интерфейс в другую зону. На файрволе уже есть зона с таким названием, поэтому сразу переместим в эту зону:

```
sudo firewall-cmd --change-interface=team0.2 --zone=dmz --permanent
sudo firewall-cmd --reload
sudo firewall-cmd --list-all --zone=dmz
```

Интерфейс переместился в эту зону, но, как видите, в ней есть сервис ssh.

<pre>[user@router1 ~]\$ sudo firewall-cmd --remove-service=ssh --zone=dmz --permanent success [user@router1 ~]\$ sudo firewall-cmd --reload success [user@router1 ~]\$ sudo firewall-cmd --list-all --zone=dmz dmz (active) target: default icmp-block-inversion: no interfaces: team0.2 sources: services: ports: protocols: forward: no masquerade: no forward-ports: source-ports: icmp-blocks: rich rules: [user@router1 ~]\$</pre>	<pre>[doctor@tardis]~\$ ssh router1 Enter passphrase for key '/home/doctor/.ssh/jumpshost': channel 0: open failed: connect failed: No route to host stdio forwarding failed kex_exchange_identification: Connection closed by remote host Connection closed by UNKNOWN port 65535 [doctor@tardis]~\$</pre>
---	---

Это значит, что из DMZ зоны можно будет подключиться по ssh к роутеру. А так как мы не очень доверяем DMZ, то лучше убрать эту возможность:

```
sudo firewall-cmd --remove-service=ssh --zone=dmz --permanent
sudo firewall-cmd --reload
sudo firewall-cmd --list-all --zone=dmz
```

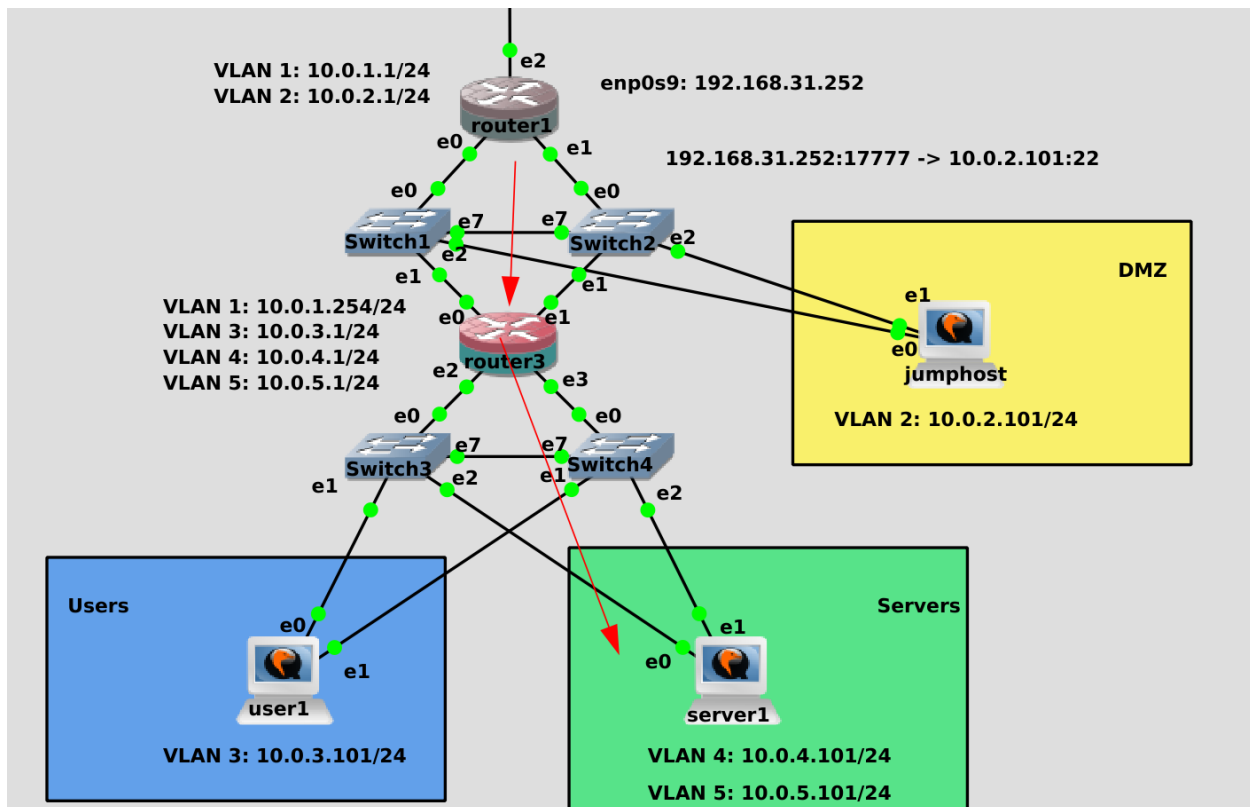
Это не отразится на текущем подключении, но больше с джампхоста мы не сможем зайти на роутер.

```
[user@router1 ~]$ ip ro sh
default via 192.168.31.1 dev ens5 proto static metric 106
10.0.1.0/24 dev team0 proto kernel scope link src 10.0.1.1 metric 350
10.0.2.0/24 dev team0.2 proto kernel scope link src 10.0.2.1 metric 400
192.168.31.0/24 dev ens5 proto kernel scope link src 192.168.31.252 metric 106
[user@router1 ~]$
```

Теперь посмотрим маршруты:

```
ip ro sh
```

Роутер 1 знает о сетях 1.0 и 2.0. И всё неизвестное он посылает в интернет через провайдера. Но за роутером 3 у нас есть ещё 3 сети - 3.0, 4.0 и 5.0. Если какой-нибудь пакет придёт с сети 3.0 на роутер1, то роутер 1 ответ будет посылать на свой default gateway.



Нас это не устраивает, мы хотим, чтобы для ответа в 3, 4 и 5 сети роутер1 обращался к роутеру 3. И вот для этого нам нужны статические маршруты. Условно, мы должны сказать роутеру1, что сеть 10.0.4.0 находится за роутером3. Адрес роутера3 для первого роутера - 10.0.1.254.

```
[user@router1 ~]$ ip ro sh
default via 192.168.31.1 dev ens5 proto static metric 106
10.0.1.0/24 dev team0 proto kernel scope link src 10.0.1.1 metric 350
10.0.2.0/24 dev team0.2 proto kernel scope link src 10.0.2.1 metric 400
192.168.31.0/24 dev ens5 proto kernel scope link src 192.168.31.252 metric 106
[user@router1 ~]$
[user@router1 ~]$ sudo nmcli connection modify team +ipv4.routes "10.0.3.0/24 10.0.1.254"
[sudo] password for user:
[user@router1 ~]$ sudo nmcli connection modify team +ipv4.routes "10.0.4.0/24 10.0.1.254"
[user@router1 ~]$ sudo nmcli connection modify team +ipv4.routes "10.0.5.0/24 10.0.1.254"
[user@router1 ~]$ sudo nmcli connection up team
Connection successfully activated (master waiting for slaves) (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/17)
[user@router1 ~]$ ip ro sh
default via 192.168.31.1 dev ens5 proto static metric 106
10.0.1.0/24 dev team0 proto kernel scope link src 10.0.1.1 metric 350
10.0.2.0/24 dev team0.2 proto kernel scope link src 10.0.2.1 metric 400
10.0.3.0/24 via 10.0.1.254 dev team0 proto static metric 350
10.0.4.0/24 via 10.0.1.254 dev team0 proto static metric 350
10.0.5.0/24 via 10.0.1.254 dev team0 proto static metric 350
192.168.31.0/24 dev ens5 proto kernel scope link src 192.168.31.252 metric 106
[user@router1 ~]$
```

Поэтому на роутере1 прописываем статические маршруты:

```
sudo nmcli connection modify team +ipv4.routes "10.0.3.0/24 10.0.1.254"
sudo nmcli connection modify team +ipv4.routes "10.0.4.0/24 10.0.1.254"
sudo nmcli connection modify team +ipv4.routes "10.0.5.0/24 10.0.1.254"
```

Мы через `nmcli` добавляем статичный маршрут, в котором пишем, что сети 3.0, 4.0 и 5.0 находятся за адресом 10.0.1.254.

После изменений поднимаем интерфейс, чтобы он применил настройки:

```
nmcli con up team
```

Ну и проверим таблицу маршрутизации:

```
ip ro sh
```

Теперь у нас в списке есть дополнительные маршруты. Без них роутер1 не смог бы ответить серверам в этих вланах, а значит, как минимум, у них не было бы интернета.

```
[user@router1 ~]$ ssh 10.0.1.254
user@10.0.1.254's password:
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Sat Dec 18 14:19:56 2021 from 10.0.1.1
[user@router3 ~]$ nmcli connection show
NAME                UUID                                  TYPE      DEVICE
team0                ea24777a-342b-4709-a618-eeed06b0cc4e team       team0
vlan3                f199c498-b4a7-4445-8b2e-153a0c8110a6 vlan       team1.3
vlan4                0cc4912c-70d7-4b91-be3a-d5fdbf0c3f02 vlan       team1.4
vlan5                4026e77b-6749-4754-bb1d-e875964a0209 vlan       team1.5
team1                c4dfb3c7-3079-412b-b1a5-342c58b8fe09 team       team1
team-slave-ens3      8e11a1e3-d542-4889-8adb-29336739fb9b ethernet  ens3
team-slave-ens4      833c746d-5c37-4984-9090-df5b8a7373c1 ethernet  ens4
team-slave-ens5      b4ce000f-947e-4535-aba8-b148da3614ac ethernet  ens5
team-slave-ens6      449014a4-5b0d-458c-998f-eaf61cfd7080 ethernet  ens6
[user@router3 ~]$ ip ro sh
default via 10.0.1.1 dev team0 proto static metric 352
10.0.1.0/24 dev team0 proto kernel scope link src 10.0.1.254 metric 352
10.0.3.0/24 dev team1.3 proto kernel scope link src 10.0.3.1 metric 400
10.0.4.0/24 dev team1.4 proto kernel scope link src 10.0.4.1 metric 401
10.0.5.0/24 dev team1.5 proto kernel scope link src 10.0.5.1 metric 402
[user@router3 ~]$
```

Что касается роутера3:

```
ssh 10.0.1.254
```

То тут у нас два тиминг интерфейса по 2 слеива:

```
nmcli con sh
```

Один интерфейс - `team0` - смотрим в сторону первого роутера, второй - `team1` - смотрит во внутреннюю сеть, там где пользователи и сервера. На втором тиминг интерфейсе мы создаём вланы 3, 4 и 5.

Что касается маршрутов:

```
ip ro sh
```

То тут всё гораздо проще. Здесь default gateway является роутер1. И так вся внутренняя сеть подключена к этому роутеру, а если он что-то не знает - скажем, интернет адреса, то он пошлёт на первый роутер, а тот уже в интернет.

```
[user@router3 ~]$ sudo firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: ens3 ens4 ens5 ens6 team0 team1 team1.3 team1.4 team1.5
  sources:
  services: cockpit dhcpv6-client ssh
  ports:
  protocols:
  forward: no
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
[user@router3 ~]$
```

Что касается деления на зоны, то тут у нас 4 зоны:

```
sudo firewall-cmd --list-all
```

- Во-первых - team0 - он у нас смотрит в сторону первого роутера.
- Во-вторых - вланы 3, 4 и 5 - для каждого из них мы выделим отдельную зону.

```
[user@router3 ~]$ sudo firewall-cmd --permanent --new-zone=routers
[sudo] password for user:
success
[user@router3 ~]$ sudo firewall-cmd --permanent --new-zone=users
success
[user@router3 ~]$ sudo firewall-cmd --permanent --new-zone=servers
success
[user@router3 ~]$ sudo firewall-cmd --permanent --new-zone=management
success
[user@router3 ~]$
```

Можно было бы использовать шаблонные зоны, но лучше создать свои с более подходящими названиями:

```
sudo firewall-cmd --permanent --new-zone=routers
sudo firewall-cmd --permanent --new-zone=users
sudo firewall-cmd --permanent --new-zone=servers
sudo firewall-cmd --permanent --new-zone=management
```

- routers - зона между первым и третьим роутером
- users - зона для локальных пользователей
- servers - зона для серверов
- management - зона для управления серверами. О ней мы поговорим чуть позже.


```
[user@router3 ~]$ sudo firewall-cmd --permanent --change-interface=team0 --zone=routers
[sudo] password for user:
The interface is under control of NetworkManager, setting zone to 'routers'.
success
[user@router3 ~]$ sudo firewall-cmd --permanent --change-interface=team1.3 --zone=users
The interface is under control of NetworkManager, setting zone to 'users'.
success
[user@router3 ~]$ sudo firewall-cmd --permanent --change-interface=team1.4 --zone=servers
The interface is under control of NetworkManager, setting zone to 'servers'.
success
[user@router3 ~]$ sudo firewall-cmd --permanent --change-interface=team1.5 --zone=management
The interface is under control of NetworkManager, setting zone to 'management'.
success
[user@router3 ~]$
```

Дальше переместим интерфейсы в соответствующие зоны:

```
sudo firewall-cmd --permanent --change-interface=team0 --zone=routers
sudo firewall-cmd --permanent --change-interface=team1.3 --zone=users
sudo firewall-cmd --permanent --change-interface=team1.4 --zone=servers
sudo firewall-cmd --permanent --change-interface=team1.5 --zone=management
```

team0 в зону routers, team1.3 в зону users, team1.4 в зону servers и team1.5 в зону management.

```
[user@router3 ~]$ sudo firewall-cmd --reload
success
[user@router3 ~]$ sudo firewall-cmd --list-all --zone=routers
routers (active)
  target: default
  icmp-block-inversion: no
  interfaces: team0
  sources:
  services:
  ports:
  protocols:
  forward: no
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
[user@router3 ~]$
```

Затем перезапустим фаервол, чтобы применились настройки:

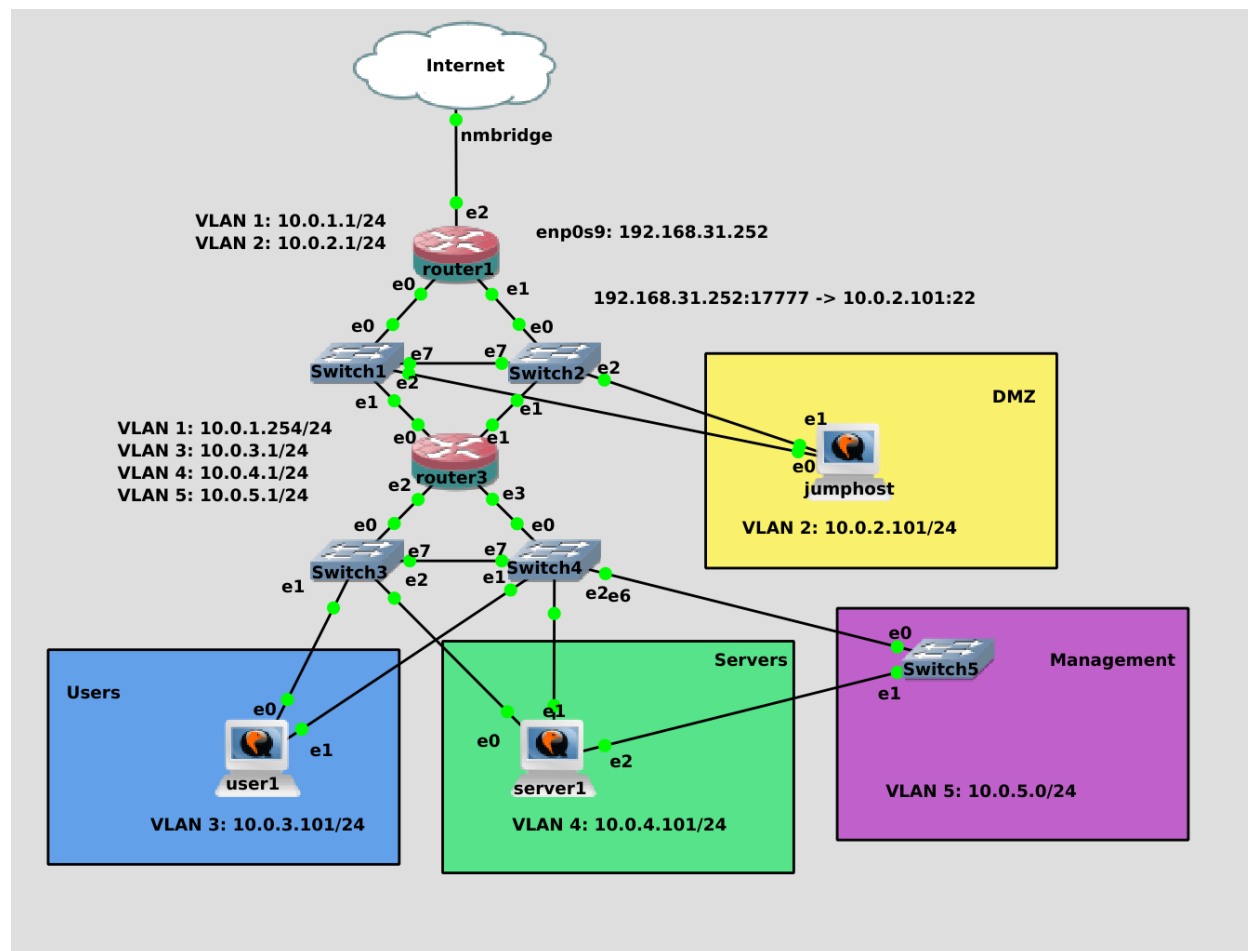
```
sudo firewall-cmd --reload
```

Ну и можем проверить настройки:

```
sudo firewall-cmd --list-all --zone=routers
```

Вроде зона есть, интерфейс в зоне и, по-умолчанию, никаких настроек, никаких сервисов в зоне. А значит никто из зоны не сможет подружиться к самому роутеру.

Вы спросите - а как же нам тогда управлять роутерами, если к ним ниоткуда подружиться нельзя? Тут уже в дело вступает management vlan.



Как мы говорили, в локальной сети бывает большое количество различного оборудования - свитчи, роутеры, сервера, стораджи и т.п. У большинства этого оборудования есть отдельный выделенный физический порт для управления - management port. Эти порты предназначены для администраторов - с помощью них можно подключаться по веб-интерфейсу, либо по ssh и делать всякие административные задачи. Это позволяет отделить управляющую часть от данных. Условно, если кто-то получит доступ к виртуалке, он не сможет по сети достигаться до самого сервера, через который можно получить доступ над всеми виртуалками. С одной стороны весь административный доступ к различным системам выносится в один vlan - management vlan - что немного сомнительно с точки зрения безопасности. Но с другой стороны мы знаем, что это единственный путь к админке - а значит в эту зону мы не пускаем никого лишнего, кроме администраторов.

В большинстве случаев на оборудовании только 1 административный порт, а значит его не подключаешь к двум свитчам. Но обычно это не проблема - даже если выйдет из строя свитч, это не отразится на пользователях - максимум админы потеряют доступ к управлению. Плюс нередко все эти подключения держат на отдельном свитче - management switch. По хорошему на схеме я должен был роутер3 отдельно подключить к этому свитчу, свитчи отдельно, сервера отдельно - но, свитчи у нас тупые, а на роутере у меня не осталось портов. Поэтому роутер3 я буду подключать через 5 vlan, а сервер1 подключу через отдельный адаптер, как и положено. Роутер1 и джампхост не будем подключать к менеджмент свитчу, так как к ним у нас меньше всего доверия.

```

GNU nano 2.9.8 /etc/ssh/sshd_config

# If you want to change the port on a SELinux system, you have to tell
# SELinux about this change.
# semanage port -a -t ssh_port_t -p tcp #PORTNUMBER
#
#Port 22
#AddressFamily any
#ListenAddress 0.0.0.0
ListenAddress 10.0.5.1
#ListenAddress ::

HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_ecdsa_key
HostKey /etc/ssh/ssh_host_ed25519_key

```

К примеру, у нас на роутер3 целых 4 адреса в 4 сетях. А мы хотим, чтобы управлять им можно было только из менеджмент сети. Для этого в настройках sshd - /etc/ssh/sshd_config - выставим в качестве ListenAddress только адрес из Management влана:

```

...
ListenAddress 10.0.5.1
...

```

И перезапустим sshd:

```
sudo systemctl restart sshd
```

```

[user@router3 ~]$ ssh 10.0.4.1
ssh: connect to host 10.0.4.1 port 22: Connection refused
[user@router3 ~]$ ssh 10.0.5.1
The authenticity of host '10.0.5.1 (10.0.5.1)' can't be established.
ECDSA key fingerprint is SHA256:xicgilYUv0x54V+1+Ur/dysJQG0XuWuHohWHPVfMJC5.
Are you sure you want to continue connecting (yes/no/[fingerprint])? ^C
[user@router3 ~]$ 

```

Для проверки можем подрубить к себе же. По адресу 4.1:

```
ssh 10.0.4.1
```

сразу видим ошибку. А по адресу 5.1:

```
ssh 10.0.5.1
```

можно понять, что подключение работает.

```
[user@router3 ~]$ sudo firewall-cmd --zone=management --add-service=ssh --permanent
success
[user@router3 ~]$ sudo firewall-cmd --reload
success
[user@router3 ~]$ sudo firewall-cmd --list-all --zone=management
management (active)
  target: default
  icmp-block-inversion: no
  interfaces: team1.5
  sources:
  services: ssh
  ports:
  protocols:
  forward: no
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
[user@router3 ~]$
```

Не забудем в зоне management разрешить сервис ssh:

```
sudo firewall-cmd --zone=management --add-service=ssh --permanent
sudo firewall-cmd --reload
sudo firewall-cmd --list-all --zone=management
```

Ну и перезапустить фаервол. По хорошему стоит убрать из других зон ssh, но всё равно в других зонах нет активных интерфейсов с айпи адресом. А там где есть - не разрешён ssh.

```
[user@router3 ~]$ ssh 10.0.4.101
user@10.0.4.101's password:
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Sat Dec 18 16:01:07 2021 from 10.0.4.1
[user@server1 ~]$ ip ro sh
default via 10.0.4.1 dev team0 proto static metric 350
10.0.4.0/24 dev team0 proto kernel scope link src 10.0.4.101 metric 350
10.0.5.0/24 dev ens5 proto kernel scope link src 10.0.5.101 metric 102
[user@server1 ~]$
```

Теперь подрубимся к серверу1:

```
ssh 10.0.4.101
```

Тут у нас есть тиминг интерфейс для 4 влана:

```
ip ro sh
```

А также 5 влан, подключенный отдельно через интерфейс ens5.

```
[user@server1 ~]$ sudo nano /etc/ssh/sshd_config
[sudo] password for user:
[user@server1 ~]$ sudo grep ListenAddress /etc/ssh/sshd_config
ListenAddress 10.0.5.101
#ListenAddress ::
[user@server1 ~]$ sudo systemctl restart sshd
[user@server1 ~]$
```

Давайте и здесь в sshd заменим ListenAddress на адрес из влана5:

```
...
ListenAddress 10.0.5.101
...
```

и перезапустим sshd:

```
sudo systemctl restart sshd
```

```
[user@server1 ~]$ sudo firewall-cmd --new-zone=management --permanent
success
[user@server1 ~]$ sudo firewall-cmd --change-interface=ens5 --zone=management --permanent
The interface is under control of NetworkManager, setting zone to 'management'.
success
[user@server1 ~]$ sudo firewall-cmd --add-service=ssh --zone=management --permanent
success
[user@server1 ~]$ sudo firewall-cmd --remove-service=ssh --permanent
success
[user@server1 ~]$ sudo firewall-cmd --reload
success
[user@server1 ~]$ sudo firewall-cmd --list-services
cockpit dhcpv6-client
[user@server1 ~]$ sudo firewall-cmd --list-interfaces
ens4 ens3 team0
[user@server1 ~]$ sudo firewall-cmd --list-services --zone=management
ssh
[user@server1 ~]$ sudo firewall-cmd --list-interfaces --zone=management
ens5
[user@server1 ~]$
```

Ну и сделаем изменения на файрволе.

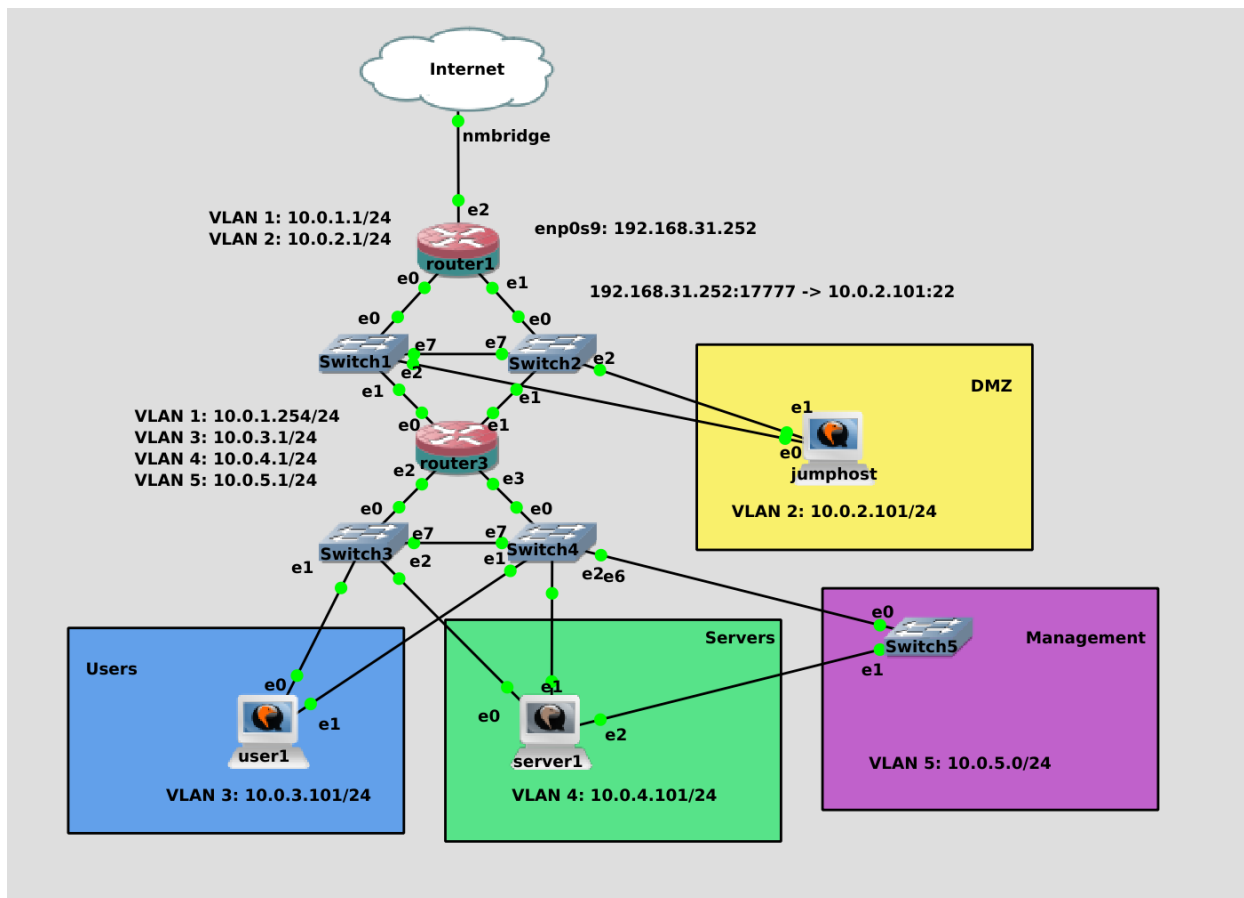
Во-первых, создадим новую зону management, переместим туда интерфейс и разрешим там ssh:

```
sudo firewall-cmd --new-zone=management --permanent
sudo firewall-cmd --change-interface=ens5 --zone=management --permanent
sudo firewall-cmd --add-service=ssh --zone=management --permanent
```

При этом, из дефолтной зоны, т.е. из public, убьём ssh и перезапустим firewall:

```
sudo firewall-cmd --remove-service=ssh --permanent
sudo firewall-cmd --reload
sudo firewall-cmd --list-services
sudo firewall-cmd --list-interfaces
sudo firewall-cmd --list-services --zone=management
sudo firewall-cmd --list-interfaces --zone=management
```

То что мы сейчас делаем на уровне виртуалки - не имеет смысла. Но просто представьте, что это физический сервер, у него на одних физических портах ходят данные, а через выделенный - management - мы управляем. И то что мы вынесли ssh на отдельный интерфейс и по файрволу настроили к нему доступ - это лишь условность. Но и из этой условности можно вынести что-то полезное - по крайней мере мы учимся работать с зонами.



И так, мы разделили нашу сеть на зоны. Где-то остались мелочи, условно, убрать лишние сервисы и интерфейсы с зон на роутерах и конечных устройствах.

```
user@server1:~  
[doctor@tardis]~  
$ ssh jumphost  
Enter passphrase for key '/home/doctor/.ssh/jumphost':  
Activate the web console with: systemctl enable --now cockpit.socket  
  
Last login: Sat Dec 18 16:27:20 2021 from 192.168.31.227  
[user@jumphost ~]$ ping 10.0.3.101  
PING 10.0.3.101 (10.0.3.101) 56(84) bytes of data.  
64 bytes from 10.0.3.101: icmp_seq=1 ttl=62 time=4.34 ms  
^C  
--- 10.0.3.101 ping statistics ---  
1 packets transmitted, 1 received, 0% packet loss, time 0ms  
rtt min/avg/max/mdev = 4.335/4.335/4.335/0.000 ms  
[user@jumphost ~]$ ssh 10.0.3.101  
ssh: connect to host 10.0.3.101 port 22: No route to host  
[user@jumphost ~]$
```

Все наши соединения сейчас держатся благодаря тому, что изменения на фаерволе не влияют на текущие ssh сессии. Если мы попытаемся заново подключиться:

```
ssh jumphost
```

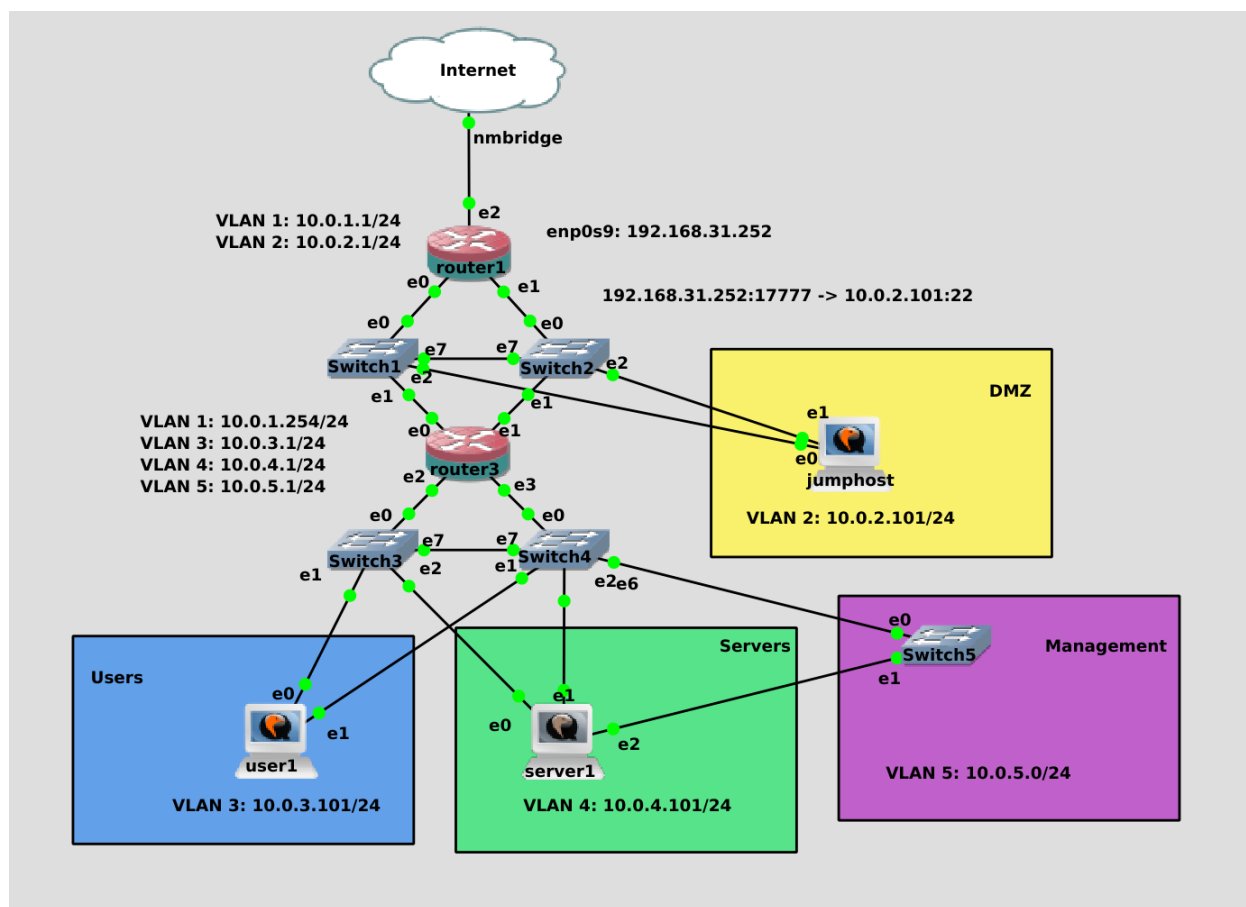
хоть пинг и будет работать:

```
ping 10.0.3.101
```

но ssh и любые другие соединения работать не будут:

```
ssh 10.0.3.101
```

Просто потому что сетевой фаервол блокирует трафик между зонами.

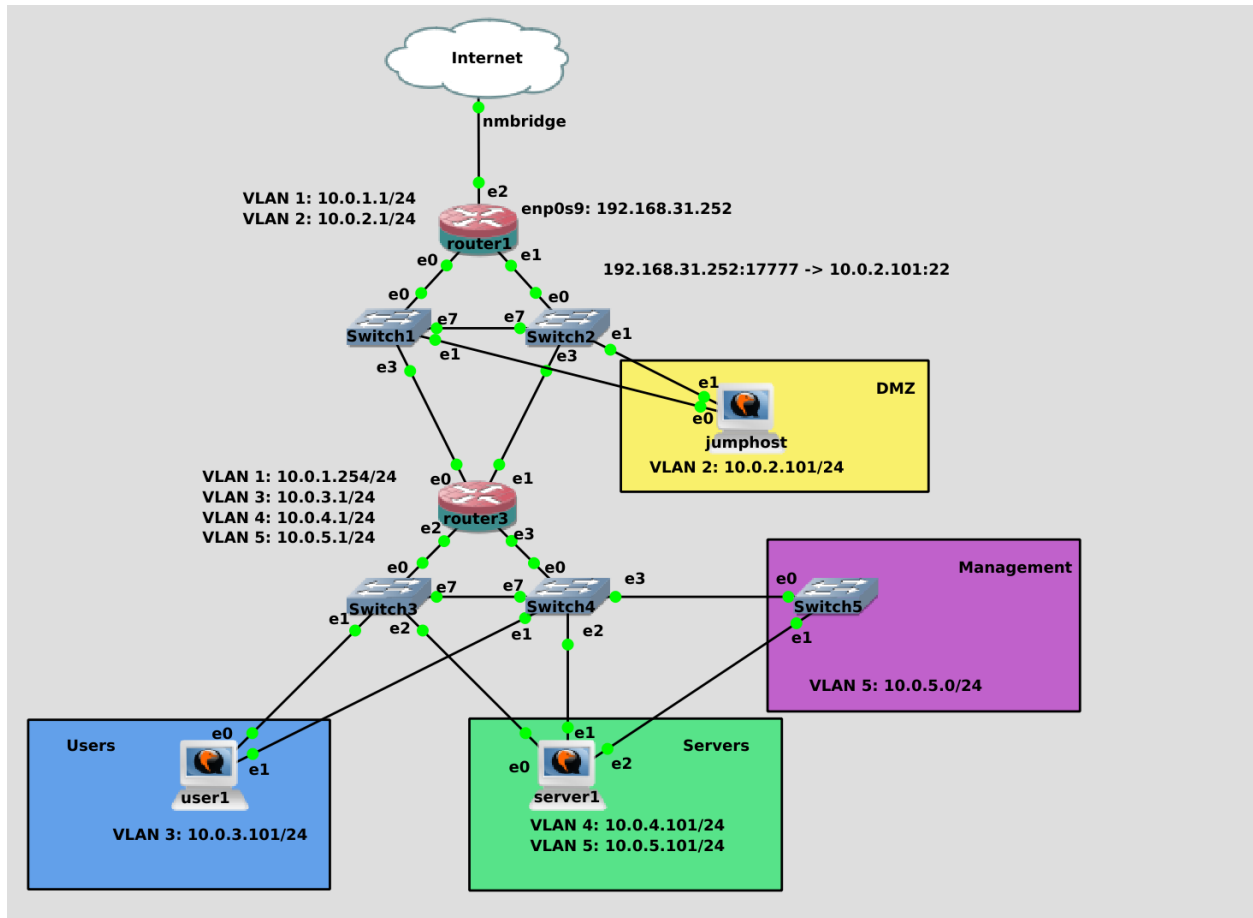


А вот как гранулярно разрешить трафик между зонами и как с джампхоста попасть на сервер1 по ssh - поговорим в следующий раз. Информации и так получилось много, схема стала комплекснее, но ничего страшного в ней нет. Постарайтесь построить такую же сеть, всё должно пинговать друг друга, за исключением адреса 5.101. Почему так - разберём в следующий раз.

Давайте подведём итоги. Сегодня мы с вами поговорили про безопасность, затронули необходимость внутреннего и внешнего файрволов, разделили нашу сеть на зоны, создали DMZ и добавили несколько вланов и подсетей.

2.8 08. Политики Firewall

2.8.1 08. Политики Firewall



В прошлый раз мы поделили нашу сеть на зоны, тем самым перекрыв все доступы. Сегодня же мы разберёмся, как точно выдавать эти доступы. В реальной инфраструктуре этим обычно занимается отдел информационной безопасности, но мы тоже неспроста это разбираем. Обычно, когда администратор настраивает и связывает какие-то сервера в разных зонах, он должен обратиться к отделу безопасности и сообщить, что ему нужно. Мол, у меня есть джампхост во втором влане с IP 10.0.2.101 и нужно, чтобы он мог подключаться к серверу1 по адресу 10.0.5.101 и по порту 22. Т.е. запрашивать доступы на файрволе - это стандартная процедура для администратора. И чтобы закрепить это у себя в голове и понимать, когда это нужно, а когда нет - мы сами будем по необходимости прописывать эти правила. Да, в реальной инфраструктуре никто это не делает через firewall, но нам главное понять саму идею, зачем и как это делать. А этот инструмент или другой - абсолютно неважно.

```

[docto@router1 ~]# firewall-cmd --list-all --zone=dmz
dmz (active)
  target: default
  icmp-block-inversion: no
  interfaces: team0.2
  sources:
  services:
  ports:
  protocols:
  forward: no
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
[docto@router1 ~]# firewall-cmd --add-service=ssh --zone=dmz
success
[docto@router1 ~]# _

```

Сегодня мы будем работать с роутером 1 и 3. В прошлый раз мы запретили с jumphost-а подключаться к роутеру1:

```
firewall-cmd --list-all --zone=dmz
```

И чтобы не настраивать всё через консоль, временно разрешим ssh:

```
firewall-cmd --add-service=ssh --zone=dmz
```

```

[docto@tardis]~$ eval $(ssh-agent)
Agent pid 30200
[docto@tardis]~$ ssh-add .ssh/jumphost .ssh/servers
Enter passphrase for .ssh/jumphost:
Identity added: .ssh/jumphost (docto@tardis)
Identity added: .ssh/servers (docto@tardis)
[docto@tardis]~$ ssh router1
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Sat Dec 25 09:41:52 2021 from 10.0.2.101
[user@router1 ~]$

```

Стартаем ssh-агент:

```
eval $(ssh-agent)
```

добавим ключи:

```
ssh-add .ssh/jumphost .ssh/servers
```

И подружимся к роутеру:

```
ssh router1
```

```
[user@router1 ~]$ sudo firewall-cmd --list-all
[sudo] password for user:
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: ens3 ens4 team0
  sources:
  services: cockpit dhcpv6-client ssh
  ports:
  protocols:
  forward: no
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

В прошлый раз на роутере 3 мы создали зону routers, а на роутере 1 всё осталось в public зоне:

```
sudo firewall-cmd --list-all
```

```
[user@router1 ~]$ sudo firewall-cmd --new-zone=routers --permanent
success
[user@router1 ~]$ sudo firewall-cmd --change-interface=team0 --zone=routers --permanent
The interface is under control of NetworkManager, setting zone to 'routers'.
success
[user@router1 ~]$ sudo firewall-cmd --add-service=ssh --zone=routers --permanent
success
[user@router1 ~]$ sudo firewall-cmd --reload
success
[user@router1 ~]$ sudo firewall-cmd --list-all --zone=routers
routers (active)
  target: default
  icmp-block-inversion: no
  interfaces: team0
  sources:
  services: ssh
  ports:
  protocols:
  forward: no
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
[user@router1 ~]$
```

Давайте это исправим. Для начала создадим зону routers:

```
sudo firewall-cmd --new-zone=routers --permanent
```

Переместим в неё интерфейс, который в одной сети с роутером 3:

```
sudo firewall-cmd --change-interface=team0 --zone=routers --permanent
```

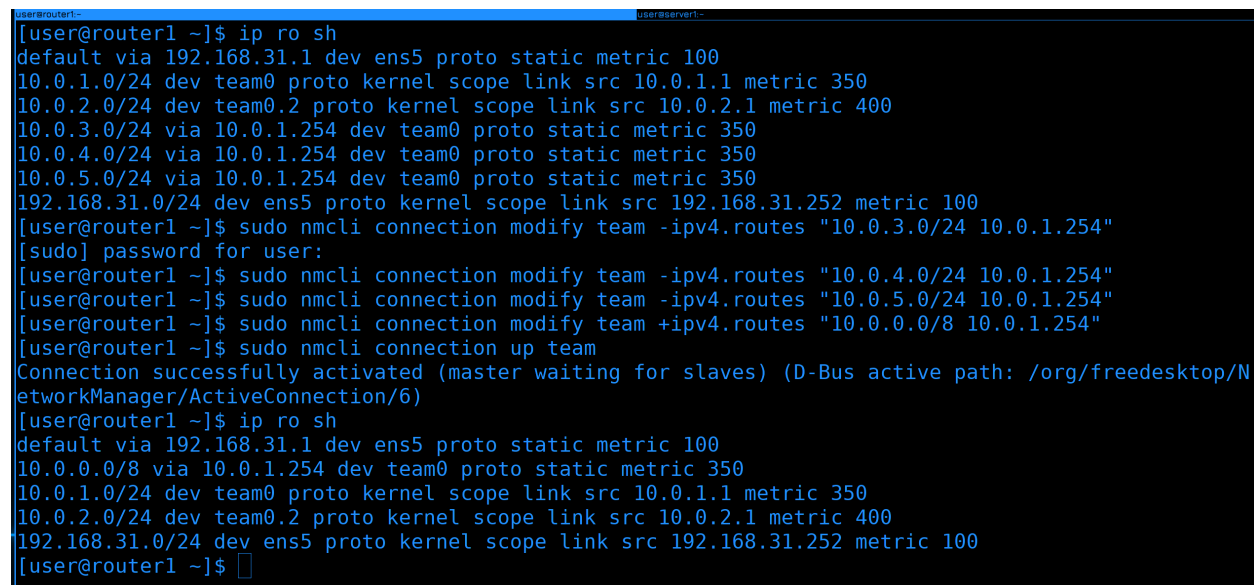
Добавим в эту зону ssh, чтобы мы могли с роутера 3 подключаться к роутеру 1:

```
sudo firewall-cmd --add-service=ssh --zone=routers --permanent
```

Потому что с джампхоста нельзя, а иначе только через консоль. Перезапустим файрвол и проверим новую зону:

```
sudo firewall-cmd --reload
sudo firewall-cmd --list-all --zone=routers
```

Всё как и должно быть.



```

[user@router1 ~]$ ip ro sh
default via 192.168.31.1 dev ens5 proto static metric 100
10.0.1.0/24 dev team0 proto kernel scope link src 10.0.1.1 metric 350
10.0.2.0/24 dev team0.2 proto kernel scope link src 10.0.2.1 metric 400
10.0.3.0/24 via 10.0.1.254 dev team0 proto static metric 350
10.0.4.0/24 via 10.0.1.254 dev team0 proto static metric 350
10.0.5.0/24 via 10.0.1.254 dev team0 proto static metric 350
192.168.31.0/24 dev ens5 proto kernel scope link src 192.168.31.252 metric 100
[user@router1 ~]$ sudo nmcli connection modify team -ipv4.routes "10.0.3.0/24 10.0.1.254"
[sudo] password for user:
[user@router1 ~]$ sudo nmcli connection modify team -ipv4.routes "10.0.4.0/24 10.0.1.254"
[user@router1 ~]$ sudo nmcli connection modify team -ipv4.routes "10.0.5.0/24 10.0.1.254"
[user@router1 ~]$ sudo nmcli connection modify team +ipv4.routes "10.0.0.0/8 10.0.1.254"
[user@router1 ~]$ sudo nmcli connection up team
Connection successfully activated (master waiting for slaves) (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/6)
[user@router1 ~]$ ip ro sh
default via 192.168.31.1 dev ens5 proto static metric 100
10.0.0.0/8 via 10.0.1.254 dev team0 proto static metric 350
10.0.1.0/24 dev team0 proto kernel scope link src 10.0.1.1 metric 350
10.0.2.0/24 dev team0.2 proto kernel scope link src 10.0.2.1 metric 400
192.168.31.0/24 dev ens5 proto kernel scope link src 192.168.31.252 metric 100
[user@router1 ~]$

```

Пока не забыл. Недавно мы на роутере1 прописали 3 маршрута:

```
ip ro sh
```

3, 4 и 5 сети. Но если у нас количество сетей будет постоянно расти, чтобы не забывать их добавлять, давайте поменяем подход. Уберём эти 3 сети:

```
sudo nmcli connection modify team -ipv4.routes "10.0.3.0/24 10.0.1.254"
sudo nmcli connection modify team -ipv4.routes "10.0.4.0/24 10.0.1.254"
sudo nmcli connection modify team -ipv4.routes "10.0.5.0/24 10.0.1.254"
```

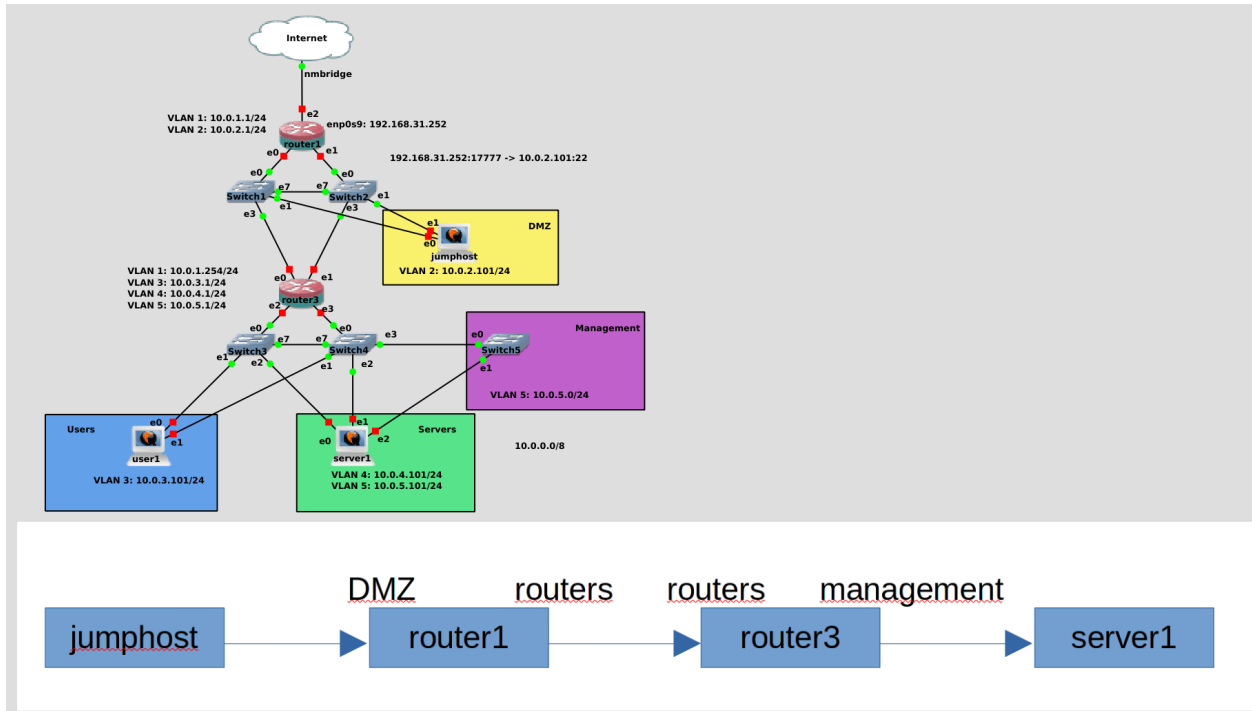
и вместо них пропишем всю подсеть 10.0.0.0/8 - сюда входит всё от 10.0.0.0 до 10.255.255.255.

```
sudo nmcli connection modify team +ipv4.routes "10.0.0.0/8 10.0.1.254"
```

Затем поднимем профиль и проверим таблицу маршрутизации:

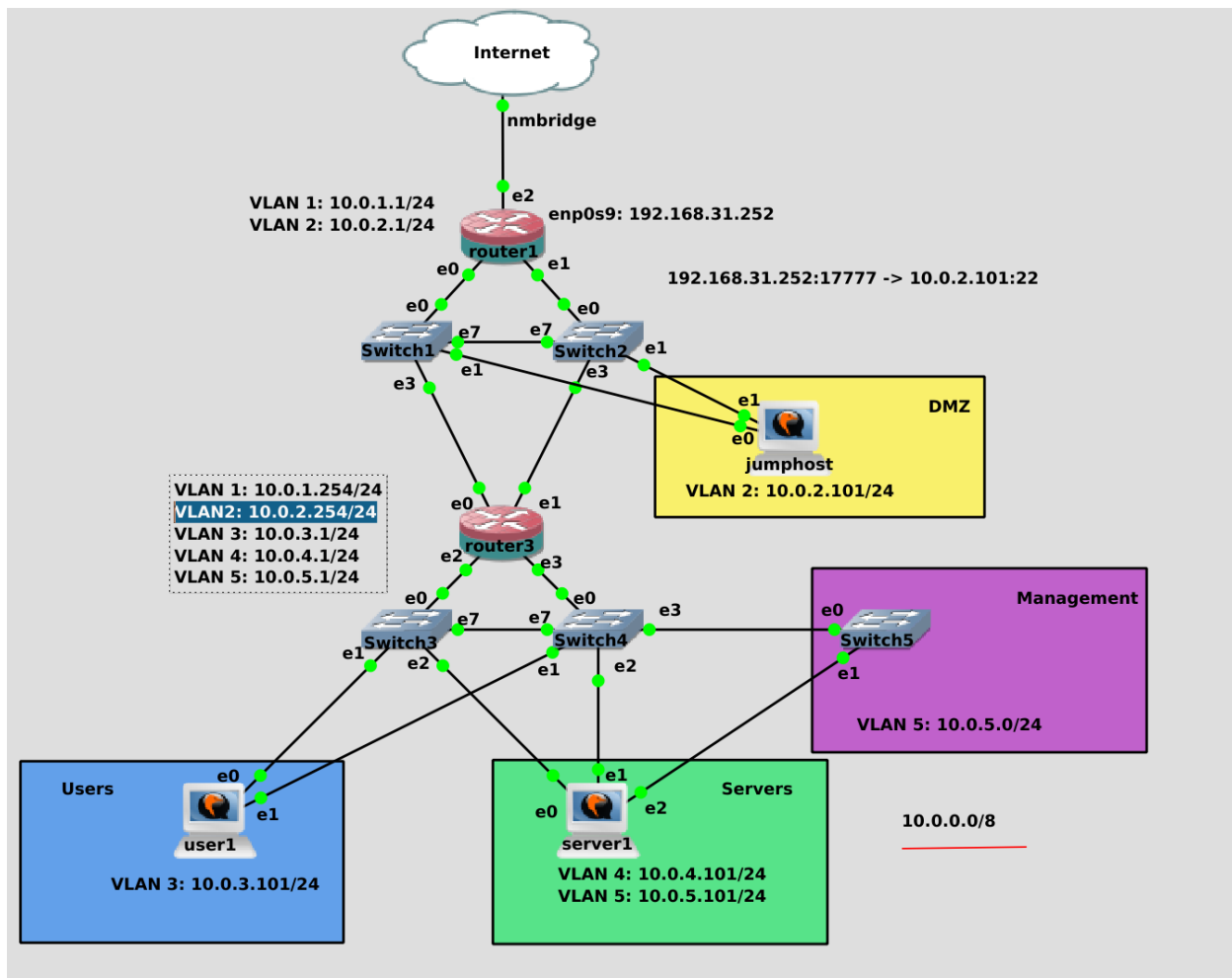
```
sudo nmcli connection up team
ip ro sh
```

Теперь всё, что касается локальной сети, будет пересылаться на третий роутер.



Давайте представим себе схему подключения jump host-а к server1. Изначально пакет идёт от jump host-а к router1. Для router1 jump host находится в зоне DMZ. Дальше router1 передаёт пакет на router3. Для router1 router3 находится в зоне routers. Тоже самое для router3 - router1 находится в зоне routers.

Что касается router3, в нашей схеме сейчас трафик из DMZ зоны, чтобы попасть во внутреннюю сеть, должен будет проходить сначала через router1, и только потом через роутер3. А это значит, что мы будем перегонять лишний трафик через роутер1. Поэтому давайте лучше добавим к роутеру3 второй влан, чтобы устройства из DMZ напрямую проходили через роутер3. Я пропущу добавление второго влана на роутере 3 и настройку транк портов на свитчах, мы это уже разбирали.



На роутере 3 я создал новый влан интерфейсы с адресом 10.0.2.254 и добавил его в зону DMZ.

```

[user@jumphost ~]$ sudo nmcli connection modify team +ipv4.routes "10.0.0.0/8 10.0.2.254"
[sudo] password for user:
[user@jumphost ~]$ sudo nmcli connection up team
Connection successfully activated (master waiting for slaves) (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/4)
[user@jumphost ~]$ ip ro sh
default via 10.0.2.1 dev team0 proto static metric 350
10.0.0.0/8 via 10.0.2.254 dev team0 proto static metric 350
10.0.2.0/24 dev team0 proto kernel scope link src 10.0.2.101 metric 350
[user@jumphost ~]$

```

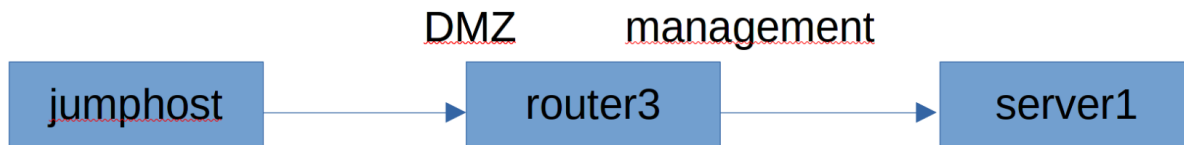
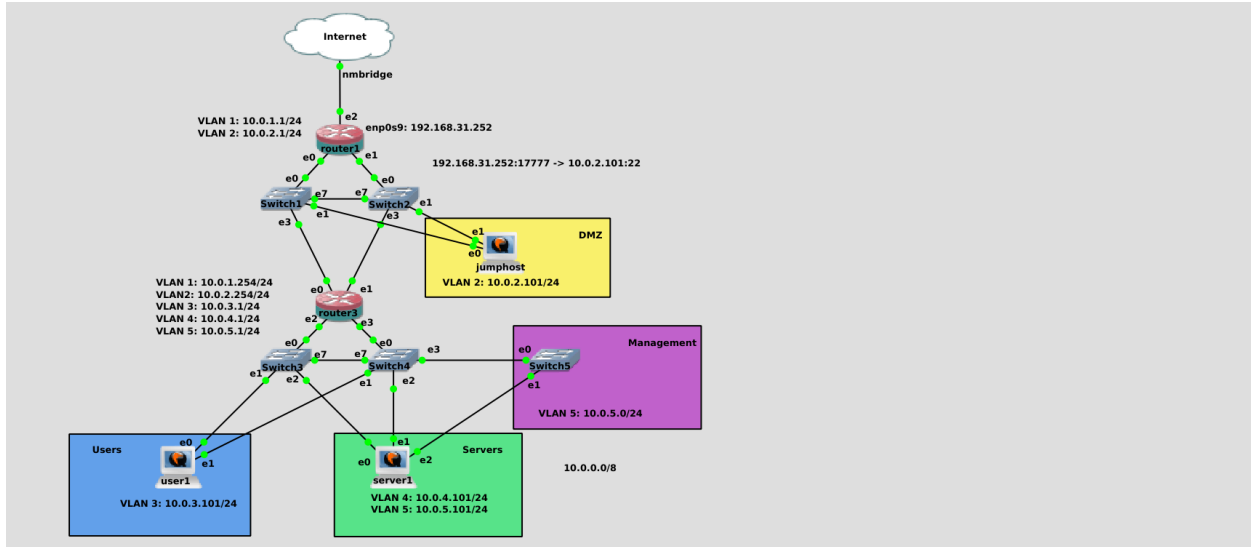
На самом jumphost-е, чтобы он направлял все пакеты во внутреннюю сеть через адрес третьего роутера, пропишем новый маршрут:

```
sudo nmcli connection modify team +ipv4.routes "10.0.0.0/8 10.0.2.254"
```

Затем поднимем интерфейс и проверим таблицу маршрутизации:

```
sudo nmcli connection up team
ip ro sh
```

Тут мы видим, что все пакеты, идущие в сеть 10.0.0.0/8 будут проходить через роутер3. А все другие адреса - через роутер 1.



Теперь подключение от jumpost-а на сервер выглядит попроще - посреди только роутер3. Jumpost для роутера3 находится в зоне DMZ, а сервер1 в зоне management. Точнее в двух зонах - servers и management, но нас сейчас интересует только вторая зона.

```

root@server1 ~# 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: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel master team0 state UP group default qlen 1000
    link/ether 0c:fe:ef:81:00:00 brd ff:ff:ff:ff:ff:ff
3: ens4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel master team0 state UP group default qlen 1000
    link/ether 0c:fe:ef:81:00:00 brd ff:ff:ff:ff:ff:ff permaddr 0c:fe:ef:81:00:01
4: ens5: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 0c:fe:ef:81:00:02 brd ff:ff:ff:ff:ff:ff
    inet 10.0.5.101/24 brd 10.0.5.255 scope global noprefixroute ens5
        valid_lft forever preferred_lft forever
    inet6 fe80::eb54:24af:5016:2b1/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
5: ens6: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state DOWN group default qlen 1000
    link/ether 0c:fe:ef:81:00:03 brd ff:ff:ff:ff:ff:ff
6: team0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 0c:fe:ef:81:00:00 brd ff:ff:ff:ff:ff:ff
    inet 10.0.4.101/24 brd 10.0.4.255 scope global noprefixroute team0
        valid_lft forever preferred_lft forever
    inet6 fe80::5a5c:65ed:ffc5:8508/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
root@server1 ~# tcpdump -i ens5
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ens5, link-type EN10MB (Ethernet), capture size 262144 bytes

```

Подключимся к серверу1. К нему по ssh можно подружиться только с 5 влана:

```
ip a
```

А он у нас на интерфейсе ens5. Давайте посмотрим tcpdump на этом интерфейсе:

```
tcpdump -i ens5
```

```

user@router1- user@jumphost-
[user@jumphost ~]$ ping 10.0.5.101
PING 10.0.5.101 (10.0.5.101) 56(84) bytes of data.
^C
--- 10.0.5.101 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1005ms

[user@jumphost ~]$ ssh 10.0.5.101
ssh: connect to host 10.0.5.101 port 22: No route to host
[user@jumphost ~]$

```

Теперь идём на jumphost и пытаемся пингануть:

```
ping 10.0.5.101
```

Пинга нет. Но дело не в файрволе, он у нас все пинги по-умолчанию разрешает. Также не получается подключиться по ssh:

```
ssh 10.0.5.101
```

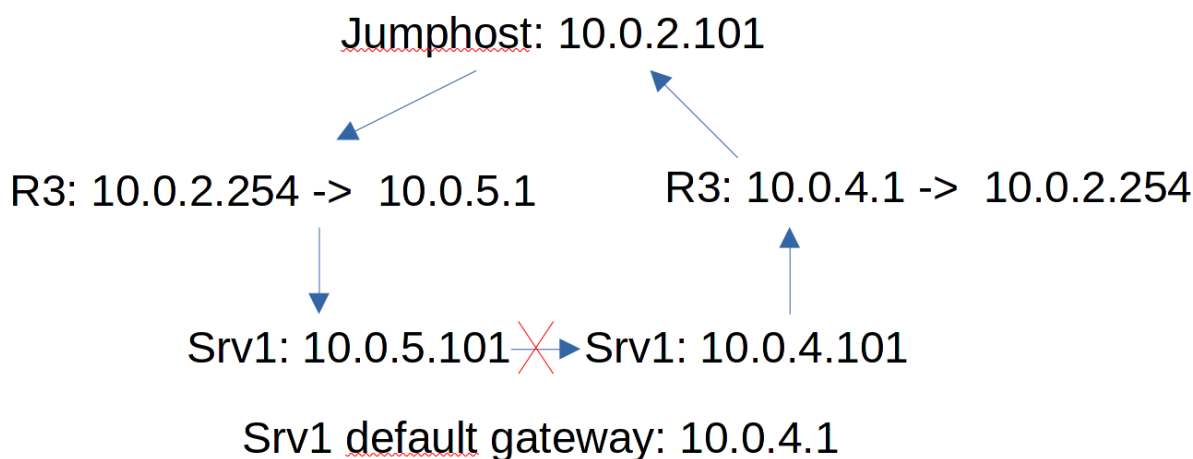
Но это ещё понятно, тут дело в файрволе. К нему мы ещё придём, а пока попытаемся разобраться, почему server1 не пингуется.

```

[root@server1 ~]# tcpdump -i ens5
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ens5, link-type EN10MB (Ethernet), capture size 262144 bytes
12:34:37.725445 IP 10.0.2.101 > server1: ICMP echo request, id 1966, seq 1, length 64
12:34:38.771275 IP 10.0.2.101 > server1: ICMP echo request, id 1966, seq 2, length 64
12:34:54.469488 IP 10.0.2.101 > server1: ICMP echo request, id 1967, seq 1, length 64
12:34:55.475398 IP 10.0.2.101 > server1: ICMP echo request, id 1967, seq 2, length 64
12:34:59.697708 ARP, Request who-has server1 tell 10.0.5.1, length 46
12:34:59.697738 ARP, Reply server1 is-at 0c:fe:ef:81:00:02 (oui Unknown), length 28
-

```

Идём и смотрим tcpdump. Как видите, пакетов по ssh не было, но вот пинги приходили. Но на них не было ответа. Почему?



Есть одна особенность, когда у вас в системе несколько IP адресов, как на нашем сервере1. У него два адреса - 10.0.4.101 и 10.0.5.101. А вот default gateway всего 1 - 10.0.4.1. И вот представьте - джампхост

посылает пинг на адрес 10.0.5.101. Сервер1 должен ответить на адрес 10.0.2.101. Но у него в таблице маршрутизации сказано, что этот адрес доступен за адресом 10.0.4.1 - а значит ответ надо посылать с другого интерфейса, на котором IP - 10.0.4.101. Т.е. пакет приходит с одного интерфейса, а ответить надо с другого. С одной стороны, мы могли бы для адреса 10.0.2.101 прописать маршрут через 10.0.5.1. Но тогда бы возникла обратная проблема - если запрос будет приходить на 4.101, то ответ будет посылаться с 5.101 - что приведёт к такой же проблеме.

- RHEL 6 and above are configured by default to apply Strict Reverse Path Forwarding filtering recommended in [RFC 3704 - Ingress Filtering for Multihomed Networks](#).
- Strict filtering means that when a packet arrives on the system, the kernel takes the source IP of the packet and makes a lookup of its routing table to see if the interface the packet arrived on is the same interface the kernel would use to send a packet to that IP. If the interfaces are the same then the packet has passed the strict filtering test and it is processed normally. If the interfaces are not the same then the packet is discarded without any further processing and in RHEL 7+, the `IPReversePathFilter` counter is incremented.
- The main effect of strict filtering is that for a given remote IP, the system will only communicate with it via a specific interface. Set up static routes to control which interface responds to a given remote IP or network.
- The filtering method is controlled globally by the `sysctl net.ipv4.conf.all.rp_filter` described in the kernel documentation: <https://www.kernel.org/doc/Documentation/networking/ip-sysctl.txt>. RHEL 6+ override the kernel default value of **0 (disabled)** for this parameter and set it to **1 (strict)**.

Raw

```
rp_filter - INTEGER
0 - No source validation.
1 - Strict mode as defined in RFC3704 Strict Reverse Path
  Each incoming packet is tested against the FIB and if the interface
  is not the best reverse path the packet check will fail.
  By default failed packets are discarded.
2 - Loose mode as defined in RFC3704 Loose Reverse Path
  Each incoming packet's source address is also tested against the FIB
  and if the source address is not reachable via any interface
  the packet check will fail.
```

Обычно так не должно быть. Поэтому ядро по-умолчанию блокирует такие ответы. Т.е. если пакет приходит с одного интерфейса, а ответ нужно послать с другого - то ядро не разрешает. В целом, конечно, можно разрешить, но с точки зрения маршрутизации это не совсем верно. Есть более правильный способ. Нам надо для интерфейса, где адрес 5.101 прописать свою таблицу маршрутизации.

```
[user@jumphost ~]$ ip route show
default via 10.0.2.1 dev team0 proto static metric 350
10.0.0.0/8 via 10.0.2.254 dev team0 proto static metric 350
10.0.2.0/24 dev team0 proto kernel scope link src 10.0.2.101 metric 350
[user@jumphost ~]$
[user@jumphost ~]$ ip rule show
0:      from all lookup local
32766:  from all lookup main
32767:  from all lookup default
[user@jumphost ~]$
[user@jumphost ~]$ ip route show table main
default via 10.0.2.1 dev team0 proto static metric 350
10.0.0.0/8 via 10.0.2.254 dev team0 proto static metric 350
10.0.2.0/24 dev team0 proto kernel scope link src 10.0.2.101 metric 350
[user@jumphost ~]$
[user@jumphost ~]$ ip route show table local
broadcast 10.0.2.0 dev team0 proto kernel scope link src 10.0.2.101
local 10.0.2.101 dev team0 proto kernel scope host src 10.0.2.101
broadcast 10.0.2.255 dev team0 proto kernel scope link src 10.0.2.101
broadcast 127.0.0.0 dev lo proto kernel scope link src 127.0.0.1
local 127.0.0.0/8 dev lo proto kernel scope host src 127.0.0.1
local 127.0.0.1 dev lo proto kernel scope host src 127.0.0.1
broadcast 127.255.255.255 dev lo proto kernel scope link src 127.0.0.1
[user@jumphost ~]$
```

Для начала разберёмся с таблицами и правилами маршрутизации. Вкратце, есть таблица маршрутизации:

```
ip route sh
```

Ничего такого. Есть ещё правила маршрутизации:

```
ip rule show
```

Числа слева - приоритет. Чем ближе к 0 - тем он выше. Дальше «от кого» - from all. Тут мы указываем исходящую сеть. И потом lookup и 3 значения - local, main и default. Это и есть таблицы маршрутизации. Та таблица маршрутизации, которую мы видим при вводе `ip ro sh` - одна из них - main. Можем убедиться в этом введя команду:

```
ip route show table main
```

Есть ещё local:

```
ip route show table local
```

Здесь, в основном, про локальные сети.

Т.е. мы можем создать несколько таблиц маршрутизации и можем написать, что для запросов с таких-то сетей нужно использовать такую-то таблицу маршрутизации, при этом учитывая ещё и приоритет. Чтобы было понятнее, сделаем это на сервере1.

```

root@server1 ~# nmcli connection modify management ipv4.routes "10.0.0.0/8 10.0.5.1 table=2" ipv4.routing-rules "priority 5 fr
om 10.0.5.101 table 2"
root@server1 ~# nmcli connection up management
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/17)
root@server1 ~# ip rule list
0:      from all lookup local
5:      from 10.0.5.101 lookup 2
32766:  from all lookup main
32767:  from all lookup default
root@server1 ~# ip route show table 2
10.0.0.0/8 via 10.0.5.1 dev ens5 proto static metric 103
root@server1 ~#

```

Добавим для интерфейса ens5 в профиле NetworkManager-а новый маршрут, при этом указав таблицу с каким-то числом, допустим, 2:

```
nmcli con mod management ipv4.routes "10.0.0.0/8 10.0.5.1 table=2" ipv4.routing-rules
↪ "priority 5 from 10.0.5.101 table 2"
```

И так, мы в профиле интерфейса ens5 сказали, что для сети 10.0.0.0/8 нужно использовать в качестве гейтвея 10.0.5.1, при этом это относится к таблице маршрутизации 2. И сразу же добавили правило с приоритетом 5, исходящим IP 10.0.5.101 и этой таблицей 2.

Поднимем интерфейс и посмотрим правила и таблицу маршрутизации:

```
nmcli con up management
ip rule list
ip route show table 2
```

Как видите, теперь у нас есть правило с приоритетом 5, которое указывает на таблицу 2, где и указан маршрут для локальной сети. Теоретически, вместо 10.0.0.0/8 можно было бы указать 0.0.0.0/0, чтобы у каждого интерфейса был свой default gateway, но так как речь касается management интерфейса, я не хочу, чтобы у него был маршрут в интернет.

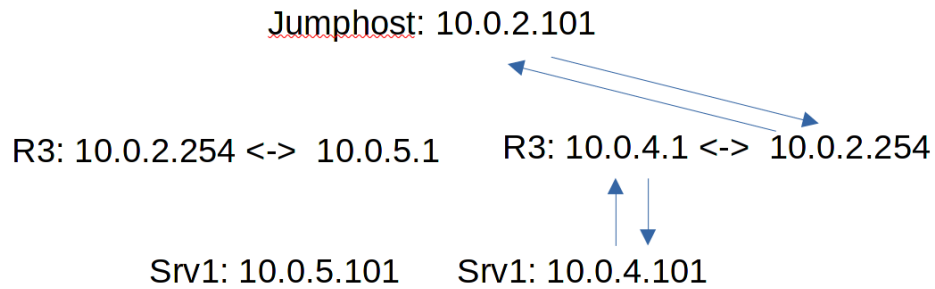
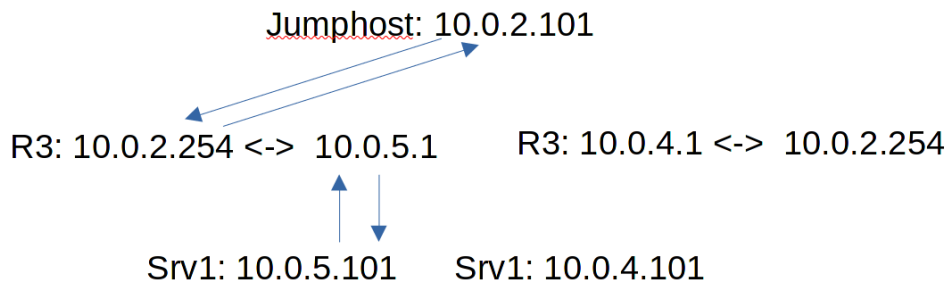
```

user@router1:-
user@jumphost:-
[user@jumphost ~]$ ping 10.0.4.101
PING 10.0.4.101 (10.0.4.101) 56(84) bytes of data.
64 bytes from 10.0.4.101: icmp_seq=1 ttl=63 time=1.01 ms
^C
--- 10.0.4.101 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.009/1.009/1.009/0.000 ms
[user@jumphost ~]$ ping 10.0.5.101
PING 10.0.5.101 (10.0.5.101) 56(84) bytes of data.
64 bytes from 10.0.5.101: icmp_seq=1 ttl=63 time=1.18 ms
64 bytes from 10.0.5.101: icmp_seq=2 ttl=63 time=3.20 ms
^C
--- 10.0.5.101 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 1.176/2.186/3.196/1.010 ms
[user@jumphost ~]$

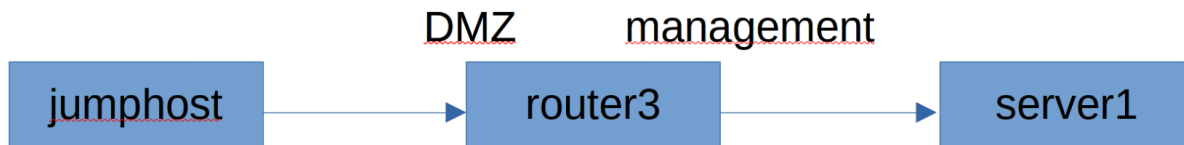
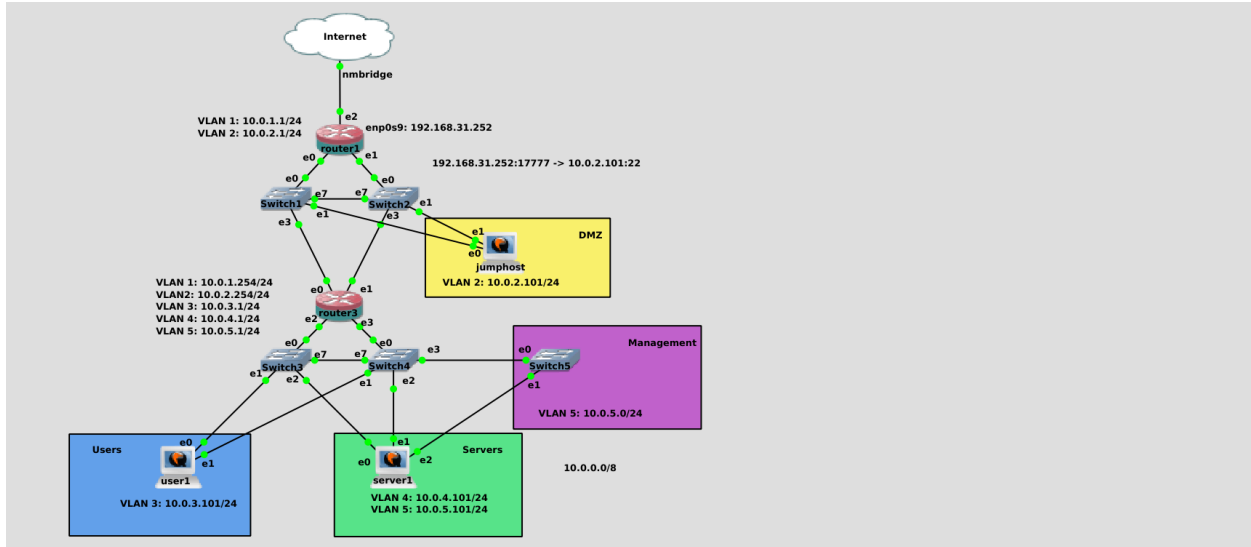
```

Теперь с самого jumphost-а можно пинговать server1 как по адресу 4.101, так и по 5.101:

```
ping 10.0.4.101
ping 10.0.5.101
```

**Table: main****Table: 2**

В итоге получается такая картина: когда запрос приходит на интерфейс с адресом 4.101, то для ответа используется таблица маршрутизации main, в которой default gateway-ем является адрес роутера 4.1. А вот если запрос придёт на адрес 5.101, то будет использоваться таблица маршрутизации 2, в которой есть обратный маршрут через адрес роутера 5.1.



Теперь, когда мы привели в порядок маршрутизацию, вернёмся к файрволу. И так, мы на внутреннем роутере должны разрешить джампхосту подключаться по ssh к серверу1. Для этого нужны политики файрвола.

```

user@router3:~$ sudo firewall-cmd --new-policy=jh_srv1_ssh --permanent
success
[user@router3 ~]$ sudo firewall-cmd --reload
success
[user@router3 ~]$ sudo firewall-cmd --info-policy=jh_srv1_ssh
jh_srv1_ssh
priority: -1
target: CONTINUE
ingress-zones:
egress-zones:
services:
ports:
protocols:
masquerade: no
forward-ports:
source-ports:
icmp-blocks:
rich rules:
[user@router3 ~]$

```

Для начала на роутере3 создадим новую политику:

```
sudo firewall-cmd --new-policy=jh_srv1_ssh --permanent
```

(continues on next page)

(продолжение с предыдущей страницы)

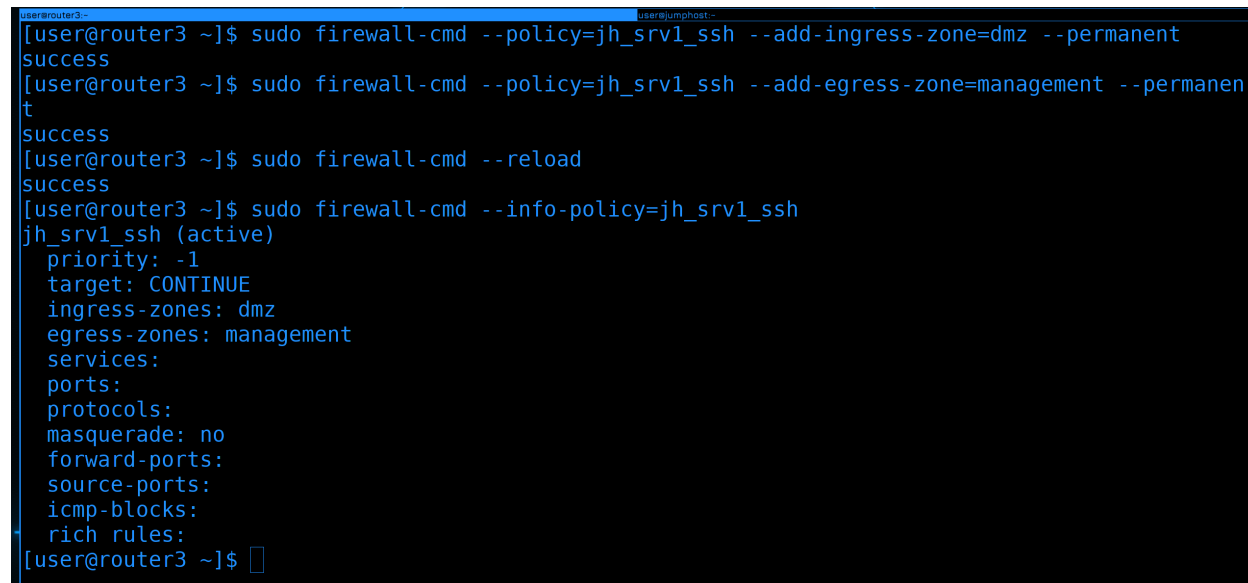
```
sudo firewall-cmd --reload
```

Она должна разрешать подключение от jump-host-а к серверу1 по ssh. Такое название и дадим, чтобы было понятнее.

После создания политика почти пустая:

```
sudo firewall-cmd --info-policy=jh_srv1_ssh
```

Политика сильно напоминает зону. Но если в зоне мы указываем порты и сервисы, по которым могут подключаться к нам, то в политике мы указываем порты и сервисы, которые мы разрешаем в проходящих пакетах от одной зоны к другой. Поэтому здесь есть два дополнительных поля - ingress-zone и egress-zone - входящая зона и исходящая.



```

[user@router3 ~]$ sudo firewall-cmd --policy=jh_srv1_ssh --add-ingress-zone=dmz --permanent
success
[user@router3 ~]$ sudo firewall-cmd --policy=jh_srv1_ssh --add-egress-zone=management --permanent
success
[user@router3 ~]$ sudo firewall-cmd --reload
success
[user@router3 ~]$ sudo firewall-cmd --info-policy=jh_srv1_ssh
jh_srv1_ssh (active)
  priority: -1
  target: CONTINUE
  ingress-zones: dmz
  egress-zones: management
  services:
  ports:
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
[user@router3 ~]$
  
```

Как вы поняли - теперь мы должны добавить ingress и egress зоны в эту политику:

```

sudo firewall-cmd --policy=jh_srv1_ssh --add-ingress-zone=dmz --permanent
sudo firewall-cmd --policy=jh_srv1_ssh --add-egress-zone=management --permanent
sudo firewall-cmd --reload
  
```

Соответственно, подключение будет из зоны dmz, значит ingress zone это dmz, а направляться пакеты будут в зону management - значит это egress зона. Проверим политику:

```
sudo firewall-cmd --info-policy=jh_srv1_ssh
```

Как видите, зоны прописались.

Но так как тут нет возможности добавить source и destination адреса, нам остаётся два варианта. Либо придётся просто прописать сервис, что приведёт к тому, что с любого адреса из зоны dmz можно будет по ssh подрубить к management зоне. А мы этого не хотим.

```

[user@router3 ~]$ sudo firewall-cmd --policy=jh_srv1_ssh --permanent --add-rich-rule='rule family=ipv4 source address=10.0.2.101/32 destination address=10.0.5.101/32 port port=22 protocol=tcp accept'
success
[user@router3 ~]$ sudo firewall-cmd --reload
success
[user@router3 ~]$ sudo firewall-cmd --info-policy=jh_srv1_ssh
jh_srv1_ssh (active)
  priority: -1
  target: CONTINUE
  ingress-zones: dmz
  egress-zones: management
  services:
  ports:
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
    rule family="ipv4" source address="10.0.2.101/32" destination address="10.0.5.101/32" port port="22" protocol="tcp" accept
[user@router3 ~]$

```

Либо придётся писать rich rule, что мы и сделаем.

```

sudo firewall-cmd --policy=jh_srv1_ssh --permanent --add-rich-rule='rule family=ipv4
↪source address=10.0.2.101/32 destination address=10.0.5.101/32 port port=22
↪protocol=tcp accept'

```

И так, здесь мы говорим, что разрешаем подключение от адреса 10.0.2.101 к адресу 10.0.5.101 по порту 22 и протоколу tcp. И это всё применяется на политику с зоны dmz на management.

Перезагрузим файрвол и проверим политику:

```

sudo firewall-cmd --reload
sudo firewall-cmd --info-policy=jh_srv1_ssh

```

Да, появилось правило.

На мой взгляд, здесь политики сделаны немного нелогично. Как минимум нужна возможность добавить source и destination адреса, а не возиться с rich rule-ами. Но это относительно новый функционал у firewalld и не то чтобы кто-то кроме нас использует firewalld как сетевой файрвол. Поэтому на это можно закрыть глаза. У политик есть приоритеты, от которых зависит порядок их проверки, но мы не будем сильно с этим заморачиваться, курс всё таки не об этом.

```

[user@jumphost ~]$ ssh 10.0.5.101
The authenticity of host '10.0.5.101 (10.0.5.101)' can't be established.
ECDSA key fingerprint is SHA256:rupl3FPDkMWQzxvXwrYc2DglAc//LxzgmN8seFjaQaE.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.5.101' (ECDSA) to the list of known hosts.
user@10.0.5.101's password:
Activate the web console with: systemctl enable --now cockpit.socket

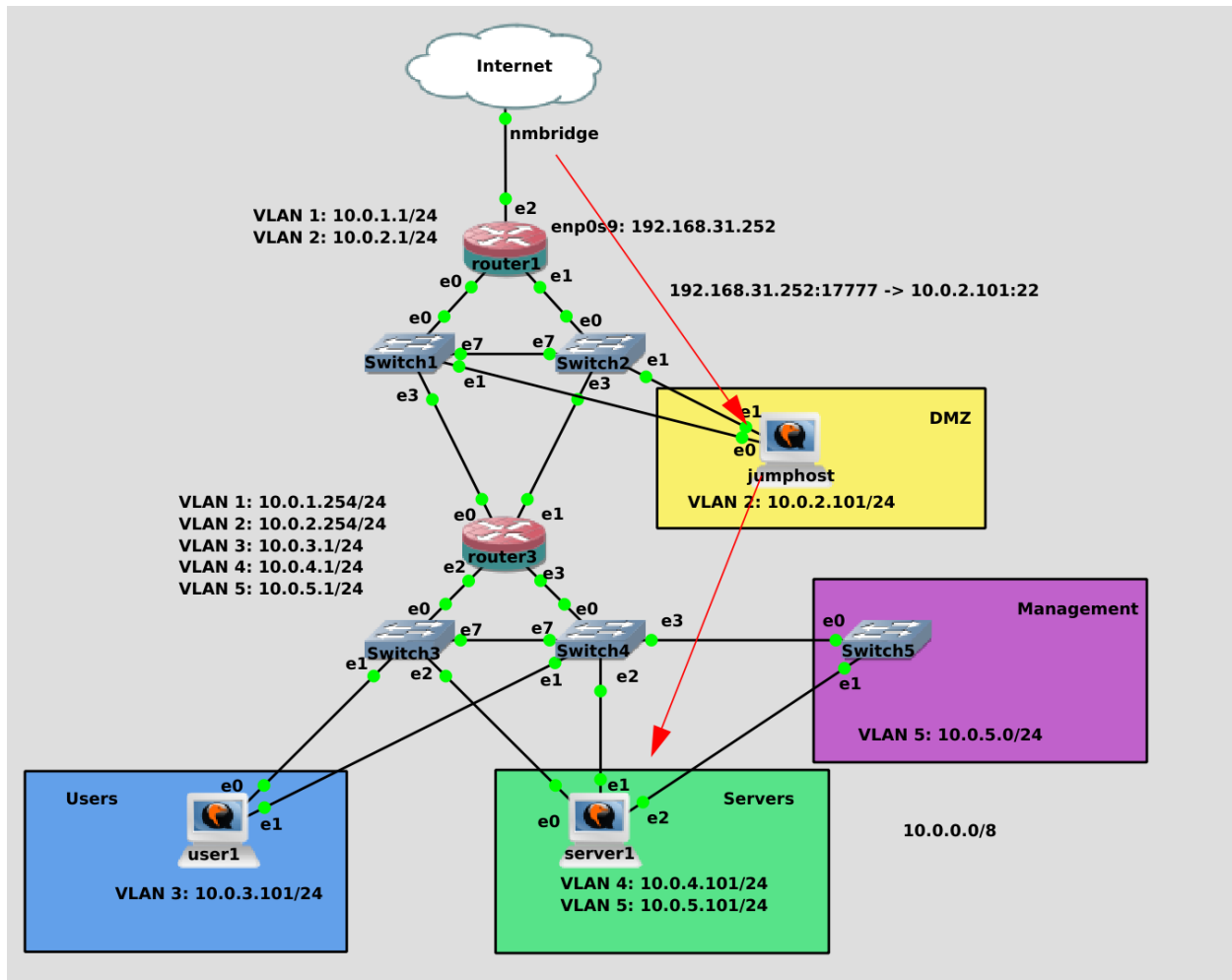
Last login: Sat Dec 18 20:58:09 2021 from 10.0.4.1
[user@server1 ~]$

```

Давайте попробуем с jumphost-а подключиться к серверу1:

```
ssh 10.0.5.101
```

И всё работает.



Теперь у нас есть возможность попасть во внутреннюю сеть, подключаясь от jumhost-а к серверу1. По сути, теперь server1 будет выступать нашим вторым jumhost-ом. Через него есть выход в 5 влан, откуда мы можем подрубить к менеджмент интерфейсам. Например, роутера3.


```
GNU nano 5.9 .ssh/config
Host jumphost
  User user
  IdentityFile ~/.ssh/jumphost
  Hostname 192.168.31.252
  Port 17777

Host server1
  User user
  IdentityFile ~/.ssh/servers
  ProxyJump jumphost
  Hostname 10.0.5.101

Host router3
  User user
  IdentityFile ~/.ssh/servers
  ProxyJump server1
  Hostname 10.0.5.1

Host router1
  User user
  IdentityFile ~/.ssh/servers
  ProxyJump router3
  Hostname 10.0.1.1
```

Теперь смотрите какую цепочку мы можем создать в ssh подключении. И так, из интернета у нас доступен только jumphost. Но сам jumphost не может подключаться куда угодно, единственный доступный ему сервер - сервер1. Сам сервер1 может видеть роутер3, так как они находятся в одном management vlan-е. А из роутера3 я могу подключиться к роутеру1. Последнее, на самом деле, не обязательно. Для роутера 1 вы также можете использовать сервер1 в качестве джамп хоста, но тогда надо будет прописать политику файрвола на роутере3, чтобы он разрешал подключение от сервера1 к роутеру1.

```

[doctor@tardis]~$ ssh router1
The authenticity of host '10.0.5.101 (<no hostip for proxy command>)' can't be established.
ED25519 key fingerprint is SHA256:Jqnmi0nLYx0kh0L2Cg8Zu9uNFb0i+wjYhXvl05y5wzs.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.5.101' (ED25519) to the list of known hosts.
user@10.0.5.101's password:
The authenticity of host '10.0.5.1 (<no hostip for proxy command>)' can't be established.
ED25519 key fingerprint is SHA256:uISKXFhIyZQ8Qv1PR0qG2X2CS008J/kQZ4K0AcAJ+yM.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.5.1' (ED25519) to the list of known hosts.
user@10.0.5.1's password:
The authenticity of host '10.0.1.1 (<no hostip for proxy command>)' can't be established.
ED25519 key fingerprint is SHA256:hzPVkgFw16+xjlLu/XH78WVwGJug6Axjj8qZ7J0s4B8.
This host key is known by the following other names/addresses:
  ~/.ssh/known_hosts:201: 10.0.2.1
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.1.1' (ED25519) to the list of known hosts.
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Sat Dec 25 09:42:38 2021 from 10.0.2.101
[user@router1 ~]$

```

Попробуем подключиться. При первом подключении надо будет сохранить ключи от всех промежуточных хостов и ввести все пароли от систем, куда мы не закинули ключи.

```

[doctor@tardis]~$ ssh router1
user@10.0.5.101's password:
user@10.0.5.1's password:
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Sat Dec 25 17:22:24 2021 from 10.0.1.254
[user@router1 ~]$ logout
Connection to 10.0.1.1 closed.
[doctor@tardis]~$ ssh-copy-id -i .ssh/servers.pub server1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: ".ssh/servers.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install all the new keys
user@10.0.5.101's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'server1'"
and check to make sure that only the key(s) you wanted were added.

[doctor@tardis]~$

```

Больше мы этих сообщений не увидим:

```
ssh router1
```

А на хосты, которые просят пароли, можем закинуть ключи:

```
ssh-copy-id -i .ssh/servers.pub server1
ssh-copy-id -i .ssh/servers.pub router3
```

```

user@router1:~$ ssh router1
[doctor@tardis]~$
[doctor@tardis]~$ ssh router1
Activate the web console with: systemctl enable --now cockpit.socket

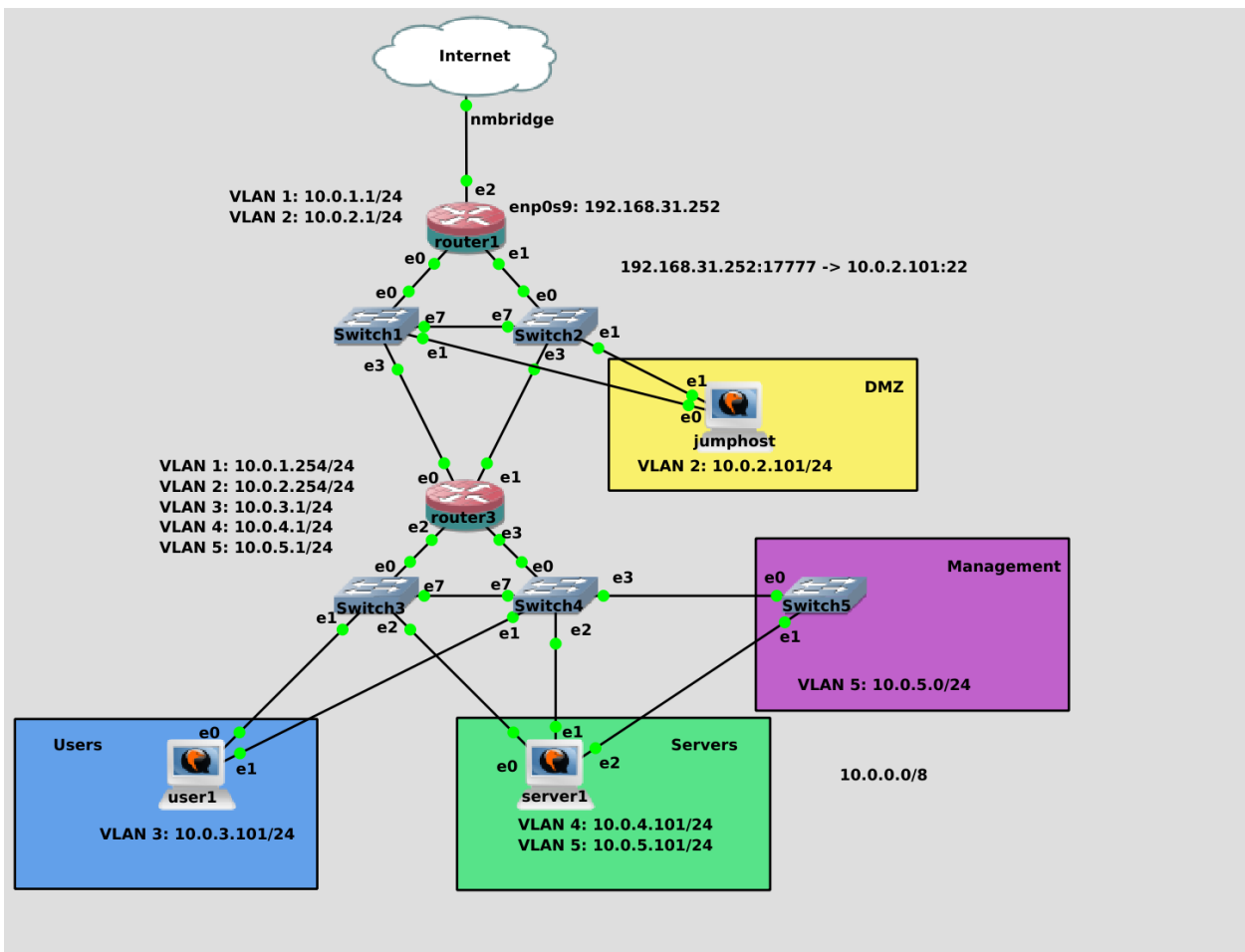
Last login: Sat Dec 25 17:26:26 2021 from 10.0.1.254
[user@router1 ~]$

```

И теперь простой командой:

```
ssh router1
```

мы через 3 джампхоста попадаем на роутер1. Да и в целом можем на любой хост в сети.



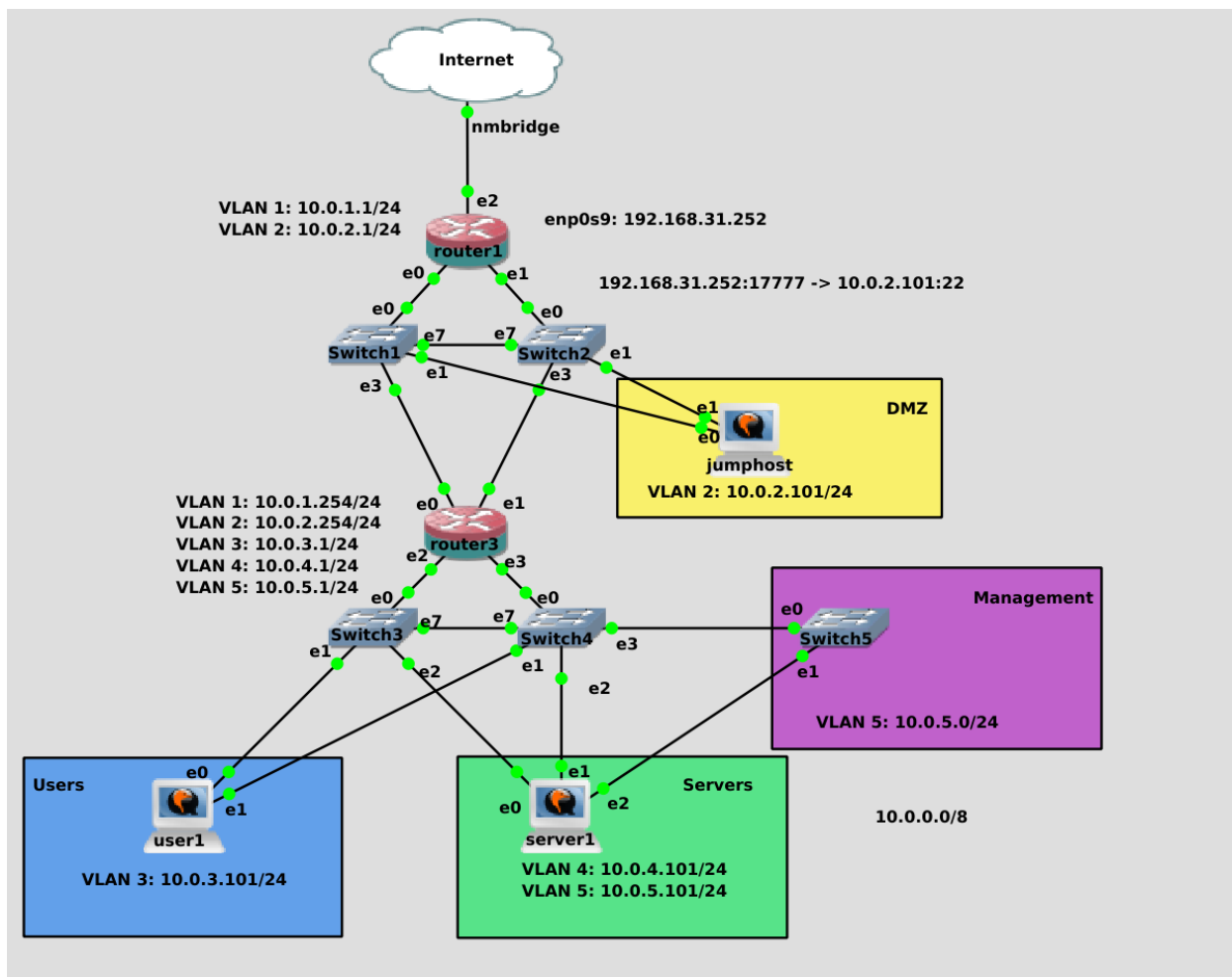
Вообще, если использовать роутер3 в качестве джампхоста, можно попасть на любой хост. Но так как настоящие роутеры не очень годятся в качестве джампхостов, вместо этого мы будем писать правила для файрвола и использовать маршрутизацию. Поэтому это будет небольшим заданием для вас - прописать на нём политики, чтобы можно было с сервера1 подключаться к юзеру1 и роутеру1.

Давайте подведём итоги. Сегодня мы с вами разобрали таблицы и правила маршрутизации, а также политики для файрволов. Хотя пока мы написали всего одну политику, но это в основном было сделано для будущих задач. Когда мы будем поднимать различные сервисы в разных зонах, нам нужно будет постоянно лезть в файрвол и писать политики. А сеть у нас почти готова - дальше можно

сконцентрироваться на самих сервисах.

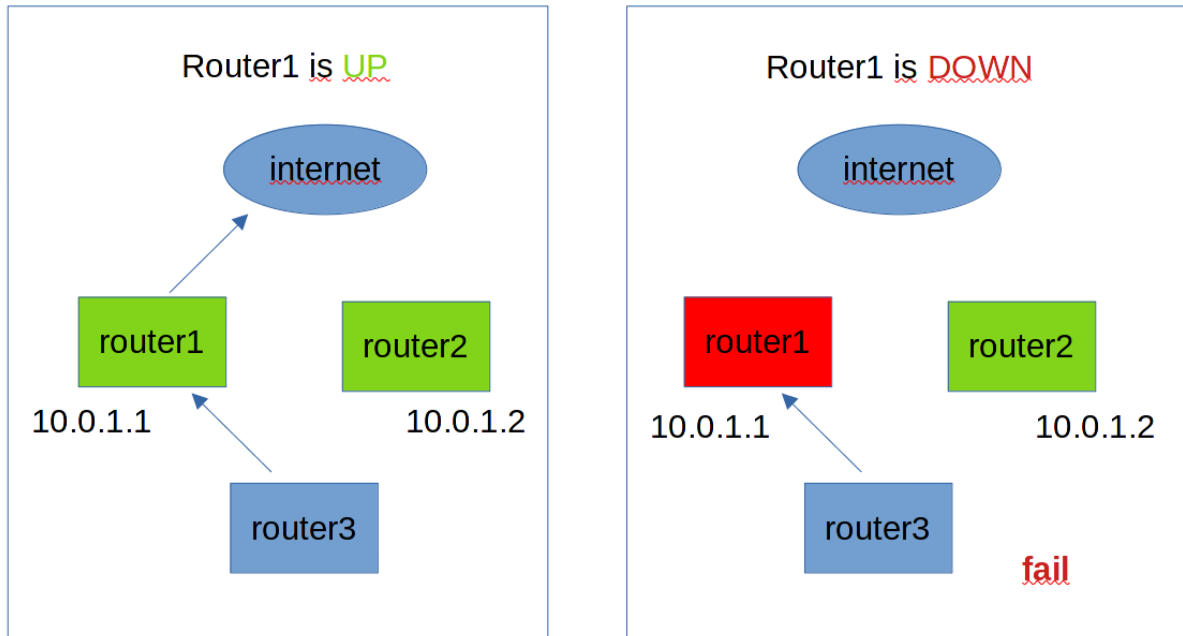
2.9 09. Плавающий IP

2.9.1 09. Плавающий IP

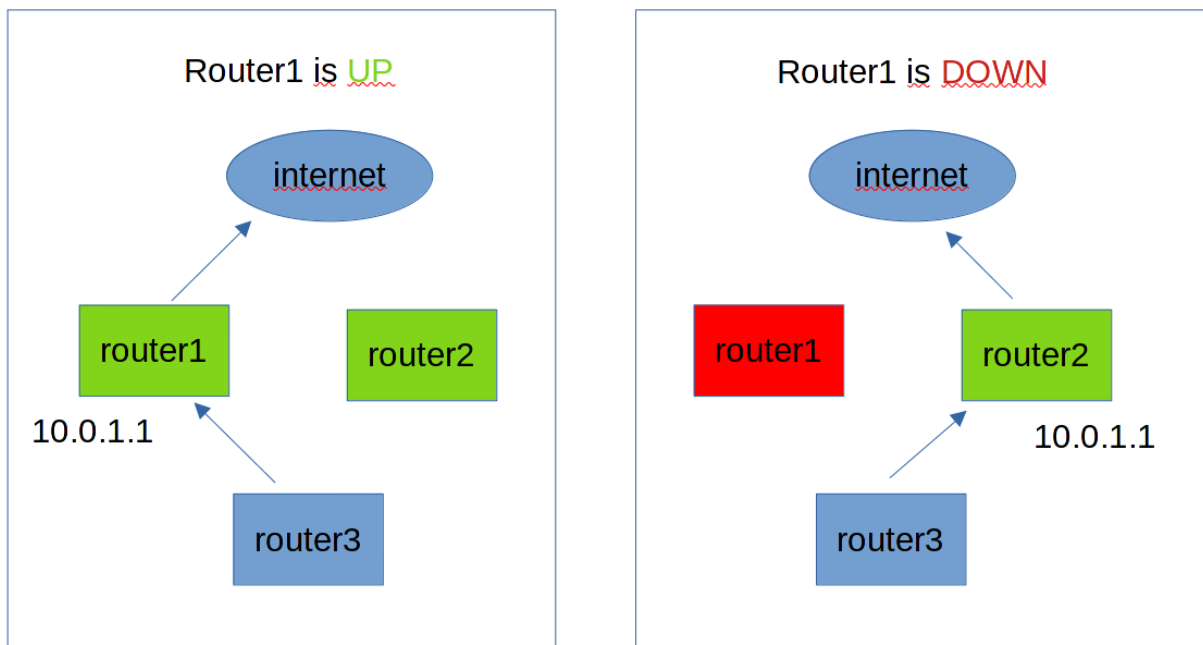


И так, сеть у нас вроде готова. Теперь давайте зададимся вопросом - а что будет, если что-то выйдет из строя? Ну, если выйдет из строя свитч - у нас есть второй, у нас есть NIC teaming - а значит всё продолжит работать. А если выйдет из строя роутер? Если роутер1, то перестанет работать интернет и не будет доступа извне, мы не сможем подключиться к джампхосту. А если роутер3 - то ляжет вся внутренняя сеть. Второе, конечно, более критично, но и того, и другого мы хотели бы избежать.

Казалось бы, ничего сложного, можно сделать как со свитчами - поставить по два роутера, если выйдет из строя один - продолжит работать другой. На деле же есть один нюанс.



Если мы поставим другой роутер, то нам надо будет дать ему другой IP адрес, так как в одной сети у двух систем не может быть одного IP адреса. Но если у второго роутера будет другой IP адрес, то хосты не будут к нему обращаться. Потому что в качестве gateway у них стоит адрес первого роутера. А два гейтвея прописать нельзя, да и смысла нет - всё равно будет использоваться только один, даже если он недоступен.



Т.е. нам нужно сделать так. Когда роутер1 работает, то IP адрес 10.0.1.1 будет у него. Если вдруг что-то с ним случится и роутер1 перестанет быть доступным, то этот же IP адрес 10.0.1.1 должен перейти на роутер2. И тогда этот IP адрес опять будет доступен для хостов и интернет продолжит работать. Такой IP, который переходит от одного хоста к другому часто называют «плавающим IP» или виртуальным ip. Этот механизм, когда IP адрес переходит с одной системы на другую работает

через протокол VRRP - virtual router redundancy protocol. Он был придуман для сети, в частности для роутеров, как раз для той ситуации, которую мы сейчас симулируем. Но такой простой и полезный механизм обрёл большую популярность и в линуксах.

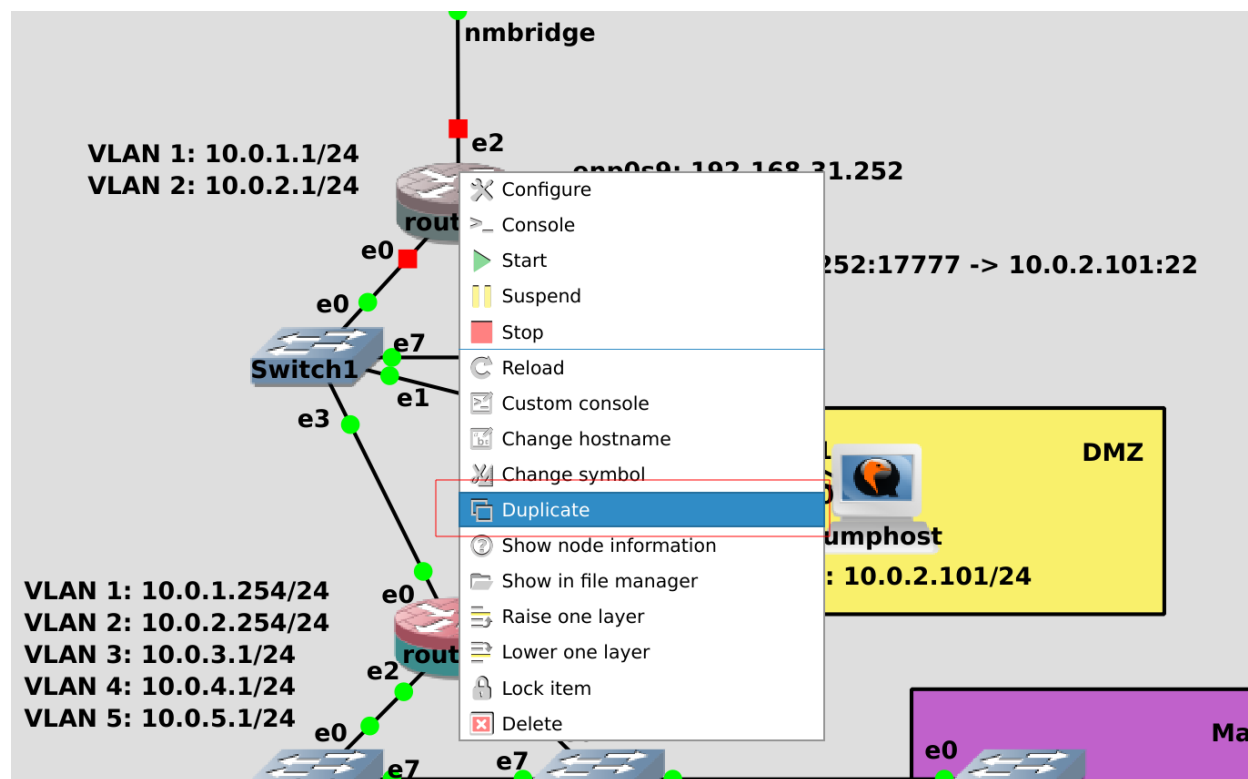
High availability (HA)
Высокая доступность

vs

Fault tolerance
Отказоустойчивость

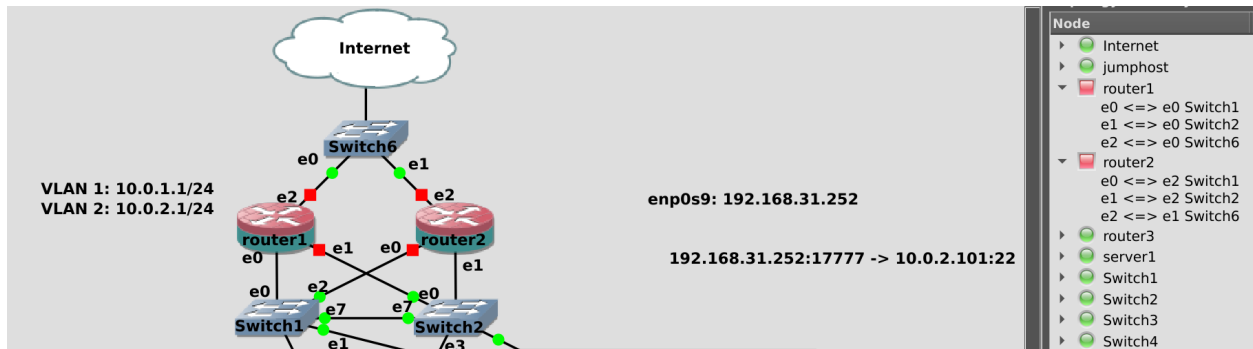
Тут стоит познакомиться с двумя терминами - высокая доступность и отказоустойчивость. Простым языком - высокая доступность - это когда почти всегда работает, но если возникнет проблема - что-то не будет работать буквально пару секунд или максимум минут. Отказоустойчивость - это когда что-бы там не поломалось, пользователи ничего не заметят и всё будет работать идеально. Но эти понятия можно применять на разных уровнях. Скажем, l2arp - это про отказоустойчивость, так как не будет никаких обрывов, а activebackup - это высокая доступность - могут потеряться пару пакетов, но потом всё продолжит работать. Или, скажем, у нас сайт отказоустойчивый, так как работает на нескольких серверах с несколькими виртуалками. И даже если один сервер ляжет, пользователи ничего не заметят. Но при этом сама виртуалка не отказоустойчивая, пока она запустится на другом сервере будет небольшой даунтайм. Но опять же, с точки зрения пользователя - сайт всё это время будет работать. Реальная инфраструктура в компаниях - это смесь отказоустойчивости и высокой доступности. Полную отказоустойчивость сделать нереально и очень дорого. И даже у крупнейших ИТ гигантов за год суммарно набирается пару часов downtime-а.

Так вот, VRRP - это про высокую доступность. Пока второй роутер заметит проблему, пока IP адрес переедет - пройдёт пару секунд. Но зачастую это вполне терпимо, поэтому в каких-то ситуациях можно использовать.



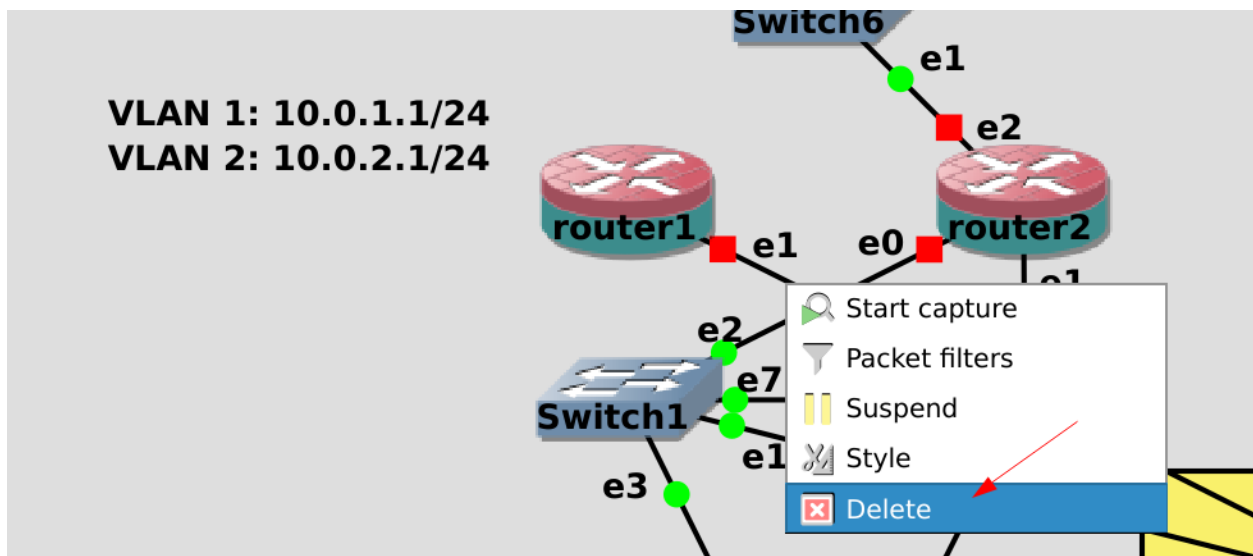
И так, возьмём router1. На нём мы настраивали NIC-teaming, firewall, ssh-ключи и другие вещи. И чтобы с нуля всё это не делать, просто скопируем виртуалку. Для начала выключаем router1, затем

правой кнопкой мыши нажимаем на нём и выбираем duplicate.

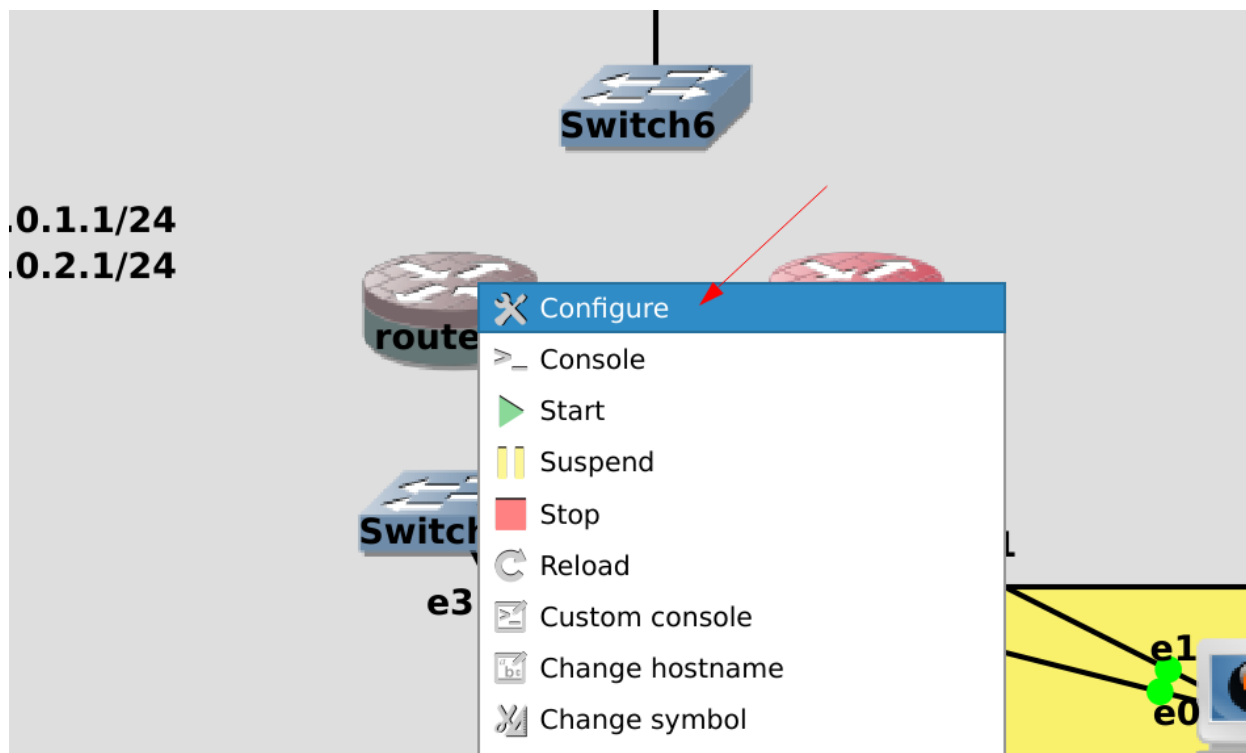


Подключаем его к нижним свитчам точно как router1 и на свитчах также делаем trunk порт. Так как у нас провайдер всего один и от него всего один кабель, добавляем ещё один свитч и подключаем оба роутера и провайдера к нему. Теоретически можно было бы через свитч1 или свитч2, но чтобы не было каши в схеме, сделаем через отдельный свитч.

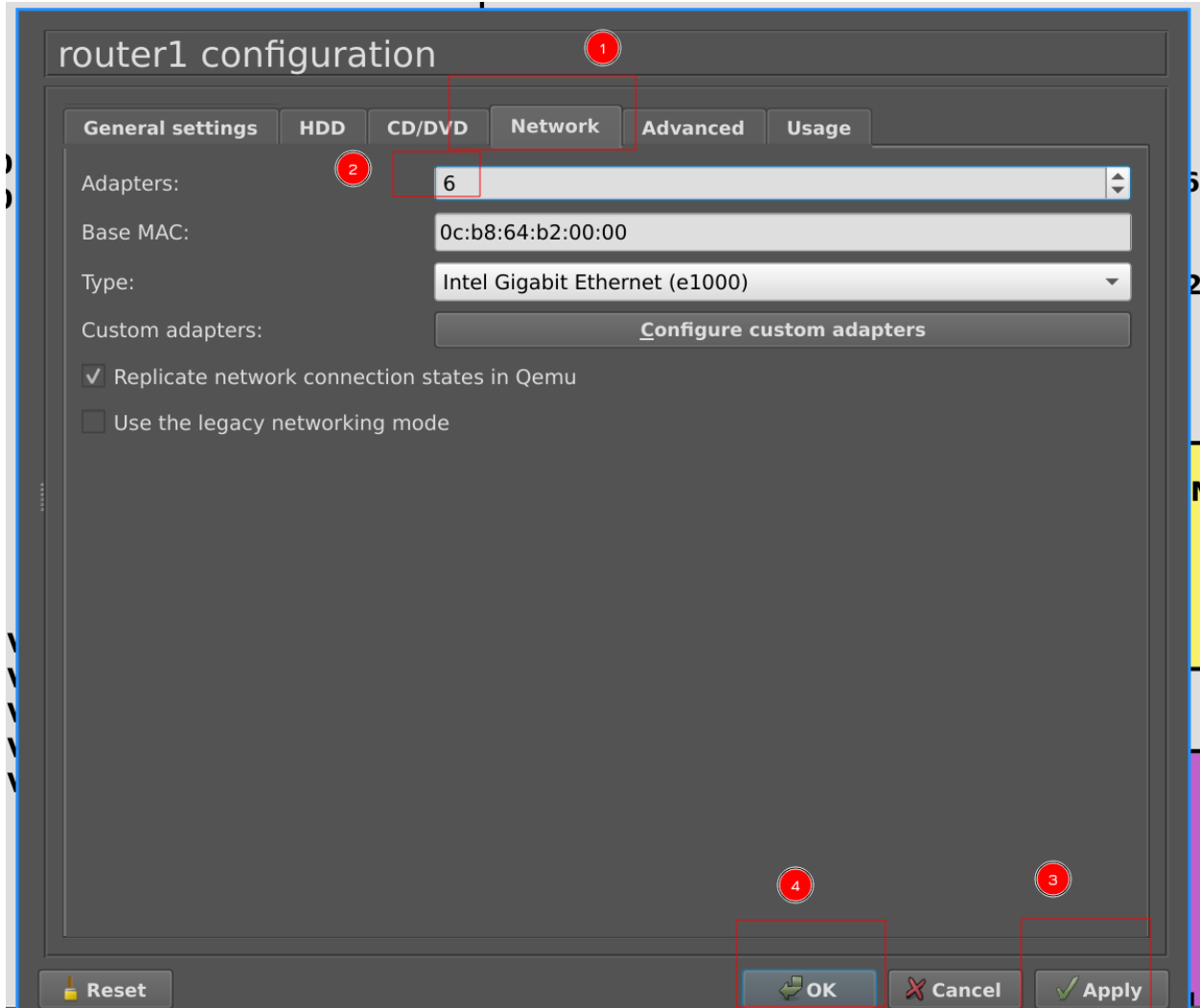
VRRP это сетевой протокол. При нём роутер1 и роутер2 будут обмениваться пакетами, чтобы понять состояние друг друга. Пакеты в сети можно подделать и таким образом какой-нибудь взломщик сможет выдать себя за один из роутеров, тем самым сделав атаку «man in the middle». Поэтому обмен информацией о VRRP стоит организовать через прямое подключение между двумя роутерами, а не через свитч.



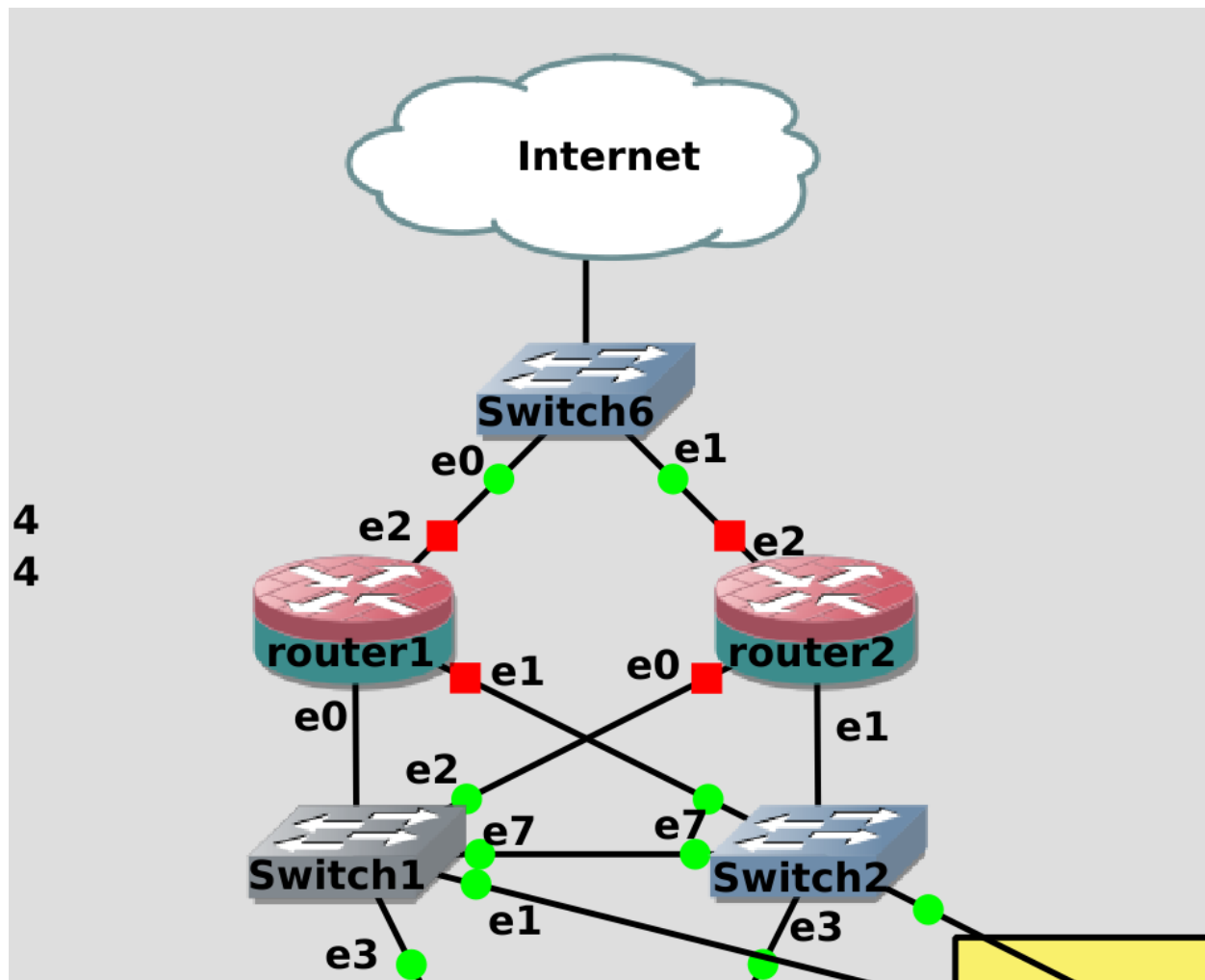
Поэтому давайте добавим ещё портов на роутеры. Но из-за особенностей GNS перед этим надо отсоединить все кабели, идущие от роутера1 и роутера2. Нажимаем правой кнопкой мыши на кабели и выбираем delete.



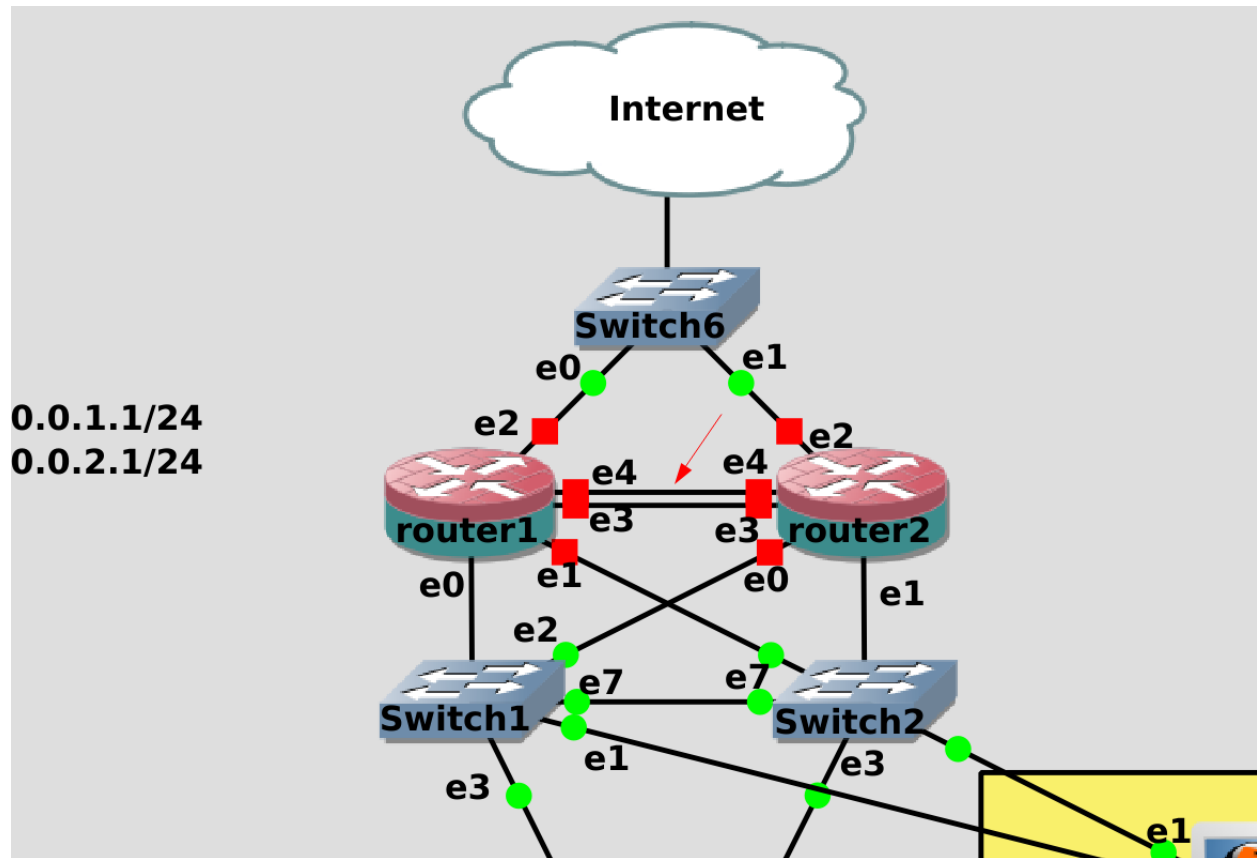
Затем нажимаем правой кнопкой на роутере1 и выбираем Configure.



Во вкладке Network меняем количество адаптеров на 6, нажимаем Apply и OK. Затем тоже самое проделываем для роутера 2.



После чего возвращаем все провода как было.



И теперь подключим роутер1 к роутеру2 двумя проводами напрямую. Теоретически, можно было бы одним, но тогда при выходе из строя адаптера или провода каждый роутер начнёт считать, что другой роутер недоступен, в итоге IP адрес пропишется на обоих роутерах и это приведёт к проблемам в сети. Такая проблема, когда системы перестают видеть друг друга, думают, что другой хост не доступен и поэтому считают себя за главного, называется split brain. Поэтому мы для надёжности подключили роутеры друг к другу двумя проводами и сделаем lасr.

```
[doctor@tardis]~$ eval $(ssh-agent)
Agent pid 123748
[doctor@tardis]~$ ssh-add .ssh/servers .ssh/jumphost
Enter passphrase for .ssh/servers:
Identity added: .ssh/servers (doctor@tardis)
Identity added: .ssh/jumphost (doctor@tardis)
[doctor@tardis]~$ ssh router1
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Sun Jan  9 09:53:04 2022 from 10.0.4.101
[user@router1 ~]$
```

Теперь касательно настройки. Для начала включим только роутер1, запустим ssh agent, добавим ключи и подключимся:

```
eval $(ssh-agent)
ssh-add .ssh/servers .ssh/jumphost
ssh router1
```

```
[user@router1 ~]$ ip --brief a
lo                UNKNOWN      127.0.0.1/8  ::1/128
ens3              UP
ens4              UP
ens5              UP          192.168.31.252/24 fe80::d6f8:5277:d81e:c71e/64
ens6              UP
ens7              UP
ens8              DOWN
team0             UP          10.0.1.1/24 fe80::fa36:eacc:485e:eb80/64
team0.2@team0     UP          10.0.2.1/24 fe80::9d8d:7e6d:fb84:7424/64
[user@router1 ~]$
```

Посмотрим список интерфейсов:

```
ip --brief a
```

Здесь мы видим, что добавились два новых интерфейса. Интерфейсы ens6 и ens7 сейчас в UP-е. На них раньше ничего не было, т.е. это те новые интерфейсы, которые смотрят на роутер2.

```
[user@router1 ~]$ nmcli connection show
NAME                UUID                                TYPE      DEVICE
Wired connection 1  2b9e6208-35a9-3dcb-ba14-3fb624c72085 ethernet ens7
internet            d0d1e84d-debf-48cd-968d-748c5d7e79dc ethernet ens5
team                900d83b8-24ff-49e2-9d4a-297dfe762217 team      team0
vlan2               ce747978-3e67-4fc3-9982-8489c68cc9ea vlan      team0.2
team-slave-ens3     4034d5a4-09b0-49c8-9168-82fdbcbac388c ethernet ens3
team-slave-ens4     6f0bc467-96c4-4305-9179-ac4586e196f8 ethernet ens4
Wired connection 2  31271cc0-db43-36a0-b7d4-320188ebd1f0 ethernet --
[user@router1 ~]$ sudo nmcli connection delete Wired\ connection\ 1 Wired\ connection\ 2
Connection 'Wired connection 1' (2b9e6208-35a9-3dcb-ba14-3fb624c72085) successfully deleted.
Connection 'Wired connection 2' (31271cc0-db43-36a0-b7d4-320188ebd1f0) successfully deleted.
[user@router1 ~]$ nmcli connection show
NAME                UUID                                TYPE      DEVICE
internet            d0d1e84d-debf-48cd-968d-748c5d7e79dc ethernet ens5
team                900d83b8-24ff-49e2-9d4a-297dfe762217 team      team0
vlan2               ce747978-3e67-4fc3-9982-8489c68cc9ea vlan      team0.2
team-slave-ens3     4034d5a4-09b0-49c8-9168-82fdbcbac388c ethernet ens3
team-slave-ens4     6f0bc467-96c4-4305-9179-ac4586e196f8 ethernet ens4
[user@router1 ~]$
```

Так как мы подключили новые интерфейсы, network manager автоматом для них создал профили - Wired connection 1, Wired connection 2:

```
nmcli con sh
```

Начнём с их удаления:

```
sudo nmcli connection delete Wired\ connection\ 1 Wired\ connection\ 2
nmcli con sh
```

```
[user@router1 ~]$ sudo nmcli connection add type team con-name teamvrrp ifname teamvrrp team.runner lacp ipv4.method manual ipv4.addresses 10.255.255.1/30
Connection 'teamvrrp' (baa93c14-9100-44c7-a73e-01399bba57fe) successfully added.
[user@router1 ~]$ sudo nmcli connection add type team-slave ifname ens6 master teamvrrp
Connection 'team-slave-ens6' (970209c6-46b2-4f4d-b3e3-e2ca35261ba4) successfully added.
[user@router1 ~]$ sudo nmcli connection add type team-slave ifname ens7 master teamvrrp
Connection 'team-slave-ens7' (b126f709-2172-457a-97af-f03074e89921) successfully added.
[user@router1 ~]$ nmcli connection show
NAME                UUID                                  TYPE      DEVICE
internet            d0d1e84d-debf-48cd-968d-748c5d7e79dc ethernet  ens5
team                900d83b8-24ff-49e2-9d4a-297dfe762217 team      team0
vlan2               ce747978-3e67-4fc3-9982-8489c68cc9ea vlan      team0.2
teamvrrp            baa93c14-9100-44c7-a73e-01399bba57fe team      teamvrrp
team-slave-ens3     4034d5a4-09b0-49c8-9168-82fdbac388c ethernet  ens3
team-slave-ens4     6f0bc467-96c4-4305-9179-ac4586e196f8 ethernet  ens4
team-slave-ens6     970209c6-46b2-4f4d-b3e3-e2ca35261ba4 ethernet  ens6
team-slave-ens7     b126f709-2172-457a-97af-f03074e89921 ethernet  ens7
[user@router1 ~]$
```

Теперь добавим новый тимиинг интерфейс:

```
sudo nmcli connection add type team con-name teamvrrp ifname teamvrrp team.runner lacp
→ipv4.method manual ipv4.addresses 10.255.255.1/30
```

Назовём профиль и интерфейс teamvrrp, в качестве runner-а - lacp, а IP адрес выдадим 10.255.255.1/30. Во-первых, такой адрес визуальнo отличается от нашей сети и нам будет легче понять, что он для чего-то особенного - общения между первым и вторым роутером. А маска /30 говорит о том, что в этой сети всего 4 адреса - 0,1,2,3. При этом использовать можно только 1 и 2, а у нас в сети как раз два адреса.

```
sudo nmcli connection add type team-slave ifname ens6 master teamvrrp
sudo nmcli connection add type team-slave ifname ens7 master teamvrrp
```

Также добавим интерфейсы ens6 и ens7 в качестве слейвов. После чего проверим список профилей в NetworkManager-e:

```
nmcli con show
```

Вроде всё верно.

```
[user@router1 ~]$ sudo dnf install keepalived
Last metadata expiration check: 0:00:29 ago on Thu 13 Jan 2022 05:42:07 PM +04.
Dependencies resolved.
=====
Package                Arch    Version                               Repository  Size
=====
Installing:
keepalived             x86_64  2.1.5-6.el8                         appstream  535 k
Installing dependencies:
=====
```

Теперь нам нужно установить пакет keepalived - он содержит функционал для работы VRRP:

```
sudo dnf install keepalived
```

```

GNU nano 2.9.8 /etc/keepalived/keepalived.conf
! Configuration File for keepalived

global_defs {
    notification_email {
        acassen@firewall.loc
        failover@firewall.loc
        sysadmin@firewall.loc
    }
    notification_email_from Alexandre.Cassen@firewall.loc
    smtp_server 192.168.200.1
    smtp_connect_timeout 30
    router_id LVS_DEVEL
    vrrp_skip_check_adv_addr
    vrrp_strict

```

Основной файл настроек - /etc/keepalived/keepalived.conf:

```
sudo nano /etc/keepalived/keepalived.conf
```

keepalived много чего может помимо vrrp и здесь примеры различных настроек. Даже в рамках vrrp функционал довольно широкий, вплоть до того, что keepalived может сам настраивать firewall, приоритет процесса vrrp, возможность попадания в swar и т.д. и т.п. В мане keepalived.conf можно очень много всего найти, но большинство из этого довольно специфично.

```

GNU nano 2.9.8 /etc/keepalived/keepalived.conf

vrrp_instance VI_1 {
    state MASTER
    interface eth0
    virtual_router_id 51
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
        192.168.200.16
        192.168.200.17
        192.168.200.18
    }
}

```

Нам это пока не нужно, поэтому мы стираем всё помимо секции vrrp_instance. Теперь давайте разбираться и поправлять.

- **vrrp_instance VI_1** - секция, в которой указаны настройки для одного интерфейса VRRP. Под интерфейсом в данном случае имеется в виду тот интерфейс, который смотрит на другой роутер. Хотя нередко общение между хостами оставляют на том же интерфейсе, на котором меняются адреса, но мы говорили об этом, что это не очень безопасно. По сути может быть, что у нас

несколько разных систем, с разными задачами и адресами, и чтобы отделять одни системы от других можно создать несколько секций. Хотя у нас будут плавать адреса на 3 интерфейсах - на интернет, vlan1 и vlan2, но всё равно всё это относится к одной задаче - все эти адреса будут переходить от роутера1 к роутеру2. VI_1 - название секции, можно поменять, но смысла нет.

- **state MASTER** - определяет, кем будет считать себя хост изначально - MASTER или BACKUP. У кого будет плавающий IP определяет приоритет, но обычно у MASTER приоритет выше, у BACKUP ниже и если всё нормально - адрес остаётся у MASTER. Но представим себе ситуацию, когда MASTER перестал работать и IP перешёл на BACKUP. Во время проблемы был небольшой даунтайм, пока всё переходило на BACKUP. И по настройкам по-умолчанию, при возвращении MASTER в нормальное состояние, IP адрес вернётся на MASTER, что опять вызовет обрыв соединений. Что не очень хорошо.

```
GNU nano 2.9.8 /etc/keepalived/keepalived.conf

vrrp_instance VI_1 {
    state BACKUP
    nopreempt
    interface eth0
    virtual_router_id 51
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
        192.168.200.16
        192.168.200.17
        192.168.200.18
    }
}
```

Если оба сервера одинаковы и мы не отдаём никому предпочтения, то можно оба сервера сделать BACKUP и добавить опцию **nopreempt**. В таком случае после возвращения нерабочей системы в нормальное состояние, IP к нему не вернётся и это не вызовет повторной потери пакетов. Т.е. IP будет меняться только когда одна из систем, будем называть их нодами - пока одна из нод не выйдет из строя. При этом учтите, что даже если прописали здесь BACKUP, на самом деле та нода, у которой сейчас адрес, считается MASTER.

- **interface eth0** - собственно, интерфейс, по которому будет проходить общение между хостами. VRRP может работать через multicast, и поэтому можно обходиться указанием интерфейса вместо IP адреса роутера2. Но, с другой стороны, мы точно знаем адрес второго роутера, а значит можем вместо всего интерфейса указать адрес второй ноды.

```

vrrp_instance VI_1 {
    state BACKUP
    nopreempt
    unicast_peer { 10.255.255.2 }
    virtual_router_id 51
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
        192.168.200.16
        192.168.200.17
        192.168.200.18
    }
}

```

Для этого уберём строчку `interface` и добавим `unicast_peer` с адресом второго роутера:

```
unicast_peer { 10.255.255.2 }
```

Соответственно, настраивая это на втором роутере, надо будет указывать адрес первого роутера.

- `virtual_router_id 51` - идентификатор `vrrp_instance`-а. Условно, если у нас несколько разных `vrrp` запущено, один для этого роутера, другой для другой системы с другими адресами - то у каждого должен быть свой уникальный `id`. А там где совпадают - `id` должен быть одинаковым. Значения могут быть от 0 до 255. И на этом роутере и на втором мы оставим один номер - 51.
- `priority 100` - приоритет. Определяет, у кого будет IP адрес. Чем выше значение - максимум 255 - тем больше приоритет. Соответственно, если у этой ноды будет приоритет 100, а у второй 50 - то адрес будет на этом сервере. Пока не возникнут проблемы и тогда адрес перейдёт на вторую ноду. А благодаря опции `nopreempt` адрес останется там, даже если первый сервер восстановится. Но в реальной среде может быть ситуация, что у вас первый сервер производительнее второго, а второй вы поставили временно, чтобы совсем всё не ломалось. Тут вам приоритет пригодится.
- `advert_int 1` - 1 - это секунда. Т.е. каждую секунду мастер ноды уведомляет другую - или других, если их несколько - что всё окей. Если вдруг мастер замолчал - вторая нода по приоритету становится мастером. Происходит это после 3 раз. Т.е. если у нас тут значение 1 секунда и от мастера нет сообщения на протяжении 3-ёх - то нода считает, что мастер недоступен. Если тут

поставить значение 2 - то это произойдёт через 6 секунд.

Skew_Time Time to skew Master_Down_Interval in seconds.
Calculated as:

$$((256 - \text{Priority}) / 256)$$

Master_Down_Interval Time interval for Backup to declare Master down (seconds). Calculated as:

$$(3 * \text{Advertisement_Interval}) + \text{Skew_time}$$

На самом деле в формуле помимо этого есть ещё небольшая задержка, но это нюансы. Можно ещё поставить миллисекунды, но сильно перебарщивать не надо, а то из-за какой-то небольшой задержки поменяется мастер нода. Поэтому оставим как есть.

```
# Note: authentication was removed from the VRRPv2 specification by
# RFC3768 in 2004.
# Use of this option is non-compliant and can cause problems; avoid
# using if possible, except when using unicast, where it can be helpful.
authentication {
    # PASS|AH
    # PASS - Simple password (suggested)
    # AH - IPSEC (not recommended)
    auth_type PASS

    # Password for accessing vrrpd.
    # should be the same on all machines.
    # Only the first eight (8) characters are used.
    auth_pass 1234
}
```

- **authentication** - для того, чтобы ноды могли убедиться, что перед ними нужный хост. Но на самом деле с этим большие проблемы - при простом PASS пароль в сети ходит в открытом виде, а при AH он хоть и шифруется, но всё равно есть способы обхода.

← → ↻ 🔍 <https://datatracker.ietf.org/doc/html/rfc3768#section-10>

9.3. Operation over ATM LANE

Operation of VRRP over ATM LANE on routers with ATM LANE interfaces and/or routers behind proxy LEC's are beyond the scope of this document.

10. Security Considerations

VRRP does not currently include any type of authentication. Earlier versions of the VRRP specification included several types of authentication ranging from none to strong. Operational experience and further analysis determined that these did not provide any real measure of security. Due to the nature of the VRRP protocol, even if VRRP messages are cryptographically protected, it does not prevent hostile routers from behaving as if they are a VRRP master, creating multiple masters. Authentication of VRRP messages could have

Hinden

Standards Track

[Page 23]

[RFC 3768](#)

VRRP

April 2004

prevented a hostile router from causing all properly functioning routers from going into backup state. However, having multiple masters can cause as much disruption as no routers, which authentication cannot prevent. Also, even if a hostile router could not disrupt VRRP, it can disrupt ARP and create the same effect as having all routers go into backup.

Поэтому в новом стандарте vrrp убрана аутентификация. Именно поэтому мы обмен информацией между роутерами сделали прямой.

```

GNU nano 2.9.8 /etc/keepalived/keepalived.conf

vrrp_instance VI_1 {
    state BACKUP
    nopreemt
    unicast_peer { 10.255.255.2 }
    virtual_router_id 51
    priority 100
    advert_int 1
    authentication {
        auth_type AH
        auth_pass 8798
    }
    virtual_ipaddress {
        192.168.200.16
        192.168.200.17
        192.168.200.18
    }
}

```

Особого смысла нам менять значение нет, но для вас, для наглядности, мол если вы захотите через общую сеть пускать VRRP и не хотите, чтобы пароль был в открытом виде, выставим значение AH и зададим пароль. Максимальная длина пароля - 8 символов.

- `virtual_ipaddress` - собственно, сами плавающие адреса. У нас их будет 3 и все они на разных интерфейсах. Поэтому нам надо при указании адреса также указывать интерфейс.

```

}
virtual_ipaddress {
    192.168.31.252/24 dev ens5
    10.0.1.1/24 dev team0
    10.0.2.1/24 dev team0.2
}
}

```

[Wrote 17 lines]

^G Get Help	^O Write Out	^W Where Is	^K Cut Text	^J Justify	^C Cur Pos
^X Exit	^R Read File	^N Replace	^U Uncut Text	^T To Spell	^L Go To Line

```

[user@router1 ~]$ ip --brief a
lo                UNKNOWN      127.0.0.1/8 ::1/128
ens3              UP
ens4              UP
ens5              UP          192.168.31.252/24 fe80::d6f8:5277:d81e:c71e/64
ens6              UP
ens7              UP
ens8              DOWN
team0             UP          10.0.1.1/24 fe80::fa36:eacc:485e:eb80/64
team0.2@team0     UP          10.0.2.1/24 fe80::9d8d:7e6d:fb84:7424/64
teamvrrp          DOWN
teamvrrp          10.255.255.1/30
[user@router1 ~]$

```

И так, предположим, провайдер даёт нам один внешний адрес - 31.252:

```
ip --brief a
```

Если роутер1 перестанет работать, нам нужно, чтобы на роутере2 был этот же адрес. Он у нас на интерфейсе ens5. Также роутер1 является гейтвеем для первого и второго влана с адресами 1.1 и 2.1. Соответственно, эти адреса мы также должны переносить на второй роутер.

На этом с конфигом роутера закончим. Сохраняем и выходим.

```
user@router1:~$ keepalived --config-test
(/etc/keepalived/keepalived.conf: Line 3) Unknown keyword 'nopreempt'
(VI_1): Unicast instances must have unicast_src_ip or interface
[user@router1 ~]$
```

Давайте проверим конфиг на ошибки, для этого можно использовать утилиту keepalived с опцией config-test:

```
keepalived --config-test
```

Как видите, у меня две ошибки. Во-первых, я опечатался в слове nopreempt. Во-вторых, при использовании unicast_peer мне нужно указывать либо unicast_src_ip, либо интерфейс. Давайте исправим:

```
GNU nano 2.9.8 /etc/keepalived/keepalived.conf

vrrp_instance VI_1 {
    state BACKUP
    nopreempt
    unicast_src_ip 10.255.255.1
    unicast_peer { 10.255.255.2 }
    virtual_router_id 51
    priority 100
    advert_int 1
    authentication {
        auth_type AN
        auth_pass 8798
    }
    virtual_ipaddress {
        192.168.31.252/24 dev ens5
        10.0.1.1/24 dev team0
        10.0.2.1/24 dev team0.2
    }
}
```

Заново открываем конфиг и исправляем ошибки. unicast_src_ip - это айпи, с которого будут отправляться запросы. Т.е. IP адрес интерфейса teamvrrp.

```
[user@router1 ~]$ sudo nano /etc/keepalived/keepalived.conf
[user@router1 ~]$ keepalived --config-test
[user@router1 ~]$
```

Ещё раз запускаем проверку:

```
keepalived --config-test
```

На этот раз никаких ошибок.

```
[user@router1 ~]$ sudo firewall-cmd --new-zone=router2 --permanent
success
[user@router1 ~]$ sudo firewall-cmd --change-interface=teamvrrp --zone=router2 --permanent
The interface is under control of NetworkManager, setting zone to 'router2'.
success
[user@router1 ~]$ sudo firewall-cmd --add-protocol={vrrp,ah} --zone=router2 --permanent
success
[user@router1 ~]$ sudo firewall-cmd --reload
success
[user@router1 ~]$ sudo firewall-cmd --list-all --zone=router2
router2 (active)
  target: default
  icmp-block-inversion: no
  interfaces: teamvrrp
  sources:
  services:
  ports:
  protocols: ah vrrp
  forward: no
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
[user@router1 ~]$
```

Чтобы роутер1 и роутер2 могли обмениваться информацией о vrrp, нам надо кое-что разрешить на файрволе. У нас есть зона routers, но отношения между роутером1 и 2 несколько иные, чем между роутером1 и 3. Можно было бы использовать зону trusted, так как соединение напрямую, но для наглядности создадим новую зону и переместим в неё новый интерфейс:

```
sudo firewall-cmd --new-zone=router2 --permanent
sudo firewall-cmd --change-interface=teamvrrp --zone=router2 --permanent
```

В этой зоне добавим протоколы vrrp и ah. Да, это не просто сервисы, а именно протоколы:

```
sudo firewall-cmd --add-protocol={vrrp,ah} --zone=router2 --permanent
```

После изменений перезагрузим и проверим зону:

```
sudo firewall-cmd --reload
sudo firewall-cmd --list-all --zone=router2
```

Теперь самое главное. Чтобы vrrp работал, нам надо, чтобы эти адреса не были прописаны статично. Их будет прописывать keeplived, поэтому в NetworkManager-е их надо убрать. Но так как помимо адреса у нас в профиле прописаны dns, gateway и прочие настройки, нам лучше не полностью убирать адреса, а заменить их на соседние. Условно, в первом влане у роутера1 выставить ip 10.0.1.2, у роутера2 выставить ip 10.0.1.3, а 10.0.1.1 будет плавать между хостами.

```
[user@router1 ~]$ sudo nmcli connection modify internet ipv4.addresses 192.168.31.250/24
[user@router1 ~]$ sudo nmcli connection modify team ipv4.addresses 10.0.1.2/24
[user@router1 ~]$ sudo nmcli connection modify vlan2 ipv4.addresses 10.0.2.2/24
[user@router1 ~]$
```

Поэтому поменяем IP адреса на всех трёх профилях:

```
sudo nmcli connection modify internet ipv4.addresses 192.168.31.250/24
sudo nmcli connection modify team ipv4.addresses 10.0.1.2/24
sudo nmcli connection modify vlan2 ipv4.addresses 10.0.2.2/24
```

Пока мы заново не поднимем интерфейсы, настройки не применятся.

```
[user@router1 ~]$ sudo nmcli con up internet && sudo nmcli con up team && sudo nmcli con up vlan
2 && sudo systemctl enable --now keepalived
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConn
ection/42)
Connection successfully activated (master waiting for slaves) (D-Bus active path: /org/freedeskto
p/NetworkManager/ActiveConnection/43)
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConn
ection/46)
Created symlink /etc/systemd/system/multi-user.target.wants/keepalived.service → /usr/lib/system
d/system/keepalived.service.
[user@router1 ~]$ ip --brief a
lo                UNKNOWN    127.0.0.1/8 ::1/128
ens3              UP
ens4              UP
ens5              UP    192.168.31.250/24 192.168.31.252/24 fe80::d6f8:5277:d81e:c71e/64
ens6              UP
ens7              UP
ens8              DOWN
teamvrrp          DOWN    10.255.255.1/30
team0             UP    10.0.1.2/24 10.0.1.1/24 fe80::fa36:eacc:485e:eb80/64
team0.2@team0     UP    10.0.2.2/24 10.0.2.1/24 fe80::9d8d:7e6d:fb84:7424/64
[user@router1 ~]$
```

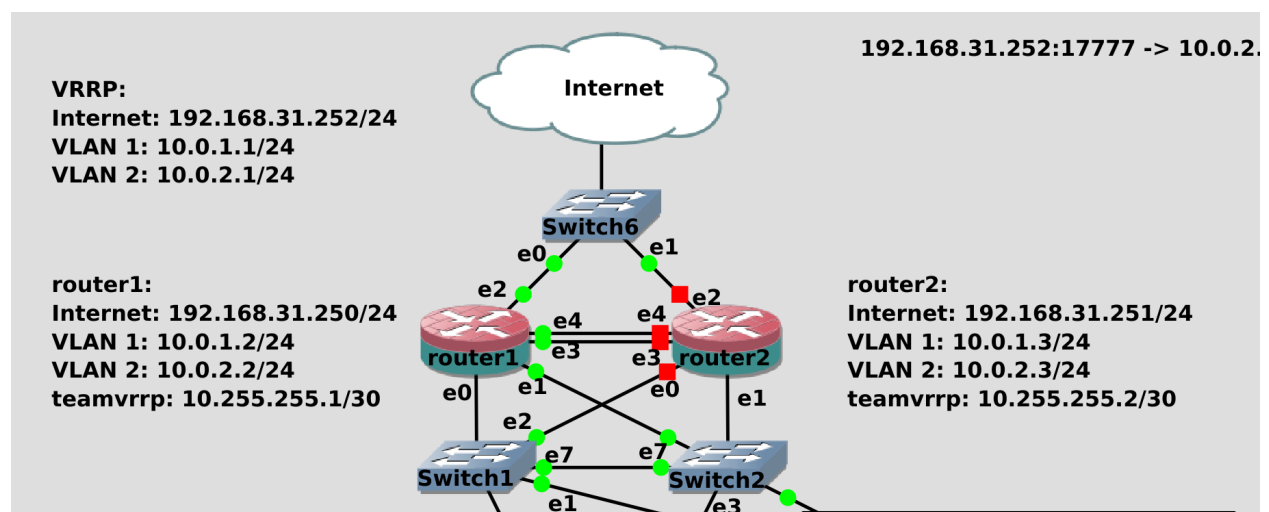
И чтобы нас не выкинуло из-за того, что мы меняем IP адреса, сделаем так. Объединим команду поднятия интерфейсов с командой запуска keepalived:

```
sudo nmcli con up internet && sudo nmcli con up team && sudo nmcli con up vlan2 && sudo
systemctl enable --now keepalived
```

Так я поднимаю все 3 интерфейса и сразу же поднимаю keepalived, который видит, что второй роутер недоступен, а значит прописывает у себя эти адреса. Так как я всё ещё подключен, всё сработало. Но всё же проверим адреса:

```
ip --brief a
```

Как видите, теперь на наших интерфейсах по два адреса. Второй прописывается сервисом keepalived, и, если его отключить, адрес пропадёт.



Теперь надо настроить router2. Вкратце - надо поменять хостнейм, адреса на интерфейсах на 10.0.1.3, 10.0.2.3, 192.168.31.251, а также создать новый тининг интерфейс для vrrp, создать зону на файрволе,

установить keepalived. Короче, всё что мы делали для первого роутера. Наверное, будет легче просто заново скопировать router1, чтобы меньше возиться с настройками. Это вы и сами можете сделать, поэтому пропустим этот шаг и сразу перейдём к настройке keepalived.

```
GNU nano 2.9.8 /etc/keepalived/keepalived.conf

vrrp_instance VI_1 {
    state BACKUP
    nopreempt
    unicast_src_ip 10.255.255.2
    unicast_peer { 10.255.255.1 }
    virtual_router_id 51
    priority 100
    advert_int 1
    authentication {
        auth_type AH
        auth_pass 8798
    }
    virtual_ipaddress {
        192.168.31.252/24 dev ens5
        10.0.1.1/24 dev team0
        10.0.2.1/24 dev team0.2
    }
}
```

На втором роутере, в конфиге `keepalived.conf` надо поправить `unicast_src_ip` и `unicast_peer`, поменяв значения местами. Так как у нас оба сервера в режиме `BACKUP` и стоит опция `nopreempt`, от различий приоритета большого смысла нет. Так как IP не будет переходить просто из-за приоритета. Лично для нас никакой разницы между первым и вторым роутером нет, поэтому приоритет можем оставить одинаковым.

```
[user@router2 ~]$ sudo systemctl restart keepalived.service
[sudo] password for user:
[user@router2 ~]$
```

После изменений рестартнём keepalived на router2 и запустим router1:

```
sudo systemctl restart keepalived
```

```
[user@router2 ~]$ ip --brief a
lo                UNKNOWN    127.0.0.1/8 ::1/128
ens3              UP
ens4              UP
ens5              UP        192.168.31.251/24 192.168.31.252/24 fe80::d6f8:5277:d81e:c71e/64
ens6              UP
ens7              UP
ens8              DOWN
team0             UP        10.0.1.3/24 10.0.1.1/24 fe80::fa36:eacc:485e:eb80/64
team0.2@team0     UP        10.0.2.3/24 10.0.2.1/24 fe80::9d8d:7e6d:fb84:7424/64
teamvrrp          UP        10.255.255.2/30 fe80::f490:128c:206e:658c/64 fe80::9ee:7a88:685:54b9/64 fe80::150a:c41d:83e9:12ca/64
[user@router2 ~]$
```

И давайте проверим. Посмотрим ip адреса на роутере2:

```
ip --brief a
```

Тут видно, что на нём и его адреса, и плавающие адреса.

```
[user@router1 ~]$ ip --brief a
lo                UNKNOWN          127.0.0.1/8 ::1/128
ens3              UP
ens4              UP
ens5              UP          192.168.31.250/24 fe80::d6f8:5277:d81e:c71e/64 fe80::3688:ad7e:e
81a:dee5/64
ens6              UP
ens7              UP
ens8              DOWN
team0             UP          10.0.1.2/24 fe80::fa36:eacc:485e:eb80/64 fe80::502c:ccd9:4ee7:89
f1/64
team0.2@team0     UP          10.0.2.2/24 fe80::9d8d:7e6d:fb84:7424/64 fe80::fac8:be11:fd2d:a0
07/64
teamvrrp          UP          10.255.255.1/30 fe80::f490:128c:206e:658c/64 fe80::9ee:7a88:685:
54b9/64 fe80::150a:c41d:83e9:12ca/64
[user@router1 ~]$
```

А вот на роутере1:

```
ip --brief a
```

нет вторых адресов. Т.е. мастером сейчас является роутер2, а роутер1 пока что просто ждёт, пока роутер2 не выйдет из строя.

```
Jan 13 18:46:24 router1 Keepalived[1156]: NOTICE: setting config option max_auto_priority should
result in better keepalived performance
Jan 13 18:46:24 router1 Keepalived[1156]: Starting VRRP child process, pid=1157
Jan 13 18:46:24 router1 Keepalived_vrrp[1157]: Registering Kernel netlink reflector
Jan 13 18:46:24 router1 Keepalived_vrrp[1157]: Registering Kernel netlink command channel
Jan 13 18:46:24 router1 Keepalived_vrrp[1157]: Opening file '/etc/keepalived/keepalived.conf'.
Jan 13 18:46:24 router1 Keepalived_vrrp[1157]: Registering gratuitous ARP shared channel
Jan 13 18:46:24 router1 Keepalived_vrrp[1157]: (VI_1) removing VIPs.
Jan 13 18:46:24 router1 Keepalived_vrrp[1157]: (VI_1) Entering BACKUP STATE (init)
Jan 13 18:46:24 router1 Keepalived_vrrp[1157]: VRRP sockpool: [ifindex( 0), family(IPv4), proto
(51), fd(11,12), unicast, address(10.255.255.1)]
Jan 13 18:46:24 router1 systemd[1]: Started LVS and VRRP High Availability Monitor.
[user@router1 ~]$
```

Если посмотреть логи на роутере1:

```
journalctl -eu keepalived --no-pager
```

можно увидеть, что роутер1 увидел Gratuitous ARP через общий канал и перешёл в режим BACKUP.

В рамках протокола ARP возможны самообращённые запросы (gratuitous ARP). При таком запросе инициатор формирует пакет, где в качестве IP-адреса используется его собственный адрес. Это бывает нужно, когда осуществляется стартовая конфигурация сетевого интерфейса. В таком запросе IP-адреса отправителя и получателя совпадают.

Самообращённый запрос позволяет ЭВМ решить две проблемы. Во-первых, определить, нет ли в сети объекта, имеющего тот же IP-адрес. Если на такой запрос придёт отклик, то ЭВМ выдаст на консоль сообщение Duplicate IP address sent from Ethernet address <...>. Во-вторых, в случае смены сетевой карты производится корректировка записи в ARP-таблицах ЭВМ, которые содержали старый MAC-адрес инициатора. Машина, получающая ARP-запрос с адресом, который содержится в её таблице, должна обновить эту запись.

Вторая особенность такого запроса позволяет резервному файловому серверу заменить основной, пошлав самообращённый запрос со своим MAC-адресом, но с IP-адресом вышедшего из строя сервера. Этот запрос вызовет перенаправление кадров, адресованных основному серверу, на резервный. Клиенты сервера при этом могут и не знать о выходе основного сервера из строя. При этом возможны и неудачи, если программные реализации в ЭВМ не в полной мере следуют регламентациям протокола ARP.

Помните что такое ARP запрос? Когда один хост знает IP адрес другого, но не знает мак адреса, он посылает ARP запрос. Так вот, представьте, что все хосты в сети запомнили, что IP адрес 1.1 находится за мак адресом первого роутера. Но вот внезапно первый роутер умирает и IP 1.1 переезжает на второй роутер. Но на нём-то другой мак адрес, а значит хосты будут пытаться обращаться к старому мак адресу, пока сами не обновят свою агр таблицу. И чтобы избежать этой проблемы и задержки, роутер2 сразу же сам отправляет всем агр запрос, в котором говорит, что теперь у адреса 1.1 такой-то мак адрес. Это такой самообращённый ARP запрос. Он также позволяет компьютерам понять, а нет ли такого же IP адреса на других компьютерах в этой сети.

```

user@router2:~ user@server1:~
[user@server1 ~]$ ping 1.1.1.1
PING 1.1.1.1 (1.1.1.1) 56(84) bytes of data:
64 bytes from 1.1.1.1: icmp_seq=1 ttl=53 time=91.7 ms
64 bytes from 1.1.1.1: icmp_seq=2 ttl=53 time=91.8 ms
64 bytes from 1.1.1.1: icmp_seq=3 ttl=53 time=92.1 ms
64 bytes from 1.1.1.1: icmp_seq=4 ttl=53 time=91.8 ms
64 bytes from 1.1.1.1: icmp_seq=5 ttl=53 time=91.9 ms

```

Ладно, вроде всё настроили, давайте тестировать. По задумке, если что-то случится с роутером2, IP адрес перейдёт на роутер1. Идём на сервер1:

```
ssh server1
```

И запускаем пинг в интернет:

```
ping 1.1.1.1
```

Дальше идём в GNS и отрубаем роутер2.

```

64 bytes from 1.1.1.1: icmp_seq=48 ttl=53 time=91.9 ms
64 bytes from 1.1.1.1: icmp_seq=49 ttl=53 time=91.6 ms
64 bytes from 1.1.1.1: icmp_seq=50 ttl=53 time=90.10 ms
64 bytes from 1.1.1.1: icmp_seq=51 ttl=53 time=92.1 ms
client_loop: send disconnect: Broken pipe
client_loop: send disconnect: Broken pipe
❗ -[doctor@tardis]-[~]
$ ssh server1
Enter passphrase for key '/home/doctor/.ssh/jumphost':
Enter passphrase for key '/home/doctor/.ssh/servers':
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Sun Jan  9 17:26:58 2022 from 10.0.2.101
[user@server1 ~]$ ping 1.1.1.1
PING 1.1.1.1 (1.1.1.1) 56(84) bytes of data.
64 bytes from 1.1.1.1: icmp_seq=1 ttl=53 time=91.7 ms
^C
--- 1.1.1.1 ping statistics ---
2 packets transmitted, 1 received, 50% packet loss, time 1002ms
rtt min/avg/max/mdev = 91.663/91.663/91.663/0.000 ms
[user@server1 ~]$

```

При этом соединение по ssh к серверу1 прервалось. Но я могу повторно подключиться и проверить пинг:

```

ssh server1
ping 1.1.1.1

```

И всё работает. Как я говорил, vrrp - это про высокую доступность, а не про отказоустойчивость. Если вырубится один роутер - то все соединения прервутся. Но через пару секунд можно будет подключиться заново, так как теперь все нужные адреса на другом роутере.

```

lo                UNKNOWN    127.0.0.1/8 ::1/128
ens3              UP
ens4              UP
ens5              UP          192.168.31.250/24 192.168.31.252/24 fe80::d6f8:5277:d81e:c71e/64
ens6              UP
ens7              UP
ens8              DOWN
team0             UP          10.0.1.2/24 10.0.1.1/24 fe80::fa36:eacc:485e:eb80/64
team0.2@team0     UP          10.0.2.2/24 10.0.2.1/24 fe80::9d8d:7e6d:fb84:7424/64 fe80::fac8:
bell:fd2d:a007/64
teamvrrp          UP          10.255.255.1/30 fe80::f490:128c:206e:658c/64
[user@router1 ~]$

lo                UNKNOWN    127.0.0.1/8 ::1/128
ens3              UP
ens4              UP
ens5              UP          192.168.31.251/24 fe80::d6f8:5277:d81e:c71e/64 fe80::3688:ad7e:e
81a:dee5/64
ens6              UP
ens7              UP
ens8              DOWN
team0             UP          10.0.1.3/24 fe80::fa36:eacc:485e:eb80/64 fe80::502c:ccd9:4ee7:89
f1/64
team0.2@team0     UP          10.0.2.3/24 fe80::9d8d:7e6d:fb84:7424/64
teamvrrp          UP          10.255.255.2/30 fe80::f490:128c:206e:658c/64
[user@router2 ~]$

```

Посмотрим адреса на обеих системах:

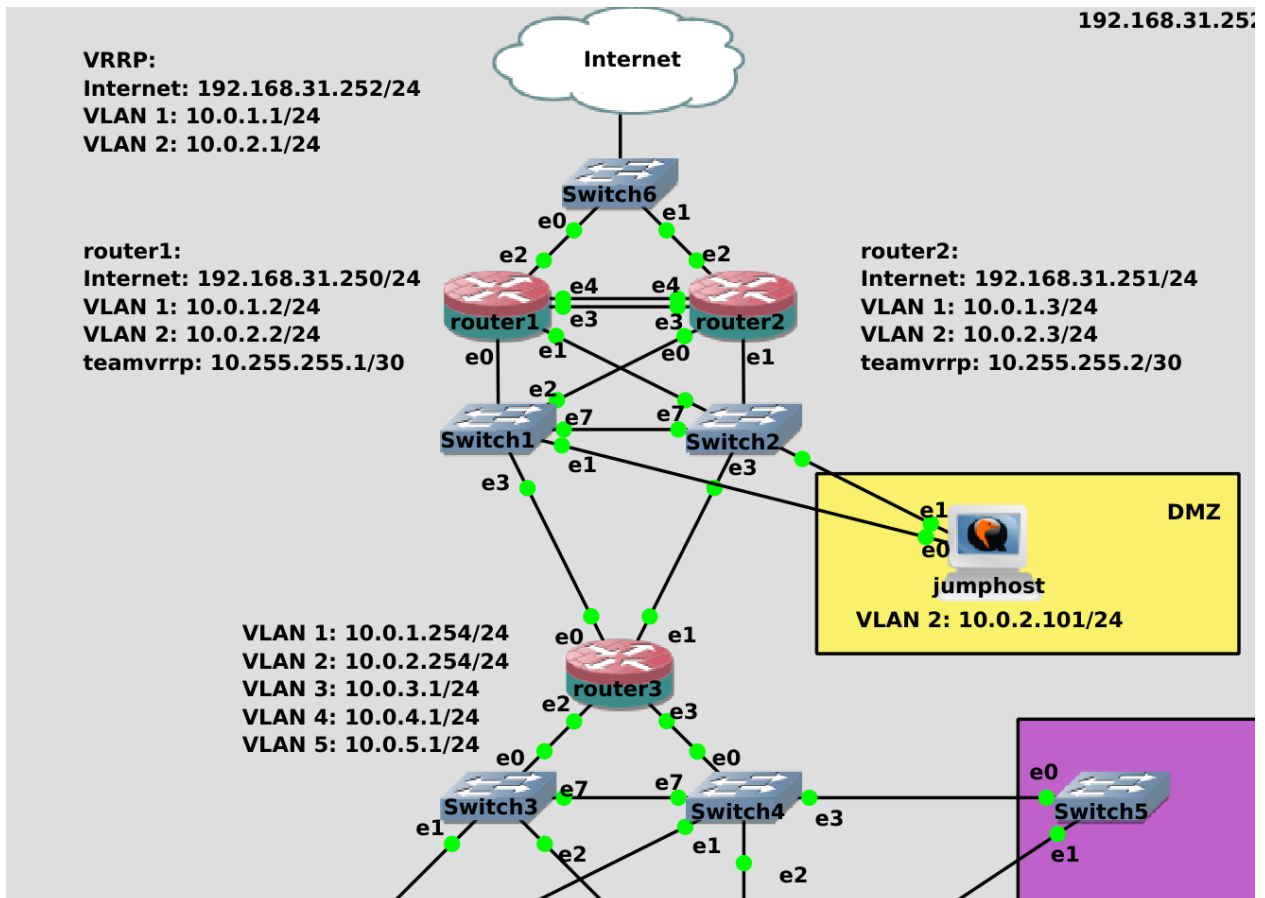
```
ip --brief a
```

Как видите, сейчас адреса на первом хосте.

```

Remote View Bookmarks Help
Connect
[roo@server1 ~]# ping 1.1.1.1
PING 1.1.1.1 (1.1.1.1) 56(84) bytes of data.
64 bytes from 1.1.1.1: icmp_seq=1 ttl=53 time=97.7 ms
64 bytes from 1.1.1.1: icmp_seq=2 ttl=53 time=115 ms
64 bytes from 1.1.1.1: icmp_seq=3 ttl=53 time=91.1 ms
64 bytes from 1.1.1.1: icmp_seq=4 ttl=53 time=91.3 ms
64 bytes from 1.1.1.1: icmp_seq=5 ttl=53 time=122 ms
64 bytes from 1.1.1.1: icmp_seq=6 ttl=53 time=96.8 ms
64 bytes from 1.1.1.1: icmp_seq=10 ttl=53 time=110 ms
64 bytes from 1.1.1.1: icmp_seq=11 ttl=53 time=92.5 ms
64 bytes from 1.1.1.1: icmp_seq=12 ttl=53 time=92.4 ms
^C
--- 1.1.1.1 ping statistics ---
12 packets transmitted, 9 received, 25% packet loss, time 11103ms
rtt min/avg/max/mdev = 91.126/100.989/122.455/11.045 ms
[roo@server1 ~]#
  
```

Чтобы увидеть, сколько пакетов мы теряем, давайте пустим пинг в интернет через консоль сервера1 и вырубим первый роутер. Как видите, потерялись всего несколько пингов, что в принципе не критично. У нас целый роутер отвалился, а сеть заработала почти моментально. Можно, конечно, немного уменьшить параметр `advert_int` до миллисекунд, но особой необходимости в этом я не вижу.



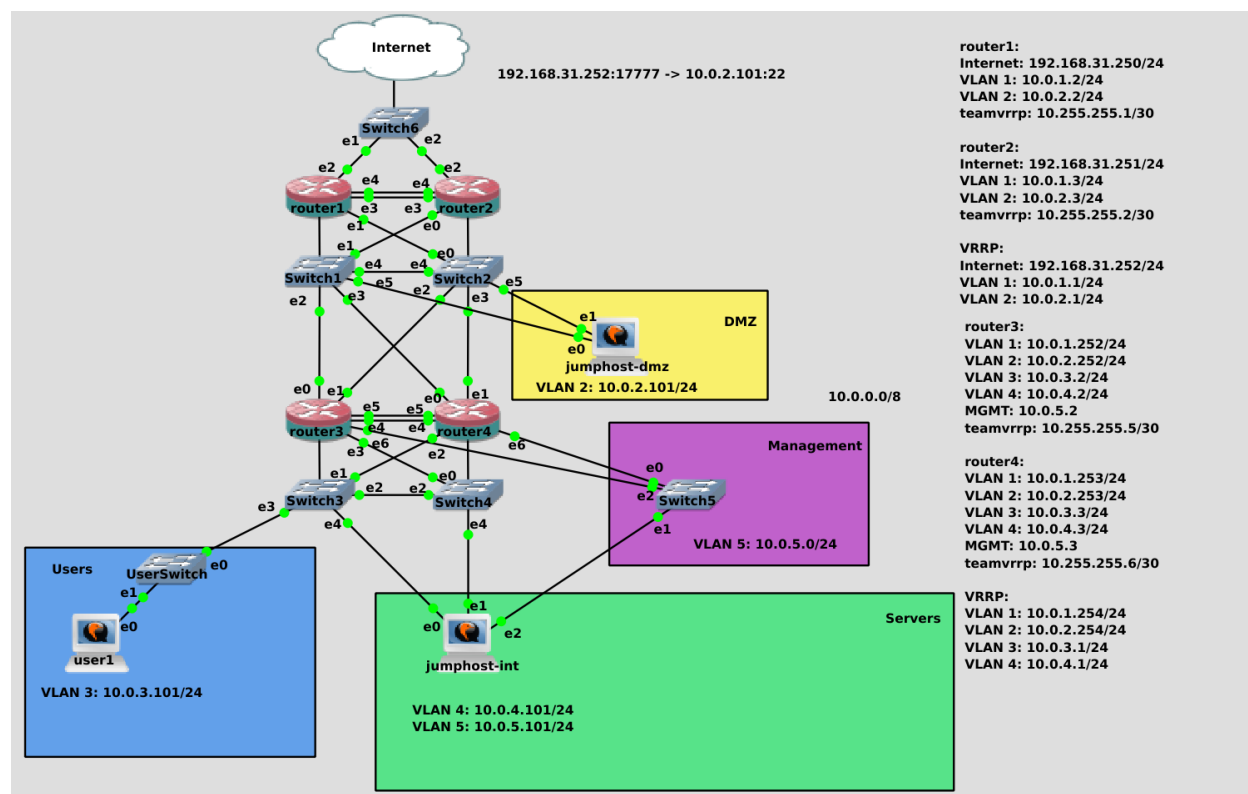
И так, теперь у нас любые проблемы с роутером1 не так страшны и буквально через пару секунд всё продолжит работать на втором. Это не только делает нашу сеть высокодоступной, но и позволит в будущем обновлять роутеры, выключать, менять и проделывать любые операции с ними, при этом это не сильно будет влиять на пользователей. Скажем, раньше при обновлении роутера у вас интернет был недоступен 5 минут, а если ещё проблема какая возникла бы - то это могло затянуться на пару часов. Сейчас же интернет недоступен буквально пару секунд.

Ваша домашняя работа будет добавить роутер4, связать его с 3 роутером и настроить VRRP для всех 5 адресов. Ну и ещё посмотрите вывод tcpdump по интерфейсу teamvrrp во время, когда всё нормально, а также во время, когда один из хостов выходит из строя, а потом возвращается. Ничего суперважного там нет, но эта практика будет полезна для работы с tcpdump.

Давайте подведём итоги. Сегодня мы разобрали, чем отличается высокая доступность от отказоустойчивости, разобрали VRRP - протокол, который позволяет одному IP плавать между хостами, настроили keepalived и всё протестировали. Как я говорил, функционал keepalived довольно большой и мы к нему ещё будем возвращаться. Но на сегодня этого хватит.

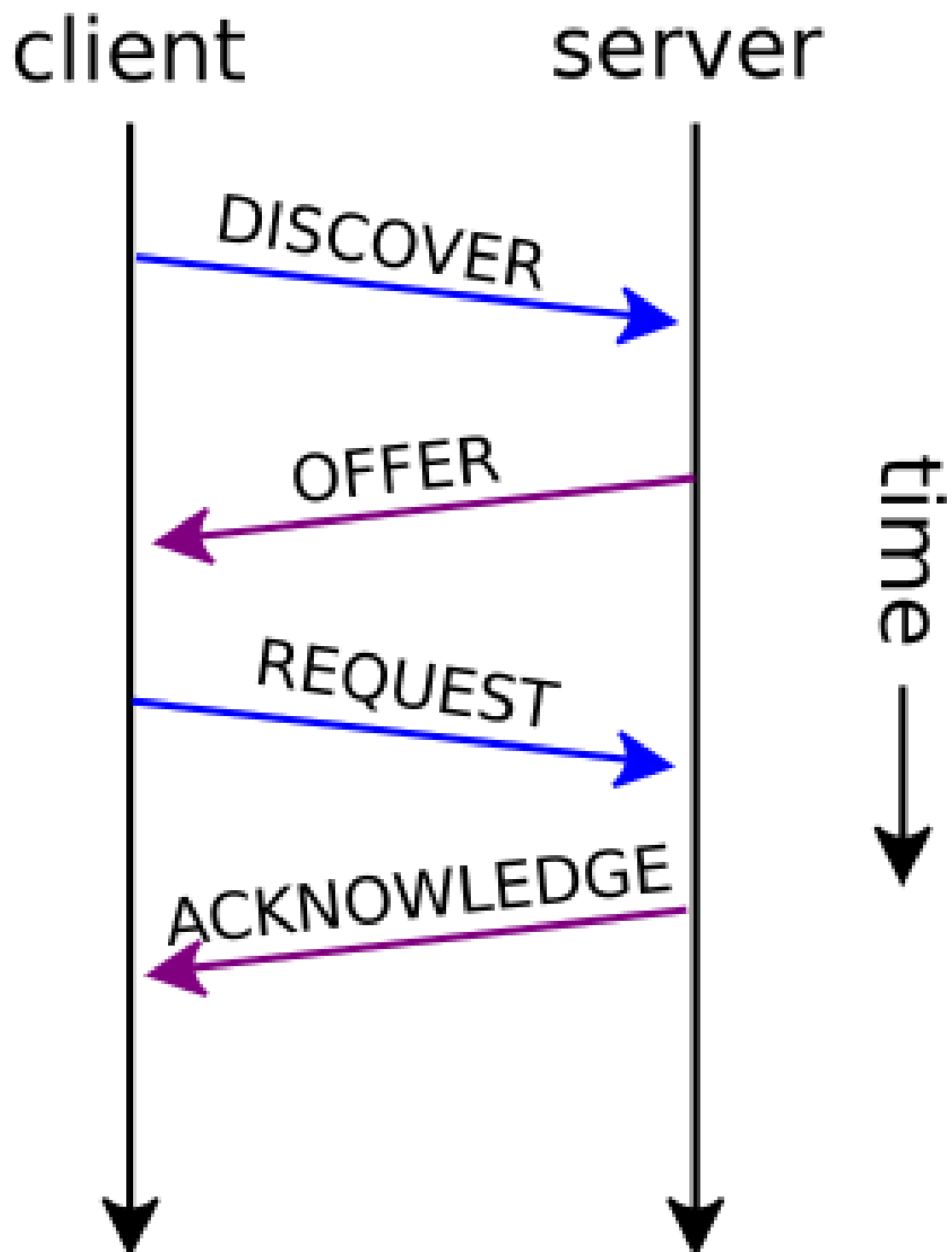
2.10 10. DHCP сервер ч.1

2.10.1 10. DHCP сервер ч.1



Чтобы какой-то компьютер мог быть частью сети и полноценно в ней работать, ему нужна информация о сети: свой IP адрес и маска, адрес gateway, адреса dns серверов, возможно, какая-то дополнительная информация. Когда в сети 1-2 компьютера, всю эту информацию можно и вручную прописать, задав статические IP адреса. Если устройств становится много, это начинает отнимать много времени. Поэтому практически в любой сети есть DHCP сервера, благодаря которым достаточно подключить устройство

к свитчу - и оно получит всю необходимую информацию о сети. Само название DHCP (Dynamic Host Configuration Protocol) говорит за себя - протокол динамической настройки хоста.



Неплохо бы иметь представление, как происходит этот процесс. Он состоит из 4 шагов и вкратце его называют DORA - DISCOVER, OFFER, REQUEST и ACKNOWLEDGE.

Представим, вы подключили какое-то новое устройство к сети. Практически в любом устройстве, которое поддерживает сеть, есть так называемый DHCP клиент - программа, которая позволяет динамически настроить сеть. Допустим, мы подключили какой-то Linux, NetworkManager увидел, что поднялся интерфейс, у этого интерфейса в профиле стоит `ipv4.method auto` - а значит NetworkManager запускает DHCP клиент. Так как в этот момент у компьютера нет никакой информации о сети, нет IP адреса или адреса DHCP сервера - dhcp клиент отправляет broadcast поиск - DISCOVER. Этот запрос придёт всем устройствам в этом broadcast домене. Если в этом влане есть DHCP сервер - то он увидит этот запрос. В зависимости от настроек и свободных адресов DHCP сервер подготовит предложение - OFFER - и пошлёт на мак адрес DHCP клиента. Теоретически, в сети может быть несколько DHCP серверов, соответственно, клиенту может прийти несколько предложений, из которых он должен выбрать одно. Но это просто техническая возможность, на деле в одном broadcast домене не размещают несколько разных dhcp серверов. Клиент, после получения предложения, посылает самообращённый ARP запрос, чтобы понять, есть ли в сети другое устройство с таким IP? И когда видит, что другого устройства нет - клиент посылает серверу запрос - REQUEST - дай мне этот адрес. Клиент, в зависимости от настроек, может также запросить дополнительную информацию. Сервер, получив такой запрос, посылает подтверждение - ACKNOWLEDGE - добавляя к ответу дополнительную информацию, в зависимости от настроек, а также время - lease time - на какое время был выдан этот IP адрес. В течении этого времени сервер не будет отдавать этот адрес другому устройству. К этому мы ещё вернёмся.

У всяких роутеров и современных свитчей есть функционал DHCP сервера. Да те же домашние роутеры - пользователи не знают ничего ни про IP адреса, ни про dns сервера, ни про маршруты. Они просто подключают компьютер к сети проводом или по WiFi - и всё работает благодаря DHCP серверу. Но функционал DHCP сервера на сетевом оборудовании зачастую довольно ограничен, неудобен и подойдёт разве что для небольших сетей. Ну и сетевое оборудование администрирует сетевик, а DHCP сервер в основном раздаёт адреса компьютерам пользователей, серверам и т.п. - то есть на то оборудование, которое администрирует сисадмин. И если DHCP сервер будет на роутере - сисадмину придётся вечно обращаться к сетевику - настрой мне то, настрой мне это. Поэтому в средних и больших сетях DHCP сервера - это отдельные виртуалки, которые админит системный администратор. Для начала мы разместим его на роутере, чтобы познакомиться с функционалом, а дальше вынесем на отдельные сервера. Логичнее будет расположить его на роутере 3 или 4, так как у них есть доступ ко всем вланам, где есть компьютеры.

```
[doctor@tardis]~$ eval $(ssh-agent )
Agent pid 1415423
[doctor@tardis]~$ ssh-add .ssh/servers .ssh/jumphost
Enter passphrase for .ssh/servers:
Identity added: .ssh/servers (doctor@tardis)
Identity added: .ssh/jumphost (doctor@tardis)
[doctor@tardis]~$ ssh router3
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Sun Jan 23 19:18:23 2022 from 10.0.5.101
[user@router3 ~]$
```

Сперва подключимся к роутеру 3:

```
eval $(ssh-agent )
ssh-add .ssh/servers .ssh/jumphost
ssh router3
```

```
[user@router3 ~]$ sudo dnf install dhcp-server
Last metadata expiration check: 19:46:44 ago on Sat 29 Jan 2022 02:36:30 PM +04.
Dependencies resolved.
=====
Package                        Architecture      Version           Repository        Size
=====
Installing:
dhcp-server                    x86_64            12:4.3.6-45.el8   baseos            529 k
Installing dependencies:
bind-export-libs               x86_64            32:9.11.26-6.el8   baseos            1.1 M
dhcp-common                    x86_64            12:4.3.6-45.el8   baseos            206 k
dhcp-libs                      x86_64            12:4.3.6-45.el8   baseos            147 k
=====
Transaction Summary
=====
Install 4 Packages

Total download size: 2.0 M
Installed size: 4.6 M
Is this ok [y/N]: y
```

Затем установим пакет dhcp-server:

```
sudo dnf install dhcp-server
```

На разных дистрибутивах пакет может называться по-разному, но всё это одна и та же реализация функционала DHCP от организации ISC - Internet Systems Consortium. Они также разрабатывают DNS сервер, который в будущем мы будем разбирать.

```
[user@router3 ~]$ sudo ls /etc/dhcp/
[sudo] password for user:
dhclient.d  dhcpd6.conf  dhcpd.conf
[user@router3 ~]$ sudo ls /etc/dhcp/dhclient.d
chrony.sh
[user@router3 ~]$ sudo cat /etc/dhcp/dhclient.d/chrony.sh
#!/bin/bash

SERVERFILE=$SAVEDIR/chrony.servers.$interface

chrony_config() {
    # Disable modifications if called from a NM dispatcher script
    [ -n "$NM_DISPATCHER_ACTION" ] && return 0

    rm -f "$SERVERFILE"
    if [ "$PEERntp" != "no" ]; then
        for server in $new_ntp_servers; do
            echo "$server ${NTPSERVERARGS:-iburst}" >> "$SERVERFILE"
        done
        /usr/libexec/chrony-helper update-daemon || :
    fi
}
```

Настройки лежат в директории /etc/dhcp :

```
sudo ls /etc/dhcp
```


Основной файл настроек DHCP сервера это `dhcpcd.conf`, также есть версия `dhcpcd6.conf` для ipv6 сетей, ну и директория `dhclient.d`:

```
sudo ls /etc/dhcp/dhclient.d
```

Где лежат настройки dhcp клиента. Например, здесь есть скрипт `chrony.sh`:

```
sudo cat /etc/dhcp/dhclient.d/chrony.sh
```

который настраивает демон ntp `chrony` по адресам ntp серверов, полученных по dhcp. Т.е. мы подключимся к сети, получим адрес и прочую информацию о сети, в том числе адреса NTP серверов, и этот скрипт настраивает `chrony`, чтобы пользователям не приходилось вручную прописывать их.

В директории `/etc/dhcp` мог быть ещё файл `dhclient.conf`, который связан с пакетом `dhclient`. В нём можно было бы прописывать недостающие настройки сети, которые не выдаст DHCP сервер, либо как-то перезаписать, что выдали, либо указать, что нужно послать DHCP серверу при запросе. Но это можно сделать в профиле `NetworkManager`, оставив `ipv4.method auto`, но при этом добавив нужную информацию.

```
[user@router3 ~]$ cat /etc/NetworkManager/
conf.d/                dnsmasq.d/            NetworkManager.conf
dispatcher.d/          dnsmasq-shared.d/    system-connections/
[user@router3 ~]$ cat /etc/NetworkManager/dispatcher.d/
20-chrony-dhcp         no-wait.d/            pre-up.d/
20-chrony-onoffline   pre-down.d/
[user@router3 ~]$ cat /etc/NetworkManager/dispatcher.d/20-chrony-dhcp
#!/bin/sh
# This is a NetworkManager dispatcher script for chronyd to update
# its NTP sources passed from DHCP options. Note that this script is
# specific to NetworkManager-dispatcher due to use of the
# DHCP4_NTP_SERVERS environment variable.

export LC_ALL=C

interface=$1
action=$2
```

Например, тот же скрипт настройки `chrony` реализован через сам `NetworkManager`. У него основная директория настроек - `/etc/NetworkManager`:

```
ls /etc/NetworkManager/
```

И здесь в директории `dispatcher.d` можно найти скрипт для `chrony`:

```
cat /etc/NetworkManager/dispatcher.d/20-chrony-dhcp
```

Я всё это к тому, что в dhcp можно настраивать не только сервер, но и клиент. Если у вас `NetworkManager`, то настройки dhcp клиента нужно делать через него. На каких-то системах может не быть `NetworkManager`-а, и там, скорее всего, настройки клиента нужно будет делать через конфиг `/etc/dhcp/dhclient.conf`.

```
[user@router3 ~]$ sudo cat /etc/dhcp/dhcpd.conf
#
# DHCP Server Configuration file.
#   see /usr/share/doc/dhcp-server/dhcpd.conf.example
#   see dhcpd.conf(5) man page
#
[user@router3 ~]$
```

Теперь перейдём к настройкам сервера. Посмотрим файл настроек:

```
sudo cat /etc/dhcp/dhcpd.conf
```

Тут настроек нет, но есть подсказка, что примеры настроек можно взять из файла dhcpd.conf.example.

```
[user@router3 ~]$ sudo cp /usr/share/doc/dhcp-server/dhcpd.conf.example /etc/dhcp/dhcpd.conf
[user@router3 ~]$ sudo nano /etc/dhcp/dhcpd.conf
```

Давайте скопируем пример настроек и заменим основной файл:

```
sudo cp /usr/share/doc/dhcp-server/dhcpd.conf.example /etc/dhcp/dhcpd.conf
```

После чего зайдём и уберём лишнее:

```
sudo nano /etc/dhcp/dhcpd.conf
```

```
GNU nano 2.9.8 /etc/dhcp/dhcpd.conf
# dhcpd.conf
#
# Sample configuration file for ISC dhcpd
#
# option definitions common to all supported networks...
option domain-name "example.org";
option domain-name-servers ns1.example.org, ns2.example.org;

default-lease-time 600;
max-lease-time 7200;

# Use this to enable / disable dynamic dns updates globally.
#ddns-update-style none;

# If this DHCP server is the official DHCP server for the local
# network, the authoritative directive should be uncommented.
#authoritative;
```

Обратите внимание, что здесь все строчки с параметрами заканчиваются на точку с запятой. Будете что-то менять - не забывайте ставить её. Также учтите, что конфигурация состоит из общих настроек, которые мы сейчас видим, а чуть ниже для каждой сети будут свои локальные секции. То что мы будем указывать здесь - можно будет перезаписывать для каждой сети.

Некоторые опции я пока разбирать не буду, например, `option domain-name` - она связана с темой DNS. Когда будем проходить DNS - вернёмся сюда и донастроим всё необходимое, а пока уберём.

Опцию `domain-name-servers` оставим. Здесь указаны адреса DNS серверов, и, обратите внимание - несколько значений разделяются запятыми. Хотя в примере они указаны по именам, на деле DHCP сервер сначала сам обращается к DNS серверам, превращает эти имена в IP адреса и клиентам отдаёт их в виде IP адресов.

```
GNU nano 2.9.8 /etc/dhcp/dhcpd.conf
option domain-name-servers 1.1.1.1, 8.8.8.8;
default-lease-time 600;
max-lease-time 7200;

# If this DHCP server is the official DHCP server for the local
# network, the authoritative directive should be uncommented.
#authoritative;

# Use this to send dhcp log messages to a different log file (you also
# have to hack syslog.conf to complete the redirection).
log-facility local7;

# No service will be given on this subnet, but declaring it helps the
# DHCP server to understand the network topology.

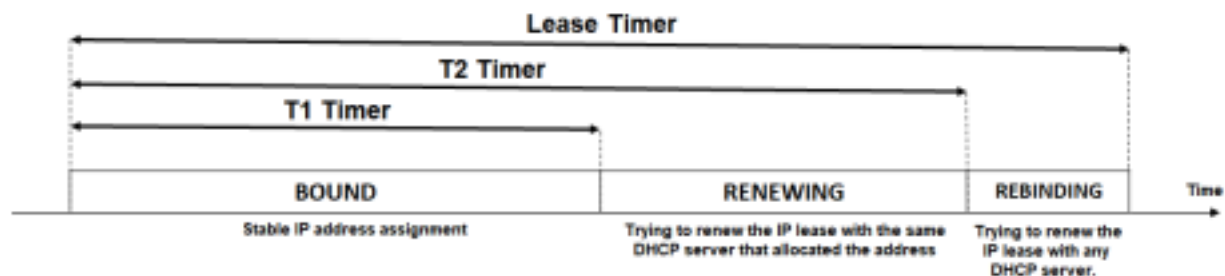
subnet 10.152.187.0 netmask 255.255.255.0 {
}

# This is a very basic subnet declaration.
```

Так как у нас пока своих DNS серверов нет, заменим эти адреса на публичные - единички и восьмёрки:

```
option domain-name-servers 1.1.1.1, 8.8.8.8;
```

Теперь касательно lease time.



IP адрес выдаётся не навсегда, а на какое-то время, в зависимости от настроек сервера. Представьте, что у вас есть гостевая сеть, в которой каждый день кто-то подключается телефоном или ноутбуком. Если всем устройствам выдавать адреса навсегда, то адреса быстро закончатся. Поэтому логичнее, чтобы адрес был привязан к устройству на какое-то время, пока устройство активно. Но при этом мы не хотели бы, чтобы в рабочей сети IP адрес менялся каждый раз, когда мы выключаем устройство или перезагружаем сеть. Вот это время, на которое выдаётся IP адрес, называется lease time - время аренды. В гостевой сети, где постоянно подключаются новые устройства, логичнее делать небольшой lease time - скажем, 2 часа. Но это не значит, что через 2 часа на устройстве сменится адрес. Пока устройство подключено к сети, оно будет периодически просить у DHCP сервера продлить это время.

Но если устройство отключить от сети на два часа, то DHCP сервер освободит этот адрес и может его выдать кому-то другому.

Table 3.1 Examples of Lease Times

Type of Subnet	Primary Use	Default Lease
GSI training lab	Students with laptops	One class period plus 10 minutes
Conference room	Visitors with laptops	2 hours
"Hotel" office	Staff members who use it daily	12 hours
Telecommuters	DSL/cable service	7 days
Staff offices	Permanent staff members	30 days
Central servers	Organization servers	3 months

В рабочей сети устройства меняются реже и чтобы лишний раз не менять адреса - lease time делают большим - скажем, месяц. Чтобы, пока устройство было выключено, никто его IP не взял. Вот примерная табличка из интернета, где и какой lease time выставлять. На деле каждая компания сама определяет свои значения. Хотя обычно там, где настроен DHCP, администратор должен выстроить сеть таким образом, чтобы смена IP адреса не вызывала каких-то проблем. Т.е. сразу надо иметь ввиду, что у устройств может смениться IP адрес. Но это не обязательно, DHCP сервер позволяет сделать так, чтобы конкретное устройство всегда получало один и тот же IP адрес. Но мы к этому ещё вернёмся.

```

GNU nano 2.9.8 /etc/dhcp/dhcpd.conf
option domain-name-servers 1.1.1.1, 8.8.8.8;
default-lease-time 600;
max-lease-time 7200;

# If this DHCP server is the official DHCP server for the local
# network, the authoritative directive should be uncommented.
#authoritative;

# Use this to send dhcp log messages to a different log file (you also
# have to hack syslog.conf to complete the redirection).
log-facility local7;

# No service will be given on this subnet, but declaring it helps the
# DHCP server to understand the network topology.

subnet 10.152.187.0 netmask 255.255.255.0 {
}

# This is a very basic subnet declaration.

```

Так вот, мы тут видим два значения:

```

default-lease-time 600;
max-lease-time 7200;

```

На самом деле их 3 - есть ещё min-lease-time, просто в примере он не указан. Т.е. максимальное, минимальное и дефолтное значение времени аренды. Значения в секундах, т.е. 600 - это 10 минут, а 7200 - 2 часа. При нормальных обстоятельствах будет выдаваться дефолтное значение. Если вдруг dhcp клиент сам попросит определённый lease time, то максимальное и минимальное значение это будут ограничивать.

Допустим, у нас есть гостевая сеть, для которой вы выделили 10 адресов и выставили lease time 2 часа. Придёт какой-нибудь человек с 10 устройствами и от dhcp клиента запросит lease time не на 2 часа, а на 3 месяца. И больше никто к этой сети не сможет подключиться, потому что в ближайшие 3 месяца все 10 адресов будут заняты этим нехорошим человеком. Да, конечно, это можно исправить, но лучше просто избежать этой проблемы.

```

GNU nano 2.9.8 /etc/dhcp/dhcpd.conf
option domain-name-servers 1.1.1.1, 8.8.8.8;
min-lease-time 3600;
default-lease-time 7200;
max-lease-time 10800;

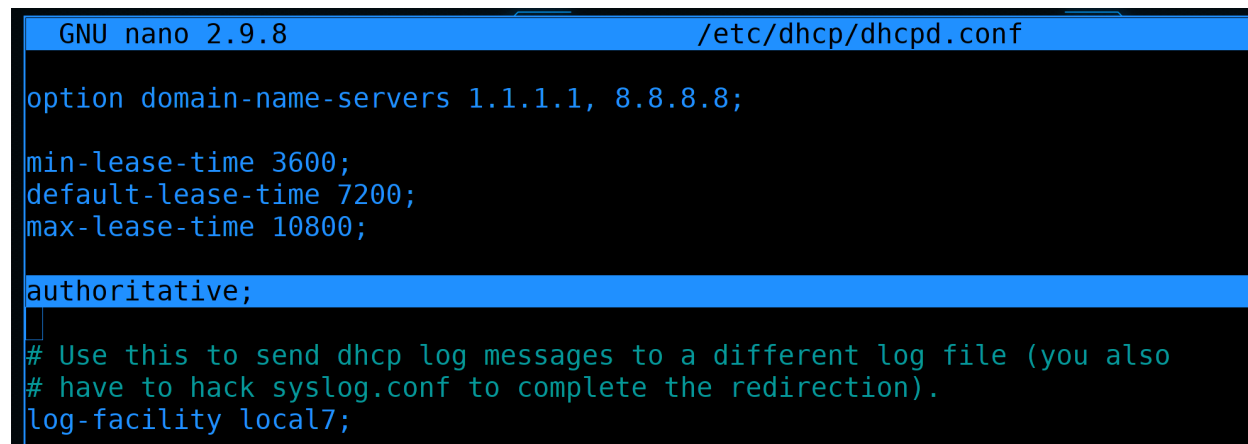
# If this DHCP server is the official DHCP server for the local
# network, the authoritative directive should be uncommented.
#authoritative;

```

Поэтому и нужны эти ограничения - пусть адрес по умолчанию выдаётся на 2 часа, если клиент захочет сразу много, то максимум 3 часа, а минимум 1:

```
min-lease-time 3600;
default-lease-time 7200;
max-lease-time 10800;
```

Но, как я сказал, в разных сетях lease time может отличаться. То что мы указываем в начале файла - это шаблонные настройки, а чуть ниже мы для каждой подсети определим более подходящие сроки.



```
GNU nano 2.9.8 /etc/dhcp/dhcpd.conf

option domain-name-servers 1.1.1.1, 8.8.8.8;

min-lease-time 3600;
default-lease-time 7200;
max-lease-time 10800;

authoritative;

# Use this to send dhcp log messages to a different log file (you also
# have to hack syslog.conf to complete the redirection).
log-facility local7;
```

Опция `authoritative` - определяет, является ли DHCP сервер авторитативным. Что это вообще значит? В каждой сети должен быть один главный DHCP сервер, который отвечает за эту сеть. Да, технически может быть несколько несвязанных серверов, но никто так не делает. Так вот, особенность авторитативного DHCP сервера в том, что он может отказывать в IP адресе и заставляя клиента запрашивать другой адрес. Это связано с тем, что иногда клиенты запоминают адрес из другой сети и пытаются принести его в эту сеть.

Условно, ноутбук был подключен к домашней сети, потом его принесли на работу и он пытается сказать рабочему DHCP серверу, что я хочу продолжить использовать домашний адрес. Авторитативный DHCP сервер имеет право посылать DHCPNAK запрос, тем самым отказывая клиенту в этом адресе и заставляя запросить новый адрес - рабочий. Если бы у нас в сети не было авторитативного сервера, то клиент послал бы запрос, его просто проигнорировали бы и клиент остался бы с домашним адресом, пока lease time домашнего адреса не закончится. И только тогда ноутбук запросил бы новый адрес. А это может занять час и в это время пользователь будет доставать администратора, мол, у меня ничего не работает.

У нас есть DHCP сервер, он у нас единственный, поэтому мы делаем его авторитативным.

```

GNU nano 2.9.8 /etc/dhcp/dhcpd.conf

option domain-name-servers 1.1.1.1, 8.8.8.8;

min-lease-time 3600;
default-lease-time 7200;
max-lease-time 10800;

authoritative;

# Use this to send dhcp log messages to a different log file (you also
# have to hack syslog.conf to complete the redirection).
log-facility local7;

# No service will be given on this subnet, but declaring it helps the
# DHCP server to understand the network topology.

subnet 10.152.187.0 netmask 255.255.255.0 {
}

```

Следующая опция - `log-facility`. Помните тему про логи и syslog? Он позволяет фильтровать логи по facility, там есть шаблонные и кастомные - `local0`, `local1` и т.п. И, допустим, в `rsyslog` можно настроить, чтобы разные facility писались в разные файлы. Так вот, мы для `dhcp` можем задать какой-то facility или оставить `local7`. А потом настроить в `rsyslog`, чтобы логи `dhcp` сервера писались в определённый файл. Оставим эту опцию как есть:

```
log-facility local7;
```

```

GNU nano 2.9.8 /etc/dhcp/dhcpd.conf

option domain-name-servers 1.1.1.1, 8.8.8.8;

min-lease-time 3600;
default-lease-time 7200;
max-lease-time 10800;

authoritative;

log-facility local7;

# No service will be given on this subnet, but declaring it helps the
# DHCP server to understand the network topology.

subnet 10.152.187.0 netmask 255.255.255.0 {
}

```

Дальше у нас дан пример пустой секции. DHCP сервер ничего не раздаёт в этой сети. Это больше нужно для общего понимания сети.

```

GNU nano 2.9.8 /etc/dhcp/dhcpd.conf

option domain-name-servers 1.1.1.1, 8.8.8.8;

min-lease-time 3600;
default-lease-time 7200;
max-lease-time 10800;

authoritative;

log-facility local7;

subnet 10.0.1.0 netmask 255.255.255.0 {
    not authoritative;
}

```

У нас есть сеть, в которой не будет DHCP - это влан 1, между внешними и внутренними роутерами. Давайте пропишем эту подсеть. Ну и так как мы в общих настройках включили authoritative, а в этой подсети мы не являемся авторитативным сервером, здесь мы эту опцию уберём:

```

subnet 10.0.1.0 netmask 255.255.255.0 {
    not authoritative;
}

```

```

GNU nano 2.9.8 /etc/dhcp/dhcpd.conf

option domain-name-servers 1.1.1.1, 8.8.8.8;

min-lease-time 3600;
default-lease-time 7200;
max-lease-time 10800;

authoritative;

log-facility local7;

subnet 10.0.1.0 netmask 255.255.255.0 {
    not authoritative;
}

subnet 10.254.239.0 netmask 255.255.255.224 {
    range 10.254.239.10 10.254.239.20;
    option routers rtr-239-0-1.example.org, rtr-239-0-2.example.org;
}

```

Чуть ниже дан пример подсети, в которой dhcp сервер работает. Самое главное - для неё указан промежуток адресов в опции **range**. Как видите, там адреса от 10 до 20 - т.е. dhcp сервер раздаёт всего 10 адресов. Всегда стоит брать промежуток с запасом. Без него в какой-то момент выйдет проблема - закончатся адреса, компьютеры не смогут получить ip адрес и попасть в сеть.

Дальше идёт опция **routers** - в ней указывается адрес gateway, который получают компьютеры. Хотя тут приведены в примере два гейтвея и компьютеры получают два default gateway-а, всё равно будет использоваться только один. И, как и в случае с адресами DNS, хоть тут и указаны имена, DHCP сервер перед выдачей адреса превратит их в IP адреса.

```

GNU nano 2.9.8 /etc/dhcp/dhcpd.conf

### DMZ
subnet 10.0.2.0 netmask 255.255.255.0 {
    range 10.0.2.10 10.0.2.50;
    option routers 10.0.2.1;
}
### Users
subnet 10.0.3.0 netmask 255.255.255.0 {
    range 10.0.3.10 10.0.3.230;
    option routers 10.0.3.1;
}
### Servers
subnet 10.0.4.0 netmask 255.255.255.0 {
    range 10.0.4.10 10.0.4.200;
    option routers 10.0.4.1;
}
### Management
subnet 10.0.5.0 netmask 255.255.255.0 {
    range 10.0.5.10 10.0.5.50;
    option routers 10.0.5.1;
}

```

Давайте пропишем здесь свои подсети. У нас их пока 4, не считая первого влана. В каких-то сетях у нас будет больше хостов, в каких-то меньше - от этого зависит **range**. И адрес роутера для каждой подсети свой.

```

### DMZ
subnet 10.0.2.0 netmask 255.255.255.0 {
    range 10.0.2.10 10.0.2.50;
    option routers 10.0.2.1;
}
### Users
subnet 10.0.3.0 netmask 255.255.255.0 {
    range 10.0.3.10 10.0.3.230;
    option routers 10.0.3.1;
}
### Servers
subnet 10.0.4.0 netmask 255.255.255.0 {
    range 10.0.4.10 10.0.4.200;
    option routers 10.0.4.1;
}
### Management

```

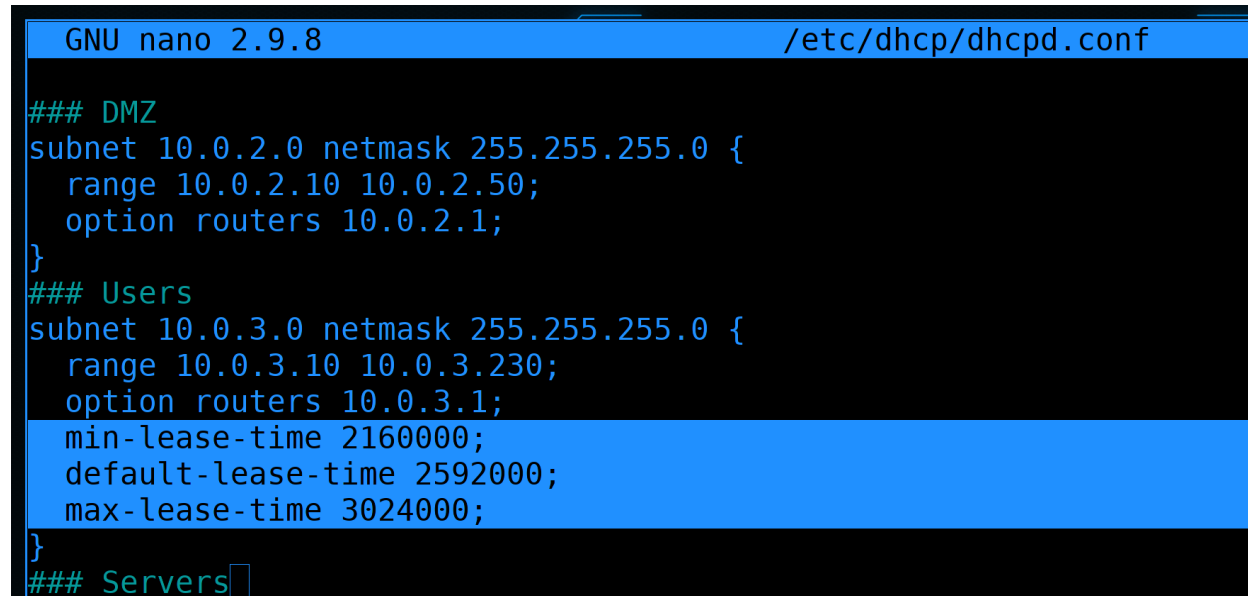
(continues on next page)

(продолжение с предыдущей страницы)

```

subnet 10.0.5.0 netmask 255.255.255.0 {
    range 10.0.5.10 10.0.5.50;
    option routers 10.0.5.1;
}

```



```

GNU nano 2.9.8 /etc/dhcp/dhcpd.conf

### DMZ
subnet 10.0.2.0 netmask 255.255.255.0 {
    range 10.0.2.10 10.0.2.50;
    option routers 10.0.2.1;
}

### Users
subnet 10.0.3.0 netmask 255.255.255.0 {
    range 10.0.3.10 10.0.3.230;
    option routers 10.0.3.1;
    min-lease-time 2160000;
    default-lease-time 2592000;
    max-lease-time 3024000;
}

### Servers


```

Теперь давайте поправлять. Начнём с lease-time. Из этих 4-ёх сетей 3 относятся к серверам и только одна к пользователям. Поэтому для пользователей пропишем внутри subnet lease-time на 25, 30 и 35 дней:

```

### Users
subnet 10.0.3.0 netmask 255.255.255.0 {
    range 10.0.3.10 10.0.3.230;
    option routers 10.0.3.1;
    min-lease-time 2160000;
    default-lease-time 2592000;
    max-lease-time 3024000;
}

```



```

GNU nano 2.9.8 /etc/dhcp/dhcpd.conf

option domain-name-servers 1.1.1.1, 8.8.8.8;

min-lease-time 6912000;
default-lease-time 7776000;
max-lease-time 8640000;

authoritative;
log-facility local7;

subnet 10.0.1.0 netmask 255.255.255.0 {
    not authoritative;
}

```

А сверху, где глобальные настройки, поправим lease-time на 80, 90 и 100 дней:

```
min-lease-time 6912000;
default-lease-time 7776000;
max-lease-time 8640000;
```

Просто интересный факт: самое большое значение - примерно 135 лет.

```
GNU nano 2.9.8 /etc/dhcp/dhcpd.conf

option domain-name-servers 1.1.1.1, 8.8.8.8;

min-lease-time 6912000;
default-lease-time 7776000;
max-lease-time 8640000;

authoritative;
log-facility local7;

option ms-classless-static-routes code 249 = array of unsigned integer 8;
option rfc3442-classless-static-routes code 121 = array of unsigned integer 8;

subnet 10.0.1.0 netmask 255.255.255.0 {
    not authoritative;
}
```

Теперь касательно DMZ. Как вы возможно помните, чтобы хост из DMZ мог достигать до внутренней сети, мы прописали на нём статичный маршрут, мол 10.0.0.0/8 находится за роутером 3. И мы хотели бы раздать этот статический маршрут по DHCP. Будет не очень банально, но копияста вам поможет. И так, в общей секции прописываем две опции:

```
option ms-classless-static-routes code 249 = array of unsigned integer 8;
option rfc3442-classless-static-routes code 121 = array of unsigned integer 8;
```

Они не изменчивы, просто пишем или копируем как есть.

```
GNU nano 2.9.8 /etc/dhcp/dhcpd.conf

subnet 10.0.1.0 netmask 255.255.255.0 {
    not authoritative;
}

### DMZ
subnet 10.0.2.0 netmask 255.255.255.0 {
    range 10.0.2.10 10.0.2.50;
    option routers 10.0.2.1;
    option ms-classless-static-routes 8, 10, 10,0,2,254;
    option rfc3442-classless-static-routes 8, 10, 10,0,2,254;
}

### Home
```

Дальше в секции DMZ прописываем ещё две опции:

```
option ms-classless-static-routes 8, 10, 10,0,2,254;
option rfc3442-classless-static-routes 8, 10, 10,0,2,254;
```

Здесь уже цифры имеют значение. Вспомните маршрут:

```
10.0.0.0/8 via 10.0.2.254
```

- 8 - это маска подсети
- 10 - это адрес целевой сети, просто мы убрали нули
- 10,0,2,254 - это адрес роутера - просто вместо точек запятые.

```
GNU nano 2.9.8 /etc/dhcp/dhcpd.conf

range 10.0.4.10 10.0.4.200;
option routers 10.0.4.1;
}
### Management
subnet 10.0.5.0 netmask 255.255.255.0 {
    range 10.0.5.10 10.0.5.50;
    option routers 10.0.5.1;
}

host fantasia {
    hardware ethernet 08:00:07:26:c0:a5;
    fixed-address fantasia.example.com;
}

```

У DHCP есть множество различных опций, всё за раз мы не разберём, да и без необходимости просто так их прописывать смысла нет. Будем проходить соответствующие темы - будем добавлять на DHCP. А пока давайте сотрём всё что ниже и оставим только секцию `host fantasia`. К ней мы ещё вернёмся. А пока сохраним и выйдем из файла.

```
[user@router3 ~]$ sudo nano /etc/dhcp/dhcpd.conf
[sudo] password for user:
[user@router3 ~]$ sudo systemctl enable --now dhcpd
[sudo] password for user:
Created symlink /etc/systemd/system/multi-user.target.wants/dhcpd.service → /usr/lib/systemd/system/dhcpd.service.
[user@router3 ~]$ sudo systemctl status dhcpd
● dhcpd.service - DHCPv4 Server Daemon
   Loaded: loaded (/usr/lib/systemd/system/dhcpd.service; enabled; vendor preset: disabled)
   Active: active (running) since Sun 2022-01-30 14:41:17 +04; 20s ago
     Docs: man:dhcpd(8)
           man:dhcpd.conf(5)
  Main PID: 2977 (dhcpd)
    Status: "Dispatching packets..."
     Tasks: 1 (limit: 2735)
    Memory: 9.9M
   CGroup: /system.slice/dhcpd.service
           └─2977 /usr/sbin/dhcpd -f -cf /etc/dhcp/dhcpd.conf -user dhcpd -group dhcpd --no-pid
```

Демон dhcp сервера называется `dhcpd`. Давайте добавим его в автозагрузку и запустим:

```
sudo systemctl enable --now dhcpd
```

А также посмотрим статус:

```
sudo systemctl status dhcpd
```

Как видите, он запущен.

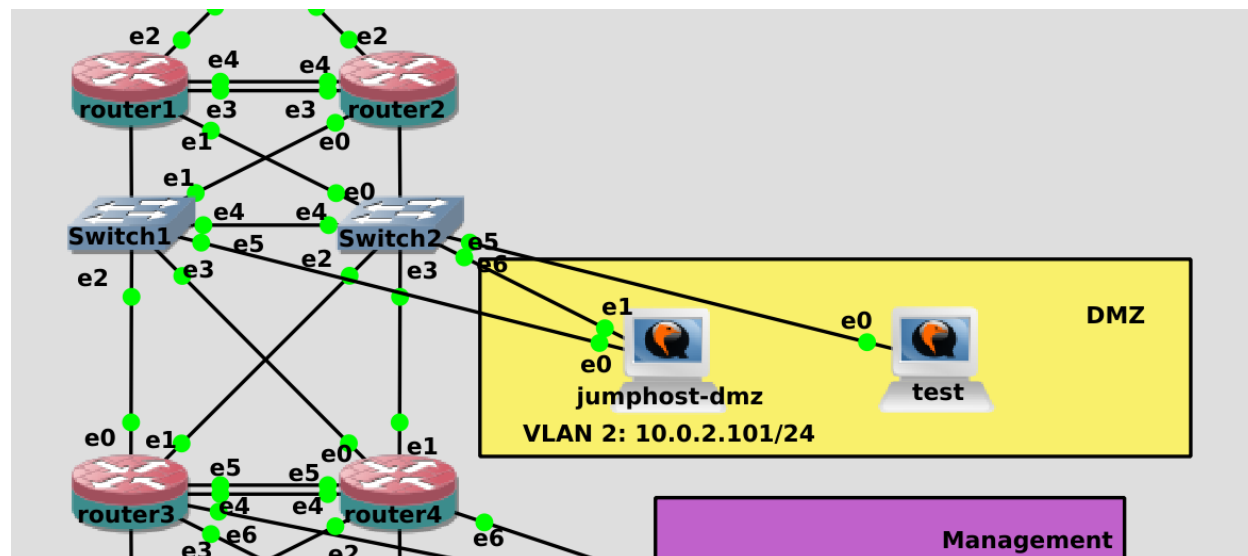
```
[user@router3 ~]$ sudo journalctl -eu dhcpd
Jan 30 14:41:17 router3 dhcpd[2977]: to which interface ens6 is attached. **
Jan 30 14:41:17 router3 dhcpd[2977]:
Jan 30 14:41:17 router3 dhcpd[2977]:
Jan 30 14:41:17 router3 dhcpd[2977]: No subnet declaration for ens5 (no IPv4 addresses).
Jan 30 14:41:17 router3 dhcpd[2977]: ** Ignoring requests on ens5. If this is not what
Jan 30 14:41:17 router3 dhcpd[2977]: you want, please write a subnet declaration
Jan 30 14:41:17 router3 dhcpd[2977]: in your dhcpd.conf file for the network segment
Jan 30 14:41:17 router3 dhcpd[2977]: to which interface ens5 is attached. **
Jan 30 14:41:17 router3 dhcpd[2977]:
Jan 30 14:41:17 router3 dhcpd[2977]: No subnet declaration for ens4 (no IPv4 addresses).
Jan 30 14:41:17 router3 dhcpd[2977]: ** Ignoring requests on ens4. If this is not what
Jan 30 14:41:17 router3 dhcpd[2977]: you want, please write a subnet declaration
Jan 30 14:41:17 router3 dhcpd[2977]: in your dhcpd.conf file for the network segment
Jan 30 14:41:17 router3 dhcpd[2977]: to which interface ens4 is attached. **
Jan 30 14:41:17 router3 dhcpd[2977]:
```

Давайте ещё посмотрим логи:

```
sudo journalctl -eu dhcpd
```

Здесь есть предупреждения, что для таких-то интерфейсов не заданы настройки в dhcpd, но ничего критичного в этом нет. Это наши slave интерфейсы для тиминга, естественно, на них нет адресов и ничего плохого в этом нет.

Стоит отметить, что на файрволе ничего открывать не надо, так как процесс DORA работает на втором уровне сети, всё общение происходит по мак адресам. И хотя в пакетах содержится информация об IP адресах, сами они не прописаны ни как source, ни как destination.



Давайте тестировать. Подключим новый компьютер к DMZ сети и включим. Так как по-умолчанию хосты берут IP по DHCP, он должен сразу взять IP адрес.

```
Jan 30 14:41:17 router3 dhcpd[2977]: Server starting service.
Jan 30 14:50:07 router3 dhcpd[2977]: DHCPDISCOVER from 0c:9e:76:54:00:00 via team0.2
Jan 30 14:50:08 router3 dhcpd[2977]: DHCP OFFER on 10.0.2.10 to 0c:9e:76:54:00:00 (alma) via team0.2
Jan 30 14:50:08 router3 dhcpd[2977]: DHCPREQUEST for 10.0.2.10 (10.0.2.252) from 0c:9e:76:54:00:00 (alma) via team0.2
Jan 30 14:50:08 router3 dhcpd[2977]: DHCPACK on 10.0.2.10 to 0c:9e:76:54:00:00 (alma) via team0.2
[user@router3 ~]$ sudo journalctl -eu dhcpd --no-pager
```

Поэтому можем глянуть логи dhcp сервера:

```
sudo journalctl -eu dhcpd --no-pager
```

И, как видно, появилось 4 сообщения DORA - DISCOVER, OFFER, REQUEST и ACK. Здесь же виден мак адрес нового устройства и выданный IP адрес.

```
[user@router3 ~]$ ssh 10.0.2.10
The authenticity of host '10.0.2.10 (10.0.2.10)' can't be established.
ECDSA key fingerprint is SHA256:mU2Gv7JHASPJY4L8tPS8Ed4ChSyf/6zR00X0nli5WN4.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.2.10' (ECDSA) to the list of known hosts.
user@10.0.2.10's password:
Activate the web console with: systemctl enable --now cockpit.socket

[user@alma ~]$ ip --brief a
lo                UNKNOWN          127.0.0.1/8 ::1/128
ens3              UP              10.0.2.10/24 fe80::2e91:40b8:d7:a634/64
ens4              DOWN
ens5              DOWN
ens6              DOWN
[user@alma ~]$ ip ro sh
default via 10.0.2.1 dev ens3 proto dhcp metric 100
10.0.0.0/8 via 10.0.2.254 dev ens3 proto dhcp metric 100
10.0.2.0/24 dev ens3 proto kernel scope link src 10.0.2.10 metric 100
[user@alma ~]$ cat /etc/resolv.conf
# Generated by NetworkManager
nameserver 1.1.1.1
nameserver 8.8.8.8
[user@alma ~]$
```

Попробуем подключиться к нему:

```
ssh 10.0.2.10
```

Подключение прошло успешно. Давайте посмотрим ip адрес, маршруты и dns сервера:

```
ip --brief a
ip ro sh
cat /etc/resolv.conf
```

У нас всё автоматом прописалось - и IP адрес, и гейтвей, и статичный маршрут, и dns сервера.

```
[user@alma ~]$ ip a show ens3
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 0c:9e:76:54:00:00 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.10/24 brd 10.0.2.255 scope global dynamic noprefixroute ens3
        valid_lft 7775602sec preferred_lft 7775602sec
    inet6 fe80::2e91:40b8:d7:a634/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
[user@alma ~]$ ip a show ens3
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 0c:9e:76:54:00:00 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.10/24 brd 10.0.2.255 scope global dynamic noprefixroute ens3
        valid_lft 7775515sec preferred_lft 7775515sec
    inet6 fe80::2e91:40b8:d7:a634/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
[user@alma ~]$
```

Если посмотрим детальнее интерфейс:

```
ip a show ens3
```

то увидим, что мы получили IP по DHCP - потому что написано `dynamic` - и видим когда у нас заканчивается `lease time` - столько-то секунд.

```
ip a show ens3
```

Если ещё раз проверить - времени осталось меньше. Но, если помните, это время будет автоматом продлеваться, пока мы подключены к сети и видим `dhcp` сервер.

```
[user@alma ~]$ nmcli connection show Wired\ connection\ 1 | grep DHCP4
DHCP4.OPTION[1]:      dhcp_lease_time = 7776000
DHCP4.OPTION[2]:      dhcp_server_identifier = 10.0.2.252
DHCP4.OPTION[3]:      domain_name_servers = 1.1.1.1 8.8.8.8
DHCP4.OPTION[4]:      expiry = 1651315806
DHCP4.OPTION[5]:      ip_address = 10.0.2.10
DHCP4.OPTION[6]:      ms_classless_static_routes = 10.0.0.0/8 10.0.2.254
DHCP4.OPTION[7]:      requested_broadcast_address = 1
DHCP4.OPTION[8]:      requested_domain_name = 1
DHCP4.OPTION[9]:      requested_domain_name_servers = 1
DHCP4.OPTION[10]:     requested_domain_search = 1
DHCP4.OPTION[11]:     requested_host_name = 1
DHCP4.OPTION[12]:     requested_interface_mtu = 1
DHCP4.OPTION[13]:     requested_ms_classless_static_routes = 1
DHCP4.OPTION[14]:     requested_nis_domain = 1
```

Чуть больше информации о полученных опциях можно увидеть через сам `NetworkManager`, если посмотреть профиль:

```
nmcli connection show Wired\ connection\ 1 | grep DHCP4
```

Тут у нас и адрес DHCP сервера, и время аренды, всякие маршруты и прочая информация. То есть всё работает как надо.

```
[user@router3 ~]$ sudo cat /var/lib/dhcpd/dhcpd.leases
[sudo] password for user:
# The format of this file is documented in the dhcpd.leases(5) manual page.
# This lease file was written by isc-dhcp-4.3.6

# authoring-byte-order entry is generated, DO NOT DELETE
authoring-byte-order little-endian;

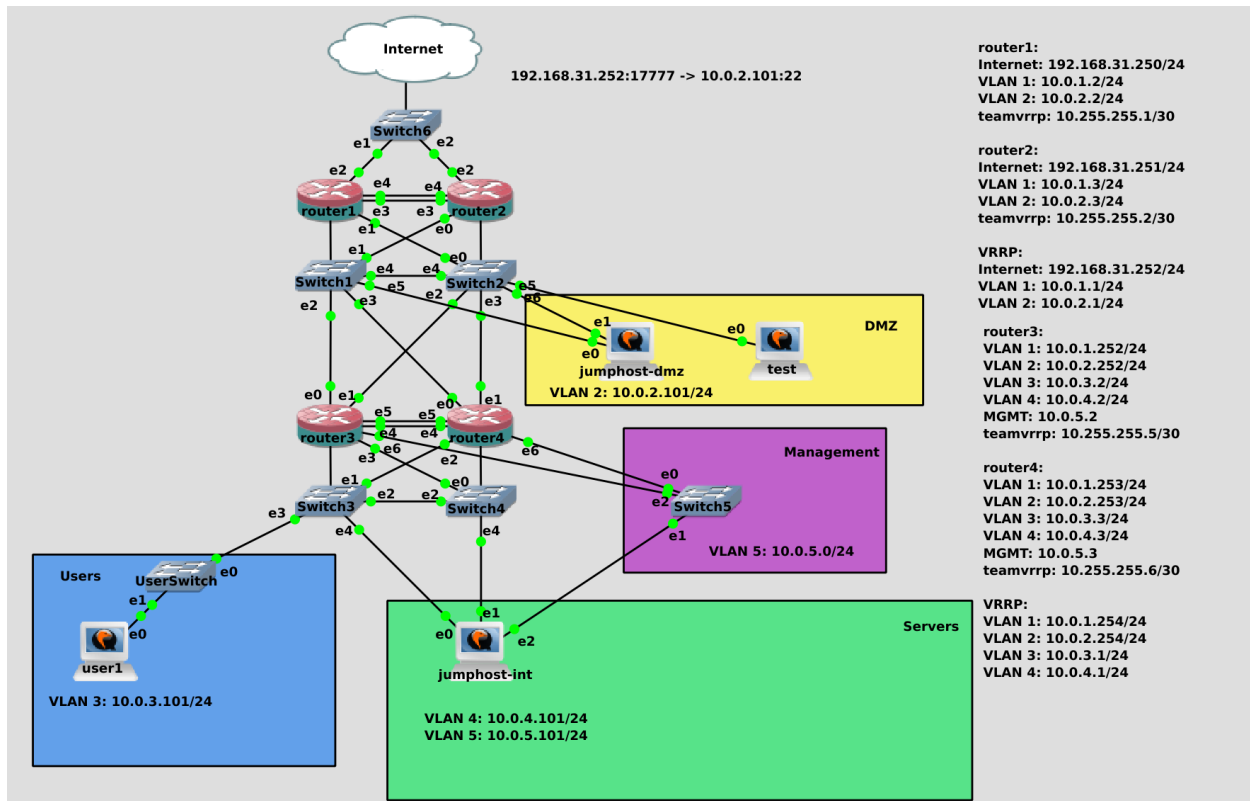
server-duid "\000\001\000\001)\211'\315\014\340\317I\000\000";

lease 10.0.2.10 {
  starts 0 2022/01/30 10:50:08;
  ends 6 2022/04/30 10:50:08;
  cltt 0 2022/01/30 10:50:08;
  binding state active;
  next binding state free;
  rewind binding state free;
  hardware ethernet 0c:9e:76:54:00:00;
  uid "\001\014\236vT\000\000";
  client-hostname "alma";
}
[user@router3 ~]$
```

Вернёмся на DHCP сервер. Он хранит информацию о текущих выданных адресах в файле `/var/lib/dhcpd/dhcpd.leases`:

```
sudo cat /var/lib/dhcpd/dhcpd.leases
```

Тут у нас видно кто когда запросил адрес и когда заканчивается время.

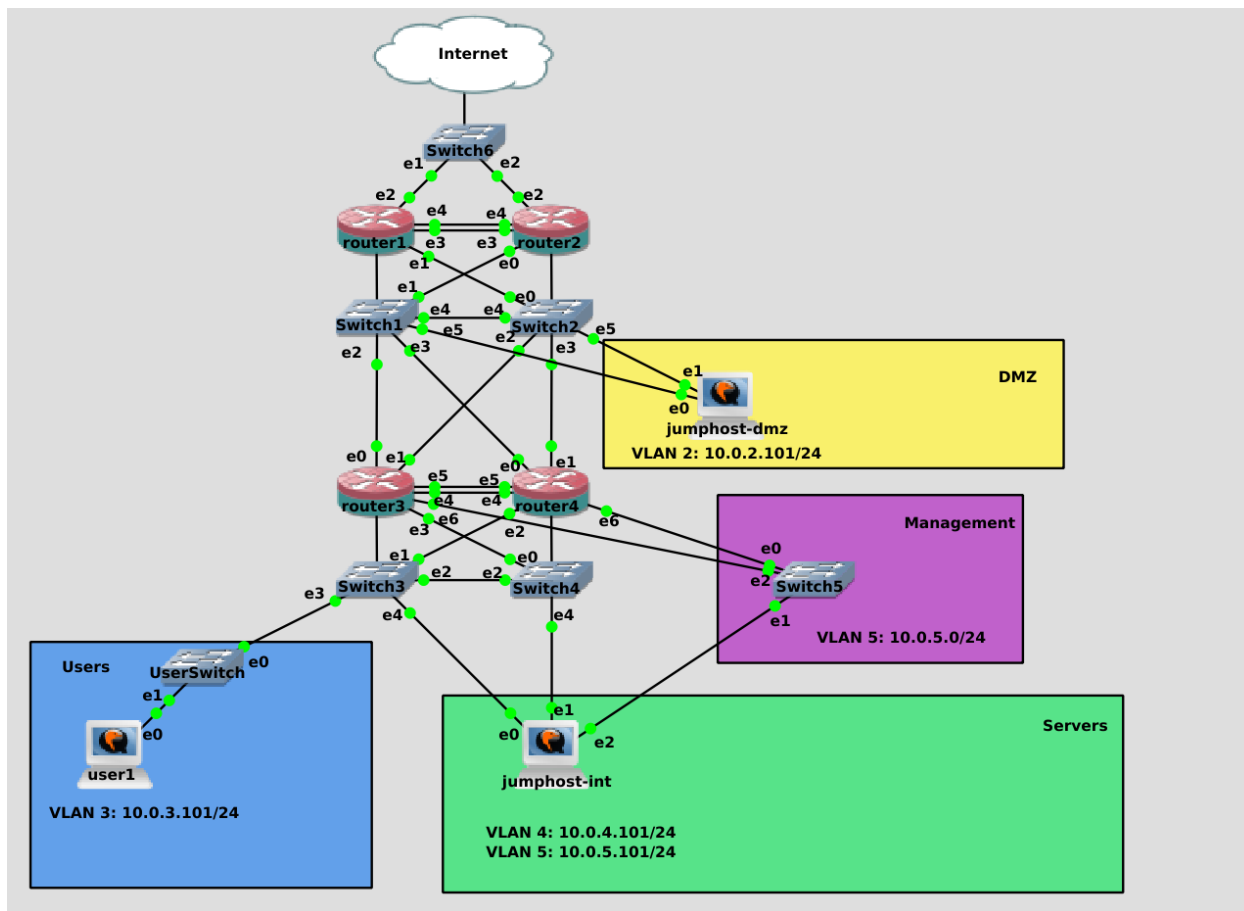


На пока этого хватит. В следующий раз мы продолжим тему DHCP - посмотрим ещё пару опций, вынесем DHCP на отдельные сервера, настроим отказоустойчивость и поговорим про безопасность. Вашим заданием будет настроить ntp сервер chrony на роутерах 3 и 4, чтобы мы в следующий раз их раздали по dhcp. Если забыли как это делать - посмотрите [54 урок](#) из курса basis. А также поиграйтесь в одной из сетей - выставите lease time в пару минут, посмотрите через tcpdump как проходит процесс DORA, как идёт запрос о продлении lease. Ну и рекомендую прочитать [статью на википедии](#), чтобы узнать чуть больше технических деталей.

2.11 11. DHCP сервер ч.2

2.11.1 11. DHCP сервер ч.2

Резервирование IP адреса



Продолжаем тему DHCP сервера. В прошлый раз мы его развернули, настроили и он уже раздаёт IP адреса. Мне хочется хосты из нашей сети перевести на него - т.е. убрать статику и получать IP по DHCP. Компьютер user1 у нас пока для красоты, никакого толку от него нет, поэтому на нём сделать это будет просто. А что будет, если я на джампхостах уберу статичный адрес и получу по DHCP? DHCP сервер их не знает, он получит запрос и выдаст первый свободный адрес из той сети, в которой они находятся. Это приведёт к тому, что на джампхостах поменяются IP адреса. А я этого совсем не хотел бы, потому что эти адреса прописаны в пробросе портов, в политиках файрвола и ssh конфигах.

Говоря про реальную инфраструктуру, есть всякие сервера и компьютеры, на которых не должны меняться IP адреса, иначе это вызовет кучу проблем. Одни сервера по IP адресам связываются с другими, есть политики на файрволах, есть какие-нибудь юзеры, которым нужны доступы к каким-нибудь серверам и всё такое. Да, большой lease time отчасти решает проблему, но это не самое надёжное решение. Некоторые из-за этого на важных хостах прописывают статику. Мы же это решим через DHCP.

Ну и наш случай, когда джампхосты адреса не получали по DHCP, а значит их в lease-ax нет. Переключим на DHCP - поменяются адреса. Чтобы этого избежать, нужно познакомить DHCP сервер с этими хостами, а для этого нужны мак адреса серверов.

```
GNU nano 2.9.8 /etc/dhcp/dhcpd.conf

range 10.0.5.10 10.0.5.50;
option routers 10.0.5.1;
}

host fantasia {
    hardware ethernet 08:00:07:26:c0:a5;
    fixed-address fantasia.example.com;
}

```

Для начала подключимся к DHCP серверу - т.е. к роутеру3 - и зайдём в настройки dhcpd.conf:

```
sudo nano /etc/dhcp/dhcpd.conf
```

Спустимся в самый низ, где мы оставили секцию host.

```
user@router3:~$ ip -c 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: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel master team0 state UP group default qlen 1000
    link/ether 0c:19:f2:58:00:00 brd ff:ff:ff:ff:ff:ff
3: ens4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel master team0 state UP group default qlen 1000
    link/ether 0c:19:f2:58:00:00 brd ff:ff:ff:ff:ff:ff permaddr 0c:19:f2:58:00:01
4: ens5: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state DOWN group default qlen 1000
    link/ether 0c:19:f2:58:00:02 brd ff:ff:ff:ff:ff:ff
5: ens6: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state DOWN group default qlen 1000
    link/ether 0c:19:f2:58:00:03 brd ff:ff:ff:ff:ff:ff
6: team0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 0c:19:f2:58:00:00 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.101/24 brd 10.0.2.255 scope global noprefixroute team0
        valid_lft forever preferred_lft forever
user@jumphost-dmz:~$

```

Теперь подключимся к jumphost-dmz и посмотрим мак адрес:

```
ip -c a
```

Мы могли бы узнать мак адрес и через arp таблицу роутера3, но мы подключились не просто так. Наши сервера подключены к сети двумя портами. Мы знаем, что у каждого порта свой MAC адрес. Но обратите внимание на мак адреса ens3, ens4 и team0 - они одинаковые. Всё дело в тининг интерфейсе - он назначает единый мак адрес на оба интерфейса - так называемый «виртуальный мак адрес». Нас это полностью устраивает, но следует знать, что, технически, для activebackup можно настроить разные мак адреса на разных интерфейсах. У нас же мак адреса одинаковые, поэтому скопируем общий мак адрес и вернёмся на DHCP сервер.

```

user@router3:~$ nano /etc/dhcp/dhcpd.conf
GNU nano 2.9.8 /etc/dhcp/dhcpd.conf

range 10.0.5.10 10.0.5.50;
option routers 10.0.5.1;
}

host jumphost-dmz {
    hardware ethernet 0c:19:f2:58:00:00;
    fixed-address 10.0.2.101;
}

```

Заменяем название хоста на jumphost-dmz, после hardware ethernet вставим скопированный мак адрес, а fixed-address заменим на IP адрес джампхоста:

```

host jumphost-dmz {
    hardware ethernet 0c:19:f2:58:00:00;
    fixed-address 10.0.2.101;
}

```

Теперь DHCP сервер при виде этого мак адреса всегда будет выдавать ему указанный IP адрес. Это называется резервирование IP адреса - мы этот IP адрес зарезервировали для хоста с таким-то мак адресом.

```

user@router3:~$ ip -c a show team0
[user@jumphost-int ~]$ ip -c a show team0
6: team0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1
000
    link/ether 0c:fe:ef:81:00:00 brd ff:ff:ff:ff:ff:ff
    inet 10.0.4.101/24 brd 10.0.4.255 scope global noprefixroute team0
        valid_lft forever preferred_lft forever
    inet6 fe80::5a5c:65ed:ffc5:8508/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
[user@jumphost-int ~]$

```

Повторим тоже самое для jumphost-int. Подключимся ко второму джампхосту и посмотрим тиминг интерфейса:

```
ip -c a show team0
```

Скопируем мак адрес.

```
host jumphost-dmz {
    hardware ethernet 0c:19:f2:58:00:00;
    fixed-address 10.0.2.101;
}

host jumphost-int {
    hardware ethernet 0c:fe:ef:81:00:00;
    fixed-address 10.0.4.101;
}
```

Затем вернёмся на DHCP сервер, скопируем секцию host и поменяем значения - имя на jumphost-int, нужный мак адрес и IP адрес 4.101:

```
host jumphost-int {
    hardware ethernet 0c:fe:ef:81:00:00;
    fixed-address 10.0.4.101;
}
```

У этого jump-хоста есть ещё один интерфейс в пятом влане.

```
user@router3:~$ ip -c a show ens5
4: ens5: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 0c:fe:ef:81:00:02 brd ff:ff:ff:ff:ff:ff
    inet 10.0.5.101/24 brd 10.0.5.255 scope global noprefixroute ens5
        valid_lft forever preferred_lft forever
    inet6 fe80::eb54:24af:5016:2b1/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
user@jumphost-int:~$
```

Вернёмся к нему и посмотрим интерфейс ens5:

```
ip -c a show ens5
```

Скопируем мак адрес.

```
host jumphost-int {
    hardware ethernet 0c:fe:ef:81:00:00;
    fixed-address 10.0.4.101;
}

host jumphost-int-mgmt {
    hardware ethernet 0c:fe:ef:81:00:02;
    fixed-address 10.0.5.101;
}
```

В настройках dhcp сервера добавим ещё одну секцию host. Имя зададим jumphost-int-mgmt, ну и вставим нужный мак адрес и пропишем IP - 5.101:

```
host jumphost-int-mgmt {
    hardware ethernet 0c:fe:ef:81:00:02;
    fixed-address 10.0.5.101;
}
```

Тут есть один нюанс. Сервер получит настройки DHCP по двум интерфейсам - vlan4 и vlan5. И если в плане адресов DNS серверов никаких проблем, то вот получить два разных gateway - это проблема.

```
### Servers
subnet 10.0.4.0 netmask 255.255.255.0 {
    range 10.0.4.10 10.0.4.200;
    option routers 10.0.4.1;
}

### Management
subnet 10.0.5.0 netmask 255.255.255.0 {
    range 10.0.5.10 10.0.5.50;
}
```

Но так как у нас в менеджмент влане gateway не нужен, мы можем просто убрать option routers в сабнете management.

```

user@router3:~$ sudo nano /etc/dhcp/dhcpd.conf
[sudo] password for user:
[user@router3 ~]$ sudo systemctl restart dhcpd
[user@router3 ~]$

```

И так, оба джампхоста мы прописали, давайте сохраним, выйдем и перезапустим dhcpd:

```
sudo systemctl restart dhcpd
```

```

[user@jumphost-dmz ~]$ sudo nmcli connection modify team ipv4.gateway ""
[user@jumphost-dmz ~]$ sudo nmcli connection modify team ipv4.routes ""
[user@jumphost-dmz ~]$ sudo nmcli connection modify team ipv4.dns ""
[user@jumphost-dmz ~]$ sudo nmcli connection modify team ipv4.method auto
[user@jumphost-dmz ~]$ sudo nmcli connection modify team ipv4.addresses ""
[user@jumphost-dmz ~]$
[user@jumphost-dmz ~]$ sudo nmcli connection up team
Connection successfully activated (master waiting for slaves) (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/10)

```

Теперь пойдём на первый джампхост. Для начала нам надо затереть то, что мы прописали статично - а именно gateway, маршруты, dns:

```

sudo nmcli connection modify team ipv4.gateway ""
sudo nmcli connection modify team ipv4.routes ""
sudo nmcli connection modify team ipv4.dns ""

```

Нужно поменять ipv4.method на auto и удалить статично прописанный адрес:

```

sudo nmcli connection modify team ipv4.method auto
sudo nmcli connection modify team ipv4.addresses ""

```

После изменений поднимем интерфейс, чтобы применить настройки:

```
sudo nmcli connection up team
```

```

[user@jumphost-dmz ~]$ ip -c a show team0
6: team0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 0c:19:f2:58:00:00 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.101/24 brd 10.0.2.255 scope global dynamic noprefixroute team0
        valid_lft 7775801sec preferred_lft 7775801sec
[user@jumphost-dmz ~]$ ip ro sh
default via 10.0.2.1 dev team0 proto dhcp metric 350
10.0.0.0/8 via 10.0.2.254 dev team0 proto dhcp metric 350
10.0.2.0/24 dev team0 proto kernel scope link src 10.0.2.101 metric 350
[user@jumphost-dmz ~]$

```

То что нас не выкинуло уже говорит о том, что всё сработало. Но давайте в этом убедимся - посмотрим интерфейс:

```
ip -c a show team0
```

Как видите, адрес 2.101 прописался по DHCP. Также посмотрим маршруты:

```
ip ro sh
```

gateway и статичный маршрут также прописались. Значит всё сработало.

```
[user@jumphost-int ~]$ sudo nmcli connection modify team ipv4.gateway ""
[user@jumphost-int ~]$ sudo nmcli connection modify team ipv4.routes ""
[user@jumphost-int ~]$ sudo nmcli connection modify team ipv4.dns ""
[user@jumphost-int ~]$ sudo nmcli connection modify team ipv4.method auto
[user@jumphost-int ~]$ sudo nmcli connection modify team ipv4.addresses ""
[user@jumphost-int ~]$
[user@jumphost-int ~]$ sudo nmcli connection up team
Connection successfully activated (master waiting for slaves) (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/5)
[user@jumphost-int ~]$
[user@jumphost-int ~]$ ip -c a show team0
6: team0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1
000
    link/ether 0c:fe:ef:81:00:00 brd ff:ff:ff:ff:ff:ff
    inet 10.0.4.101/24 brd 10.0.4.255 scope global dynamic noprefixroute team0
        valid_lft 7775990sec preferred_lft 7775990sec
    inet6 fe80::5a5c:65ed:ffc5:8508/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
[user@jumphost-int ~]$
```

Теперь перейдём на второй джампхост. Делаем тоже самое - удаляем гейтвей, маршруты, dns, меняем метод на auto и удаляем адрес:

```
sudo nmcli connection modify team ipv4.gateway ""
sudo nmcli connection modify team ipv4.routes ""
sudo nmcli connection modify team ipv4.dns ""
sudo nmcli connection modify team ipv4.method auto
sudo nmcli connection modify team ipv4.addresses ""
```

Затем поднимаем интерфейс, чтобы применить настройки и смотрим результат:

```
sudo nmcli connection up team
ip -c a show team0
```

Всё работает - прописался нужный IP адрес. Остался интерфейс влана 5 - и для него мы делаем тоже самое:

```
sudo nmcli connection modify management ipv4.gateway ""
sudo nmcli connection modify management ipv4.routes ""
sudo nmcli connection modify management ipv4.dns ""
sudo nmcli connection modify management ipv4.method auto
sudo nmcli connection modify management ipv4.addresses ""
sudo nmcli connection up management
ip -c a show ens5
```

Ну и в конце убедимся, что с маршрутами всё в порядке:

```
ip ro sh
```



```

[doctor@tardis]~$ ssh router3
Enter passphrase for key '/home/doctor/.ssh/jumphost':
Enter passphrase for key '/home/doctor/.ssh/servers':
Enter passphrase for key '/home/doctor/.ssh/servers':
Activate the web console with: systemctl enable --now cockpit.socket

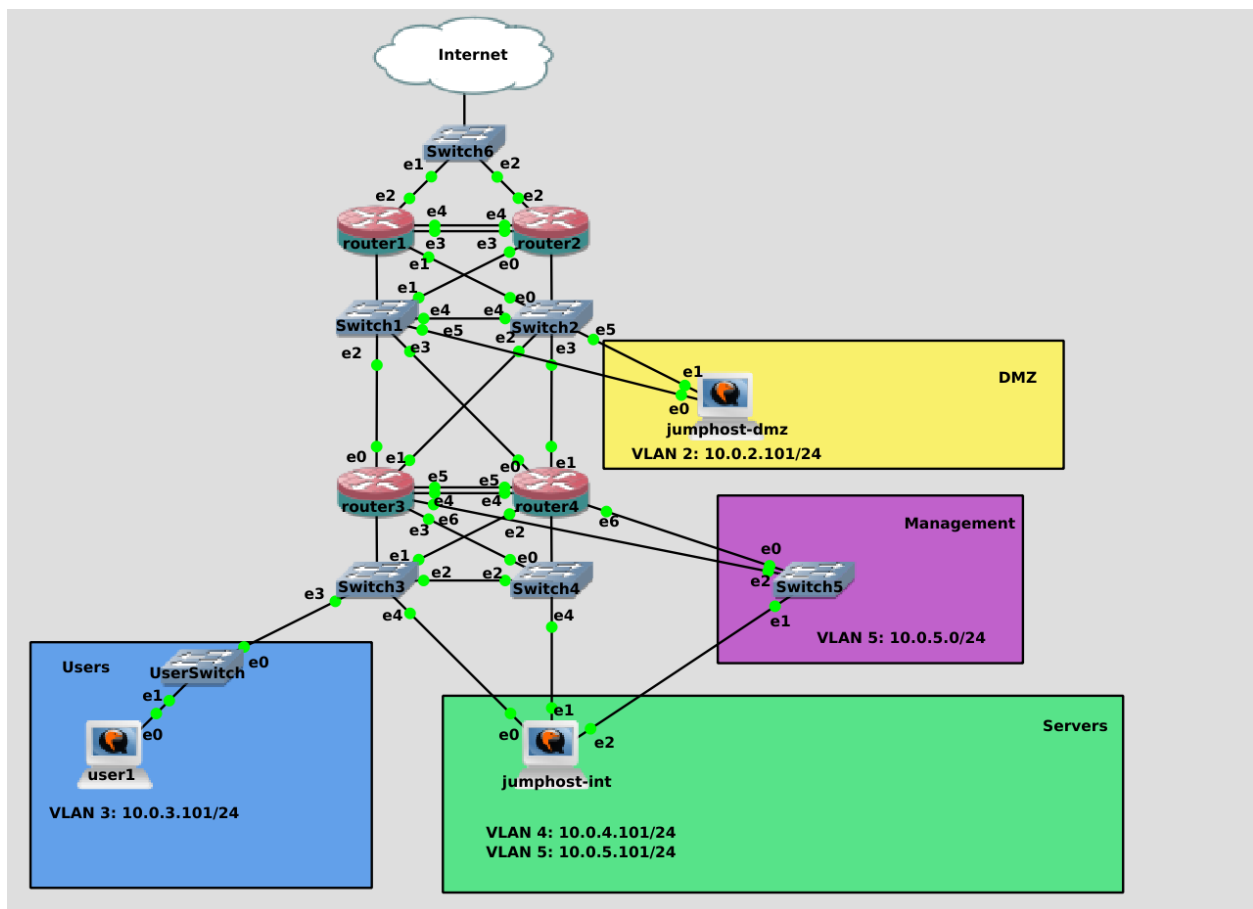
Last login: Sun Feb 13 10:25:07 2022 from 10.0.5.101
[user@router3 ~]$
[user@router3 ~]$

```

Для полной уверенности попробуем подключиться с компа к роутеру 3, чтобы проверить соединение через оба джампхоста:

```
ssh router3
```

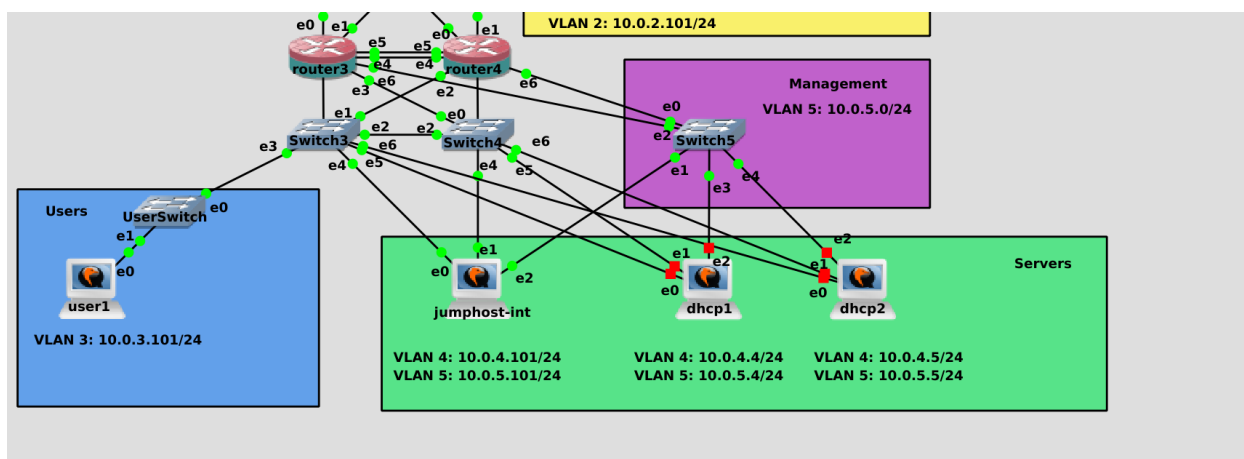
И всё работает.



Остался юзерский комп, но на него сейчас я не буду тратить время, его адрес резервировать смысла нет, а поменять статику на dhcp - вы и сами теперь можете.

Отказоустойчивость

Теперь что касается отказоустойчивости. Представим, что DHCP сервер перестал работать. Новые компьютеры не смогут получить IP адрес, а значит у них сеть не будет работать. Если dhcp сервер не будет работать очень долго, то на компьютерах, получивших адрес до проблемы, lease time закончится и на них также перестанет работать сеть. Ну или если компьютер перезагрузить - при подключении к сети DHCP клиент попытается связаться с сервером и тут всё зависит от DHCP клиента. Некоторые клиенты при недоступности DHCP сервера будут использовать старый адрес, а некоторые просто его потеряют. Да, вы, скорее всего, сможете восстановить DHCP сервер до того как истечёт lease time, но вот в гостевой сети или там где компы периодически выключаются или переподключаются к сети - проблема будет ощутима. Решение - второй DHCP сервер. Ну и заодно вынесем функционал DHCP на отдельные сервера.



Для этого добавим две новые виртуалки в серверную сеть. Чтобы не заморачиваться, я просто скопирую jump-host-int и назову новые виртуалки dhcp1 и dhcp2. Также подключу двумя проводами к свитчам 3 и 4, и по одному кабелю на менеджмент свитч. Адреса дадим следующие - на dhcp1 адреса 10.0.4.4, 10.0.5.4, а на dhcp2 - 10.0.4.5, 10.0.5.5. Так как это DHCP сервера, тут нам нужно будет прописать статику. Gateway прописываем только в 4 влане. Этап настройки имён и адресов я пропущу, вы и сами справитесь. И не забудьте в sshd_config поправить ListenAddress.

Чтобы мы могли скачать и установить DHCP сервер на новые сервера, нужно будет разрешить интернет в политиках файрвола. Вообще, давать серверам интернет не очень безопасно, но мы только поставили систему, у нас не хватает пакетов, своего репозитория пока нет, прокси сервера нет - поэтому временно, для удобства, разрешим интернет.

```
[user@router3 ~]$ sudo firewall-cmd --permanent --new-policy=src_to_internet
success
[user@router3 ~]$ sudo firewall-cmd --permanent --policy=src_to_internet --add-ingress-zone=servers
success
[user@router3 ~]$ sudo firewall-cmd --permanent --policy=src_to_internet --add-egress-zone=routers
success
[user@router3 ~]$ sudo firewall-cmd --permanent --policy=src_to_internet --add-service={dns,http,https}
success
[user@router3 ~]$ sudo firewall-cmd --reload
success
[user@router3 ~]$
```

Для этого идём на роутер3 и создаём новую политику:

```
sudo firewall-cmd --permanent --new-policy=srv_to_internet
```

В этой политике мы разрешим серверам подключаться в интернет. Поэтому входящей зоной будет «сервера», а исходящей - «роутеры», так как за ней находится роутер1, который смотрит в интернет.

```
sudo firewall-cmd --permanent --policy=srv_to_internet --add-ingress-zone=servers
sudo firewall-cmd --permanent --policy=srv_to_internet --add-egress-zone=routers
```

Чтобы сервера могли достучаться до репозитория, они сначала должны узнать их адреса - это DNS, а также подключиться к репозиториям - это http и https. Добавим такие сервисы:

```
sudo firewall-cmd --permanent --policy=srv_to_internet --add-service={dns,http,https}
```

После чего перезагрузим файрвол:

```
sudo firewall-cmd --reload
```



```

user@router3:~$ sudo firewall-cmd --info-policy=srv_to_internet
srv_to_internet (active)
  priority: -1
  target: CONTINUE
  ingress-zones: servers
  egress-zones: routers
  services: dns http https
  ports:
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
[user@router3 ~]$
  
```

Посмотрим политику:

```
sudo firewall-cmd --info-policy=srv_to_internet
```

Зоны есть, сервисы есть - вроде всё нормально.

```

[user@router3 ~]$ sudo dnf install dhcp-server
Last metadata expiration check: 0:00:05 ago on Sun 13 Feb 2022 11:50:03 AM +04.
Dependencies resolved.
=====
Package                                Architecture      Version           Repository        Size
=====
Installing:
dhcp-server                            x86_64            12:4.3.6-45.el8   baseos            529 k
Installing dependencies:
bind-export-libs                       x86_64            32:9.11.26-6.el8   baseos            1.1 M
dhcp-common                             noarch            12:4.3.6-45.el8   baseos            206 k
dhcp-libs                               x86_64            12:4.3.6-45.el8   baseos            147 k
=====
Transaction Summary
=====
Install 4 Packages

Total download size: 2.0 M
Installed size: 4.6 M
Is this ok [y/N]: 

```

Установим DHCP сервер на dhcp1 и dhcp2:

```
sudo dnf install dhcp-server
```

```

[user@router3 ~]$ sudo scp /etc/dhcp/dhcpd.conf root@10.0.5.4:/etc/dhcp/dhcpd.conf
[sudo] password for user:
The authenticity of host '10.0.5.4 (10.0.5.4)' can't be established.
ECDSA key fingerprint is SHA256:rupl3FPDkMWQzxvXwrYc2DglAc/LxzgmN8seFjaQaE.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.5.4' (ECDSA) to the list of known hosts.
root@10.0.5.4's password:
dhcpd.conf                                100% 1262      1.0MB/s   00:00
[user@router3 ~]$ 

```

Чтобы заново конфиг не писать, скопируем его с роутера3 на dhcp1:

```
sudo scp /etc/dhcp/dhcpd.conf root@10.0.5.4:/etc/dhcp/dhcpd.conf
```

Ещё один момент. У нас сейчас конфиг разрастается. Тут у нас и сабнеты, и резервация, и всякие настройки. Плюс нам нужно будет для отказоустойчивости добавить ещё несколько строк. При этом, всё кроме настроек отказоустойчивости на обоих серверах будет одинаковое. Поэтому я предлагаю сделать следующее - вынести конфиги в отдельную директорию и дальше в ней делить. Затем эту директорию можно будет спокойно копировать на второй сервер. А в самом dhcpd.conf держать только настройки отказоустойчивости, потому что они уникальны для каждого сервера.

```

[user@router3 ~]$ sudo mkdir /etc/dhcp/dhcpd.conf.d/
[sudo] password for user:
[user@dhcp1 ~]$ 

```

На сервере dhcp1 создадим директорию /etc/dhcp/dhcpd.conf.d/:

```
sudo mkdir /etc/dhcp/dhcpd.conf.d/
```

```

[user@dhcp1 ~]$ sudo cat -n /etc/dhcp/dhcpd.conf | grep host
42 host jumphost-dmz {
47 host jumphost-int {
52 host jumphost-int-mgmt {
[user@dhcp1 ~]$ sudo tail -n +42 /etc/dhcp/dhcpd.conf | sudo tee /etc/dhcp/dhcpd.conf.d/reservation.conf
host jumphost-dmz {
    hardware ethernet 0c:19:f2:58:00:00;
    fixed-address 10.0.2.101;
}

host jumphost-int {
    hardware ethernet 0c:fe:ef:81:00:00;
    fixed-address 10.0.4.101;
}

host jumphost-int-mgmt {
    hardware ethernet 0c:fe:ef:81:00:02;
    fixed-address 10.0.5.101;
}
[user@dhcp1 ~]$

```

Определим, с какой строчки начинается описание хостов:

```
sudo cat -n /etc/dhcp/dhcpd.conf | grep host
```

С 42 строчки. Перенесём всё что ниже 42 строчки в файл dhcpd.conf.d/reservation.conf:

```
sudo tail -n +42 /etc/dhcp/dhcpd.conf | sudo tee /etc/dhcp/dhcpd.conf.d/reservation.conf
```

В этом файле мы будем указывать те адреса, которые зарезервировали.

```

[user@dhcp1 ~]$ sudo head -41 /etc/dhcp/dhcpd.conf | sudo tee /etc/dhcp/dhcpd.conf.d/subnets.conf
option domain-name-servers 1.1.1.1, 8.8.8.8;

min-lease-time 6912000;
default-lease-time 7776000;
max-lease-time 8640000;

authoritative;
log-facility local7;

option ms-classless-static-routes code 249 = array of unsigned integer 8;
option rfc3442-classless-static-routes code 121 = array of unsigned integer 8;

subnet 10.0.1.0 netmask 255.255.255.0 {
    not authoritative;
}

### DMZ
subnet 10.0.2.0 netmask 255.255.255.0 {

```

А то что выше 41 строки вынесем в файл subnets.conf:

```
sudo head -41 /etc/dhcp/dhcpd.conf | sudo tee /etc/dhcp/dhcpd.conf.d/subnets.conf
```

там мы будем держать всё, что касается подсетей.

```
GNU nano 2.9.8 /etc/dhcp/dhcpd.conf

include "/etc/dhcp/dhcpd.conf.d/subnets.conf";
include "/etc/dhcp/dhcpd.conf.d/reservation.conf";
```

И чтобы основной конфиг файл ссылался на то, что лежит в этих директориях, сотрём всё в основном файле dhcpd.conf и пропишем в нём опции include, указывающие на все конфиг файлы в этой директории:

```
include "/etc/dhcp/dhcpd.conf.d/subnets.conf";
include "/etc/dhcp/dhcpd.conf.d/reservation.conf";
```

```
dhcpd.conf(5) File Formats Manual dhcpd
NAME
    dhcpd.conf - dhcpd configuration file

DESCRIPTION
    The dhcpd.conf file contains configuration information for dhcpd, the Internet
    Consortium DHCP Server.

    The dhcpd.conf file is a free-form ASCII text file. It is parsed by the re-
    descent parser built into dhcpd. The file may contain extra tabs and new
    formatting purposes. Keywords in the file are case-insensitive. Comments
    placed anywhere within the file (except within quotes). Comments begin w
    character and end at the end of the line.

    The file essentially consists of a list of statements. Statements fall into
    categories - parameters and declarations.

    Parameter statements either say how to do something (e.g., how long a lease t
    whether to do something (e.g., should dhcpd provide addresses to unknown clie
    what parameters to provide to the client (e.g., use gateway 220.177.244.7).

    Declarations are used to describe the topology of the network, to describe c
    the network, to provide addresses that can be assigned to clients, or to appl
    of parameters to a group of declarations. In any group of parameters and
    /failover peer "foo"
```

Теперь мы должны прописать опции касательно отказоустойчивости. Легче всего зайти в ман:

```
man dhcpd.conf
```

И поискать failover peer «foo»:

```
/failover peer "foo"
```

```
failover peer "foo" {
    primary;
    address anthrax.rc.example.com;
    port 647;
    peer address trantor.rc.example.com;
    peer port 847;
    max-response-delay 60;
    max-unacked-updates 10;
    mclt 3600;
    split 128;
    load balance max seconds 3;
}
```

Так вы найдёте пример секции с настройками. Скопируйте всю секцию.

```
GNU nano 2.9.8 /etc/dhcp/dhcpd.conf

failover peer "foo" {
primary;
address anthrax.rc.example.com;
port 647;
peer address trantor.rc.example.com;
peer port 847;
max-response-delay 60;
max-unacked-updates 10;
mclt 3600;
split 128;
load balance max seconds 3;
}

include "/etc/dhcp/dhcpd.conf.d/subnets.conf";
include "/etc/dhcp/dhcpd.conf.d/reservation.conf";
```

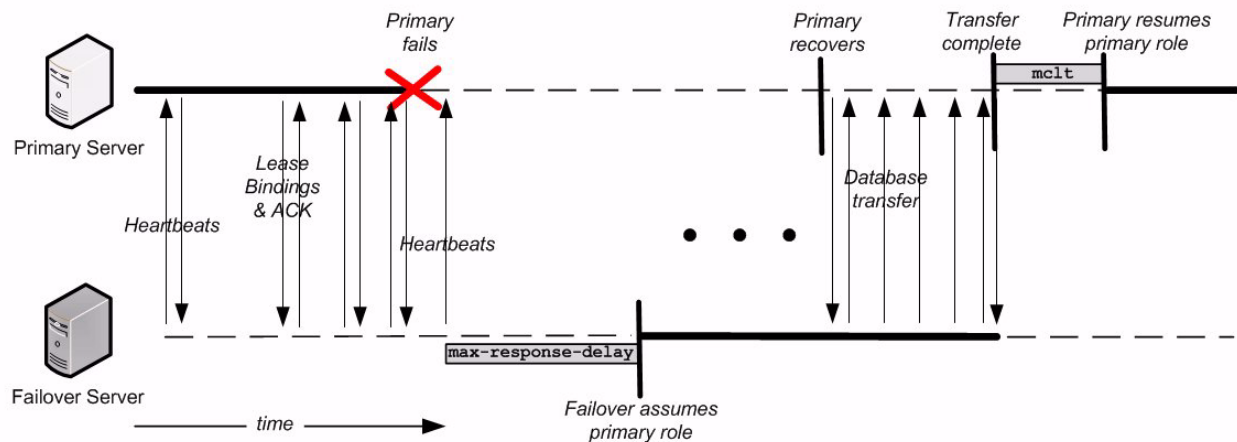
Затем вернёмся в dhcpd.conf и вставим скопированное перед опциями include.

Давайте разбираться, что здесь написано.

- failover peer «foo» - здесь определяется второй DHCP сервер, с которым этот будет обмениваться данными. Может быть несколько различных DHCP серверов, с которыми этот будет обмениваться подсетями. Скажем, для отказоустойчивости DHCP в рамках одной подсети он будет обмениваться с одним сервером, а для другой подсети - с другим. «foo» - просто название, чтобы различать

одни сервера от других. Поменяем его на dhcp12.

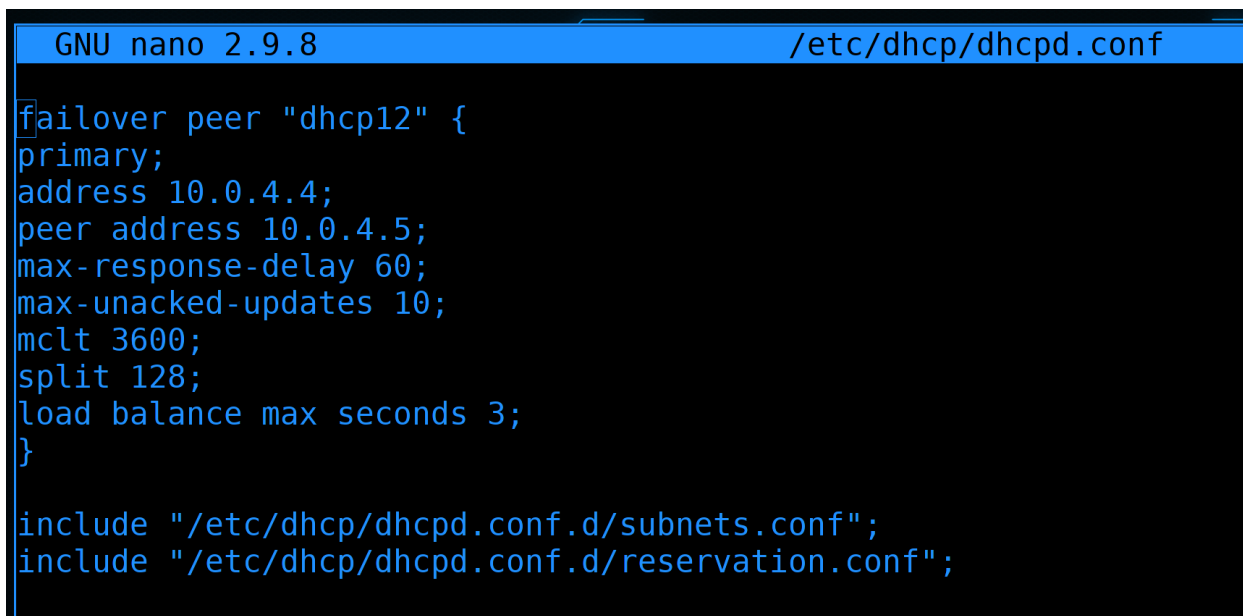
- primary - говорит о том, что это основной сервер. На втором сервере прописывается secondary. Некоторые опции настраиваются только на primary, есть какие-то нюансы на уровне соединений, но в основном между primary и secondary разница небольшая.
- address - здесь указывается адрес текущего сервера, который будет использоваться для обмена информацией между dhcp серверами. В нашем случае это 10.0.4.4.
- port 647 - тот порт, на котором будет слушать этот сервер. По-умолчанию это 647 tcp порт. Эту настройку можно убрать, чтобы оставить только необходимое, давайте так и сделаем.
- peer address - адрес другого сервера. В нашем случае - 10.0.4.5.
- peer port 847 - порт второго сервера. Вообще, в стандартах написано, что оба сервера должны слушать на 647 порту. Теоретически, можно использовать другие порты. IANA говорит, что 847 порт тоже относится к dhcp failover, но никакого объяснения, для чего на secondary сервере использовать другой порт я не нашёл. Это один из вопросов, на которые я не могу найти ответ. Если уберём эту опцию - будет использоваться 647 порт. Давайте уберём эту опцию.



- max-response-delay 60 - сколько секунд прождать, прежде чем посчитать второй сервер недоступным. Через 60 секунд, не получив ответа от primary сервера, второй сервер возьмёт на себя роль primary, а по возвращению primary в строй передаст ему информацию о всех выданных адресах и обновлённых lease time.

другую половину - secondary. Если значение 0 - все хосты обслуживает secondary, если 256 - все хосты обслуживает primary. Ну и всякие вариации - кто больше адресов выдаёт. Это значение указывается только на primary.

- load balance max seconds 3 - если пришёл DHCP Discover или Request и в течении 3 секунд второй хост не ответил, хотя очередь была его, то балансировка игнорируется и этому клиенту отвечает этот сервер. Это имеет смысл, когда сервера вроде считают друг друга доступными, но другой сервер почему-то игнорирует запросы.



```
GNU nano 2.9.8 /etc/dhcp/dhcpd.conf

failover peer "dhcp12" {
primary;
address 10.0.4.4;
peer address 10.0.4.5;
max-response-delay 60;
max-unacked-updates 10;
mclt 3600;
split 128;
load balance max seconds 3;
}

include "/etc/dhcp/dhcpd.conf.d/subnets.conf";
include "/etc/dhcp/dhcpd.conf.d/reservation.conf";
```

В итоге у нас получается такой конфиг. Общая часть, которая должна быть и на этом сервере, и на том, вынесена в различные файлы в директории. То что будет отличаться в зависимости от primary и secondary прописано здесь. Но мало прописать failover peer, его ещё надо указать для сабнетов.

```

GNU nano 2.9.8 /etc/dhcp/dhcpd.conf.d/subnets.conf

subnet 10.0.1.0 netmask 255.255.255.0 {
    not authoritative;
}

### DMZ
subnet 10.0.2.0 netmask 255.255.255.0 {
    pool {
        range 10.0.2.10 10.0.2.50;
        failover peer "dhcp12";
    }
    option routers 10.0.2.1;
    option ms-classless-static-routes 8, 10, 10,0,2,254;
    option rfc3442-classless-static-routes 8, 10, 10,0,2,254;
}

### Users
subnet 10.0.3.0 netmask 255.255.255.0 {
    pool {
        range 10.0.3.10 10.0.3.230;
        failover peer "dhcp12";
    }
}

```

Поэтому заходим в файл `dhcpd.conf.d/subnets.conf`, каждый `range` берём в `pool` и в нём же прописываем `failover peer`:

```

...
pool {
    range 10.0.2.10 10.0.2.50;
    failover peer "dhcp12";
}
...

```

Т.е. для разных адресов могут быть различные группы `dhcp` серверов. После того, как везде пропишем `pool` и `failover peer`, сохраним настройки и выйдем.

```

[user@dhcp1 ~]$ sudo scp -r /etc/dhcp/dhcpd.conf /etc/dhcp/dhcpd.conf.d root@10.0.5.5:/etc/dhcp/
[sudo] password for user:
The authenticity of host '10.0.5.5 (10.0.5.5)' can't be established.
ECDSA key fingerprint is SHA256:rupl3FPDkMWQzxvXwrYc2DglAc//LxzgmN8seFjaQaE.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.5.5' (ECDSA) to the list of known hosts.
root@10.0.5.5's password:
dhcpd.conf                               100% 215    170.6KB/s   00:00
reservation.conf                         100% 274    368.1KB/s   00:00
subnets.conf                             100% 1229   1.4MB/s     00:00
[user@dhcp1 ~]$

```

Скопируем настройки `dhcpd.conf` и директорию на второй `dhcp` сервер:

```
sudo scp -r /etc/dhcp/dhcpd.conf /etc/dhcp/dhcpd.conf.d root@10.0.5.5:/etc/dhcp/
```

Да, если мы будем делать какие-то изменения на ДНСП, то постоянно нужно будет копировать файлы на второй сервер. Т.е. `lease`-ы между серверами синхронизируются, а вот конфиги нет. Можно, конечно,

через какой-нибудь rsync и стоп настроить синхронизацию директорий или каким-нибудь другим способом. Я ничего изобретать не буду, так как планирую в будущем применить подход «Инфраструктура как код», и он решит данную проблему.

```

user@dhcp1:~$ nano /etc/dhcp/dhcpd.conf
GNU nano 2.9.8 /etc/dhcp/dhcpd.conf

failover peer "dhcp12" {
secondary;
address 10.0.4.5;
peer address 10.0.4.4;
max-response-delay 60;
max-unacked-updates 10;
#mclt 3600;
#split 128;
load balance max seconds 3;
}

include "/etc/dhcp/dhcpd.conf.d/subnets.conf";
include "/etc/dhcp/dhcpd.conf.d/reservation.conf";

```

А пока перейдём на второй сервер и откроем dhcpd.conf:

```
sudo nano /etc/dhcp/dhcpd.conf
```

Здесь мы поменяем primary на secondary, значения address и peer address поменяем местами, а также закомментируем mclt и split.

```

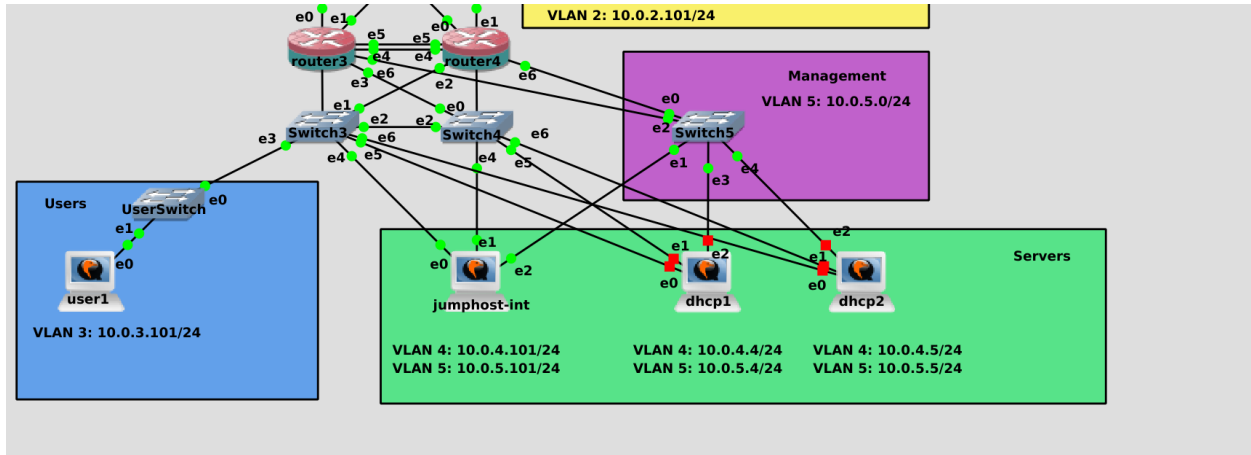
user@router3:~$ sudo firewall-cmd --permanent --add-port=647/tcp
[sudo] password for user:
success
[user@dhcp1 ~]$ sudo firewall-cmd --reload
success
[user@dhcp1 ~]$

```

Кроме того, на обоих серверах в файрволах добавим порт 647/tcp, чтобы dhcp сервера могли обмениваться данными.

```
sudo firewall-cmd --permanent --add-port=647/tcp
sudo firewall-cmd --reload
```

DHCP Relay



Но прежде чем включать dhcp сервера, давайте вернёмся к схеме. Когда у нас DHCP сервером был router3 было легко - на него приходят все вланы, а значит все broadcast запросы доходили до dhcp сервера. Но сейчас DHCP сервера работают в 4 влане, а значит они не смогут обслуживать другие подсети. Теоретически, можно было бы выдать на DHCP сервера транком все вланы, но выдавать все вланы на одно устройство считается плохой практикой. Потому что в случае уязвимости, с любого влана можно будет попасть на DHCP сервер, а с него на все остальные хосты во всей сети.

То есть, нам нужно, чтобы DHCP сервер обслуживал все подсети, но делал это не напрямую. Для этого есть так называемый DHCP relay. Это функционал, который зачастую есть на всяких роутерах и свитчах, и он позволяет пересылать DHCP запросы на DHCP сервера. Вкратце - на роутере мы настраиваем DHCP relay, при виде DHCP запросов он пересылает их на DHCP сервера. DHCP сервера отвечают DHCP-relay, а он пересылает клиентам. Нам нужно поставить relay на router3 и router4.

```

user@router3:~$ sudo systemctl disable --now dhcpd
[sudo] password for user:
Removed /etc/systemd/system/multi-user.target.wants/dhcpd.service.
[user@router3 ~]$

```

Для начала подключаемся к роутер3, сразу же отключаем и убираем из автозапуска DHCP сервер, так как здесь он больше не нужен:

```
sudo systemctl disable --now dhcpd
```

```

[user@router3 ~]$ sudo dnf install dhcp-server
Last metadata expiration check: 0:00:05 ago on Sun 13 Feb 2022 11:50:03 AM +04.
Dependencies resolved.
=====
Package                        Architecture      Version           Repository        Size
=====
Installing:
dhcp-server                    x86_64            12:4.3.6-45.el8   baseos            529 k
Installing dependencies:
bind-export-libs              x86_64            32:9.11.26-6.el8   baseos            1.1 M
dhcp-common                   noarch            12:4.3.6-45.el8   baseos            206 k
dhcp-libs                     x86_64            12:4.3.6-45.el8   baseos            147 k
=====
Transaction Summary
=====
Install 4 Packages

Total download size: 2.0 M
Installed size: 4.6 M
Is this ok [y/N]: 

```

Затем ставим пакет dhcp-relay:

```
sudo dnf install dhcp-relay
```

```

[user@router3 ~]$ sudo systemctl edit dhcrelay.service --full

```

У этой программы нет файла настроек и все опции передаются через командную строку. И вроде у неё есть свой сервис, но в нём не указаны адреса DHCP серверов. Поэтому этот сервис нужно редактировать, можно сразу через systemctl:

```
sudo systemctl edit dhcrelay.service --full
```

```

GNU nano 2.9.8 /etc/systemd/system/.#dhcrelay.service7a9492ace3472a6f

[Unit]
Description=DHCP Relay Agent Daemon
Documentation=man:dhcrelay(8)
Wants=network-online.target
After=network-online.target

[Service]
Type=notify
ExecStart=/usr/sbin/dhcrelay -d --no-pid 10.0.4.4 10.0.4.5
StandardError=null

[Install]
WantedBy=multi-user.target

```

Здесь, в конце строки ExecStart прописываем адреса DHCP серверов через пробел:

```
ExecStart=/usr/sbin/dhcrelay -d --no-pid 10.0.4.4 10.0.4.5
```

Сохраняем и выходим.

```
[user@router3 ~]$ sudo systemctl edit dhcrelay.service --full
[user@router3 ~]$ sudo systemctl enable --now dhcrelay
Created symlink /etc/systemd/system/multi-user.target.wants/dhcrelay.service → /etc/systemd/system/dhcrelay.service.
[user@router3 ~]$ sudo systemctl status dhcrelay
● dhcrelay.service - DHCP Relay Agent Daemon
   Loaded: loaded (/etc/systemd/system/dhcrelay.service; enabled; vendor preset: disabled)
   Active: active (running) since Sun 2022-02-13 18:04:24 +04; 6s ago
     Docs: man:dhcrelay(8)
  Main PID: 26775 (dhcrelay)
    Status: "Dispatching packets..."
     Tasks: 1 (limit: 2735)
    Memory: 5.3M
    CGroup: /system.slice/dhcrelay.service
            └─26775 /usr/sbin/dhcrelay -d --no-pid 10.0.4.4 10.0.4.5
```

Затем запускаем сервис и добавляем в автозапуск:

```
sudo systemctl enable --now dhcrelay
sudo systemctl status dhcrelay
```

На этом с настройкой DHCP Relay на router3 мы закончили. Также нужно поднять его на router4, так как в случае, если адрес будет на нём, у нас DHCP relay перестанет работать полноценно.

```
user@router4:~$ sudo dnf install dhcp-relay
[sudo] password for user:
Last metadata expiration check: 0:20:36 ago on Mon 14 Feb 2022 06:55:13 PM +04
Dependencies resolved.
=====
Package                                Architecture Version                                Repos
=====
Installing:
  dhcp-relay                           x86_64      12:4.3.6-45.el8                        baseos
Installing dependencies:
  bind-export-libs                      x86_64      32:9.11.26-6.el8                        baseos
  dhcp-common                           noarch      12:4.3.6-45.el8                        baseos
  dhcp-libs                             x86_64      12:4.3.6-45.el8                        baseos
Transaction Summary
=====
Install 4 Packages

Total download size: 1.7 M
Installed size: 3.7 M
Is this ok [y/N]: y
```

Подключаемся на 4 роутер, устанавливаем пакет dhcp-relay:

```
sudo dnf install dhcp-relay
```

```

GNU nano 2.9.8 /etc/systemd/system/.#dhcrelay.service2b133e45ad6
[Unit]
Description=DHCP Relay Agent Daemon
Documentation=man:dhcrelay(8)
Wants=network-online.target
After=network-online.target

[Service]
Type=notify
ExecStart=/usr/sbin/dhcrelay -d --no-pid 10.0.4.4 10.0.4.5
StandardError=null

[Install]
WantedBy=multi-user.target

```

Затем редактируем сервис:

```
sudo systemctl edit dhcrelay.service --full
```

Прописывая в нём адреса DHCP серверов.

```

[user@router4 ~]$ sudo systemctl edit dhcrelay.service --full
[user@router4 ~]$ sudo systemctl enable --now dhcrelay.service
Created symlink /etc/systemd/system/multi-user.target.wants/dhcrelay.service → /etc/systemd/system/dhcrelay.service.
[user@router4 ~]$

```

Запускаем сервис и добавляем в автозапуск:

```
sudo systemctl enable --now dhcrelay
```

С релями закончили.

Тестирование

```

[user@dhcp1 ~]$ sudo systemctl enable --now dhcpd
Created symlink /etc/systemd/system/multi-user.target.wants/dhcpd.service → /usr/lib/systemd/system/dhcpd.service.
[user@dhcp1 ~]$ sudo systemctl status dhcpd
● dhcpd.service - DHCPv4 Server Daemon
   Loaded: loaded (/usr/lib/systemd/system/dhcpd.service; enabled; vendor preset: disabled)
   Active: active (running) since Mon 2022-02-14 19:22:11 +04; 22s ago
     Docs: man:dhcpd(8)
           man:dhcpd.conf(5)
  Main PID: 1403 (dhcpd)
    Status: "Dispatching packets..."
    Tasks: 1 (limit: 2735)
   Memory: 5.4M

```

Возвращаемся на dhcp1 и запускаем сервис:


```
sudo systemctl enable --now dhcpd
```

В автозапуск добавилось. Заодно проверим статус:

```
sudo systemctl status dhcpd
```

Всё запущено, всё работает. Также сразу запускаем на dhcp2.

```
one>
Feb 13 18:29:18 dhcp1 dhcpd[25101]: balanced pool 56005e2ca540 10.0.2.0/24 total 41 free 21 b
ackup 20 lts 0 max-misbal 6
Feb 13 18:29:18 dhcp1 dhcpd[25101]: Sending updates to dhcp12.
Feb 13 18:29:18 dhcp1 dhcpd[25101]: failover peer dhcp12: peer moves from recover-done to normal
Feb 13 18:29:18 dhcp1 dhcpd[25101]: failover peer dhcp12: Both servers normal
[user@dhcp1 ~]$ sudo journalctl -eu dhcpd --no-pager
```

Затем идём в логи dhcp сервера:

```
sudo journalctl -eu dhcpd --no-pager
```

В логах мы должны увидеть сообщение «Both servers normal». Именно оно говорит нам о том, что сервера видят друг друга.

```
user@router3:~$ sudo nmcli connection modify ens3 ipv4.gateway ""
user@router3:~$ sudo nmcli connection modify ens3 ipv4.dns ""
user@router3:~$ sudo nmcli connection modify ens3 ipv4.method auto
user@router3:~$ sudo nmcli connection modify ens3 ipv4.addresses ""
user@router3:~$
user@router3:~$ sudo nmcli connection up ens3
```

Давайте тестировать. Возьмём, к примеру, компьютер user1. Уберём здесь статику и пропишем получение адреса по DHCP:

```
sudo nmcli connection modify ens3 ipv4.gateway ""
sudo nmcli connection modify ens3 ipv4.dns ""
sudo nmcli connection modify ens3 ipv4.method auto
sudo nmcli connection modify ens3 ipv4.addresses ""
```

Когда поднимем интерфейс:

```
sudo nmcli connection up ens3
```

Нас выкинет. Скорее всего поменялся IP адрес.

```

Feb 14 21:13:27 dhcp1 dhcpd[2145]: balanced pool 564011cf1540 10.0.2.0/24 total 41 free 21 ba
ckup 20 lts 0 max-misbal 6
Feb 14 21:13:27 dhcp1 dhcpd[2145]: Sending updates to dhcp12.
Feb 14 21:13:27 dhcp1 dhcpd[2145]: failover peer dhcp12: peer moves from recover-done to normal
Feb 14 21:13:27 dhcp1 dhcpd[2145]: failover peer dhcp12: Both servers normal
Feb 14 21:14:46 dhcp1 dhcpd[2145]: DHCPDISCOVER from 0c:fc:ce:ac:00:00 via 10.0.3.2: load balanc
e to peer dhcp12
Feb 14 21:14:46 dhcp1 dhcpd[2145]: DHCPDISCOVER from 0c:fc:ce:ac:00:00 via 10.0.3.3: load balanc
e to peer dhcp12
Feb 14 21:14:47 dhcp1 dhcpd[2145]: DHCPREQUEST for 10.0.3.10 (10.0.4.5) from 0c:fc:ce:ac:00:00 v
ia 10.0.3.2: lease owned by peer
Feb 14 21:14:47 dhcp1 dhcpd[2145]: DHCPREQUEST for 10.0.3.10 (10.0.4.5) from 0c:fc:ce:ac:00:00 v
ia 10.0.3.3: lease owned by peer
[user@dhcp1 ~]$

```

Пойдём на dhcp1 и посмотрим логи:

```
sudo journalctl -eu dhcpd --no-pager
```

Здесь мы видим, что пришёл запрос DHCPDISCOVER, но написано - loadbalancing to peer - т.е. запрос будет обслуживать второй сервер. Мы видим по два запроса, потому что оба DHCP relay перенаправили запросы.

```

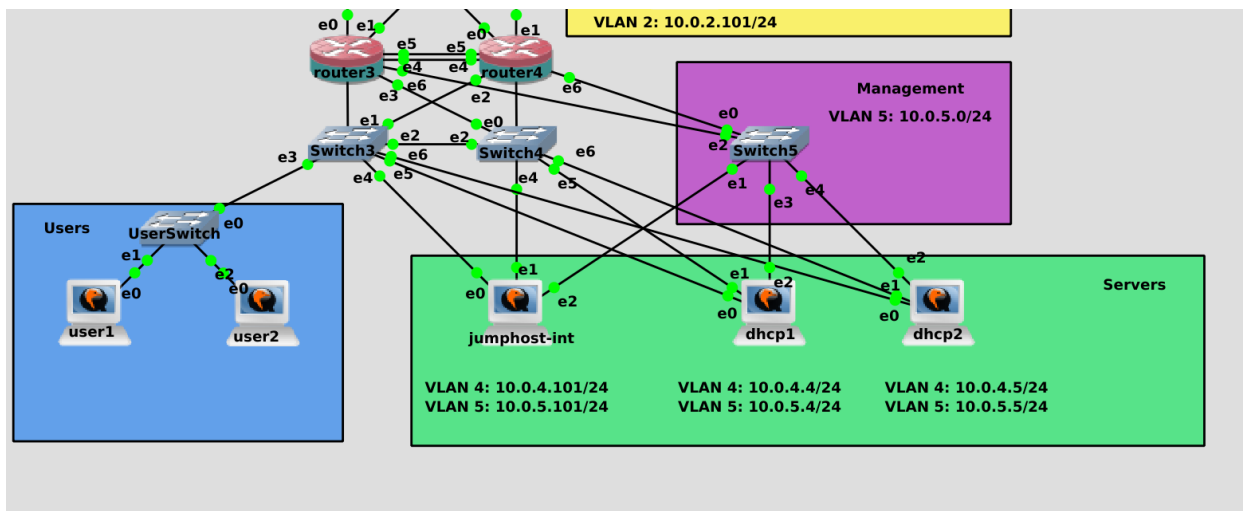
Feb 13 18:36:42 dhcp2 dhcpd[24600]: DHCPDISCOVER from 0c:fc:ce:ac:00:00 via 10.0.3.2
Feb 13 18:36:43 dhcp2 dhcpd[24600]: DHCPDISCOVER from 0c:fc:ce:ac:00:00 via 10.0.3.2
Feb 13 18:36:43 dhcp2 dhcpd[24600]: DHCPREQUEST for 10.0.3.10 (10.0.4.5) from 0c:fc:ce:ac:00:00
(user1) via 10.0.3.2
Feb 13 18:36:43 dhcp2 dhcpd[24600]: DHCPACK on 10.0.3.10 to 0c:fc:ce:ac:00:00 (user1) via 10.0.3
.2
[user@dhcp2 ~]$

```

Посмотрим логи второго сервера:

```
sudo journalctl -eu dhcpd --no-pager
```

И здесь мы видим полноценный DORA процесс. Пришёл запрос через адрес 10.0.3.2 - это адрес router3, т.е. DHCP relay. На него же был послан ответ. И теперь у user1 адрес 10.0.3.10.



Для теста подключим ещё один компьютер user2 и включим.

```
Feb 13 18:36:43 dhcp2 dhcpd[24600]: DHCPREQUEST for 10.0.3.10 (10.0.4.5) from 0c:fc:ce:ac:00:00 (user1) via 10.0.3.2
Feb 13 18:36:43 dhcp2 dhcpd[24600]: DHCPACK on 10.0.3.10 to 0c:fc:ce:ac:00:00 (user1) via 10.0.3.2
Feb 13 18:43:54 dhcp2 dhcpd[24600]: DHCPDISCOVER from 0c:5f:3e:f0:00:00 via 10.0.3.2: load balancing to peer dhcp12
Feb 13 18:43:55 dhcp2 dhcpd[24600]: DHCPREQUEST for 10.0.3.120 (10.0.4.4) from 0c:5f:3e:f0:00:00 via 10.0.3.2: lease owned by peer
[user@dhcp2 ~]$
```

Теперь начнём с логов dhcp2:

```
sudo journalctl -eu dhcpd --no-pager
```

На этот раз опять IP выдал второй DHCP сервер, поэтому я удалил и заново добавил виртуалку user2. И теперь мы видим такое же сообщение - loadbalancing to peer. Т.е. запрос обслуживает другой сервер - dhcp1.

```
via 10.0.3.2: lease owned by peer
Feb 13 18:43:54 dhcp1 dhcpd[25101]: DHCPDISCOVER from 0c:5f:3e:f0:00:00 via 10.0.3.2
Feb 13 18:43:55 dhcp1 dhcpd[25101]: DHCP OFFER on 10.0.3.120 to 0c:5f:3e:f0:00:00 (alma) via 10.0.3.2
Feb 13 18:43:55 dhcp1 dhcpd[25101]: DHCPREQUEST for 10.0.3.120 (10.0.4.4) from 0c:5f:3e:f0:00:00 (alma) via 10.0.3.2
Feb 13 18:43:55 dhcp1 dhcpd[25101]: DHCPACK on 10.0.3.120 to 0c:5f:3e:f0:00:00 (alma) via 10.0.3.2
[user@dhcp1 ~]$ sudo journalctl -eu dhcpd --no-pager
```

Посмотрим его логи:

```
sudo journalctl -eu dhcpd --no-pager
```

А на этот запрос ответил dhcp1 и выдал адрес 3.120. Т.е. первый адрес выдал dhcp2, второй тоже dhcp2, а третий - dhcp1. Порядок, в общем-то, не важен, главное что оба сервера выдают адреса.

```
[user@dhcp1 ~]$ sudo dnf install perl
Last metadata expiration check: 2:17:27 ago on Sun 13 Feb 2022 04:35:43 PM +04.
Dependencies resolved.
=====
Package                Arch    Version                               Repo      Size
=====
Installing:
perl                   x86_64  4:5.26.3-420.el8                     appstream 72 k
Installing dependencies:
dwz                    x86_64  0.12-10.el8                           appstream 108 k
efi-srpm-macros        noarch  3-3.el8                                appstream 21 k
ghc-srpm-macros        noarch  1.4.2-7.el8                           appstream 9.2 k
gn-srpm-macros         noarch  2-17.el8                              appstream 12 k
=====
```

Кстати, есть относительно удобный способ смотреть выданные IP адреса, но для этого нужно установить пакет perl:

```
sudo dnf install perl
```

```

[user@dhcp1 ~]$ perl /usr/share/doc/dhcp-server/dhcp-lease-list.pl
To get manufacturer names please download http://standards.ieee.org/regauth/oui/oui.txt to /usr/
local/etc/oui.txt
Reading leases from /var/lib/dhcpd/dhcpd.leases
=====
MAC                IP            hostname      valid until      manufacturer
=====
0c:5f:3e:f0:00:00  10.0.3.120    alma          2022-02-13 15:43:55 -NA-
0c:fc:ce:ac:00:00  10.0.3.10     -NA-          2022-02-13 15:36:43 -NA-
[user@dhcp1 ~]$

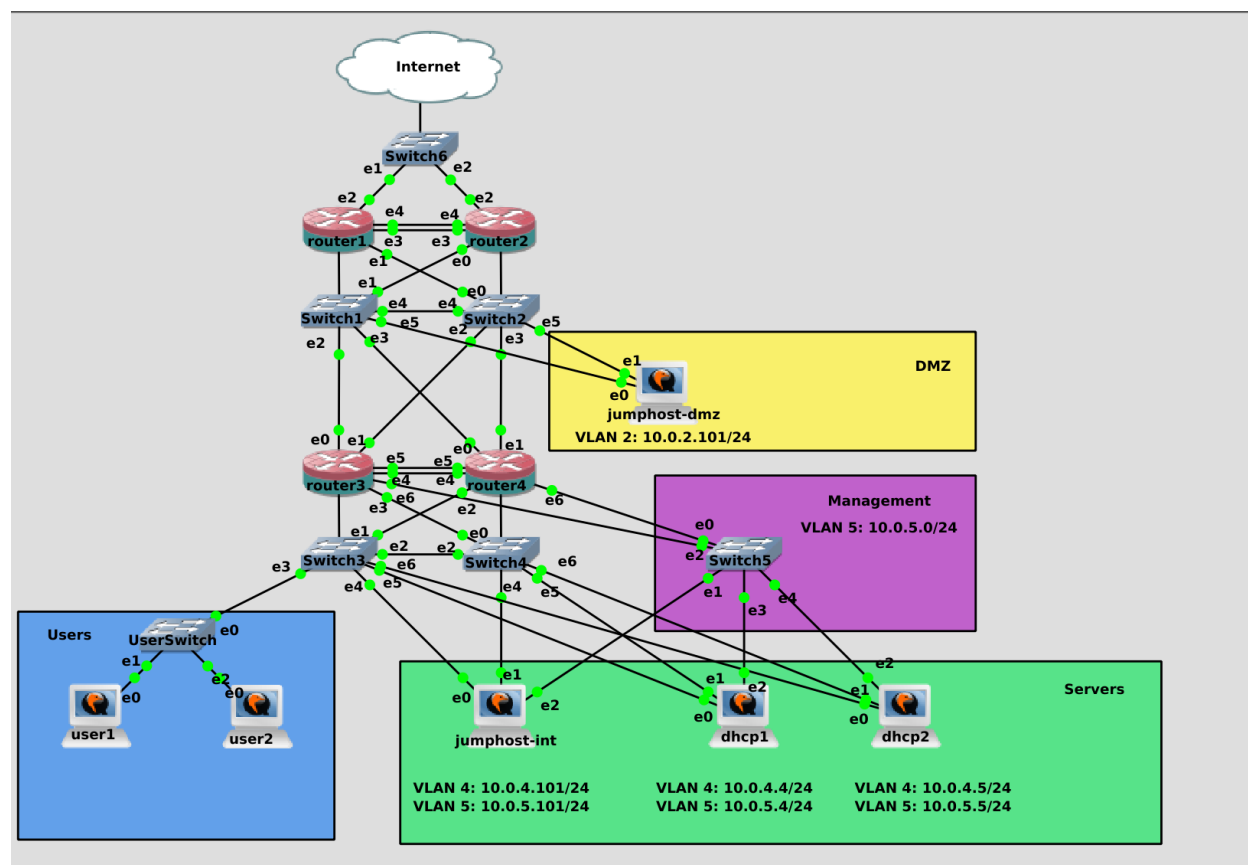
```

В директории `/usr/share/doc/dhcp-server` лежит скрипт, который может в удобном виде показать текущие lease:

```
perl /usr/share/doc/dhcp-server/dhcp-lease-list.pl
```

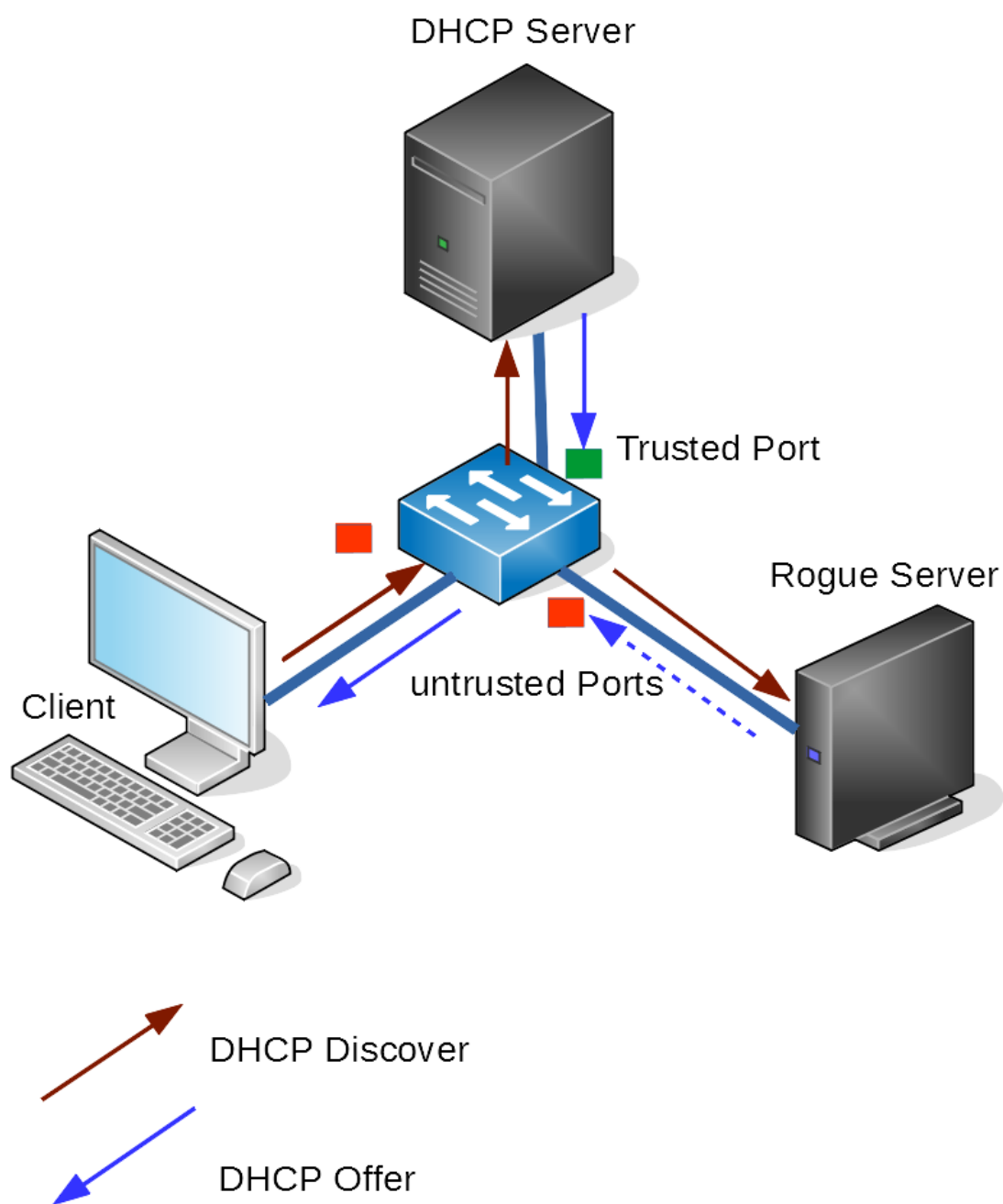
От того, что оба dhcp relay перенаправляют запросы особой проблемы нет, но логи загрязняются одинаковыми сообщениями. Чтобы это решить, можем сделать так, чтобы DHCP relay работал только на роутере, у которой плавающий адрес. Т.е. чтобы при vtr переезжал не только IP адрес, но и сервис dhcprelay. Но это будет вашим домашним заданием.

Безопасность



Под конец стоит упомянуть про безопасность. DHCP, как и многие другие базовые сетевые протоколы, был придуман давно и действует низкоуровнево - на уровне мак и IP адресов. На этих уровнях очень легко выдать себя за кого-то другого, потому что и мак адрес, и IP адрес можно подделать. Есть конечно механизмы защиты, но нужно понимать, что в сам протокол такую защиту не встроить и на уровне самого DHCP сервера не всё можно исправить.

Например, подключается кто-то левый к сети и генерирует DHCP запросы на огромное количество случайных MAC адресов. DHCP сервер резервирует все адреса под несуществующие компьютеры - у больше у него адресов не остаётся. Это называется DHCP starvation.



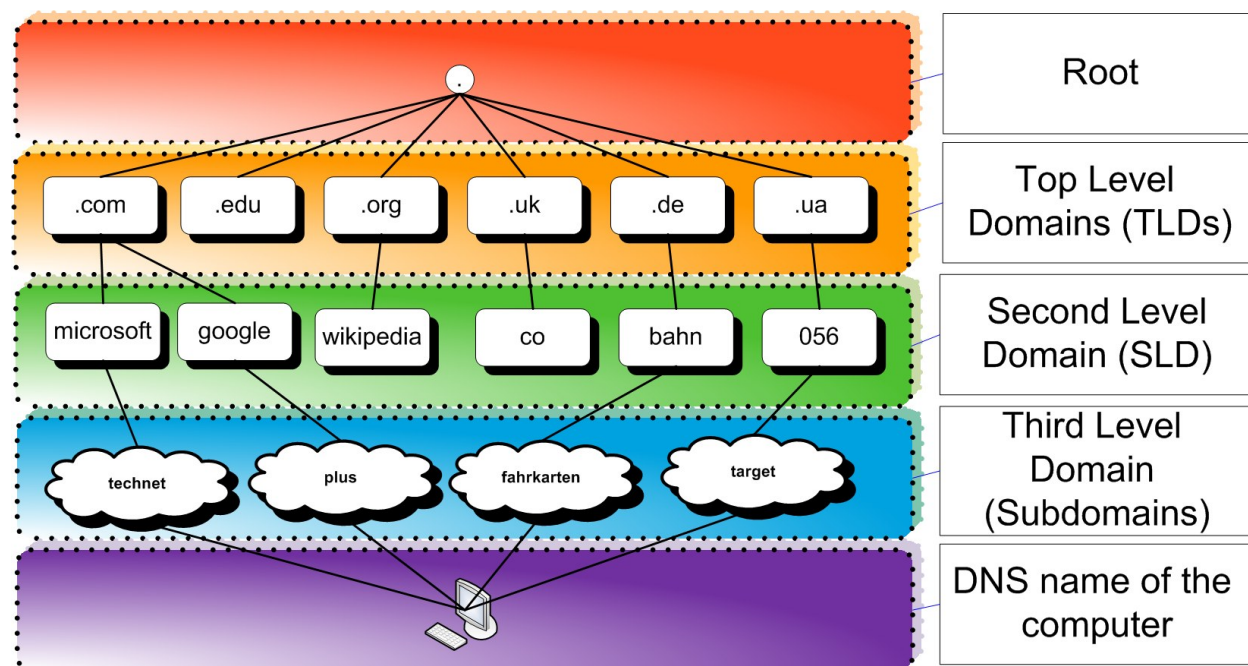
Дальше, скажем, можно поднять левый DHCP сервер - обычно его называют rogue, который будет выдавать те настройки сети, которые нужны взломщику. Скажем, вместо рабочего роутера указать компьютер взломщика. Тогда, компьютеры, получившие адрес от плохого DHCP будут выходить через плохой роутер - и получится MiTM атака.

И от первого, и от второго защищает DHCP snooping, который настраивается на умных свитчах. За этим обычно следит сетевой администратор или отдел информационной безопасности. Но советую почитать про DHCP snooping, чтобы иметь представление.

Давайте подведём итоги. Мы подняли 2 dhcp сервера, раскидали конфиги по нескольким файлам, настроили отказоустойчивость, подняли DHCP relay, убедились, что всё работает и немного поговорили про безопасность. Если говорить в рамках небольшой сети, DHCP довольно простой и понятный. Чем больше сеть - тем больше нюансов появляется и тем больше нужно знать. Так или иначе к DHCP мы ещё будем возвращаться, но слишком углубляться я не планирую.

2.12 12. Основы DNS ч.1

2.12.1 12. Основы DNS ч.1



Прежде чем начать разбирать DNS - систему доменных имён - следует понять, что такое «домен». В целом, под доменом можно понимать область, в частности - область имён. И всё это завязано под иерархию.

Всё как с директориями в линуксе. Есть корневая директория. Аналог - корневой домен:

```
/
.
```

В корневой директории есть другие директории, например, home, bin, usr и т.д.. В корневом домене есть другие домены, например, com, ru, org и т.п. - они называются доменами первого уровня:

```
/home  
.com
```

В директории /home есть другие директории, например, user1. В домене com есть другие домены, к примеру, google. Это домены второго уровня:

```
/home/user1  
.com.google
```

В директории user1 есть файлы, например, file1. В домене google есть имена, например, drive:

```
/home/user1/file1  
.com.google.drive
```

Вместо file1 могла быть другая директория с другими поддиректориями, а внутри drive могли быть домены четвёртого уровня и так далее по той же логике.

Правда доменные имена принято писать наоборот, т.е. сначала домены третьего уровня, потом второго, потом первого. Ну а корневой домен практически нигде не пишут, кроме как в настройках dns серверов:

```
drive.google.com
```

В этом адресе, условно, drive - имя хоста, google.com - домен. А drive.google.com называется полным доменным именем или fqdn. Правда за этим адресом стоит огромное количество серверов и поэтому не совсем верно говорить, что drive - это имя хоста.

```
[root@alma ~]# ping drive.google.com  
PING drive.google.com (216.58.214.142) 56(84) bytes of data.  
64 bytes from sof02s42-in-f14.1e100.net (216.58.214.142): icmp_seq=1 ttl=116 time=50.6 ms  
^C  
--- drive.google.com ping statistics ---  
1 packets transmitted, 1 received, 0% packet loss, time 0ms  
rtt min/avg/max/mdev = 50.609/50.609/50.609/0.000 ms  
[root@alma ~]#
```

Если попробуем пингануть этот адрес:

```
ping drive.google.com
```

мы увидим, что он ссылается на другой. И даже это не факт что адрес хоста, но не будем пока лезть в такие дебри.

И чтобы все эти области имён были не только на словах, а действительно работали, существует система доменных имён - DNS. То что DNS используется для превращения имён в IP адреса вы уже знаете. Вы знаете имя, вы знаете адрес DNS сервера, DNS сервер знает какой адрес стоит за этим именем и используя таблицу маршрутизации вы можете добраться до целевого сервера.

Но функции DNS этим не ограничиваются. Чтобы разобраться, чем же занимается эта система доменных имён нам нужно поговорить с ней на её языке. И в этом нам поможет утилита nslookup. Но прежде чем начнём, немного вспомним про саму утилиту.


```
[root@alma ~]# nslookup google.com
Server:                192.168.31.1
Address:               192.168.31.1#53

Non-authoritative answer:
Name:   google.com
Address: 172.217.169.110
Name:   google.com
Address: 2a00:1450:4017:808::200e

[root@alma ~]#
```

В самом простом варианте, если просто передать ей имя:

```
nslookup google.com
```

мы увидим адрес сервера, который нам ответил, а также адрес, которые мы запросили. В данном ответе их два: первый - IPv4 адрес, второй - IPv6.



В моей домашней сети DNS сервером выступил мой роутер. Большинство домашних роутеров имеют в себе и DNS сервер, и DHCP сервер. Который при выдаче адресов среди dhcp опций указывает DNS сервером себя. Естественно, DNS сервер на вашем роутере не знает все адреса в мире, он лишь перенаправляет ваши запросы на другие DNS сервера (рекурсивный DNS сервер). Иногда у молодых специалистов формируется ошибочное мнение, что можно указать в качестве DNS сервера любой gateway в любом месте. Но это заблуждение из-за многофункциональности домашних роутеров, где просто на роутере есть DNS сервер.

```
[root@alma ~]# nslookup google.com 1.1.1.1
Server:                1.1.1.1
Address:               1.1.1.1#53

Non-authoritative answer:
Name:   google.com
Address: 142.250.185.78
Name:   google.com
Address: 2a00:1450:4017:801::200e

[root@alma ~]#
```

Можем запросить адрес у другого DNS сервера, просто указав его в команде nslookup:

```
nslookup google.com 1.1.1.1
```

Как видите, теперь адрес DNS сервера другой и ответ по адресам тоже отличается. То что ответ другой - ничего удивительного, так как за гуглом стоит множество адресов.

Так вот, чтобы понять DNS, нужно поговорить с ним на его языке. Этот язык состоит из записей и их типов.

```
[root@alma ~]# nslookup -type=A google.com
Server:                192.168.31.1
Address:               192.168.31.1#53

Non-authoritative answer:
Name:   google.com
Address: 172.217.169.110

[root@alma ~]#
```

То что DNS сервер нам отвечал до этого - т.е. выдавал IP адрес google.com - это был тип записи A - адрес. Добавим в команду nslookup опцию type=A:

```
nslookup -type=A google.com
```

Ответ, который мы видим - это DNS запись.

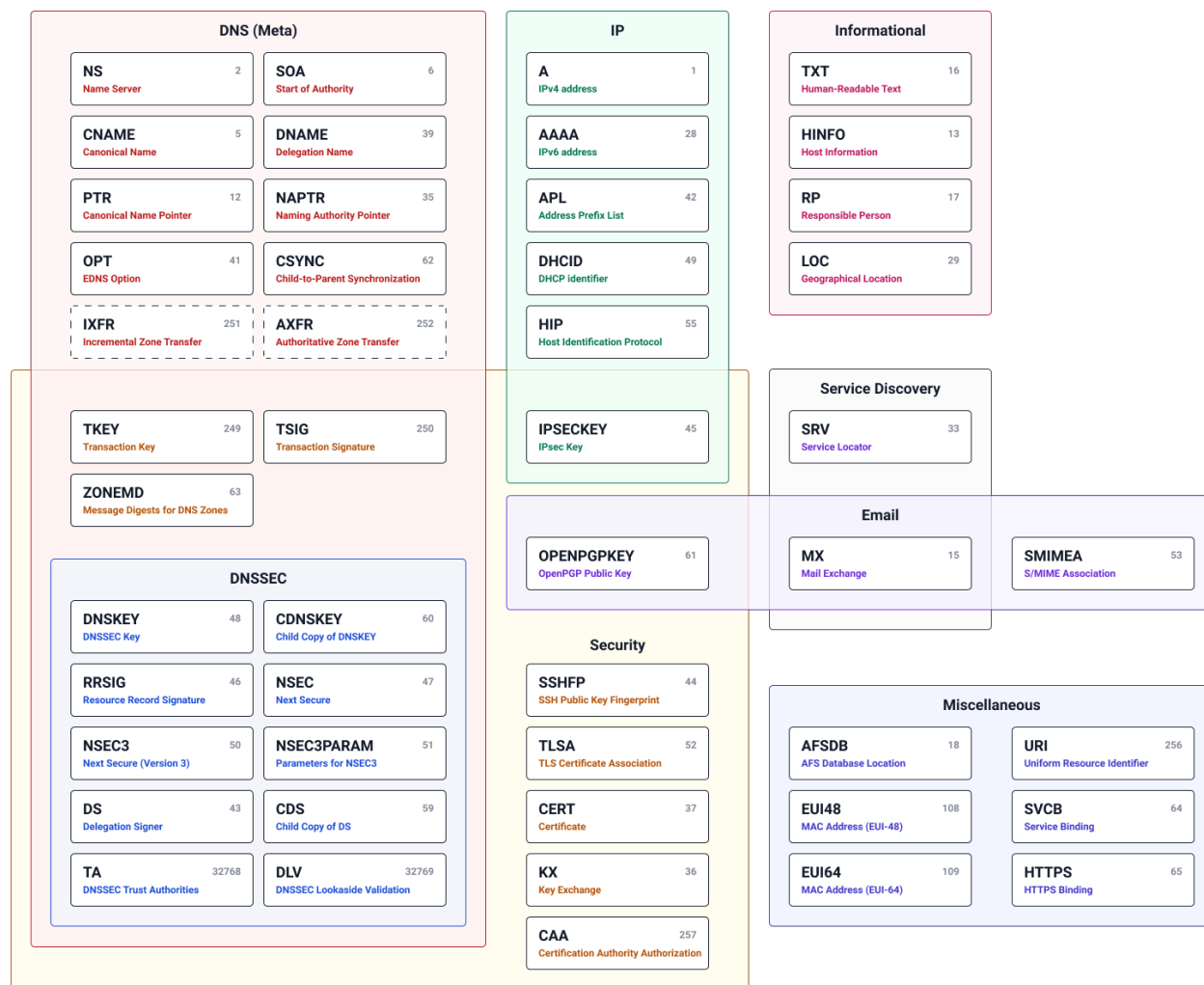
```
[root@alma ~]# nslookup -type=AAAA google.com
Server:                192.168.31.1
Address:                192.168.31.1#53

Non-authoritative answer:
Name:   google.com
Address: 2a00:1450:4017:808::200e

[root@alma ~]#
```

Помните, мы ещё видели IPv6 адрес? У неё другой тип - 4A (AAAA):

```
nslookup -type=AAAA google.com
```



Т.е. у DNS есть разные типы записей и выполняют они разные задачи, тем самым координально

расширяют задачи DNS сервера. К примеру, A запись позволяет найти, какой IP адрес стоит за таким-то именем. В тоже время MX запись позволяет найти адрес почтового сервера не зная ни имени, ни адреса.

```
[root@alma ~]# nslookup -type=SOA google.com
Server:          192.168.31.1
Address:         192.168.31.1#53

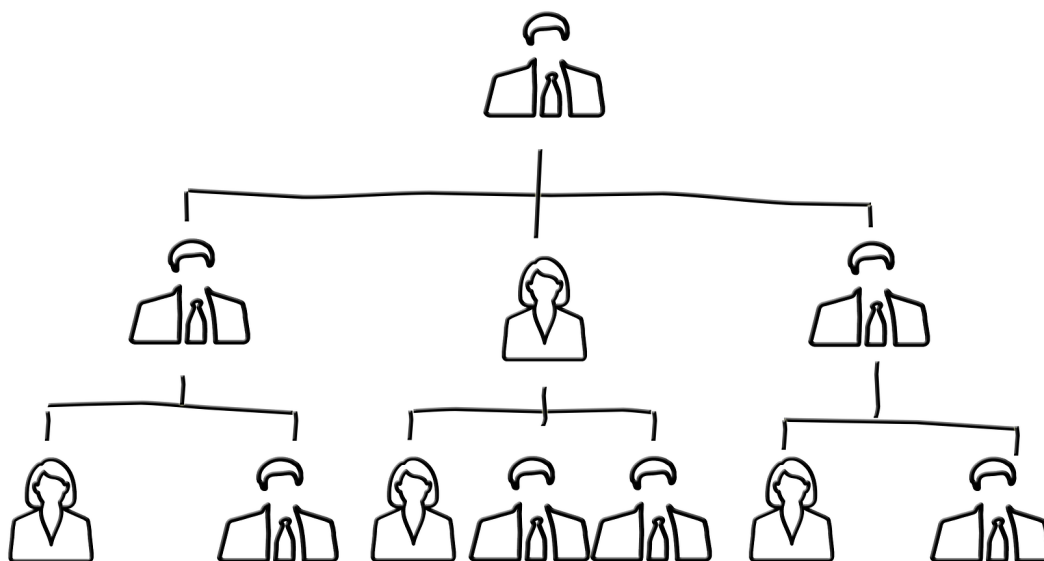
Non-authoritative answer:
google.com
    origin = ns1.google.com
    mail addr = dns-admin.google.com
    serial = 440320270
    refresh = 900
    retry = 900
    expire = 1800
    minimum = 60

Authoritative answers can be found from:
google.com      nameserver = ns1.google.com.
google.com      nameserver = ns4.google.com.
```

И чтобы понять, как это всё связано и в чём основная суть DNS нам нужен один тип записи - SOA - start of authority:

```
nslookup -type=SOA google.com
```

На русский это переводят как «начальная запись зоны», но мне этот перевод не нравится, так как он не передаёт главную идею - авторитет. Под авторитетом я подразумеваю кого-то, у кого есть полномочия и к кому есть доверие.



Представим себе банковскую организацию. Условно, вы приходите в банк, хотите открыть счёт и положить деньги. По сути, вы передаёте свои деньги какому-то человеку, которого вы видите впервые в жизни. Но вы понимаете, что это работник банка, а у вас есть некое доверие к банку. Если в целом проследить цепочку, есть некая высшая точка, условно, центробанк. Центробанк доверяет высшему руководству какого-то банка, высшее руководство доверяет своим подчинённым менеджерам, менеджеры доверяют своим работникам. А вы, в свою очередь, доверяете работникам банка. Эти работники могут абсолютно разными задачами заниматься - кто-то денежные переводы делает, кто-то кредиты выдаёт и т.п. Т.е. услуги могут быть разные и всё завязано на том, что есть некий авторитет, который подтверждается цепочкой полномочий и доверия.

В этом же и суть DNS. Типы записей - это типы услуг. Да, конечные пользователи обращаются к DNS серверу за этими услугами, но все эти услуги так или иначе завязаны на авторитете. Эти полномочия и доверие распространяются не на всё вокруг, а только на ограниченную область, которая называется зоной. Т.е. есть авторитет, который отвечает за какую-то зону. К примеру, вы не знаете IP адреса гугла - но доверяете его DNS серверам, которые и сообщают вам нужный адрес. Или, скажем, вы не знаете, какой почтовый сервер у такого-то домена - DNS сервер вам подскажет и вы этому доверитесь. Обратный пример - google не знает, действительно ли вы владелец какого-то домена. Он просит вас создать DNS запись в своей зоне. И затем google проверяет наличие этой записи. И только потому, что google доверяет всей этой цепочке DNS, он доверяет и записи, которую увидел в вашей зоне.

```
[root@alma ~]# nslookup -type=SOA google.com
Server:          192.168.31.1
Address:         192.168.31.1#53

Non-authoritative answer:
google.com
    origin = ns1.google.com
    mail addr = dns-admin.google.com
    serial = 440320270
    refresh = 900
    retry = 900
    expire = 1800
    minimum = 60

Authoritative answers can be found from:
google.com      nameserver = ns4.google.com.
google.com      nameserver = ns3.google.com.
google.com      nameserver = ns2.google.com.
```

Вернёмся к SOA записи:

```
nslookup -type=SOA google.com
```

В данном запросе мы спрашиваем, кто является авторитетным лицом зоны google.com. В ответе мы видим, что главным DNS сервером этой зоны является ns1.google.com. А связаться с администратором этой зоны можно по адресу dns-admin@google.com. Да, в самой записи вместо собачки - точка, но на самом деле это собачка, просто так сложилось, что пишут точку. Дальше идёт немного технической информации для других DNS серверов, к ней мы ещё вернёмся, а пока опустим такие детали.

Ещё стоит обратить внимание на запись «Non-authoritative answer» - т.е. ответ, который мы получили, не является авторитетным. В чём суть - одно дело прийти в банк и там получить какую-то услугу от сотрудника банка - тем самым и вы доверяете сотруднику, и банк. Другое дело вы просите кого-то пойти в банк вместо вас. Тут уж вы сами доверились этому человеку и если он вас кинет, банк здесь не при чём. И google.com здесь не при чём, поэтому вы видите, что этот ответ не от авторитетного источника, а от кого-то третьего. А авторитетные источники - те кто отвечают за зону google.com - указаны ниже: ns1.google.com, ns2.google.com и т.д.

```
[root@alma ~]# nslookup -type=SOA google.com ns1.google.com
Server:          ns1.google.com
Address:         216.239.32.10#53

google.com
    origin = ns1.google.com
    mail addr = dns-admin.google.com
    serial = 440320270
    refresh = 900
    retry = 900
    expire = 1800
    minimum = 60

[root@alma ~]#
```

Т.е. если запросить информацию напрямую у авторитетного источника:

```
nslookup -type=SOA google.com ns1.google.com
```

такого предупреждения не будет.

```
[root@alma ~]# nslookup -type=SOA pikabu.ru ns1.google.com
Server:          ns1.google.com
Address:         216.239.32.10#53

** server can't find pikabu.ru: REFUSED

[root@alma ~]# nslookup -type=SOA pikabu.ru
Server:          192.168.31.1
Address:         192.168.31.1#53

Non-authoritative answer:
pikabu.ru
    origin = alla.ns.cloudflare.com
    mail addr = dns.cloudflare.com
```

Зачастую авторитетные сервера не отвечают за других и не перенаправляют на другие зоны:

```
nslookup -type=SOA pikabu.ru ns1.google.com
```

Как видите, здесь пишется refused. Т.е. сервер ns1.google.com отказался направлять мой запрос на другие сервера, чтобы выдать мне ответ, т.е. он не рекурсивный. Хотя рекурсивные и авторитетные DNS сервера можно совмещать, когда это касается публичных адресов и интернета - их разделяют.

Условно, у вас есть домен и вы хотите поднять авторитетный dns сервер для пользователей из интернета, чтобы указывал на ваш вебсайт и почтовый сервер. Не стоит этот сервер использовать как рекурсивный. Так как ваш dns сервер всякие боты начнут использовать для DDOS атак на другие сервера, т.е. будут использовать ваш сервер для запросов на другие сервера, чтобы нагрузить их и сделать

недоступными. DNS сервера - один из самых популярных векторов атак. Также у публичных серверов всегда есть риск взлома и всякими атаками могут навредить вашим локальным пользователям, которые используют этот сервер в качестве рекурсивного.

На этом месте я бы хотел показать разницу между доменами и зонами. Грубо говоря, домены - это про области имён, а зоны - это про то, кто отвечает и как это технически реализовано.

```
[root@alma ~]# nslookup -type=SOA yandex.ru
Server:          192.168.31.1
Address:         192.168.31.1#53

Non-authoritative answer:
yandex.ru
    origin = ns1.yandex.ru
    mail addr = sysadmin.yandex-team.ru
    serial = 2022040924
    refresh = 600
    retry = 300
    expire = 2592000
    minimum = 900
```

К примеру, есть домен yandex.ru:

```
nslookup -type=SOA yandex.ru
```

У него авторитетным DNS сервером является ns1.yandex.ru, а адресом администратора - sysadmin@yandex-team.ru.

```
[root@alma ~]# nslookup -type=SOA cloud.yandex.ru
Server:          192.168.31.1
Address:         192.168.31.1#53

Non-authoritative answer:
cloud.yandex.ru
    origin = ns3.yandex.ru
    mail addr = sysadmin.yandex.ru
    serial = 2016082652
    refresh = 900
    retry = 600
    expire = 3600000
    minimum = 300
```

А есть домен cloud.yandex.ru:

```
nslookup -type=SOA cloud.yandex.ru
```

У него главным DNS сервером является ns3.yandex.ru и адрес администратора другой - sysadmin@yandex.ru. cloud.yandex.ru находится в домене yandex.ru, но вот зона - другая. Т.е. условно есть некая организация, у неё есть какие-то подразделения. Разные подразделения администрируют разные люди. И чтобы они могли добавлять любые записи в рамках своего поддомена можно наделить их авторитетом этой зоны. Это называется делегированием домена. Делегирование домена работает через NS записи, которые указывают на DNS сервера.

```
[root@alma ~]# nslookup -type=SOA .
Server:                192.168.31.1
Address:                192.168.31.1#53

Non-authoritative answer:
.
    origin = a.root-servers.net
    mail addr = nstld.verisign-grs.com
    serial = 2022040900
    refresh = 1800
    retry = 900
    expire = 604800
    minimum = 86400

Authoritative answers can be found from:
.
    nameserver = h.root-servers.net.
.
    nameserver = k.root-servers.net.
```

В качестве примера давайте посмотрим цепочку, кто отвечает за зону и кому делегирует следующую. Начнём с корневой зоны:

```
nslookup -type=SOA .
```

За корневую зону отвечают корневые DNS сервера. Про них можно почитать в интернете, советую в свободное время ознакомиться.

```
[root@alma ~]# nslookup -type=NS ru. c.root-servers.net
Server:          c.root-servers.net
Address:         192.33.4.12#53

Non-authoritative answer:
*** Can't find ru.: No answer

Authoritative answers can be found from:
ru      nameserver = f.dns.ripn.net.
ru      nameserver = a.dns.ripn.net.
ru      nameserver = b.dns.ripn.net.
ru      nameserver = d.dns.ripn.net.
ru      nameserver = e.dns.ripn.net.
a.dns.ripn.net internet address = 193.232.128.6
b.dns.ripn.net internet address = 194.85.252.62
```

А мы берём любой из корневых DNS серверов и ищем там запись NS, ведущую на домен первого уровня ru:

```
nslookup -type=NS ru. c.root-servers.net
```

Так мы находим DNS сервера, которые содержат информацию об этой зоне.

```
[root@alma ~]# nslookup -type=SOA ru. f.dns.ripn.net
Server:          f.dns.ripn.net
Address:         193.232.156.17#53

ru
    origin = a.dns.ripn.net
    mail addr = hostmaster.ripn.net
    serial = 4050924
    refresh = 86400
    retry = 14400
    expire = 2592000
    minimum = 3600

[root@alma ~]# █
```

Затем спросим у любого из них, кто отвечает за зону ru:

```
nslookup -type=SOA ru. f.dns.ripn.net
```

Другие сервера тоже отвечают, но главным считается сервер с буквой а.

```
[root@alma ~]# nslookup -type=NS yandex.ru. a.dns.ripn.net
Server:          a.dns.ripn.net
Address:         193.232.128.6#53

Non-authoritative answer:
*** Can't find yandex.ru.: No answer

Authoritative answers can be found from:
YANDEX.RU        nameserver = ns2.yandex.RU.
YANDEX.RU        nameserver = ns9.z5h64q92x9.net.
YANDEX.RU        nameserver = ns1.yandex.RU.
ns1.YANDEX.RU    internet address = 213.180.193.1
ns2.YANDEX.RU    internet address = 93.158.134.1
ns1.YANDEX.RU    has AAAA address 2a02:6b8::1
ns2.YANDEX.RU    has AAAA address 2a02:6b8:0:1::1

[root@alma ~]#
```

Пойдёмте дальше и спросим у этих серверов, кто содержит информацию о домене yandex.ru:

```
nslookup -type=NS yandex.ru. a.dns.ripn.net
```

К примеру, сервер ns1.yandex.RU.

```
[root@alma ~]# nslookup -type=SOA yandex.ru. ns1.yandex.RU.
Server:          ns1.yandex.RU.
Address:         213.180.193.1#53

yandex.ru
    origin = ns1.yandex.ru
    mail addr = sysadmin.yandex-team.ru
    serial = 2022040925
    refresh = 600
    retry = 300
    expire = 2592000
    minimum = 900

[root@alma ~]#
```

Посмотрим, кто отвечает за зону yandex.ru:

```
nslookup -type=SOA yandex.ru. ns1.yandex.RU.
```

Ну это мы уже видели, когда смотрели про cloud.yandex.ru.

```
[root@alma ~]# nslookup -type=NS cloud.yandex.ru. ns1.yandex.RU.
Server:          ns1.yandex.RU.
Address:         213.180.193.1#53

Non-authoritative answer:
*** Can't find cloud.yandex.ru.: No answer

Authoritative answers can be found from:
cloud.yandex.ru nameserver = ns4.yandex.ru.
cloud.yandex.ru nameserver = ns3.yandex.ru.

[root@alma ~]#
```

Ну и с сервера ns1.yandex.ru посмотрим, на какой NS сервер делегирован cloud.yandex.ru:

```
nslookup -type=NS cloud.yandex.ru. ns1.yandex.RU.
```

На ns3 и ns4.

Так устроена структура DNS. Помимо SOA, NS и A записей есть и другие, но их мы разберём в следующий раз.

intranet.	Used for Private/internal DNS Namespaces	RFC 6762 (Appendix G)
internal.		
private.		
corp.		
home.		
lan.	Used for Multicast DNS	RFC 6762 (Section 3)
local.		
localhost.	Reserved to avoidance of confusion and errors	RFC 6761
onion.	Anonymous onion service	RFC 7686
test.	Avoidance of confusion, allowing use for documentation and network testing	RFC 6761

Вопрос - а где достать домены? Для внутренней сети можно использовать один из выделенных доменов первого уровня, допустим, lan. Так как dns сервера свои, можно использовать любые домены, хоть google.com. Единственное что, для локальных пользователей такая идея будет создавать проблемы, так как на настоящий гугл уже не получится зайти по этому имени. Но зато подобным образом можно запрещать доступ к каким-то доменам - создавать зоны на локальных серверах, которые ведут вникуда.

Но локальные записи работают только в локальной инфраструктуре, там, где вы контролируете, какой DNS сервер прописан у пользователей. Если же у вас сайт, почта или другие сервисы, которые должны быть доступны всем в интернете, для них нужно отдельное публичное имя, которое надо покупать. Их продают регистраторы доменных имён. По крайней мере почти все имена второго уровня продаются. Есть какие-то сервисы, которые могут дать домен третьего уровня бесплатно, но с такими доменами будут определённые проблемы. К примеру, бесплатными или очень дешёвыми доменами зачастую пользуются спамеры, из-за чего в целом падает репутация домена, в следствии чего многие спам-фильтры предвзято относятся к таким доменам. И есть большой шанс, что ваша почта на таком домене почти всегда будет попадать в спам.

Что касается покупки нормального домена, то всё не так плохо, но и не то чтобы супер. Есть сайты, которые продают домены, к примеру - reg.ru. В целом домены продаются на год или несколько лет.

linux

linux-tor.ru

5 900 ₺

Магазин доменов

linuxvps.ru

35 000 ₺

Магазин доменов

linuxcom.ru

5 900 ₺

Магазин доменов

Домены по вашему запросу

Это подборка доменов, похожих на те, что вы искали. Мы составили её на основе предложений из других зон и созвучных адресов.

linux.online	Домен занят	Как купить занятый домен?
linux.com	Домен занят	Как купить занятый домен?
linux.shop	★ Премиум	487 500 ₽
linux.site		Как купить занятый домен?

Короткое красивое доменное имя больше привлекает пользователей, поэтому на них всегда есть конкуренция. Некоторые покупают такие имена сотнями и перепродают по огромным ценам.

🔍 sndaisjnd

Домены по вашему запросу

Это подборка доменов, похожих на те, что вы искали. Мы составили её на основе предложений из других зон и созвучных адресов.

sndaisjnd.com	1 309 ₺	🛒
sndaisjnd.shop	468 249 ₺	🛒
sndaisjnd.site	143 119 ₺	🛒
sndaisjnd.store	689 299 ₺	🛒
sndaisjnd.pw	384 349 ₺	🛒
sndaisjnd.tech	689 599 ₺	🛒
sndaisjnd.space	143 119 ₺	🛒
sndaisjnd.info	2 506 749 ₺	🛒
sndaisjnd.org	1 549 999 ₺	🛒

👉 Показать еще 9 доменов

В целом можно найти имена, которые стоят не так дорого. Но всегда обращайте внимание на цену продления. Потому что нередко можно купить неплохие имена по скидке за очень дёшево, а через год или три сумма продления будет стоять в несколько раз дороже.

🔍 https://www.name.com/account/domain/details/gnulinux.pro/dns

BACK TO DOMAIN DETAILS

Manage DNS

NOTE: DNS changes can take up to 48 hours to apply. [More info](#)

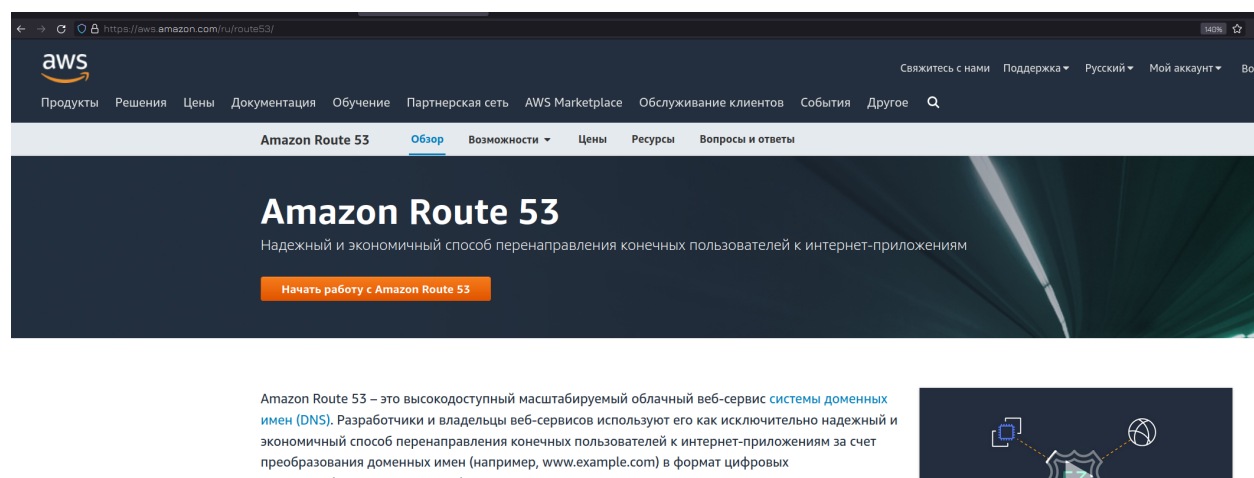
SHOW DNS TEMPLATES

DELETE SELECTED RECORDS

EXPORT DNS RECORDS (CSV)

<input type="checkbox"/>	TYPE	HOST	ANSWER	TTL	PRIO	ACTIONS
<input type="checkbox"/>	A	docs.infra.gnulinux.pro	3.66.226.169	300	N/A	<div>EDITDELETE</div> <div>CREATED: 2021-12-08</div>
<input type="checkbox"/>	ANAME	gnulinux.pro	gnulinuxpro.github.io	300	N/A	<div>EDITDELETE</div> <div>CREATED: 2021-11-04</div>
<input type="checkbox"/>	CNAME	basis.gnulinux.pro	readthedocs.io	300	N/A	<div>EDITDELETE</div> <div>CREATED: 2021-11-03</div>
<input type="checkbox"/>	CNAME	faq.gnulinux.pro	hosting.gitbook.io	300	N/A	<div>EDITDELETE</div> <div>CREATED: 2021-11-22</div>
<input type="checkbox"/>	CNAME	infra.gnulinux.pro	hosting.gitbook.io	300	N/A	<div>EDITDELETE</div> <div>CREATED: 2021-11-16</div>
<input type="checkbox"/>	CNAME	www.gnulinux.pro	gnulinuxpro.github.io	300	N/A	<div>EDITDELETE</div> <div>CREATED: 2021-11-04</div>

После покупки домена вы можете настроить NS записи на свои сервера, либо использовать публичные DNS сервера. Зачастую, публичные DNS записи можно выставить у регистраторов доменных имён, то есть вам не нужно будет иметь свой DNS сервер, который смотрит в интернет.



Нередко вместо DNS серверов регистратора люди предпочитают использовать DNS сервера облачных провайдеров. У них могут быть свои преимущества, особенно когда вы используете их облака для других сервисов, например, для почты.

Но если вы купили публичный домен, не стоит его использовать в локальной инфраструктуре. Иначе у вас получится каша с локальными и публичными днс записями. Вам либо придётся все локальные записи выставлять в интернет, либо делать копию зоны на локальном сервере с другими записями. Ну и от этого всякие проблемы будут, например, одна из самых частых проблем - многие используют адрес домена как адрес сайта, допустим, google.com. Но в сети, где есть домен контроллер, адрес домена должен быть на нём. Можно конечно эту проблему решить граблями, но лучше так не делать.

Но если прям хочется публичный домен использовать с локальной инфраструктуре, можно делегировать домен третьего уровня. Скажем, публичный домен - google.com, а в локальной сети использовать домен соgr.google.com, указывающий на DNS сервера, доступные только в локальной сети.

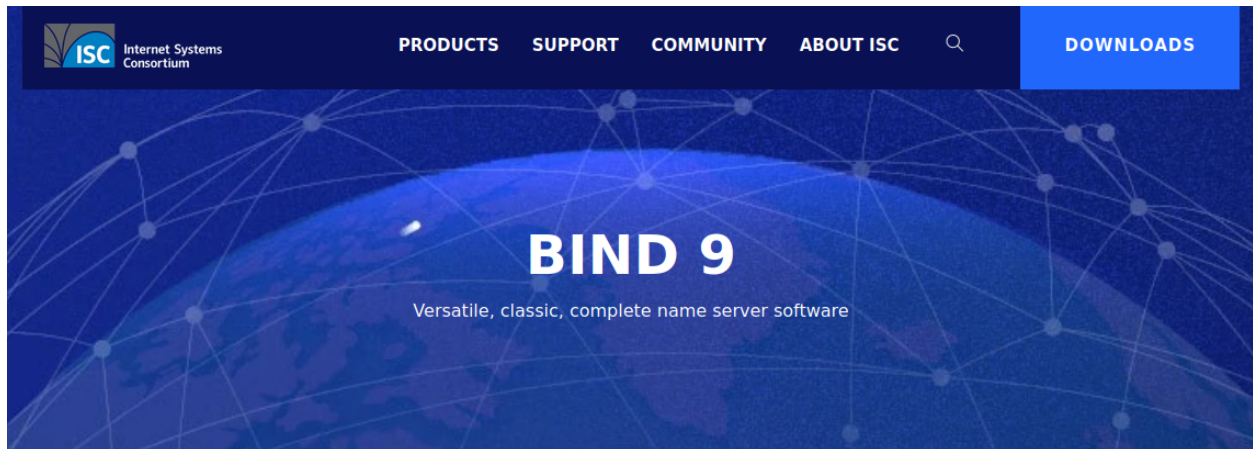
На пока это всё, но в теме DNS ещё многое предстоит изучить. Сегодня мы разобрали, что такое домен, что такое зона, поговорили о делегировании, обсудили локальные и публичные домены.

2.13 13. Основы DNS ч.2

2.13.1 13. Основы DNS ч.2

Вступление

В прошлый раз мы с вами разобрали, что такое домен, что такое зона, делегирование, SOA, NS записи и некоторые другие основы DNS. Предлагаю не тратить время на очередную теорию про другие основные типы DNS записей, этого в интернете и так полно, мы же их будем рассматривать по мере практической необходимости. К примеру, будем разбирать почтовый сервер - зайдём на DNS сервер, пропишем необходимые типы записей для почтового сервера, там же их и разберём.



Why use BIND 9?

BIND 9 has evolved to be a very flexible, full-featured [DNS](#) system. Whatever your application is, BIND 9 probably has the required features. As the first, oldest, and most commonly deployed solution, there are more network engineers who are already familiar with BIND 9 than with any other system.

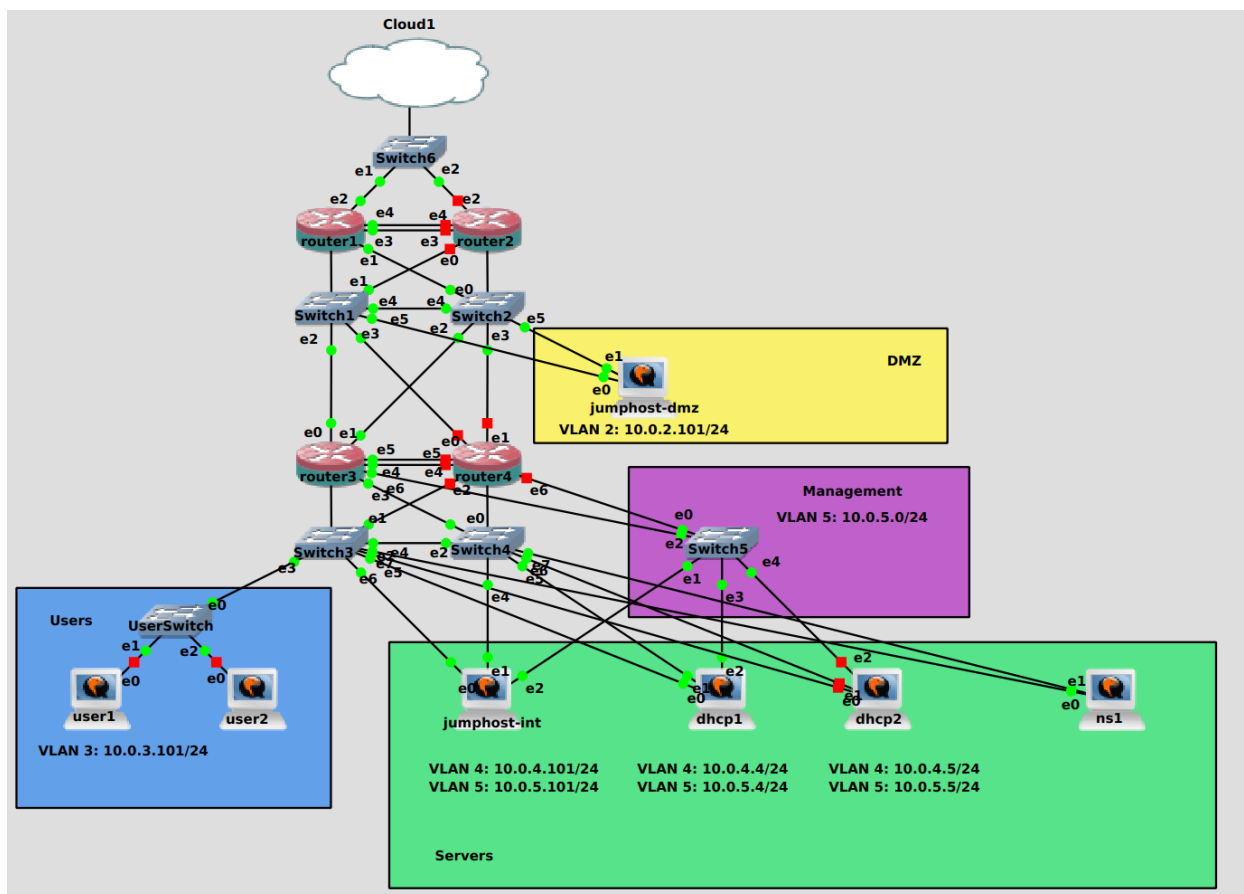
BIND 9 is transparent [open source](#), licensed under the [MPL 2.0 license](#). Users are free to add functionality to BIND 9 and contribute back to the community through our open [Gitlab](#).

If you want source code, download a current version from the [ISC website](#) or [our FTP site](#). Or, install our updated ISC packages for [Ubuntu](#), [CentOS/Fedora](#), and the standard [Debian package](#). If you prefer Docker, get our [official Docker image](#).

Help is available via our [community mailing list](#), or you may purchase a [support subscription](#) for expert, confidential, 24x7 support from the ISC team.

А пока, давайте поднимем DNS сервер. Есть разные реализации, мы будем учиться на bind. Это сервер от той же организации, которая разрабатывает dnsmc сервер, разбираемый в предыдущих темах. Почему именно bind? С одной стороны он известный, много где применяется, а с другой - это хороший пример для обучения работы с сервисами в командой строке Linux. Если раньше в отношении сервисов мы просто редактировали конфиг файл и рестартовали сервис, то с bind всё чуть сложнее - помимо настроек самого демона, у него есть ещё файлы зон и утилиты, которые управляют DNS сервером.

Подготовка виртуальной машины



Начнём с того, что добавим ещё одну виртуалку в нашу схему, назовём её ns1 и подключим в серверный влан используя оба свитча. Настройку тининга и выдачу хостнейма сделайте сами.

DNS сервер, в некотором роде, будет связывать все системы внутри нашей инфраструктуры. И если вдруг у него поменяется IP адрес - это может привести к плохим последствиям. Поэтому давайте зарезервируем IP адрес на dhcp сервере.

```
[user@dhcp1 ~]$ perl /usr/share/doc/dhcp-server/dhcp-lease-list.pl
To get manufacturer names please download http://standards.ieee.org/regauth/oui/oui.txt to /usr/
local/etc/oui.txt
Reading leases from /var/lib/dhcpd/dhcpd.leases
MAC                IP                hostname          valid until       manufacturer
=====
0c:35:c4:37:00:00   10.0.4.108        ns1               2022-12-17 16:19:54 -NA-
0c:35:c4:37:00:01   10.0.4.109        alma             2022-12-17 15:58:38 -NA-
0c:38:2c:d3:00:00   10.0.4.12         ch1              2023-02-28 15:10:50 -NA-
0c:3e:fa:05:00:00   10.0.4.74         database         2023-02-28 15:10:53 -NA-
0c:d6:65:e0:00:00   10.0.4.54         ch3              2023-02-28 15:10:52 -NA-
0c:f5:75:41:00:00   10.0.4.106        ch2              2023-02-28 15:10:51 -NA-
[user@dhcp1 ~]$
```

Заходим на dhcp сервер и смотрим список выданных адресов:

```
perl /usr/share/doc/dhcp-server/dhcp-lease-list.pl
```

В табличке с помощью колонки hostname находим наш сервер ns1 и копируем его мак адрес.

```
GNU nano 2.9.8 /etc/dhcp/dhcpd.conf.d/reservation.conf

host jumphost-dmz {
    hardware ethernet 0c:19:f2:58:00:00;
    fixed-address 10.0.2.101;
}

host jumphost-int {
    hardware ethernet 0c:fe:ef:81:00:00;
    fixed-address 10.0.4.101;
}

host jumphost-int-mgmt {
    hardware ethernet 0c:fe:ef:81:00:02;
    fixed-address 10.0.5.101;
}

host ns1 {
    hardware ethernet 0c:35:c4:37:00:00;
    fixed-address 10.0.4.53;
}
```

Дальше заходим в конфиг с резервациями нашего dhcp сервера и добавляем новую запись.

/etc/dhcp/dhcpd.conf.d/reservation.conf

```
host ns1 {
    hardware ethernet 0c:35:c4:37:00:00;
    fixed-address 10.0.4.53;
}
```

Я дал адрес 53, так как DNS работает на 53 порту. Так будет легче запомнить, что за этим адресом DNS сервер.

```
[root@dhcp1 user]# systemctl restart dhcpd
```

Не забываем после изменений скопировать конфиг файл с резервациями на второй dhcp сервер, а дальше перезапускаем сервис dhcpd:

```
systemctl restart dhcpd
```

```
[root@ns1 ~]#
[root@ns1 ~]# nmcli connection up team
Connection successfully activated (master waiting for slaves) (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/18)
[root@ns1 ~]# ip a show team0
2: team0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 0c:35:c4:37:00:00 brd ff:ff:ff:ff:ff:ff
    inet 10.0.4.53/24 brd 10.0.4.255 scope global dynamic noprefixroute team0
        valid_lft 7775993sec preferred_lft 7775993sec
    inet6 fe80::bf0e:4e8:7f05:3c74/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
[root@ns1 ~]#
```

Чтобы наш новый сервер получил новый адрес, переподключаем на нём сетевой интерфейс и проверяем IP адрес.

```
nmcli con up team
ip a show team0
```

```
[doctor@tardis]~$ ssh ns1
The authenticity of host '10.0.4.53 (<no hostip for proxy command>)' can't be established.
ED25519 key fingerprint is SHA256:8s1EIlv2v3LmRwuL0reNltFM2j+vEZ4ZAubZ/XNatEs.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.4.53' (ED25519) to the list of known hosts.
user@10.0.4.53's password:
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Sat Dec 17 19:16:16 2022 from 10.0.4.4
[user@ns1 ~]$
```

Правим ssh конфиг и подключаемся к новому DNS серверу:

```
ssh ns1
```

Установка bind

```
[user@ns1 ~]$ sudo yum install bind
Last metadata expiration check: 0:19:11 ago on Sat 17 Dec 2022 07:27:10 PM +04.
Dependencies resolved.
=====
Package                Architecture Version                      Repository      Size
=====
Installing:
bind                   x86_64      32:9.11.36-5.el8_7.2        appstream       2.1 M
Upgrading:
bind-libs              x86_64      32:9.11.36-5.el8_7.2        appstream       174 k
bind-libs-lite         x86_64      32:9.11.36-5.el8_7.2        appstream       1.2 M
bind-license           noarch      32:9.11.36-5.el8_7.2        appstream       103 k
bind-utils             x86_64      32:9.11.36-5.el8_7.2        appstream       451 k
json-c                 x86_64      0.13.1-3.el8                baseos          40 k
python3-bind           noarch      32:9.11.36-5.el8_7.2        appstream       150 k
Transaction Summary
=====
Install  1 Package
Upgrade  6 Packages

Total download size: 4.2 M
Is this ok [y/N]:
```

Установим bind:

```
sudo yum install bind -y
```

Тут с названиями довольно запутано - пакет называется bind, его сервис в systemd называется named, а в файрволе сервис называется просто dns.

```
[user@ns1 ~]$ sudo systemctl enable --now named
Created symlink /etc/systemd/system/multi-user.target.wants/named.service → /usr/lib/systemd/system/named.service.
[user@ns1 ~]$ sudo firewall-cmd --add-service=dns --permanent
success
[user@ns1 ~]$ sudo firewall-cmd --reload
success
[user@ns1 ~]$
```

Так как мы в тестовой среде, можем сразу запустить сервис и добавить его на фаервол:

```
sudo systemctl enable --now named
sudo firewall-cmd --add-service=dns --permanent
sudo firewall-cmd --reload
```

Но в реальной среде так делать не стоит, лучше предварительно настроить, а уже потом запускать сервис и добавлять его на фаерволе. Если вы будете разворачивать сервис в интернете, то может сложиться ситуация, что у какого-то демона, по-умолчанию, не безопасные настройки. И пока вы будете с ними возиться, кто-то уже успеет сделать что-то плохое.

Разбор настроек

```
GNU nano 2.9.8 /etc/named.conf
//
// named.conf
//
// Provided by Red Hat bind package to configure the ISC BIND named(8) DNS
// server as a caching only nameserver (as a localhost DNS resolver only).
//
// See /usr/share/doc/bind*/sample/ for example named configuration files.
//
options {
    listen-on port 53 { 127.0.0.1; };
    listen-on-v6 port 53 { ::1; };
    directory "/var/named";
    dump-file "/var/named/data/cache_dump.db";
    statistics-file "/var/named/data/named_stats.txt";
    memstatistics-file "/var/named/data/named_mem_stats.txt";
    secroots-file "/var/named/data/named.secroots";
    recursing-file "/var/named/data/named.recursing";
    allow-query { localhost; };
}
```

Основной файл настроек bind-a - /etc/named.conf. Тут у нас всё что касается самого демона - на каких портах он работает, всякие настройки кэширования, безопасности и прочее. Ещё здесь указываются зоны, но всё что касается самих DNS записей прописывается в других файлах, до них мы ещё дойдём. Пока разберём этот файл.

```
[user@ns1 ~]$ ls /usr/share/doc/bind*/sample/
etc var
[user@ns1 ~]$ ls /usr/share/doc/bind*/sample/etc
named.conf named.rfc1912.zones
[user@ns1 ~]$ cat /usr/share/doc/bind*/sample/etc/named.conf
/*
 * Sample named.conf BIND DNS server 'named' configuration file
 * for the Red Hat BIND distribution.
 *
 * See the BIND Administrator's Reference Manual (ARM) for details, in:
 *   file:///usr/share/doc/bind-{version}/arm/Bv9ARM.html
 * Also see the BIND Configuration GUI : /usr/bin/system-config-bind and
 * its manual.
 */

options
{
    // Put files that named is allowed to write in the data/ directory:
    directory                "/var/named";           // "Working" directory
```

Сверху мы видим небольшую подсказку. Она может привести нас к примерам, где можно найти чуть более детальные конфиги:

```
ls /usr/share/doc/bind*/sample/
ls /usr/share/doc/bind*/sample/etc
cat /usr/share/doc/bind*/sample/etc/named.conf
```

Да, всё это можно найти и в интернете, но иногда, если что-то вылетело из головы, легче взглянуть на пример и вспомнить, чем пытаться это нагуглить. Ну и взглянув на готовые примеры можно встретить то, что и не думал искать раньше. К примеру, пока я не взглянул на этот конфиг, я и не думал, что у bind есть графическая утилита для настройки.

Сам конфиг напоминает конфиг для DHCP сервера - также используются точки с запятыми как разделители, комментарии со слешами и звёздочками, секции с фигурными скобками, также есть возможность расширить настройки на несколько файлов с помощью include. Основные настройки самого демона находятся в секции options, есть отдельная секция logging, есть секции под специфичные нужды, такие как view, key, trusted-keys, zone. Всё это мы будем разбирать по мере необходимости.

```
options {
    listen-on port 53 { 127.0.0.1; };
    listen-on-v6 port 53 { ::1; };
    directory "/var/named";
    dump-file "/var/named/data/cache_dump.db";
    statistics-file "/var/named/data/named_stats.txt";
    memstatistics-file "/var/named/data/named_mem_stats.txt";
    secroots-file "/var/named/data/named.secroots";
    recursing-file "/var/named/data/named.recursing";
    allow-query { localhost; };
```

Вернёмся к основному конфигу и пройдемся по секции options:

```
sudo nano /etc/named.conf
```

- **listen-on** - указывает, на каких адресах и портах работает этот демон. DNS по умолчанию

работает на 53 порту, и вы, скорее всего, никогда не поменяете стандартный порт. Большинство DNS клиентов даже не позволяют указать другой порт. То есть, если вы настроите здесь не стандартный порт - то ваши клиенты просто не смогут достучаться до DNS сервера. Да, есть DNS-over-TLS или DNS-over-HTTPS, они используют не 53 порт, но это настраивается по другому и не связано с этим параметром. Разве что DNS сервера могут обращаться друг к другу по нестандартным портам, и это может быть полезно в каких-то специфичных случаях.

В фигурных скобках мы видим 127.0.0.1. Это означает, что этот DNS сервер слушает только на этом адресе, а значит он доступен только сам для себя. Чтобы другие компьютеры могли к нему обращаться, давайте добавим сюда также его основной адрес. И не забудьте поставить точку с запятой после:

```
listen-on port 53 { 127.0.0.1; 10.0.4.53; };
```

- **listen-on-v6** - тоже самое, что и **listen-on**, но предназначенное для **ipv6**.

Дальше идут несколько параметров, которые полезно понимать, но зачастую нет смысла трогать:

- **directory** - основная директория с файлами. Здесь обычно всякие файлы с DNS записями, временные файлы и прочее. Мы ещё сюда зайдём.
- **dump-file** - Файл, куда сохраняется кэш во время работы DNS сервера. Огромная часть работы DNS сервера - работа с кэшем, чтоб не узнавать ответ на каждый запрос повторно, что занимает гораздо больше времени и ресурсов. Если у вас есть высоконагруженный DNS сервер, на него приходит огромное количество DNS запросов, и вдруг сервис падает, даже если он моментально заново поднимется, восстановление кэша с нуля может быть лишней тратой времени и ресурсов. А при наличии такого **dump-файла** процесс восстановления будет гораздо быстрее.
- **statistics-file** - файл, куда сливается статистика с работы сервиса - количество и типы DNS запросов, сколько заняли времени и т.п.
- **memstatistics-file** - ещё один файл статистики, но больше связанный с потреблением памяти.
- **secroots-file** - файл, куда кэшируются ключи для проверки подписей для DNSSEC. DNSSEC это отдельная тема, которую я не хочу пока мешать с основами DNS, её мы разберём отдельно.
- **recursing-file** - файл, куда будет записан вывод команды **rndc recursing**, который покажет текущие рекурсивные запросы. Может пригодится при поиске проблем с рекурсивными запросами.

Теперь перейдём к параметрам поинтереснее:

- **allow-query** - разрешить запросы - и в скобках мы указываем откуда. **listen-on** просто позволяет сервису слушать на указанных адресах, но это не значит, что он будет отвечать всем, кто пришёл на этот адрес. В нашем случае, сейчас:

```
allow-query { localhost; };
```

настроено так, что сервер будет отвечать только на свои запросы, а на все запросы с других IP адресов он будет отвечать отказом. В скобках мы можем указать конкретные IP адреса, подсети или просто **any**, чтобы отвечать всем.

Так как мы поднимаем сервис для нашей локальной сети, давайте укажем 10.0.0.0/8:

```
allow-query { localhost; 10.0.0.0/8; };
```

И ещё раз напомним - не забывайте ставить точку с запятой в конце.


```

/*
- If you are building an AUTHORITATIVE DNS server, do NOT enable recursion.
- If you are building a RECURSIVE (caching) DNS server, you need to enable
  recursion.
- If your recursive DNS server has a public IP address, you MUST enable access
  control to limit queries to your legitimate users. Failing to do so will
  cause your server to become part of large scale DNS amplification
  attacks. Implementing BCP38 within your network would greatly
  reduce such attack surface
*/
recursion yes;

```

Обратите внимание, что нигде в этом конфиг файле нет какого-то комментария с объяснением параметра кроме одного специфичного места:

- **recursion** - позволяет сделать DNS сервер рекурсивным или отрубить этот функционал. Рекурсивный DNS сервер - это такой DNS сервер, который, получая запрос от клиента, ищет у себя в кэше, и если не найдёт - он идёт на другой DNS сервер, чтобы узнать ответ там. Узнав ответ, он отвечает клиенту и сохраняет ответ у себя в кэше, на случай, если будут ещё такие запросы.

То есть задача рекурсивного DNS сервера - находиться поближе к клиентам и быстро отвечать на запросы, чтобы ускорить работу с DNS и не гонять трафик просто так. Поэтому его также называют кэширующим DNS сервером.

Если есть DNS сервера, которые перенаправляют запросы, значит есть и сервера, которые предоставляют ответы - авторитетные DNS сервера. Задача таких DNS серверов - предоставлять информацию о зонах. К примеру, если у вас есть домен и вы хотите для него создать DNS записи, то вам нужен авторитетный DNS сервер.

Скажем, DNS на вашем домашнем дешёвом роутере может быть только кэширующим. Если вы поднимаете DNS сервер в интернете, чтобы делиться информацией о вашем домене, то не стоит его делать рекурсивным, иначе с помощью вашего сервера будут делать DDOS атаки. Об этом и предупреждают в комментариях в этом конфиге.

Но не то чтобы авторитетный и рекурсивный сервера были взаимоисключающими. Например, немало компаний используют Microsoft Active Directory, где встроена служба DNS, которая одновременно и рекурсивная, и авторитетная. Да, к DNS сервису Active Directory из интернета доступ давать не стоит, но вот внутри сети компании можно пользоваться.

В этой теме мы сделаем кэширующий DNS сервер, поэтому оставляем этот параметр со значением **yes**.

```

*/
recursion yes;

forwarders { 1.1.1.1; 8.8.8.8; };

```

И так, мы решили, что наш DNS сервер будет рекурсивным. А рекурсивный должен узнавать ответы от других серверов. А из каких? Правильно, нам нужен параметр:

- **forwarders** - указывает, к каким DNS серверам нужно обращаться за ответами. Здесь вы можете указать DNS адреса провайдеров, так как они обычно ближе по сети, или какие-либо другие более доверенные сервера. Тут уже на ваше усмотрение. По личному опыту, я встречал несколько раз ситуации, когда у провайдеров возникают проблемы с DNS серверами и от этого страдает сеть

компании. Но на вопрос «кому доверять DNS запросы компании в интернете» нет простого ответа, и, для примера, я просто указал сервера Cloudflare и Google.

```
forwarders { 1.1.1.1; 8.8.8.8; };
```

Дальше идёт несколько параметров, связанных с DNSSEC, но мы это разберём в отдельной теме, поэтому сейчас их пропустим.

```
logging {
    channel default_debug {
        file "data/named.run";
        severity dynamic;
    };
};
```

Перейдём к секции logging. bind делит логи на различные категории.

client

Processing of client requests.

cname

Name servers that are skipped for being a CNAME rather than A/AAAA records.

config

Configuration file parsing and processing.

database

Messages relating to the databases used internally by the name server to store zone and cache data.

default

Logging options for those categories where no specific configuration has been defined.

dispatch

Dispatching of incoming packets to the server modules where they are to be processed.

dnssec

DNSSEC and TSIG protocol processing.

dnstap

The **dnstap** DNS traffic capture system.

Категории могут быть связаны с клиентскими запросами, конфиг файлами, dnssec и прочим.

Эти категории можно направить в различные каналы.

8.2.8.1. The `channel` Phrase

`channel`

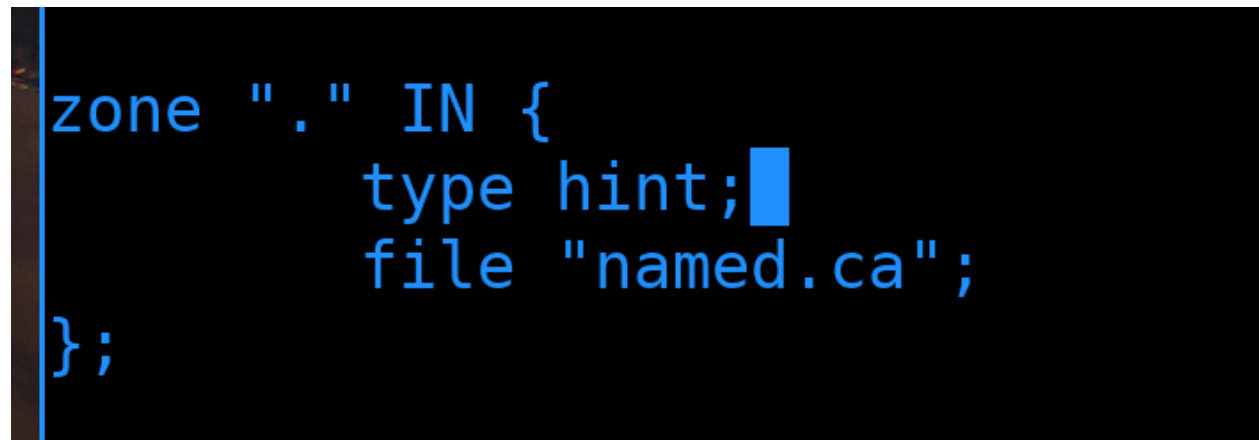
Grammar:

```
channel <string> {
    buffered <boolean>;
    file <quoted_string> [ versions ( unlimited | <integer> ) ] [ size <size> ] [ suffix
    null;
    print-category <boolean>;
    print-severity <boolean>;
    print-time ( iso8601 | iso8601-utc | local | <boolean> );
    severity <log_severity>;
    stderr;
    syslog [ <syslog_facility> ];
}; // may occur multiple times
```

Вы создаёте каналы, где указываете, что делать с логом - записать в файл, отправить в syslog, добавить ли время лога, или может в целом ничего не делать с этим логом.

И в секции logging вы указываете, какую категорию в какие каналы стоит направлять:

```
logging {
    category default { default_logfile; default_debug; };
    category unmatched { null; };
};
```



```
zone "." IN {
    type hint;
    file "named.ca";
};
```

Дальше у нас секция про корневую зону «.».

```

.          518400  IN      NS      m.root-servers.net.

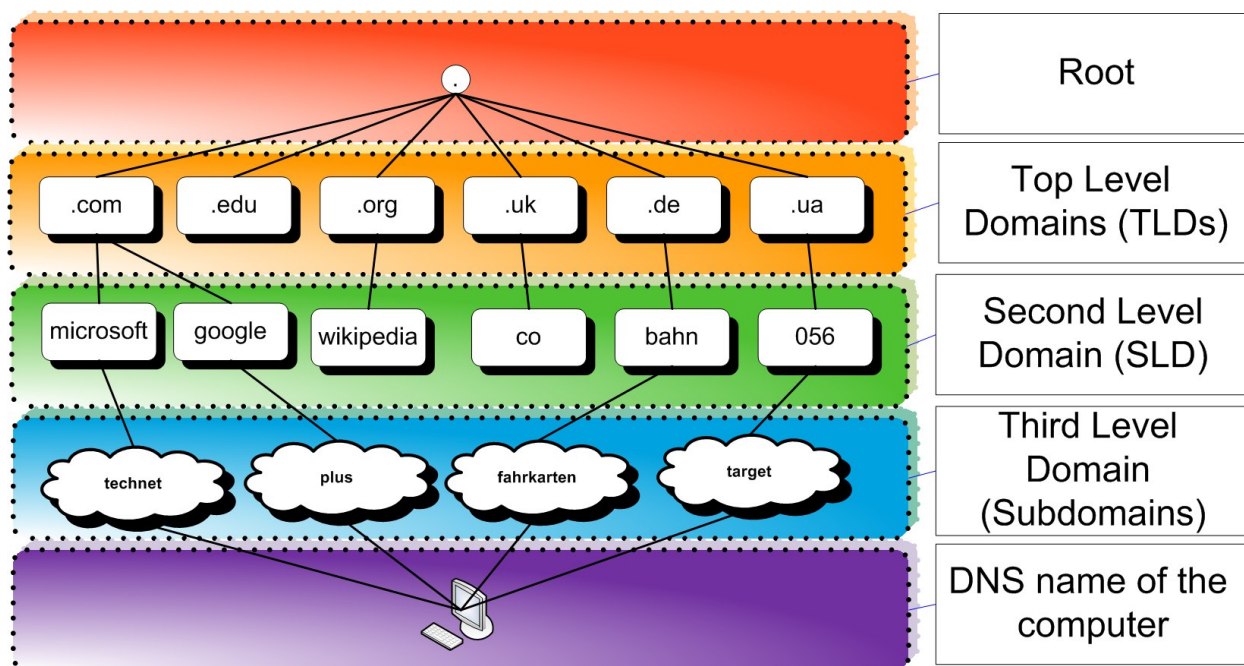
;; ADDITIONAL SECTION:
a.root-servers.net. 518400  IN      A        198.41.0.4
b.root-servers.net. 518400  IN      A        199.9.14.201
c.root-servers.net. 518400  IN      A        192.33.4.12
d.root-servers.net. 518400  IN      A        199.7.91.13
e.root-servers.net. 518400  IN      A        192.203.230.10
f.root-servers.net. 518400  IN      A        192.5.5.241
g.root-servers.net. 518400  IN      A        192.112.36.4
h.root-servers.net. 518400  IN      A        198.97.190.53

```

Давайте посмотрим на содержимое указанного файла `named.ca`:

```
sudo cat /var/named/named.ca
```

Здесь мы видим адреса корневых DNS серверов.



Помните, мы говорили, что DNS адрес как путь в Linux, только читается наоборот и вместо slash - точка? Благодаря этому файлу наш DNS сервер знает об адресах корневых DNS серверов. Если бы мы оставили параметр `recursion yes`, при этом не добавив `forwarders`, то наш DNS сервер сначала бы послал запрос на эти корневые сервера, чтобы узнать DNS сервера для доменов первого уровня, затем обращался к тем, чтобы найти адреса DNS серверов для доменов второго уровня и дальше в зависимости от необходимой зоны. В прошлой теме мы с `nslookup` проходились по всей этой цепочке, поэтому советуем предварительно посмотреть предыдущую тему.

Так вот, если бы за каждым запросом мы бы ходили через такую огромную цепочку, то всё работало бы в разы медленнее, не говоря уже о том, что нагрузка на все эти сервера была бы безумной. Вот почему важно указывать `forwarders`.

Запуск сервиса

```

user@ns1:~$ sudo systemctl restart named
[sudo] password for user:
user@ns1:~$

```

И так, давайте сохраним файл. Так как мы поменяли пару настроек, не забудем перезапустить named для применения наших изменений:

```
sudo systemctl restart named
```

Мы сделали restart, так как пока что нашим DNS сервером никто не пользуется и перезагрузка сервиса не страшна. Но в будущем, когда нужно будет применить настройки на рабочем DNS сервисе, которым уже пользуются клиенты, так делать не стоит, потому что это приведёт к недоступности DNS сервера, а это может привести к серьезным последствиям.

```

[user@ns1 ~]$ sudo systemctl cat named | grep Reload
ExecReload=/bin/sh -c 'if /usr/sbin/rndc null > /dev/null 2>&1; then /usr/sbin/rndc reload; else /bin/kill -HUP $MAINPID; fi'
[user@ns1 ~]$

```

Для безболезненного применения настроек можем использовать reload вместо restart:

```
sudo systemctl cat named | grep Reload
```

Как видите, reload задействует утилиту rndc - с помощью неё можно управлять bind, при чем не только локальным.

```

user@ns1:~$ sudo journalctl -eu named
Jan 21 21:57:00 ns1 named[1569]: host unreachable resolving './NS/IN': 198.97.190.53#53
Jan 21 21:57:00 ns1 named[1569]: timed out resolving './DNSKEY/IN': 1.1.1.1#53
Jan 21 21:57:00 ns1 named[1569]: host unreachable resolving './NS/IN': 192.58.128.30#53
Jan 21 21:57:00 ns1 named[1569]: network unreachable resolving './DNSKEY/IN': 2001:500:1::53#53
Jan 21 21:57:00 ns1 named[1569]: network unreachable resolving './DNSKEY/IN': 2001:503:c27::2:30#53
Jan 21 21:57:00 ns1 named[1569]: network unreachable resolving './DNSKEY/IN': 2001:500:a8::e#53
Jan 21 21:57:00 ns1 named[1569]: network unreachable resolving './DNSKEY/IN': 2001:500:9f::42#53

```

Если вы где-то опечатались, а очень часто новички забывают где-то точку с запятой и сервис потом не стартует, то легче всего посмотреть в логи, где можно будет найти номер строки с ошибкой:

```
sudo journalctl -eu named
```

В моих логах я вижу другую ошибку - host unreachable resolving, timeout и прочее. Это говорит о том, что у моего DNS сервера нет доступа к указанным DNS серверам.

Настройка firewall

```
user@ns1:~$ ping 1.1.1.1
PING 1.1.1.1 (1.1.1.1) 56(84) bytes of data.
64 bytes from 1.1.1.1: icmp_seq=1 ttl=56 time=11.5 ms
64 bytes from 1.1.1.1: icmp_seq=2 ttl=56 time=12.2 ms
^C
--- 1.1.1.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 11.492/11.838/12.185/0.363 ms
[user@ns1 ~]$
```

Давайте попытаемся разобраться. Пустим ping на 1.1.1.1, чтобы понять, есть ли сетевой доступ:

```
ping 1.1.1.1
```

Да, единички пингуются, значит у этой системы есть доступ в интернет.

```
user@ns1:~$ ping google.com
ping: google.com: Name or service not known
[user@ns1 ~]$ ^C
[user@ns1 ~]$
```

Однако, если мы попытаемся пингануть google.com то увидим ошибку Name or service not known:

```
ping google.com
```

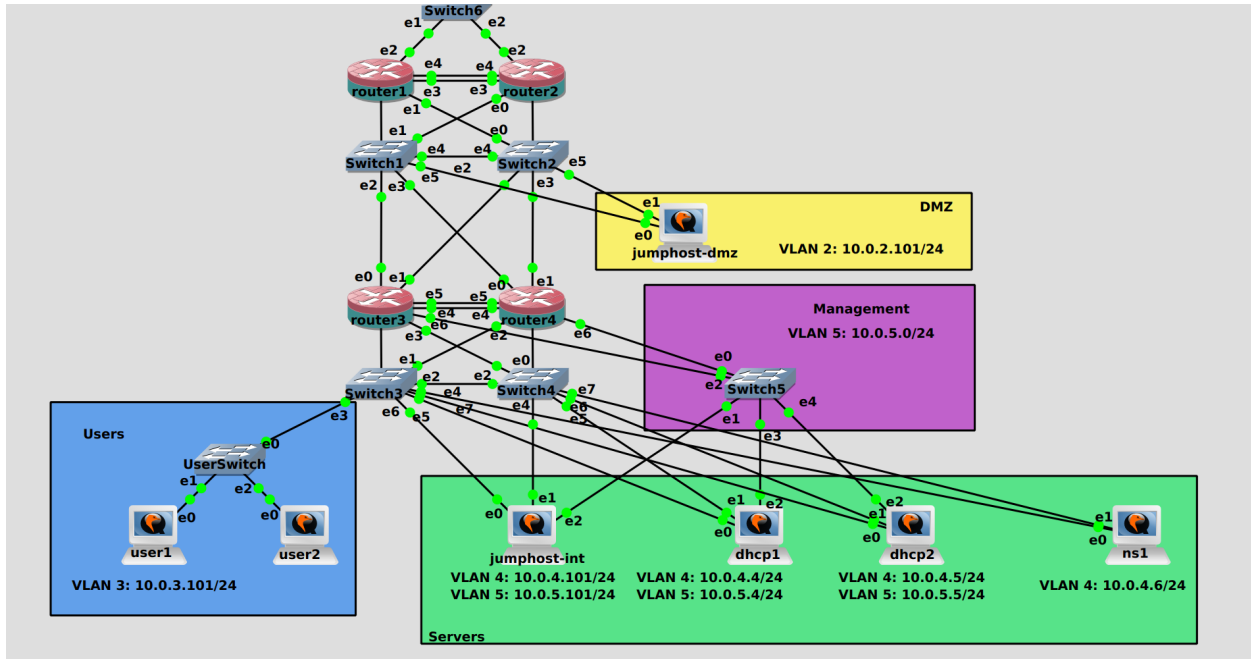
Это нам говорит о том, что не резолвятся имена.

```
user@ns1:~$ cat /etc/resolv.conf
# Generated by NetworkManager
nameserver 1.1.1.1
nameserver 8.8.8.8
[user@ns1 ~]$
```

Давайте проверим resolv.conf:

```
cat /etc/resolv.conf
```

И здесь мы видим, что у нас прописаны эти самые единички, и они у нас пингуются. И так, пинг у нас доходит, но DNS запросы нет. Почему?



Самое время вспомнить, что у нас сервера существуют не в вакууме, а в нашей сети. И в нашей сети есть файрволы, которые не пускают ничего, кроме пингов. Значит нам надо разрешить нашим DNS серверам доступ в интернет, хотя бы для сервиса DNS.

```
srv_to_internet (active)
priority: -1
target: CONTINUE
ingress-zones: servers
egress-zones: routers
services: dns http https
ports:
protocols:
masquerade: no
forward-ports:
source-ports:
icmp-blocks:
rich rules:
```

```
[user@router3 ~]$ sudo firewall-cmd --list-all-policies
```

Идём на router3 и смотрим все политики:


```
sudo firewall-cmd --list-all-policies
```

И обнаруживаем, что оказывается, мы уже разрешали нашим серверам доступ в интернет, в том числе по DNS.

```
srv1_all_ssh (active)
  priority: -1
  target: CONTINUE
  ingress-zones: management servers
  egress-zones: ANY
  services:
  ports:
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
    rule family="ipv4" source address="10.0.5.101/32" destination
    = "tcp" accept
    rule family="ipv4" source address="10.0.4.101/32" destination
    = "tcp" accept

[user@router4 ~]$ sudo firewall-cmd --list-all-policies
```

Однако, если зайти на router4, можем заметить, что такой политики нет.

```
sudo firewall-cmd --list-all-policies
```

Мы забыли её добавить. Ладно, не страшно, быстренько вспоминаем:

```
[user@router4 ~]$ sudo firewall-cmd --permanent --new-policy=srv_to_internet
success
[user@router4 ~]$ sudo firewall-cmd --permanent --policy=srv_to_internet --add-ingress-zone=servers
success
[user@router4 ~]$ sudo firewall-cmd --permanent --policy=srv_to_internet --add-egress-zone=routers
success
[user@router4 ~]$
[user@router4 ~]$ sudo firewall-cmd --permanent --policy=srv_to_internet --add-service={dns,http,https}
success
[user@router4 ~]$ sudo firewall-cmd --reload
success
[user@router4 ~]$
```

Просто скопируем из [второй темы по DHCP](#) команды для создания политики, добавления в неё зон, добавление сервисов и, собственно, рестарт firewall.

```
sudo firewall-cmd --permanent --new-policy=srv_to_internet
sudo firewall-cmd --permanent --policy=srv_to_internet --add-ingress-zone=servers
sudo firewall-cmd --permanent --policy=srv_to_internet --add-egress-zone=routers
sudo firewall-cmd --permanent --policy=srv_to_internet --add-service={dns,http,https}
sudo firewall-cmd --reload
```



```
[user@ns1 ~]$ ping google.com
PING google.com (216.58.215.78) 56(84) bytes of data.
64 bytes from waw02s16-in-f14.1e100.net (216.58.215.78): icmp_seq=1 ttl=116 time=10.5 ms
64 bytes from waw02s16-in-f14.1e100.net (216.58.215.78): icmp_seq=2 ttl=116 time=12.6 ms
^C
--- google.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 10.521/11.560/12.599/1.039 ms
[user@ns1 ~]$
```

Возвращаемся на ns1 и проверяем пинг:

```
ping google.com
```

Теперь всё пингуется и резолвится. Но, обратите внимание, пинг идёт не благодаря нашему новому DNS сервису, в `/etc/resolv.conf` были прописаны единички. То есть мы пока что не тестировали наш DNS сервер.

То что мы подняли DNS сервер не значит, что все компьютеры в нашей сети автоматом стали его использовать. Они даже его не знают. А как они могут узнать, что у нас в сети есть DNS сервер? Правильно, DHCP! Это DHCP сервер говорит всем нашим компьютерам адреса DNS серверов.

```
[user@dhcp1 ~]$ nslookup google.com 10.0.4.53
Server:                10.0.4.53
Address:                10.0.4.53#53

Non-authoritative answer:
Name:   google.com
Address: 142.250.203.206
Name:   google.com
Address: 2a00:1450:401b:80e::200e

[user@dhcp1 ~]$
```

Пойдём на DHCP сервера и прежде всего проверим, могут ли они использовать наш DNS сервер. Как проверить, работает ли DNS сервер? Правильно, к примеру с помощью `nslookup` с указанием сервера:

```
nslookup google.com 10.0.4.53
```

Мы видим, что есть ответ. Значит что? Значит наш DNS работает. Но наш DHCP сервер находится в той же подсети, что и DNS сервер, а это значит, что мы не учитываем firewall.

```
AlmaLinux 8.5 (Arctic Sphynx)
Kernel 4.18.0-348.2.1.el8_5.x86_64 on an x86_64

Activate the web console with: systemctl enable --now cockpit.socket

user1 login: root
Password:
Last login: Sun Jan 23 18:46:09 on tty1
[root@user1 ~]# nslookup google.com 10.0.4.53
;; connection timed out; no servers could be reached

[root@user1 ~]# _
```

Попробуем тоже самое с пользовательского компьютера.

```
nslookup google.com 10.0.4.53
```

И на этот раз мы видим `connection timed out`. Как мы и предполагали, все кто находится в серверной сети могут достучаться до DNS, а вот для остальных стоит прописать политику на наших файрволах.

```
[user@router3 ~]$ sudo firewall-cmd --new-policy=dns --permanent
success
[user@router3 ~]$
[user@router3 ~]$ sudo firewall-cmd --policy=dns --add-ingress-zone=ANY --permanent
success
[user@router3 ~]$ sudo firewall-cmd --policy=dns --add-egress-zone=servers --permanent
success
[user@router3 ~]$ sudo firewall-cmd --policy=dns --permanent --add-rich-rule='rule family=ipv4 source address=10.0.0.0/8 destination address=10.0.4.53/32 port port=53 protocol=udp accept'
success
[user@router3 ~]$ sudo firewall-cmd --reload
success
[user@router3 ~]$
```

Идём на router3. Добавим политику, назовём её просто - dns:

```
sudo firewall-cmd --new-policy=dns --permanent
```

Для этой политики входящей зоной будет любая - ANY, а исходящей - сервера:

```
sudo firewall-cmd --policy=dns --add-ingress-zone=ANY --permanent
sudo firewall-cmd --policy=dns --add-egress-zone=servers --permanent
```

Мы не хотим, чтобы любой пользователь мог подключиться к любому серверу по DNS, а только к одному определённом. Поэтому добавляем rich-rule:

```
sudo firewall-cmd --policy=dns --permanent --add-rich-rule='rule family=ipv4 source
↪address=10.0.0.0/8 destination address=10.0.4.53/32 port port=53 protocol=udp accept'
```

Не забываем перезапустить файрвол:

```
sudo firewall-cmd --reload
```

Ну и самое главное, на этот раз, не забудем добавить тоже самое на втором файрволе. Будь у нас какой-нибудь коммерческий firewall, он бы в режиме репликации копировал настройки на оба наших файрвола, но вряд ли firewalld рассчитан на такое применение. Но я в своих темах показываю идею, что и для чего делать, и в данном случае: поднял сервис - надо добавить на файрволе.

```
[root@user1 ~]# nslookup google.com 10.0.4.53
Server:      10.0.4.53
Address:     10.0.4.53#53

Non-authoritative answer:
Name:   google.com
Address: 142.250.203.206
Name:   google.com
Address: 2a00:1450:401b:80d::200e

[root@user1 ~]# _
```

Вернёмся и проверим снова - и теперь мы видим, что user1 получает ответ от DNS сервера.

Настройка DHCP

Отлично, значит мы можем обновить настройки на DHCP серверах.

```
GNU nano 2.9.8 /etc/dhcp/dhcpd.conf.d/subnets.conf

option domain-name-servers 10.0.4.53;

min-lease-time 6912000;
default-lease-time 7776000;
max-lease-time 8640000;
```

Идём на dhcp1, заходим в настройки подсетей:

```
sudo nano /etc/dhcp/dhcpd.conf.d/subnets.conf
```

Раньше в параметре `domain-name-servers` у нас было два DNS сервера, но мы пока что подняли только один, так что укажем только его:

```
option domain-name-servers 10.0.4.53;
```

```
[user@dhcp1 ~]$ sudo nano /etc/dhcp/dhcpd.conf.d/subnets.conf
[user@dhcp1 ~]$ sudo systemctl restart dhcpd
[user@dhcp1 ~]$
```

Сохраняем изменения, выходим и перезапускаем сервис для применения настроек.

```
[user@dhcp2 ~]$ sudo nano /etc/dhcp/dhcpd.conf.d/subnets.conf
[sudo] password for user:
[user@dhcp2 ~]$ sudo systemctl restart dhcpd
[user@dhcp2 ~]$
```

Продельываем ту же операцию на втором DHCP сервере.

```
[root@user1 ~]# nmcli connection up ens3
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/178)
[root@user1 ~]# cat /etc/resolv.conf
# Generated by NetworkManager
nameserver 10.0.4.53
[root@user1 ~]# nslookup google.com
Server:      10.0.4.53
Address:     10.0.4.53#53

Non-authoritative answer:
Name:   google.com
Address: 142.250.203.206
Name:   google.com
Address: 2a00:1450:401b:80e::200e

[root@user1 ~]#
```

Мы могли бы дождаться, пока dhcp клиенты свяжутся с сервером по поводу обновления lease time и получают новые настройки, но это может занять время. И чтобы ускорить этот процесс, я просто переподключусь к сети на пользовательском компьютере:

```
nmcli con up ens3
```

Проверю /etc/resolv.conf:

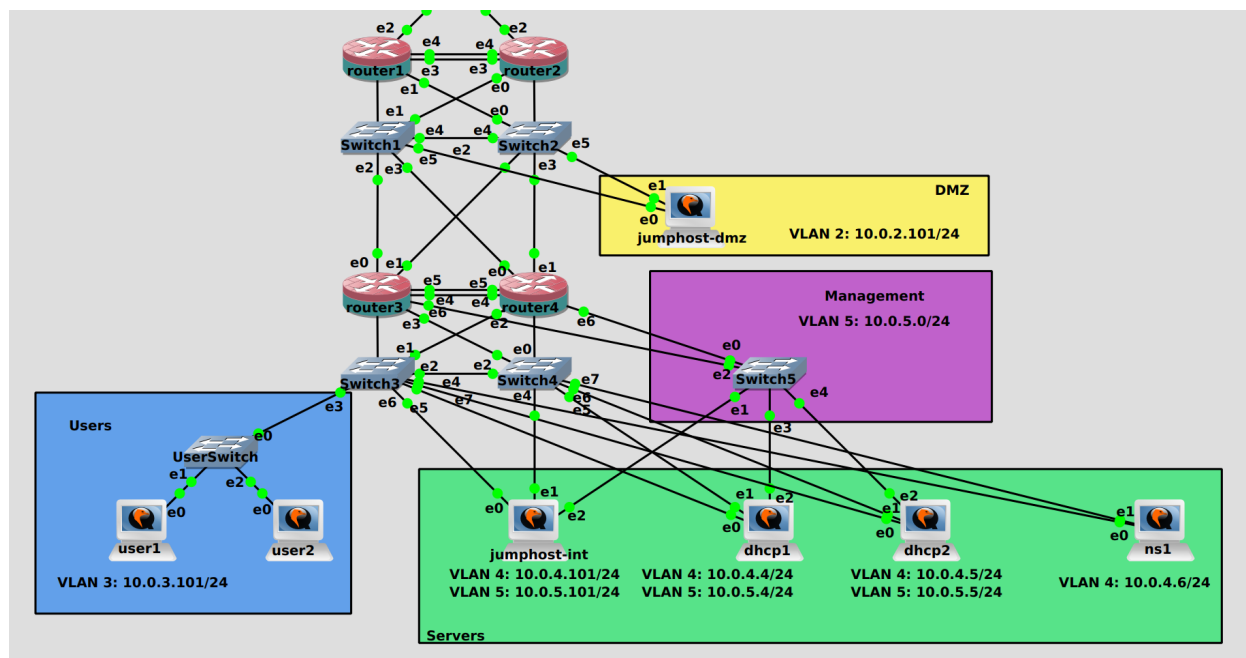
```
cat /etc/resolv.conf
```

И здесь мы видим, что адрес DNS сервера обновился. Можем для проверки ещё раз пустить nslookup:

```
nslookup 10.0.4.53
```

И всё работает.

Итоги



По хорошему, нам теперь надо вернуться на файрвол, убрать доступ к DNS в интернете всем серверам и прописать специфичное правило - чтобы только ns1 мог подключаться в интернет только к своим forwarders. Но это будет заданием для вас.

Ну и другое задание - поднять ns2. Поднять второй кэширующий DNS сервер будет довольно просто - нужно поднять точно такой же сервис, точно с такими же настройками, добавить его на файрволе и в DHCP. Сервера между собой никак не связаны, не делятся никакими пулами, как тот же DHCP, не нужно разделять один и тот же IP адрес. Все DNS клиенты поддерживают несколько DNS серверов, благодаря чему можно прописав на клиентах два DNS адреса сделать отказоустойчивость.

И так, подытожим. Мы сегодня поставили bind, прошлись по настройкам и настроили его как кэширующий DNS сервер. При этом интегрировали его в нашу сеть - обновили правила на файрволах, обновили настройки DHCP серверов и проверили работоспособность. В будущих темах мы разберём, как поднять авторитетный DNS сервер, поговорим про DNSSEC, DNS-over-TLS и прочее.