# Лабораторные работы

- Лабораторная работа 0. Изучение работы текстового редактора VIM.
- Лабораторная работа 1. Работа с консолью в Linux.
- Лабораторная работа 2. Работа с дисковой подсистемой ОС Linux.
- Лабораторная работа 3. Работа с сетью в Linux. Виртуальный коммутатор Linux Bridge.
- Лабораторная работа 4. Основы виртуализации в Linux. QEMU/KVM.
- Лабораторная работа 5. Основы виртуализации в Linux. Libvirt.
- Лабораторная работа 6. Основы виртуализации в Linux. Отказоустойчивый кластер на базе Corosync/Pacemaker.
- Лабораторная работа 7. Основы виртуализации в Linux. Динамическая миграция ресурсов в отказоустойчивом кластере на базе Corosync/Pacemaker.

# Лабораторная работа 0. Изучение работы текстового редактора VIM.

Цель

Получение базовых навыков при работе с основным текстовым редактором операционной системе Linux (CentOS 7).

Задачи

1. Подключиться к облачной платформе
2. Установить пакет vim
3. Пройти обучение работе с vim.

Схема виртуального стенда




Виртуальный сервер labnode

Для работы с облачной платформой необходимо прочитать инструкцию. Переключиться на проект [GROUP]:[team]-lab:1-2. Подключиться к `labnode`, логин - `labuser`, пароль - `labpass1!`

## Задание 1. Подключиться к облачной платформе.

Для работы с облачной платформой необходимо прочитать `инструкцию`.

## Задание 2. Установить пакет `vim`

Для установки, удаления, обновления пакетов в ОС CentOS 7 используется утилита yum (аббр. Yellowdog Updater, Modified). В процессе установки/обновления/удаления пакетов

нужно будет подтвердить установку новых версий, нажав `y`. Или же использовать флаг `-y`. Необходимо установить пакет vim, в котором содержится необходимое обучающее руководство.

```
sudo yum install vim
```

## Задание 3. Пройти обучение работе с vim.

Ввести в консоли следующую команду, для запуска интерактивного обучающего курса по работе с VIM:

```
vimtutor ru
```

Пройти интерактивный курс до конца.

# Лабораторная работа 1. Работа с консолью в Linux.

Цель

Получение базовых навыков при работе с консолью в операционной системе Linux (CentOS 7).

Задачи

1. Подключиться к виртуальной машине в RESDS Dashboard.
2. Выполнить базовые действия с файлами и папками.
3. Установить пакет wget.
4. Научиться работать с переменными окружения.
5. Создать нового пользователя и подключиться к нему. Поменять shell у пользователя.
6. Изучить работу с текстом в Bash.

Схема виртуального стенда




Виртуальный сервер labnode

Для работы с облачной платформой необходимо прочитать инструкцию.

Переключиться на проект [GROUP]:[team]-lab:1-2. Подключиться к `labnode`, логин - `labuser`, пароль - `labpass1!`

## Задание 1. Работа с файловой системой.

1. Создать директорию task01. Перейти в неё.

```
mkdir task01
cd task01
```

2. Создать в домашнем каталоге текстовый файл user при помощи текстового редактора `vi`, и заполнить его произвольными символами.

```
vi user
```

3. Скопировать файл `user` в новый файл с именем `root`.

```
cp user root
```

4. Посмотреть права доступа на файлы можно с помощью команды:

```
ls -l
```

5. Задать владельца root и группу root на файл root. (sudo позволяет выполнять команды от `root`; ввести пароль в диалоговом окне, при этом символы отображаться не будут; пароль - `labpass1!`)

```
sudo chown root:root root
```

6. Переименовать (т.е. переместить с новым именем) файл `user` в файл `lock`.

```
mv user lock
```

7. На файл root поставить доступ на чтение и запись группе и владельцу остальным только на чтение.

```
sudo chmod 664 root
```

8. На файл lock поставить доступ на чтение владельцу, группе и остальным пользователям убрать доступ на чтение запись и исполнение.

```
chmod 600 lock
```

9. Вывести содержимое файла root в терминал.

```
cat root
```

10. Отредактировать файл root случайным образом и вывести его содержимое в консоль.

```
sudo vi root
cat root
```

11. Удалить каталог task01.

```
cd
sudo rm -rf task01
```

# Задание 2. Установка пакетов.

Для установки, удаления, обновления пакетов в ОС CentOS 7 используется утилита yum (аббр. Yellowdog Updater, Modified). В процессе установки/обновления/удаления пакетов нужно будет подтвердить установку новых версий, нажав `y`. Или же использовать флаг `-y`.

Для установки/удаления пакетов существуют команды:

```
sudo yum install -y package_name
sudo yum remove -y package_name
```

В качестве примера необходимо установить программу для скачивания файлов по web-протоколам - `wget`.

```
sudo yum install wget
```

Можно проверить версию установленного пакета:

```
wget –-version
```

# Задание 3. Создание пользователей.

Создать нового пользователя newuser с паролем `newpass1!`. Сделать его администратором.

```
sudo adduser newuser
sudo passwd newuser # Ввести пароль newpass1!, при этом пароль отображаться не будет
sudo usermod -aG wheel newuser # Добавить пользователя newuser в группу wheel, что даст ему права на исполнение команд с sudo
```

Для того, чтобы выполнять команды от имени пользователя `newuser`, необходимо сменить текущее окружение на окружение пользователя newuser командой - `su`. При этом нужно будет ввести пароль пользователя `newuser`.

```
su - newuser
```

Имя пользователя изменилось с `labuser` на `newuser`. Также, имя пользователя хранится в переменной окружения `$USER`. Просмотреть значение переменной можно с помощью команды `echo`.

```
echo $USER
```

Чтобы узнать, в каких группах состоит текущий пользователь, используется команда `groups`. В данном случае это должны быть группы `newuser` и `wheel`.

```
groups
```

Для выхода из оболочки используется команда `exit`. При этом выход идет в оболочку, запущенную пользователем `labuser`.

```
exit
```

Теперь нужно сменить оболочку, запускаемую по умолчанию при входе пользователя `newuser`. Чтобы узнать список доступных в системе оболочек, используется следующая команда:

```
cat /etc/shells
```

Сейчас должна быть запущена оболочка `/bin/bash`. Чтобы узнать, какая оболочка запущена сейчас, можно вывести на экран значение переменной `$0`.

```
echo $0
```

Чтобы просто запустить оболочку, достаточно просто набрать путь к ней. Запустить оболочку - `sh`, и вывести на экран переменную - `$0`. После выйти обратно в `bash`.

```
/bin/sh
echo $0
exit
```

Чтобы сменить оболочку по умолчанию для пользователя, можно отредактировать файл `/etc/passwd`, либо можно использовать утилиту `usermod`. Необходимо посмотреть, какая оболочка сейчас является оболочкой по умолчанию для пользователя `newuser`, сменить её на `/bin/sh`, и проверить изменения.

```
sudo grep newuser /etc/passwd # grep - программа для поиска по тексту. В данном случае, выведет все
строки, содержащие newuser.
sudo usermod --shell /bin/sh newuser
sudo grep newuser /etc/passwd # Данная команда произведет поиск по файлу
```

```
# /etc/passwd, и отобразит в консоли все
# строки, в которых присутствует слово
# newuser. В этой же строке будет указано,
# какая оболочка используется данным
# пользователем
```

Теперь необходимо переключиться в режим работы от имени пользователя `newuser`, и проверить, какая оболочка используется. После выйти из этого режима, и удалить пользователя.

```
su - newuser
echo $0
exit
sudo userdel -r newuser # флаг r используется, когда вы хотите удалить также домашний каталог
пользователя.
```

# Задание 4. Работа с текстом в bash.

Для практики нужно воспользоваться текстовым файлом, специально созданным для выполнения задания - таблица подключений OpenVPN, состоящая из имени клиента, IP адреса, MAC адреса устройства и внешнего IP, с которого клиент подключился. Необходимо загрузить файл, воспользовавшись специальной утилитой s3cmd, загружающего файлы с облачного хранилища, работающему по специальному протоколу s3(simple storage server):

```
sudo yum install s3cmd -y
sudo yum upgrade -y
cd ~
cp /var/lib/cloud/s3cfg .s3cfg
s3cmd get s3://lab1/clients.txt ~/
```

Вывести содержимое файла в консоль (Тут для удобства можно пользоваться встроенными в консоль горячими клавишами: Ctrl+L - Очистить содержимое консоли, Shift+PgUp - Прокрутить консоль вверх, Shift+PgDn - Прокрутить консоль вниз).

```
cat ~/clients.txt
```

Для того, чтобы вывести строки, содержащие подстроку, можно использовать grep. Необходимо вывести информацию о клиенте под номером 24 (grep - регистрозависимый, Client24 начинается с заглавной буквы).

```
grep Client24 ~/clients.txt
```

В результате исполнения команды должна быть выведена одна строка. Если было несколько строк, содержащих подстроку Client24, то в результате выведется несколько строк. Далее необходимо ввести команду:

```
grep Client2 ~/clients.txt
```

Необходимо попробовать понять, сколько будет выведено строк и почему именно они.

Помимо обычных строк grep поддерживает также регулярные выражения. При этом регулярные выражения должны экранироваться символом . С помощью регулярных выражений можно задать абсолютно любой паттерн. Относительно простой - вывести всех клиентов, имя которых заканчивается на 4. Точка в выражении означает любой символ.

```
grep Client\.4 ~/clients.txt
```

grep также умеет работать с пайплайном (вертикальная черта). С помощью пайплайна можно передавать вывод от одной программы другой, по принципу конвейера. Например, можно вывести текст через cat, и передать этот вывод на вход команды grep, для его обработки этой командой.

```
cat ~/clients.txt  |  grep Client\.4
```

Пайплайнов может быть несколько. Так, например, воспользовавшись программой AWK можно вывести только имя клиента и его внешний адрес - то есть первый и четвертый столбцы. (Внимание на пробел в двойных кавычках между номерами столбцов. Это разделитель, который будет между столбцами в конечном результате. Пробел можно заменить на любой другой символ, например, на дефис -, или символ табуляции \t. Можно попробовать это сделать)

```
cat ~/clients.txt  |  grep Client\.4 | awk ‘{print $1” “$4}’
```

AWK также поддерживает функции в своем синтаксисе. Например, чтобы вывести имя клиента и MAC, при этом чтобы MAC печатался заглавными буквами, нужно использовать следующую конструкцию.

```
cat ~/clients.txt  |  grep Client\.4 | awk ‘{print $1” “toupper($3)}’
```

Далее необходимо вывести MAC адрес, чтобы он был разделён не двоеточиями, а дефисами, тогда можно воспользоваться sed. Нужно лишь задать параметры для замены. После этого результат можно вывести не в консоль, а в новый файл (набирать в одну строку):

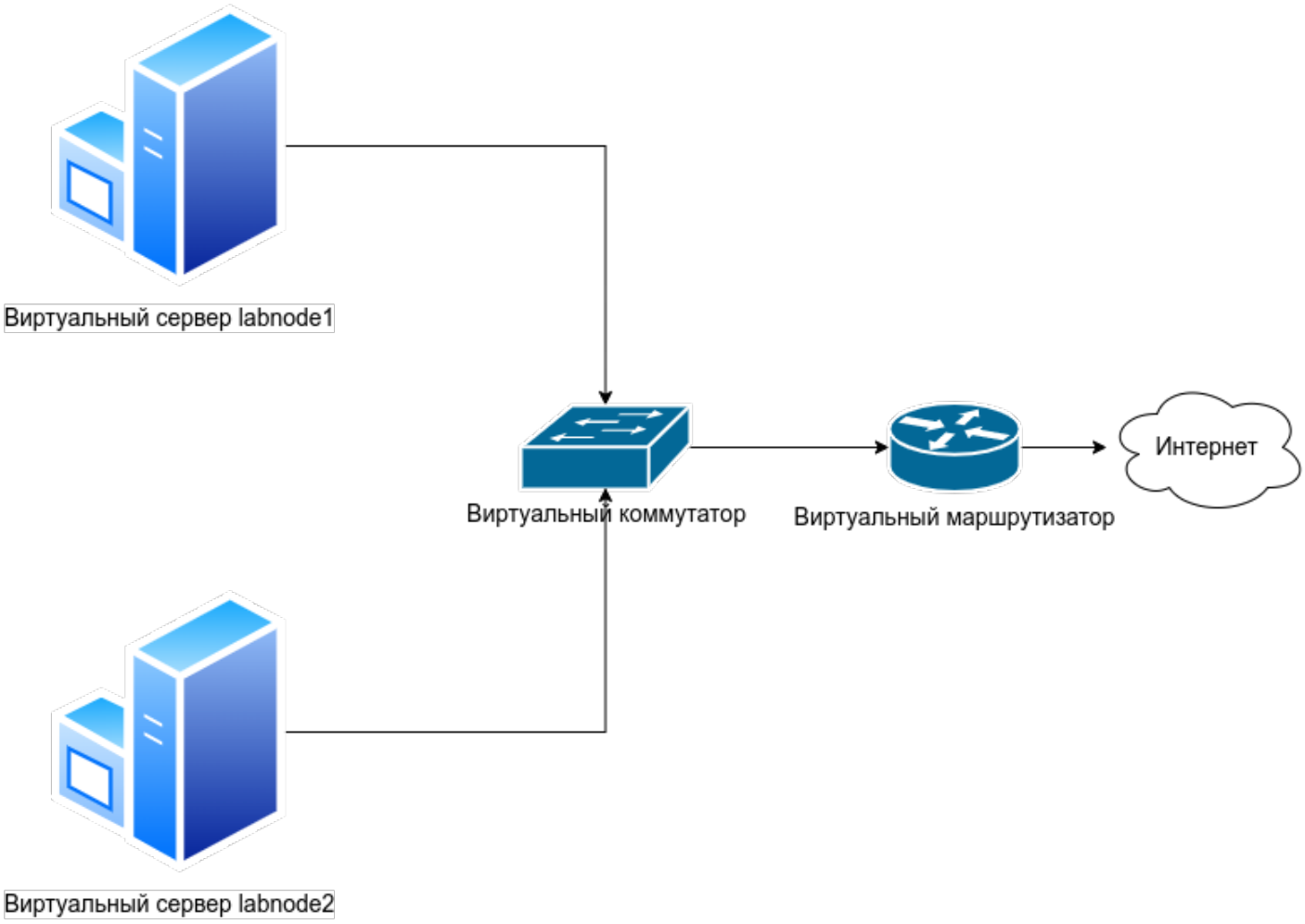
```
cat ~/clients.txt  |  grep Client\.4 | awk ‘{print $1” “toupper($3)}’ | sed -r ‘s/:/-/g’ > newfile.txt
```

# Лабораторная работа 2. Работа с дисковой подсистемой ОС Linux.

Цель:
Получение базовых навыков при работе с дисковой подсистемой в операционной системе Linux (CentOS 7).
Задачи

1. Разметить диск как DOS (MBR), создать на его разделах файловую систему.
2. Удалить разметку с диска.
3. Разметить диск как GPT.
4. Используя LVM создать физический том, группу томов и поверх них логические тома. Провести настройку этих томов.
5. Научиться работать с файловой системой на логическом томе, с её созданием и монтированием.
6. Использовать fstab для автоматизации монтирования при загрузке.

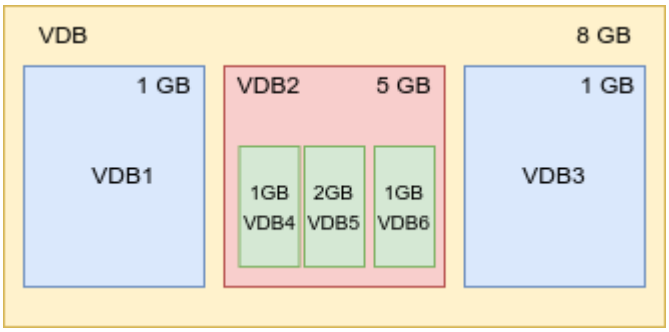
   Схема виртуального стенда




# Задание 1. Создание разделов с использованием fdisk на MBR.

Подключиться к `labnode1`, логин - `labuser`, пароль - `labpass1!`

Необходимо сделать на диске следующую разметку:




Для этого запустить fdisk в интерактивном режиме, в качестве аргумента передавая путь к блочному устройству.

```
sudo fdisk /dev/vdb
```

1. Создать разметку DOS (MBR), с помощью команды `o`.

2. Создать primary раздел. Нажать `n`, для создания нового раздела. Нажать `p`, указывая, что нужен именно primary. Номер раздела выбрать - `1` (можно ничего не выбирать, так как этот номер раздела используется по умолчанию). Утилита fdisk автоматически рассчитывает свободный сектор, с которого можно начать создание раздела. Для первого сектора первого раздела это будет сектор 2048 (можно ничего не выбирать, а просто нажать `enter` так как этот номер раздела используется по умолчанию. Для всех последующих разделов `fdisk` будет сам вычислять первый незанятый сектор, и предлагать его по умолчанию). При указании последнего сектора необходимо указать `+1G` (утилита `fdisk` автоматически рассчитает нужное количество секторов). В итоге должен получиться primary раздел на 1 ГБ. Проверить, что раздел добавлен в таблицу разделов, с помощью команды `p`.
3. Создать ещё один раздел на 5 ГБ, но с типом extended. Сделать всё то же самое как в пункте 2, но выбрать вместо primary, extended, набрав `e`, и последний сектор указать `+5G` от первого рекомендуемого.
4. Создать два логических раздела по 1 ГБ и один на 2 ГБ (логические тома могут быть созданы только при наличии extended раздела, и размещаются “внутри” extended раздела). Для этого выбрать тип раздела - logical, нажав `l`.
5. Создать еще один primary раздел на 1 ГБ.
6. Проверить получившуюся таблицу разделов, с помощью `p`. Если всё было сделано правильно, должен получиться следующий результат:

```
Command (m for help): p

Disk /dev/vdb: 8589 MB, 8589934592 bytes, 16777216 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x9878ddea

   Device Boot      Start         End      Blocks   Id  System
/dev/vdb1            2048     2099199     1048576   83  Linux
/dev/vdb2         2099200    12584959     5242880    5  Extended
/dev/vdb3        12584960    14682111     1048576   83  Linux
/dev/vdb5         2101248     6295551     2097152   83  Linux
/dev/vdb6         6297600     8394751     1048576   83  Linux
/dev/vdb7         8396800    10493951     1048576   83  Linux
```

7. После этого, необходимо записать эту таблицу на диск, нажав w. После выполнения этой команда утилита fdisk завершит работу и вернет вас в оболочку пользователя.

Проверить, что все изменения применились можно с помощью следующей команды:

```
lsblk
```

В результате на диске vdb должно отображаться 6 новых разделов.

# Задание 2. Создание файловой системы.

Создать файловые системы на разделах, созданных в предыдущем задании. Пусть:

1. На vdb1 ext4.
2. На vdb3 - xfs.
3. На vdb5 - btrfs.

Делается это так:

```
sudo mkfs.ext4 /dev/vdb1
sudo mkfs.xfs /dev/vdb3
sudo mkfs.btrfs /dev/vdb5
```

Проверить, что файловые системы были созданы.

```
sudo lsblk -f
```

Теперь, в каталоге пользователя, создать каталог media.

```
mkdir ~/media
```

Примонтировать раздел.

```
sudo mount /dev/vdb5 /home/labuser/media/
```

После этого можно размещать файлы в каталоге /home/labuser/media/ и они будут размещаться на диске vdb. Проверить куда и какие разделы примонтированы можно с помощью команды sudo mount без аргументов.

Для выполнения дальнейших этапов работы необходимо отмонтировать данный раздел:

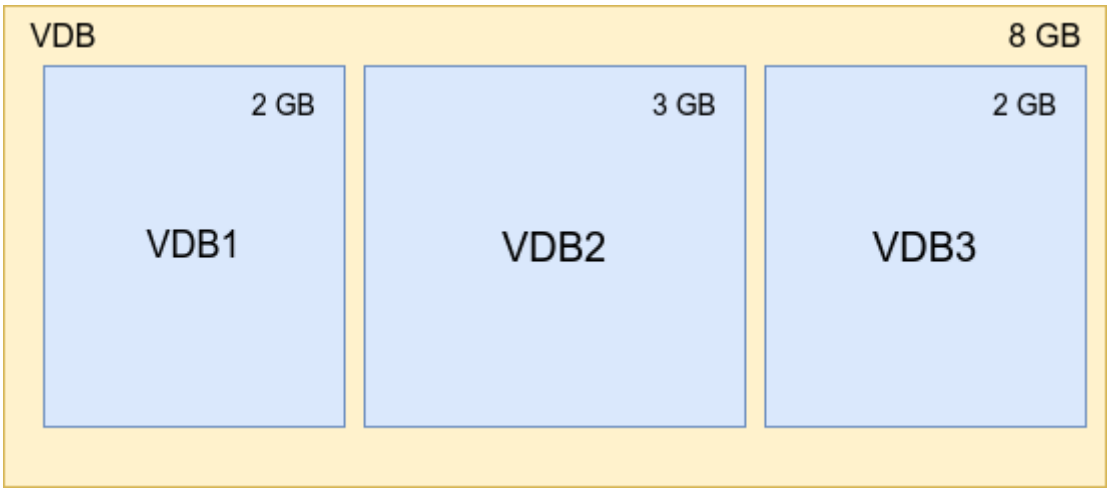
```
sudo umount /dev/vdb5
```

# Задание 3. Создание разделов с использованием fdisk на GPT.

Запустить fdisk в интерактивном режиме, в качестве аргумента передавая путь к блочному устройству.

```
sudo fdisk /dev/vdb
```

1. Выбрать таблицу разметки GPT. Делается это, нажав `g`.
2. Создать 3 раздела, согласно схеме. Сделать это с помощью команды `n`, как и в случае с MBR.




# Задание 4. Работа с LVM.

В этом задании используется разметка из задания 3. Для выполнения этого задания потребуется пакет lvm2. Установить его можно с помощью yum.

```
sudo yum install lvm2
```

Сначала необходимо изменить системный id раздела. Он влияет на то, какая метка файловой системы будет отображаться в fdisk в столбце Type. Запустить `fdisk` в интерактивном режиме, в качестве аргумента передавая путь к блочному устройству.

```
sudo fdisk /dev/vdb
```

1. Необходимо изменить тип раздела всем разделам. Нажав t, выбрать номер раздела, метку которого нужно поменять. Далее необходимо нажать L, чтобы просмотреть все доступные метки. Нужно найти Linux LVM (поиск в текстовой консоли может быть затруднен из-за длинного списка. Пролистать список вверх можно с помощью клавиш shift + PgUp, либо просто ввести значение метки - 31). Набрать id, под которым стоит нужная метка.
2. Проделать эту операцию со всеми разделами на диске.
3. Необходимо проверить введенные значения, а после записать их, нажав w.
4. Необходимо сделать все три раздела физическими томами (В процессе будет сообщение, что файловая система на томе будет уничтожена. Нужно согласиться, набрав `y`).

```
sudo pvcreate /dev/vdb1
sudo pvcreate /dev/vdb2
sudo pvcreate /dev/vdb3
```

Проверить успешность можно с помощью команды `sudo pvdisplay`. Эта команда должна вывести список всех PV(физических устройств), на которых могут быть размещены тома LVM.

5. На этих физических томах создать группу томов, и задать ей имя. Имя можно выбрать любое, например vg1:

```
sudo vgcreate vg1 /dev/vdb1 /dev/vdb2 /dev/vdb3
```

Проверить с помощью `sudo vgdisplay`. В результате должна отобразиться 1 VG (группа томов), в данном случае vg1. 6. Теперь в группе томов можно создать логические тома lv1 и lv2 размером 1 ГБ и 2 ГБ соответственно.

```
sudo lvcreate -n lv1 -L 1G vg1
sudo lvcreate -n lv2 -L 2G vg1
```

Проверить с помощью `sudo lvdisplay`. В результате должны отобразиться 2 LV (логических тома) lv1 и lv2.

7. Теперь в системе появились блочные устройства /dev/vg1/lv1 и /dev/vg1/lv2. Осталось создать на них файловую систему. Тут различий с обычными разделами нет.

```
sudo mkfs.ext4 /dev/vg1/lv1
sudo mkfs.ext4 /dev/vg1/lv2
```

8. Удаление физических томов. Удалить из группы том /dev/vdb1. Чтобы убрать из работающей группы томов раздел, сначала необходимо перенести все данные с него на другие разделы:

```
sudo pvmove /dev/vdb1
```

Затем удалить его из группы томов:

```
sudo vgreduce vg1 /dev/vdb1
```

И, наконец, удалить физический том:

```
sudo pvremove /dev/vdb1
```

Последняя команда просто убирает отметку о том, что диск является членом lvm, и не удаляет ни данные на разделе, ни сам раздел. После удаления физического тома из LVM для дальнейшего использования диск придётся переразбивать/переформатировать.

9. Добавление физических томов. Необходимо расширить VG, добавив к нему том /dev/vdb1. Чтобы добавить новый том в группу томов, создать физический том:

```
sudo pvcreate /dev/vdb1
```

Добавить его в группу:

```
sudo vgextend vg1 /dev/vdb1
```

Теперь можно создать ещё один логический диск (lvcreate) или увеличить размер существующего (lvresize).

10. Изменение размеров LVM позволяет легко изменять размер логических томов. Для этого нужно сначала изменить сам логический том:

```
sudo lvresize -L 3G vg1/lv2
```

Так как логический том является обычным дисковым (блочным) устройством, расширение этого дискового устройство никак не скажется на файловой системе, находящейся на нем, и не приведет к увеличению ее размера. Чтобы файловая система заняла все свободное место на блочном устройстве, необходимо её расширить отдельной командой: А затем файловую систему на нём:

```
sudo resize2fs /dev/vg1/lv2
```

# Задание 5. Монтирование разделов.

Удалить существующие логические LVM разделы, и создать новый.

```
sudo lvremove /dev/vg1/lv1
sudo lvremove /dev/vg1/lv2


sudo lvcreate -n media -L 6G vg1
sudo mkfs.ext4 /dev/vg1/media
```

Далее необходимо примонтировать раздел.

```
sudo mount /dev/vg1/media /home/labuser/media/
```

Записать тестовый файл test.

```
echo "string" | sudo tee ~/media/test
```

Если проблемы с доступом к записи, сменить владельца каталога. После выполнить команду заново.

```
sudo chown labuser:labuser /home/labuser/media/*
```

Отмонтировать раздел.

```
sudo umount /home/labuser/media/
```

Зайти внутрь созданного каталога, и удостовериться, что файла test там нет. Он остался на разделе. Чтобы после перезагрузки не монтировать раздел заново, нужно добавить автомонтирование в конфигурационный файл /etc/fstab (удалять из этого файла ничего нельзя! В случае ошибки в конфигурационном файле операционная система не загрузится, делать очень внимательно!). Для начало необходимо сохранить резервную копию конфигурационного файла:

```
sudo cp /etc/fstab /etc/fstab.old
```

Далее необходимо открыть его, чтобы изменить содержимое:

```
sudo vi /etc/fstab
```

Добавить туда следующую строку и сохранить.

```
/dev/vg1/media    /home/labuser/media    ext4    defaults    0    0
```

Для того, чтобы убедиться в корректности сохранения, необходимо вывести в консоль содержимое файла /etc/fstab командой:

```
sudo cat /etc/fstab
```

Необходимо проверить, что изначальное содержимое файла сохранено, и так же в нём есть добавленная запись. Если содержимое отличается от ожидаемого, то необходимо восстановить сохраненную версию, и произвести все изменения ещё раз. После этого повторить проверку. Восстановить содержимое можно следующей командой:

```
sudo cp /etc/fstab.old /etc/fstab
```

Перезагрузить систему с помощью команды `reboot`. После загрузки зайти в каталог ~/media, требуется увидеть файл test.

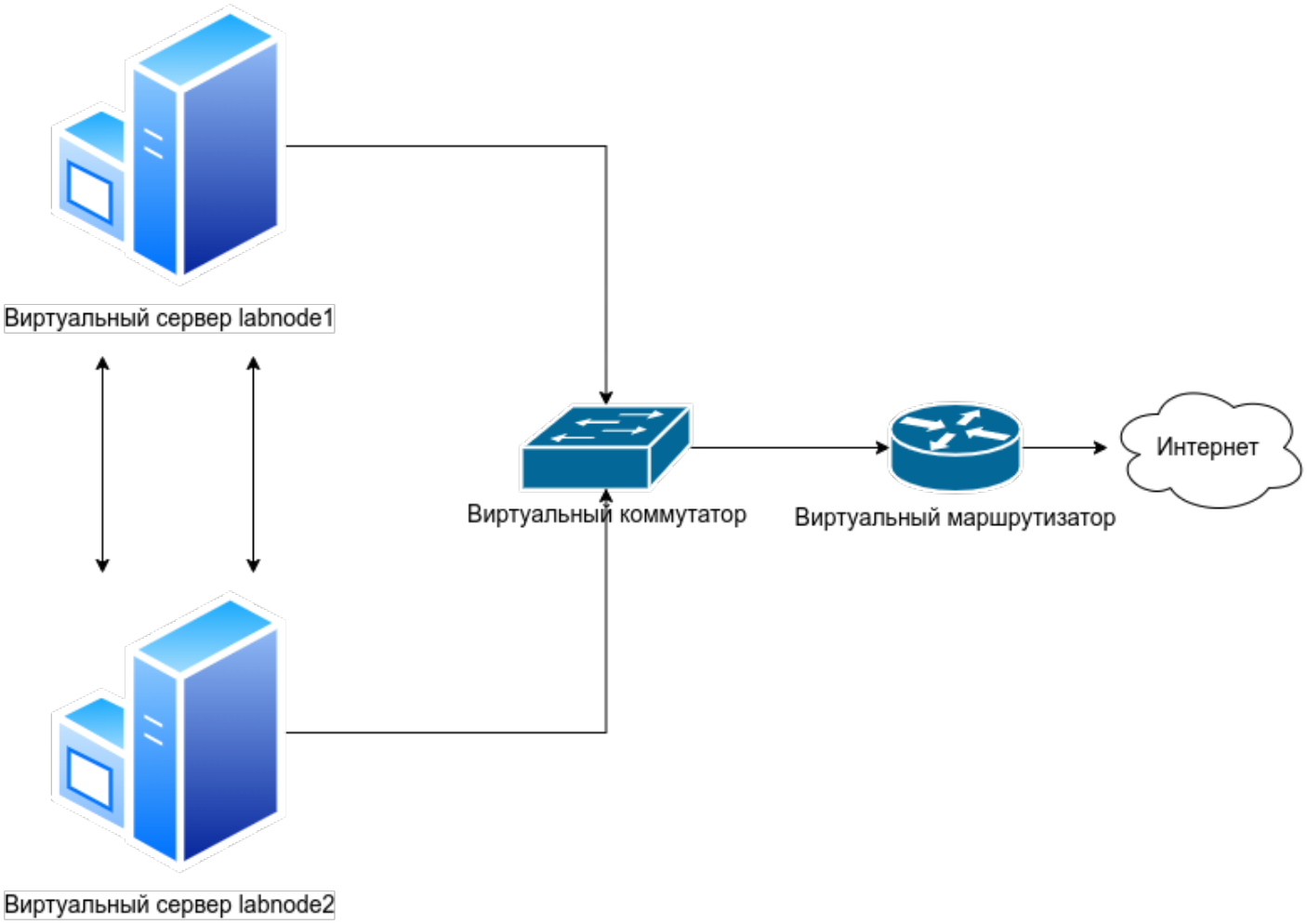
# Лабораторная работа 3. Работа с сетью в Linux. Виртуальный коммутатор Linux Bridge.

Цель
Получить представления о работе сетевой подсистемы в операционной системе Linux (CentOS 7), и научиться выполнять базовые действия с ней. Научиться работать с виртуальным коммутатором Linux Bridge.

Задачи

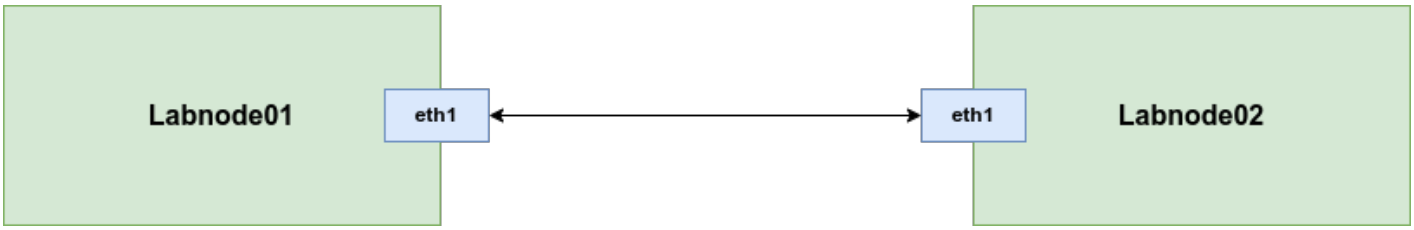
1. Задать статический IP адрес на интерфейс.
2. Настроить объединение интерфейсов.
3. Настроить сетевой мост.
4. Настроить VxLAN(виртуальные сети).
5. Ознакомиться с возможностями сетевых утилит.

Схема виртуального лабораторного стенда




# Задание 1. Установка статического IP адреса физическому интерфейсу

Переключиться на проект [GROUP]:[team]-lab:3. Подключиться к `labnode-1`. логин - `labuser`, пароль - `labpass1!` Подключение должно быть выполнено по следующей схеме (рис. 1).




Воспользоваться утилитой ip. Для того, чтобы увидеть существующие в системе интерфейсы, набрать команду:

```
ip address
```

Там же будут отображены основные параметры этих сетевых интерфейсов.
Команда ip позволяет использовать короткие имена команд. в данном случае, вместо `ip address` можно использовать команду

```
ip a
```

*В случае правильного выполнения команд (для всех команд кроме ip address) утилита `ip` не будет возвращать никакого значения. В случае, если команда выполнена неправильно, будет возвращена соответствующая ошибка.*

Задать интерфейсу `eth1` IP адрес:

```
sudo ip address add 10.0.12.20/24 dev eth1
```

Изменить состояние на `up`.

```
sudo ip link set up dev eth1
```

Посмотреть изменения (состояние устройства eth1 должно измениться на UP):

```
ip address
```

Подключиться к **labnode-2**. Установить интерфейсу **eth1** IP адрес:

```
sudo ip address add 10.0.12.30/24 dev eth1
```

Изменить состояние на `up`:

```
sudo ip link set up dev eth1
```

С **labnode-1** проверить доступность **labnode-2**:

```
ping -c 4 10.0.12.30
```

После перезагрузки сервера, или сервиса сети все изменения отменяются. Перезагрузить **оба** сервера, и посмотреть на состояние интерфейсов (необходимо проверить, сохранились ли на интерфейсе адреса, заданные предыдущими командами):

```
reboot
ip address
```

# Задание 2. Настройка статического адреса через конфигурационные файлы

Подключение должно быть выполнено по следующей схеме (рис. 1). Для того, чтобы изменения оставались в силе, нужно настроить интерфейс через конфигурационный файл.

Тогда настройки будут загружаться при старте системы. Создать конфигурацию интерфейса eth1 на **labnode-1**:

```
sudo vi /etc/sysconfig/network-scripts/ifcfg-eth1
```

И привести его к следующему виду:

```
TYPE=Ethernet
DEVICE=eth1
BOOTPROTO=static
IPADDR=10.0.12.20
PREFIX=24
ONBOOT=yes
```

Основные параметры:

1. TYPE - Тип сетевого интерфейса
2. NAME - Имя интерфейса
3. DEVICE - Устройство, которое интерфейс использует
4. BOOTPROTO - если этот параметр static, то интерфейс не будет автоматически получать адрес от сети, маску и другие параметры подключения. В случае необходимости автоматического получения адреса – необходимо указать значение этого параметра -dhcp.
5. ONBOOT - включать ли интерфейс при загрузке.
6. IPADDR - IP-адрес
7. DNS1 - DNS, через который обращаться к доменам. Можно указать несколько параметров: DNS1, DNS2...
8. PREFIX - префикс, другой способ задания маски сети. Для префикса 24 маска будет 255.255.255.0
9. GATEWAY - шлюз

*Все остальные параметры являются необязательными в данной лабораторной работе, и могут быть удалены.*

Скопировать файл **ifcfg-eth1** с именем **ifcfg-eth2**:

```
sudo cp /etc/sysconfig/network-scripts/ifcfg-eth1 /etc/sysconfig/network-scripts/ifcfg-eth2
```

Отредактировать его с помощью редактора **vi**:

```
sudo vi /etc/sysconfig/network-scripts/ifcfg-eth2
```

И привести его содержимое к следующему виду:

```
TYPE=Ethernet
DEVICE=eth2
BOOTPROTO=static
IPADDR=10.0.12.21
PREFIX=24
ONBOOT=yes
```

Перезагрузить сеть и посмотреть интерфейсы:

```
sudo systemctl restart network
ip address
```

Далее необходимо настроить интерфейсы на узле **labnode-2**, по такому же принципу, как и **labnode-1**. Создать конфигурацию интерфейса **eth1**:

```
sudo vi /etc/sysconfig/network-scripts/ifcfg-eth1
```

И привести его к следующему виду:

```
TYPE=Ethernet
DEVICE=eth1
BOOTPROTO=static
IPADDR=10.0.12.30
PREFIX=24
ONBOOT=yes
```

Скопировать файл **ifcfg-eth1** с именем **ifcfg-eth2**:

```
sudo cp /etc/sysconfig/network-scripts/ifcfg-eth1 /etc/sysconfig/network-scripts/ifcfg-eth2
```

В оболочке **bash** (и во многих других оболочках) для упрощения ввода текста выше можно использовать упрощенную форму ввода текста выше:

```
sudo cp /etc/sysconfig/network-scripts/ifcfg-eth{1,2}
```

Отредактировать его через vi:

```
sudo vi /etc/sysconfig/network-scripts/ifcfg-eth2
```

И привести к следующему виду:

```
TYPE=Ethernet
DEVICE=eth2
BOOTPROTO=static
IPADDR=10.0.12.31
PREFIX=24
ONBOOT=yes
```

Перезагрузить сеть и посмотреть интерфейсы:

```
sudo systemctl restart network
ip address
```

Проверить доступность интерфейсов:

```
(labnode-1) ping -c 4 10.0.12.30
(labnode-1) ping -c 4 10.0.12.31
(labnode-2) ping -c 4 10.0.12.20
(labnode-2) ping -c 4 10.0.12.21
```

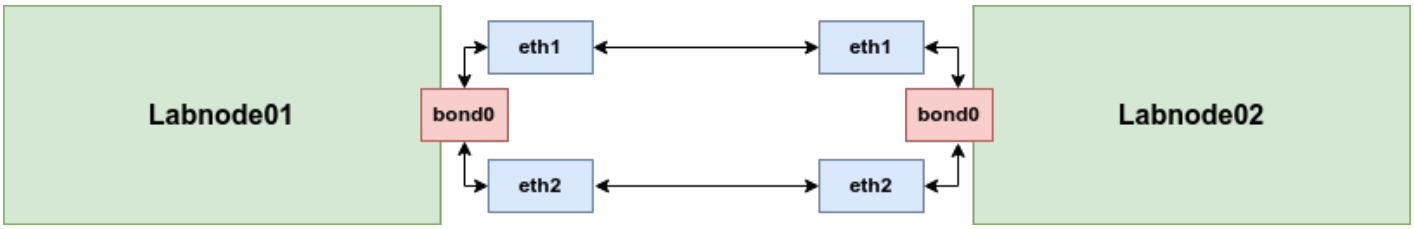
# Задание 3. Настройка объединения интерфейсов.

"Объединение" (bonding) сетевых интерфейсов - позволяет совокупно собрать несколько портов в одну группу, эффективно объединяя пропускную способность в одном направлении. Например, вы можете объединить два порта по 100 мегабит в 200 мегабитный магистральный порт.

<I>В некоторых случаях интерфейсы после перезапуска сетевой службы не удаляют ip адреса, что связано с особенностью работы up/down скриптов. В таком случае в системе на разных интерфейсах может присутствовать одинаковый ip адрес (проверить можно командой `ip address`). Для решения этой проблемы необходимо отчистить все адреса на интерфейсе. Сделать это можно как просто удалив конкретный адрес с интерфейса, так и воспользоваться специальной командой, которая очистит все имеющиеся на нём адреса:

```
sudo ip address flush dev eth1
sudo ip address flush dev eth2
```

Подключение должно быть выполнено по следующей схеме (рис. 2).



Для того, чтобы создать интерфейс **bond0**, нужно создать файл конфигурации:

```
sudo vi /etc/sysconfig/network-scripts/ifcfg-bond0
```

Конфигурация **bond0** интерфейса на **labnode-1** будет следующей:

```
TYPE=Bond
DEVICE=bond0
BOOTPROTO=static
IPADDR=10.0.12.20
PREFIX=24
BONDING_MASTER=yes
BONDING_OPTS="mode=0 miimon=100"
ONBOOT=yes
```

Это создаст сам **bond0** интерфейс. Но нужно также назначить физические интерфейсы **eth1** и **eth2**, как подчиненные ему. Необходимо изменить конфигурационный файл **eth1**:

```
sudo vi /etc/sysconfig/network-scripts/ifcfg-eth1
```

И привести его к следующему виду:

```
TYPE=Ethernet
DEVICE=eth1
MASTER=bond0
SLAVE=yes
```

То же самое сделать и с **eth2**:

```
sudo vi /etc/sysconfig/network-scripts/ifcfg-eth2
```

```
TYPE=Ethernet
DEVICE=eth2
MASTER=bond0
SLAVE=yes
```

Перезагрузить сеть.

```
sudo systemctl restart network
```

Посмотреть, что получилось:

```
ip address
```

В полученном выводе интерфейсы **eth1** и **eth2** должны быть в подчиненном режиме (SLAVE), а интерфейс **bond0** должен иметь ip адрес и находиться в состоянии **UP**. Также необходимо проверить, что на интерфейсах **eth1** и **eth2** нет никаких ip адресов. Теперь необходимо проделать тоже самое на labnode-2.

```
sudo vi /etc/sysconfig/network-scripts/ifcfg-bond0
```

Конфигурация **bond0** интерфейса будет следующей:

```
TYPE=Bond
DEVICE=bond0
BOOTPROTO=static
IPADDR=10.0.12.30
PREFIX=24
BONDING_MASTER=yes
BONDING_OPTS="mode=0 miimon=100"
ONBOOT=yes
```

Конфиг **eth1**:

```
TYPE=Ethernet
DEVICE=eth1
MASTER=bond0
SLAVE=yes
```

Конфиг **eth2**:

```
TYPE=Ethernet
DEVICE=eth2
MASTER=bond0
SLAVE=yes
```

Перечитать конфигурационные файлы сетевых устройств и проверить после этого настройки сетевых интерфейсов:

```
sudo systemctl restart network
```

С **labnode-1** необходимо проверить доступность **labnode-2**:

```
ping -c 4 10.0.12.30
```

Теперь на **labnode-1** необходимо отключить **eth1** и посмотреть его состояние:

```
sudo ip link set down eth1
ip address
```

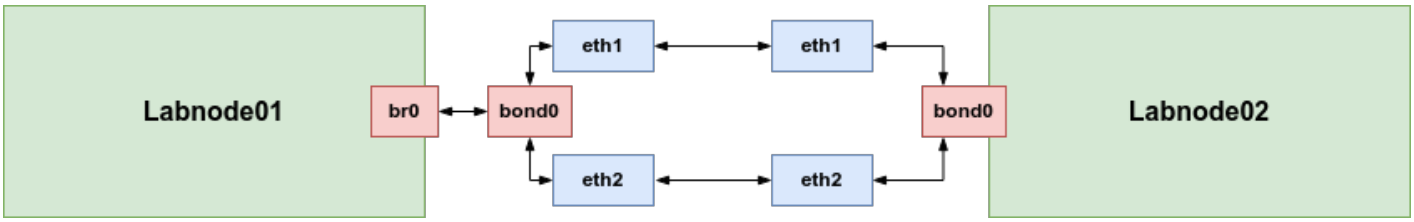
И с **labnode-2** проверить его доступность:

```
ping 10.0.12.20 -c 4
```

Если объединение интерфейсов настроено правильно, то узел будет доступен, даже после выключения одного из интерфейсов.

# Задание 4. Настройка bridge интерфейса.

Ядро Linux имеет встроенный механизм коммутации пакетов между интерфейсами, и может функционировать как обычный сетевой коммутатор. Интерфейс Bridge представляет собой как сам виртуальный сетевой коммутатор, так и сетевой интерфейс с ip адресом, назначенным на порт этого коммутатора. Подключение должно быть выполнено по следующей схеме (рис. 3).




На **labnode-1** требуется создать конфиг **ifcfg-br0**:

```
sudo vi /etc/sysconfig/network-scripts/ifcfg-br0
```

Мост будет иметь следующую конфигурацию:

```
TYPE=Bridge
DEVICE=br0
BOOTPROTO=static
IPADDR=10.0.12.20
PREFIX=24
STP=on
ONBOOT=yes
```

Spanning Tree Protocol (STP) нужен, чтобы избежать петель коммутации.

Интерфейс **bond0**, настроенный до этого, может быть интерфейсом этого сетевого коммутатора, но в таком случае ip адрес уже будет назначен на интерфейс виртуального сетевого коммутатора, и все настройки ip с интерфейса **bond0** можно будет убрать. Для

этого в конфигурационный файл интерфейса bond0 также нужно добавить параметр BRIDGE=br0. Также удалить из него параметры `BOOTPROTO, IPADDR, PREFIX, ONBOOT` (можно просто закомментировать с помощью символа #, когда пригодятся, раскомментировать их, убрав символ #):

```
sudo vi /etc/sysconfig/network-scripts/ifcfg-bond0
```

Перезагрузить сеть:

```
sudo systemctl restart network
```

Можно проверить результат. Для этого на **labnode-2** выполнить следующую команду:

```
ip address
```

Проверить, что в результате вывода этой команды ip адрес будет назначен только на интерфейс br0, и он будет в состоянии UP. Если все правильно, то проверить доступность соседнего узла командой:

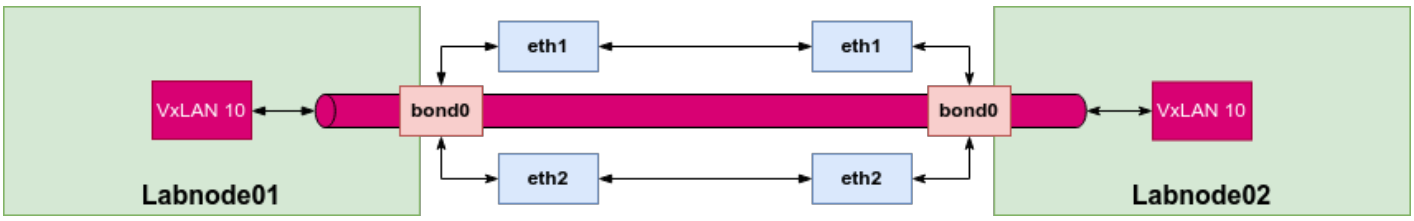
```
ping -c 4 10.0.12.20
```

*Мост может подняться не сразу. Если что, необходимо подождать.*

# Задание 5. Создание VxLAN интерфейсов.

VxLAN является механизмом построения виртуальных сетей на основаниях тоннелей, поверх реальных сетей, но при этом позволяющим их разграничивать.

Подключение должно быть выполнено по следующей схеме (рис. 4).




На **labnode-1** удалить конфигурацию моста **br0**:

```
sudo rm /etc/sysconfig/network-scripts/ifcfg-br0
```

И привести **bond0** к прежнему виду:

```
TYPE=Bond
DEVICE=bond0
BOOTPROTO=static
```

```
IPADDR=10.0.12.20
PREFIX=24
BRIDGE=br0
BONDING_MASTER=yes
BONDING_OPTS="mode=0 miimon=100"
ONBOOT=yes
```

И перезагрузить сервер:

```
sudo reboot
```

После загрузки сервера проверить работу сети с узла labnode-2:

```
ping -c 4 10.0.12.20
```

На **labnode-1** добавить интерфейс **vxlan10**:

```
sudo ip link add vxlan10 type vxlan id 10 dstport 0 dev bond0
```

Настроить коммутацию Linux Bridge:

```
  sudo bridge fdb append to 00:00:00:00:00:00 dst 10.0.12.30 dev vxlan10
```

Назначить vxlan10 IP адрес и перевести его в состояние up:

```
sudo ip addr add 192.168.1.20/24 dev vxlan10
sudo ip link set up dev vxlan10
```

VxLAN работает как приложение. Пакеты инкапсулируются в udp, и для работы VxLAN требуется udp порт 8472. Открыть его в фаерволе:

```
sudo firewall-cmd --permanent --add-port=8472/udp
sudo firewall-cmd --reload
```

После нужно сделать все то-же самое на *labnode-2*. Добавить интерфейс vxlan10:

```
sudo ip link add vxlan10 type vxlan id 10 dstport 0 dev bond0
```

Настроить коммутацию Linux Bridge:

```
sudo bridge fdb append to 00:00:00:00:00:00 dst 10.0.12.20 dev vxlan10
```

Назначить vxlan10 IP адрес и перевести его в состояние up:

```
sudo ip addr add 192.168.1.30/24 dev vxlan10
sudo ip link set up dev vxlan10
```

Открыть порт 8472/udp в фаерволе:

```
sudo firewall-cmd --permanent --add-port=8472/udp
sudo firewall-cmd --reload
```

Протестировать соединение через vxlan. Для этого на **labnode-1**:

```
ping -c 4 192.168.1.30
```

Посмотреть на arp таблицу. Там можно увидеть соответствие mac адресов с ip адресами.

```
sudo arp
```

Нужно убедиться, что появилось приложение, которое слушает порт 8472.

```
ss -tulpn | grep 8472
```

Теперь необходимо добавить vxlan с другим тегом (20), и убедиться в том, что из него не будет доступа к vxlan с тегом 10 (пакеты будут отбрасываться из-за разных тегов). Перезагрузить **labnode-2**. Текущая настройка vxlan сбросится.

```
sudo reboot
```

На **labnode-2** добавить интерфейс vxlan20 и настроить его:

```
sudo ip link add vxlan20 type vxlan id 20 dstport 0 dev bond0
sudo bridge fdb append to 00:00:00:00:00:00 dst 10.0.12.20 dev vxlan20
sudo ip addr add 192.168.1.30/24 dev vxlan20
sudo ip link set up dev vxlan20
ss -tulpn | grep 8472
```

Протестировать соединение через vxlan. Для этого на **labnode-1**:

```
ping 192.168.1.30 -c 4
```

# Лабораторная работа 4. Основы виртуализации в Linux. QEMU/KVM.

## Цель

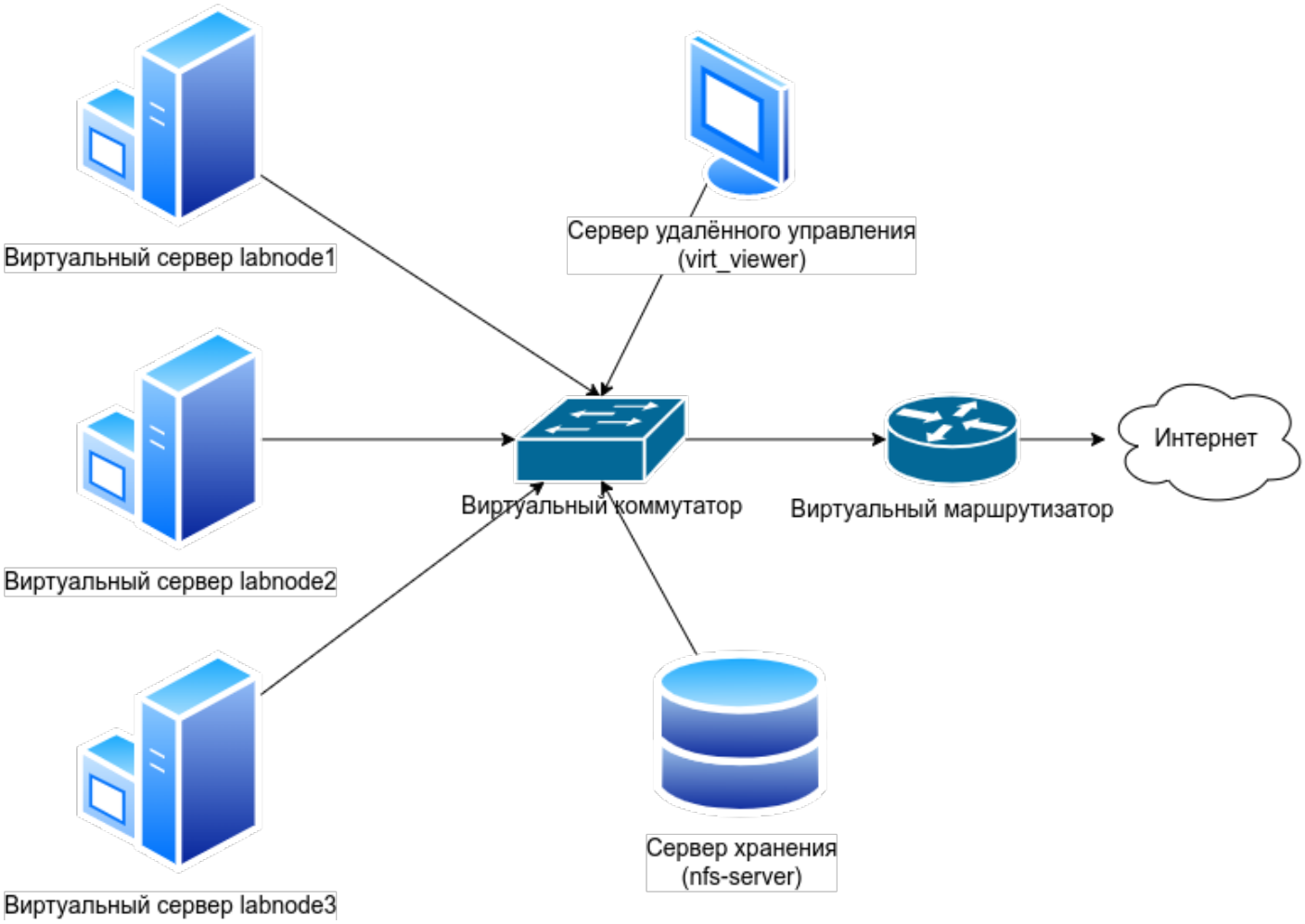
Получить понимание принципа работы программ, используемых для виртуализации в операционной системе CentOS 7. Научиться работать с qemu.

## Задачи

1. Установить qemu.
2. С использованием qemu-img провести базовые операции над образами.
3. Скачать образ Cirros, запустить виртуальную машину и установить операционную систему на диск.

**Note: Авторизация на всех узлах**
**Логин:** labuser
**Пароль:** labpass1!

Схема виртуального лабораторного стенда




# Задание 1. Установка QEMU.

Переключиться на проект [GROUP]:[team].lab:4-7. Включить **labnode-1** и **virt_viewer**.

На **labnode-1**:

1. Установить эмулятор аппаратного обеспечения различных платформ:

```
sudo yum install -y qemu-kvm
```

2. Убедиться, что модуль KVM загружен (с помощью команд lsmod и grep):

```
lsmod | grep -i kvm
```

```
kvm                   636921  1 kvm_intel
```

# Задание 2. Управление образами дисков при помощи qemu-img.

Чтобы запускать виртуальные машины, QEMU требуются образы для хранения определенной файловой системы данной гостевой ОС. Такой образ сам по себе имеет тип некоторого файла, и он представляет всю гостевую файловую систему, расположенную в некотором виртуальном диске. QEMU поддерживает различные образы и предоставляет инструменты для создания и управления ими. Можно построить пустой образ диска с помощью утилиты qemu-img, которая должна быть установлена.

1. Проверить какие типы образов поддерживаются qemu-img:

```
sudo qemu-img -h | grep Supported
```

2. Создать образ qcow2 с названием system.qcow2 и размером 5 ГБ:

```
sudo qemu-img create -f qcow2 system.qcow2 5G
```

3. Проверить что файл был создан:

```
ls -lah system.qcow2
```

4. Посмотреть дополнительную информацию о данном образе:

```
sudo qemu-img info system.qcow2
```

# Задание 3. Изменение размера образа.

Не все типы образов поддерживают изменение размера. Чтобы изменить размер такого образа необходимо преобразовать его вначале в образ raw при помощи команды преобразования qemu- img.

1. Конвертировать образ диска из формата qcow2 в raw:

```
sudo qemu-img convert -f qcow2 -O raw system.qcow2 system.raw
```

2. Добавить дополнительно 5 ГБ к образу:

```
sudo qemu-img resize -f raw system.raw +5GB
```

3. Проверить новый текущий размер образа:

```
sudo qemu-img info system.raw
```

4. Конвертировать образ диска обратно из raw в qcow2:

```
sudo qemu-img convert -f raw -O qcow2 system.raw system.qcow2
```

# Задание 4. Загрузка образа Cirros.

Для загрузки образов с общедоступных репозиториев требуется утилита **s3cmd**. Загрузить необходимый образ, воспользовавшись **s3cmd**:

```
sudo yum install s3cmd
cd ~
sudo cp /var/lib/cloud/s3cfg .s3cfg
s3cmd -f get s3://lab3/cirros.img /tmp/
```

# Задание 5. Создание виртуального окружения с помощью qemu-system.

1. Для того, чтобы подключиться к виртуальной машине по протоколу удаленного рабочего стола **Spice**, нужно открыть порт **5900**.

```
sudo firewall-cmd --permanent --add-port=5900-5930/tcp
sudo firewall-cmd --reload
```

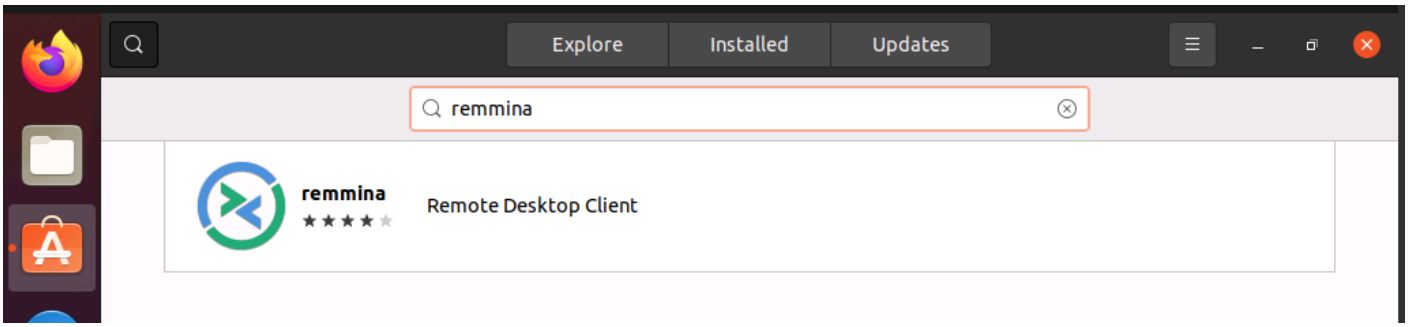
2. Посмотреть ip адрес вашего сервера

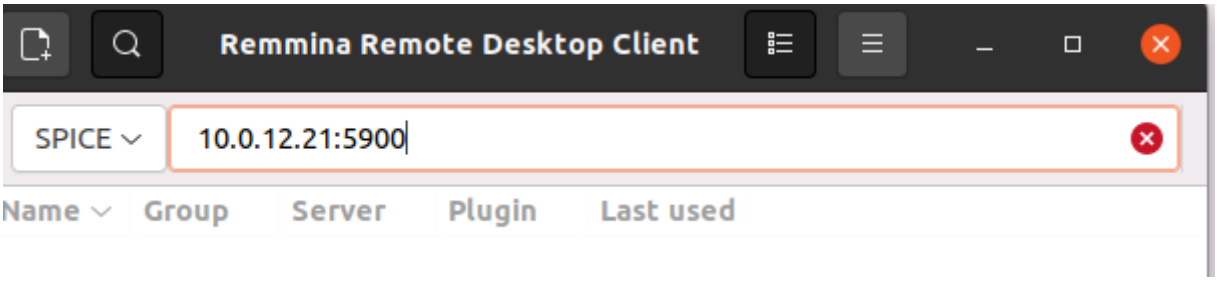
```
ip a
```

3. Запустить систему при помощи qemu-system:

```
sudo /usr/libexec/qemu-kvm -hda /tmp/cirros.img \
-m 1024 -vga qxl -spice port=5900,disable-ticketing
```

4. Открыть консоль виртуальной машины virt-viewer. Данная вирутальная машина имеет графическую оболочку, и позволяет взаимодействовать с графическими приложениям, создавая рабочее окружение клиента. Подключиться из виртуальной машины virt_viewer (Пользователь - **labuser**, пароль -**labpass1!**) к виртуальной машине. Для этого скачать программу remmina. Через менеджер Ubuntu Software (в списке слева) установить утилиту **remmina**, введя в поисковой строке её название, и нажав **install**.




Установить и открыть программу (название - **Remmina**). Для открытия программы **Remmina** – открыть меню приложений в левом нижнем углу, и из открывшегося списка приложений выбрать - **Remmina**. Подключаться по адресу `spice://10.0.12.21:5900`

5. Залогиниться в Cirros. Дефолтные логин и пароль написаны в консоли (log - **cirros**/ pass- **gocubsgo**) ОС. Набрать команду **uname -a**. Посмотреть на версию ядра ОС. Выключить виртуальную машину, набрав

```
sudo poweroff
```

# Лабораторная работа 5. Основы виртуализации в Linux. Libvirt.

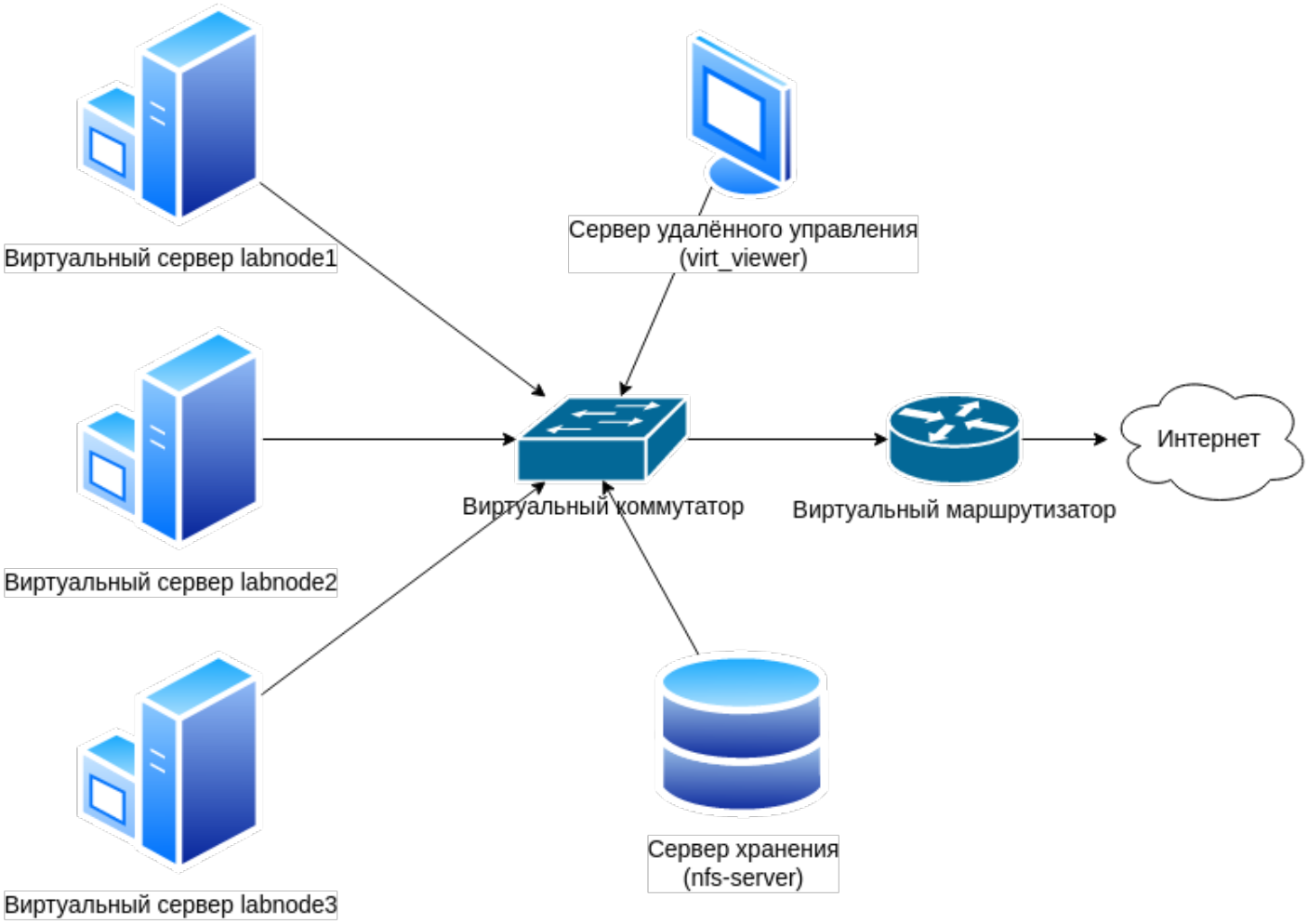
Цель
Научиться работать с Libvirt и Virsh

Задачи

1. Установить Libvirt.
2. Настроить сетевой мост.
3. Создать виртуальную машину.
4. Провести базовые операции с виртуальной машиной.

**Note: Авторизация на всех узлах**
**Логин:** labuser
**Пароль:** labpass1!

Схема виртуального лабораторного стенда




# Задание 1. Установка Libvirt и Virsh.

Необходимо установить несколько пакетов для виртуализации, которые не входят в базовую комплектацию системы. В проекте [GROUP]:[team]-lab:4-7, на labnode-1 нужно выполнить следующую команду:

```
sudo yum install -y libvirt virt-install
```

# Задание 2. Настройка моста.

Установить пакет **bridge-utils**:

```
sudo yum install -y bridge-utils
```

Вывести на экран имеющиеся интерфейсы:

```
ip -c address
```

Открыть файл **/etc/sysconfig/network-scripts/ifcfg-br0**:

```
sudo vi /etc/sysconfig/network-scripts/ifcfg-br0
```

И добавить в него следующее содержимое:

```
TYPE=Bridge
DEVICE=br0
BOOTPROTO=static
ONBOOT=yes
IPADDR=10.0.12.21
PREFIX=24
GATEWAY=10.0.12.1
```

А в файл **/etc/sysconfig/network-scripts/ifcfg-eth0** добавить параметр `BRIDGE`, убрать `BOOTPROTO` и `ONBOOT`, `GATEWAY`, `IPADDR`, `NETMASK`:

```
DEVICE=eth0
BRIDGE=br0
USERCTL=no
```

Перезагрузить сервер:

```
sudo reboot
```

# Задание 3. Создание виртуальной машины.

Переместить образ **cirros** в **/var/lib/libvirt/images/**

```
sudo mv /tmp/cirros.img /var/lib/libvirt/images/
```

Следующая команда создаст новую KVM виртуальную машину

```
sudo virt-install --name cirros \
--ram 1024 \
--disk path=/var/lib/libvirt/images/cirros.img,cache=none \
--boot hd \
--vcpus 1 \
--network bridge:br0 \
--graphics spice,listen=0.0.0.0 \
--wait 0
```

*Символ \ - обратная косая черта используется для экранирования специальных символов в строковых и символьных литералах. В данном случае нужна, чтобы переместить каретку на*

*новую строку, для наглядности. После ее добавления в команду можно нажать Enter, но строка не отправится на выполнение, а ввод команды продолжится.* При ошибке в наборе команды, можно не набирать ее заново, а нажать стрелку вверх, исправить ее, и снова нажать `Enter` Подробнее о параметрах:

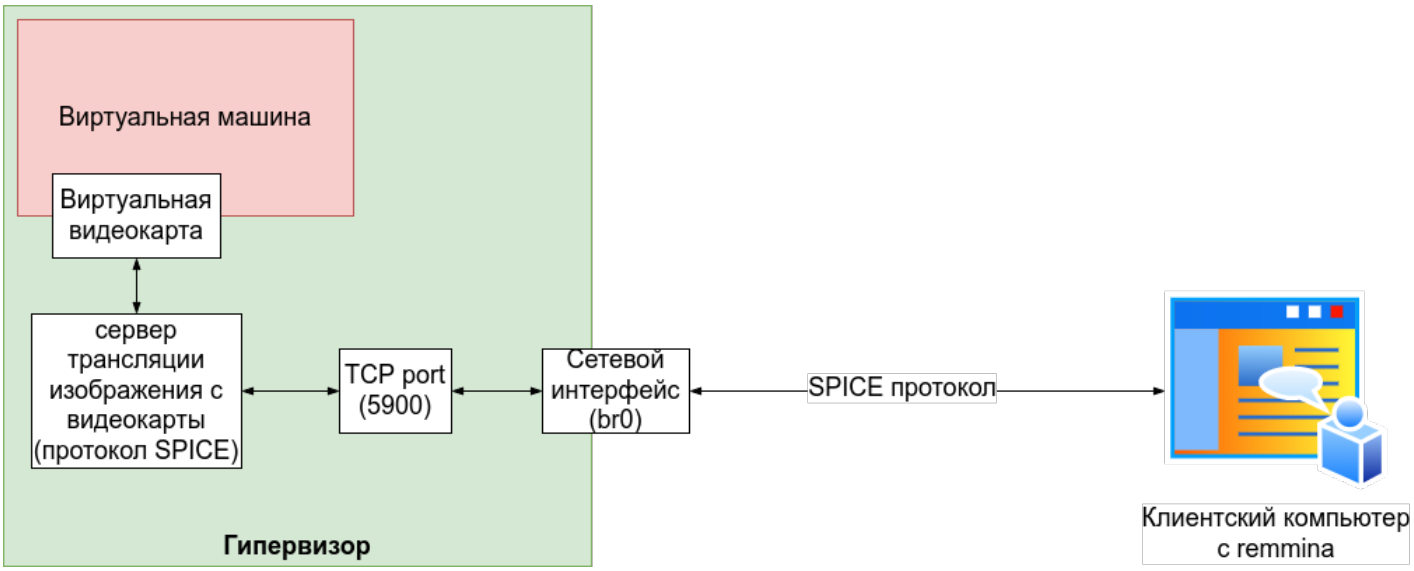
| Название парметра | Описание параметра |
|---|---|
| name | Имя виртуальной машины, которое будет отображаться в virsh |
| ram | Размер оперативной памяти в МБ |
| disk | Диск, который будет создан и подключен к виртуальной машине |
| vcpus | Количество виртуальных процессоров, которые нужно будет настроить для гостя |
| os-type | Тип операционной системы |
| os-variant | Название операционной системы |
| network | Определение сетевого интерфейса, который будет подключен к виртуальной машине |
| graphics | Определяет графическую конфигурацию дисплея. |
| cdrom | CD ROM устройство |

Далее необходимо подключиться из виртуальной машине virt_viewer (Пользователь - `labuser`, пароль - `labpass1!`) к виртуальной машине через программу **Remmina**.

Открыть её (название - Remmina). Подключиться по адресу **10.0.12.21:5900**, выбрав протокол SPICE Вернуться в консоль **labnode-1**. Проверить состояние гостевой системы, используя команду (Если в консоли написано “Domain installation still in progress”, то нажмите `^C`):

```
sudo virsh list --all
```

# Задание 4. Операции с виртуальной машиной.




Рассмотрим работу утилиты **virsh**. Чтобы подключиться к ВМ по протоколу удаленного доступа, используется следующая команда:

```
sudo virsh domdisplay cirros
```

Результатом исполнения этой команды будет адрес для подключения к графическому интерфейсу ВМ, с указанием номера порта. Получить информацию о конкретной ВМ можно так:

```
sudo virsh dominfo cirros
```

В результате чего будет выведена информация, об основных параметрах виртуальной машины. Выключить/включить ВМ можно с помощью команды:

```
sudo virsh destroy cirros
sudo virsh start cirros
```

Добавление ВМ в автозапуск происходит следующим образом:

```
sudo virsh autostart cirros
```

Теперь, виртуальная машина будет автоматически запускаться, после перезагрузки сервера. Кроме того, может потребоваться отредактировать XML конфигурацию ВМ:

```
sudo virsh edit cirros
```

* *Чтобы выйти из редактора без сохранения - :q!*

Необходимо выгрузить конфигурацию ВМ в XML в файл, используя команду:

```
sudo virsh dumpxml cirros | tee cirros.xml
```

Необходимо удалить ВМ, и убедиться, что её нет в списке виртуальных машин:

```
sudo virsh undefine cirros
sudo virsh destroy cirros
sudo virsh list --all
```

Для создания ВМ из XML существует следующая команда:

```
sudo virsh define cirros.xml
sudo virsh list --all
```

# Лабораторная работа 6. Основы виртуализации в Linux. Отказоустойчивый кластер на базе Corosync/Pacemaker.

Цель

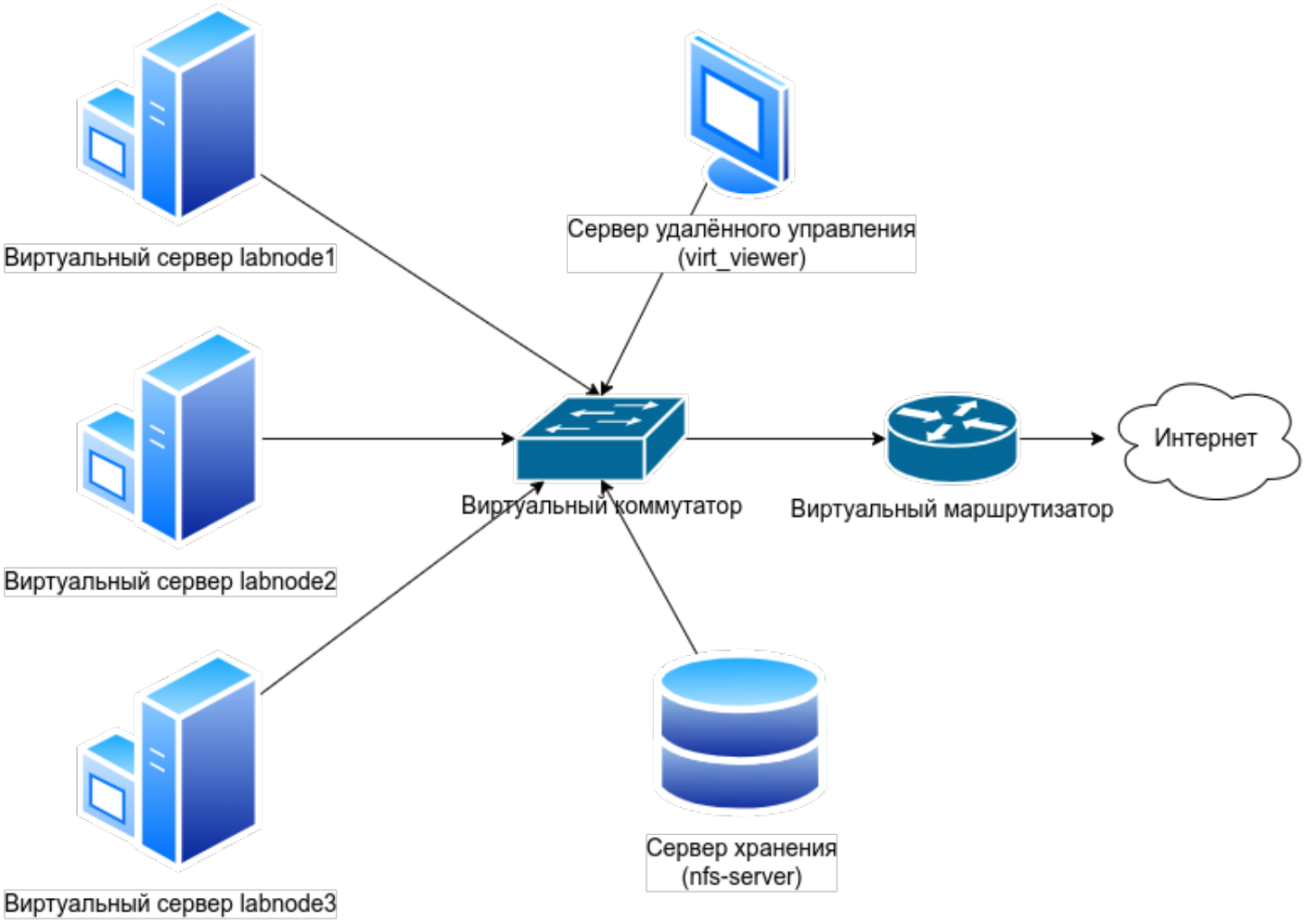
Получить базовые навыки в работе с пакетом управления виртуализацией Libvirt.

Задачи

1. Настроить nfs клиент.
2. Установить и настроить Corosync/Pacemaker.
3. Подготовить XML ВМ.
4. Создать ресурс.

**Note: Логин/пароль на всех узлах**
**Логин:** labuser
**Пароль:** labpass1!

Схема виртуального лабораторного стенда




# Задание 1. Настройка nfs клиента

В проекте [GROUP]:[team]-lab:4-7, на labnode-1, labnode-2 и labnode-3 нужно зайти в файл **/etc/fstab**:

```
sudo vi /etc/fstab
```

И раскомментировать следующую строку (уберите символ # в начале строки):

```
10.0.12.18:/home/nfs/ /media/nfs_share/ nfs rw,sync,hard,intr 0 0
```

На всех трёх машинах установить пакет для работы с nfs и перемонтировать разделы, используя комманды:

```
sudo yum install -y nfs-utils
sudo mkdir /media/nfs_share
sudo mount -a
```

# Задание 2. Установка Pacemaker и Corosync

Установка очень проста. На **всех** узлах нужно выполнить команду:

```
sudo yum install -y pacemaker corosync pcs resource-agents qemu-kvm libvirt virt-install
```

Далее поднять pcs. Тоже, на всех узлах:

```
sudo systemctl start pcsd
sudo systemctl enable pcsd
```

Открыть порты, необходимые для работы кластера (на всех узлах):

```
sudo firewall-cmd --permanent --add-port=5900-5930/tcp
sudo firewall-cmd --permanent --add-port=49152-49216/tcp
sudo firewall-cmd --permanent --add-service={high-availability,libvirt,libvirt-tls}
sudo firewall-cmd --reload
```

Для обращения к узлам по имени, а не по адресу удобнее прописать на ***всех узлах*** сопоставление ip адреса и его имени. В таком случае, для сетевого взаимодействия между узлами можно будет обращаться по его имени. Для того чтобы прописать это соответствие, необходимо открыть файл ***/etc/hosts***:

```
sudo vi /etc/hosts
```

Прописать в нем следующее:

```
10.0.12.21 labnode-1 labnode-1.novalocal
10.0.12.22 labnode-2 labnode-2.novalocal
10.0.12.23 labnode-3 labnode-3.novalocal
```

Задайте пользователю **hacluster** пароль на всех узлах(сам пользователь был автоматически создан в процессе установки pacemaker).

```
echo password | sudo passwd --stdin hacluster
```

И, с помощью pcs создать кластер (на одном из узлов):

```
sudo pcs cluster auth labnode-1 labnode-2 labnode-3 -u hacluster -p password --force
sudo pcs cluster setup --force --name labcluster labnode-1 labnode-2 labnode-3
sudo pcs cluster start --all
```

Отключить fencing (в рамках работы он не рассматривается)

```
sudo pcs property set stonith-enabled=false
```

Включить автозапуск сервисов на всех трех машинах:

```
sudo systemctl enable pacemaker corosync --now
sudo systemctl status pacemaker corosync
```

Просмотреть информацию о кластере и кворуме:

```
sudo pcs status
sudo corosync-quorumtool
```

# Задание 3. Настройка моста.

На **labnode-1** уже создан мост. Сделать то же на **labnode-2** и **labnode-3**. Открыть файл:

```
sudo vi /etc/sysconfig/network-scripts/ifcfg-br0
```

И добавить в него следующее содержимое для labnode-2:

```
TYPE=Bridge
DEVICE=br0
BOOTPROTO=static
ONBOOT=yes
IPADDR=10.0.12.22
PREFIX=24
GATEWAY=10.0.12.1
```

И добавить в него следующее содержимое для labnode-3:

```
TYPE=Bridge
DEVICE=br0
BOOTPROTO=static
ONBOOT=yes
IPADDR=10.0.12.23
PREFIX=24
GATEWAY=10.0.12.1
```

А в файл **/etc/sysconfig/network-scripts/ifcfg-eth0** добавить параметр **BRIDGE** и убрать **BOOTPROTO** и **ONBOOT**:

```
DEVICE=eth0
#HWADDR=”оставить как было”
#ONBOOT=yes
TYPE=Ethernet
NAME=eth0
USERCTL=no
BRIDGE=br0
```

Перезагрузить сеть:

```
sudo systemctl restart network
```

# Задание 4. Создание ресурса

Для начала, нужно отключить Selinux. Требуется зайти в файл **/etc/selinux/config**:

```
sudo vi /etc/selinux/config
```

И заменить значение параметра SELINUX с enforcing на permissive:

```
SELINUX = permissive
```

После чего перезагрузить ВМ:

```
sudo reboot
```

Сделать это нужно на **labnode-1, labnode-2 и labnode-3**.

В предыдущих заданиях был сделан дамп (копия) конфигурации виртуальной машины **cirros** на **labnode-1**.

```
ls -lah /home/labuser/cirros.xml
```

Необходимо зайти в него через vi:

```
sudo vi cirros.xml
```

И изменить в разделе  путь до диска с /var/lib/libvirt/images/ на /media/nfs_share/, удалив одну строку, и заменив её другой.

```
<source file='/var/lib/libvirt/images/cirros.img'/> ### эту строчку необходимо удалить
```

```
<source file='/media/nfs_share/cirros.img'/>
```

Можно воспользоваться поиском по файлу, набрав /, а затем то, что необходимо найти. Искать нужно <disk. То есть, набрать /<disk. Скопировать cirros.xml с **labnode-1** на **labnode-2** и **labnode-3**:

```
scp cirros.xml labnode-2:~
scp cirros.xml labnode-3:~
```

На labnode-1, labnode-2 и labnode-3 также переместить файл в /etc/pacemaker/

```
sudo mv cirros.xml /etc/pacemaker/
sudo chown hacluster:haclient /etc/pacemaker/cirros.xml
```

Теперь добавить сам ресурс:

```
sudo pcs resource create cirros VirtualDomain \
config="/etc/pacemaker/cirros.xml" \
migration_transport=tcp meta allow-migrate=true
```

Просмотреть список добавленных ресурсов

```
sudo pcs status
sudo pcs resource show cirros
```

Проверить список виртуальных машин на узле, на котором запустился ресурс:

```
sudo virsh list --all
```

Проверить, что ресурс успешно запустился. Для этого из virt_viewer (Пользователь - `Admin`, пароль - `labpass1!`) подключиться к нему через программу Reminna. Подключаться по адресу spice://[address]:5900. *[address] - это IP адрес узла, на котором находится ресурс. Узнать его можно, набрав в консоли соответствующего узла ip -c a.*

# Лабораторная работа 7. Основы виртуализации в Linux. Динамическая миграция ресурсов в отказоустойчивом кластере на базе Corosync/Pacemaker.

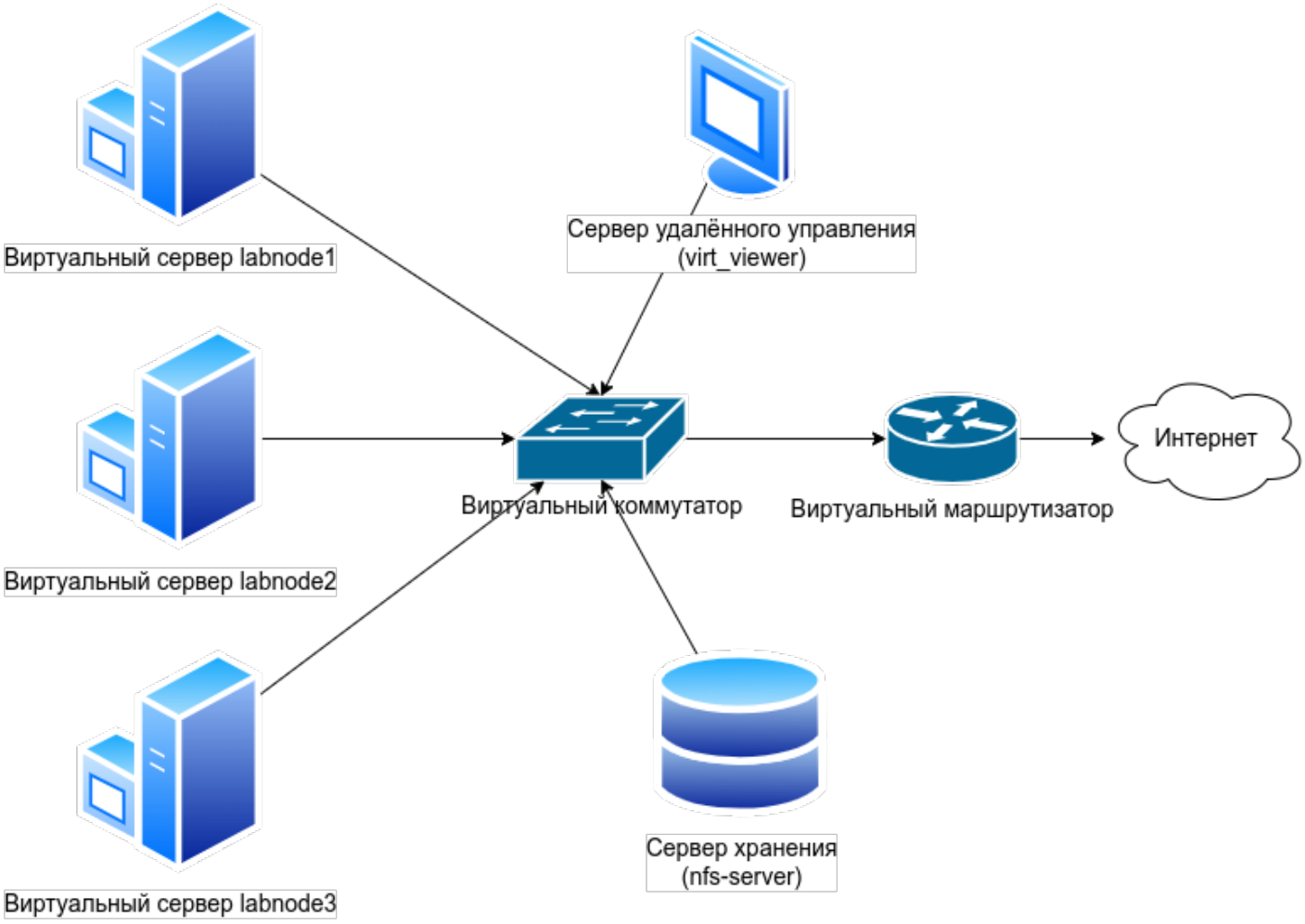
Цель
Получить базовые навыки в работе с пакетом управления виртуализацией Libvirt.

Задачи

1. Настроить динамическую миграцию.
2. Провести миграцию ресурса.

**Note: Логин/пароль на всех узлах**
**Логин:** labuser
**Пароль:** labpass1!

Проект: [GROUP]:[team]-lab:4-7 Схема виртуального лабораторного стенда




# Задание 1. Настройка динамической миграции

Порты в фаерволе уже открыты, после этого настроить **libvirt**. Необходимо перейти в файл **/etc/libvirt/libvirtd.conf**

```
sudo vi /etc/libvirt/libvirtd.conf
```

Добавить туда три параметра:

```
listen_tls = 0
listen_tcp = 1
auth_tcp = "none"
```

Сохранить файл. После этого необходимо изменить файл **/etc/sysconfig/libvirtd**

```
sudo vi /etc/sysconfig/libvirtd
```

Добавить параметр:

```
LIBVIRTD_ARGS="--listen --config /etc/libvirt/libvirtd.conf"
```

Перезагрузить **libvirt**.

```
sudo systemctl restart libvirtd
```

Проделать эти операции на всех узлах.

# Задание 2. Миграция ресурса

Нужно переместить ресурс на **labnode-2**:

```
sudo pcs resource move cirros labnode-2
```

На labnode-2 посмотреть статус кластера, и проверить список запущенных гостевых машин можно следующими командами:

```
sudo pcs status
sudo virsh list --all
```

Команда move добавляет ресурсу правило, заставляющее его запускаться только на указанном узле. Для того, чтобы очистить все добавленные ограничения - clear:

```
sudo pcs resource clear cirros
```

Из Remote Viewer необходимо проверить доступность ВМ на labnode-2. ( **spice://10.0.12.22:5900**). Необходимо дождаться загрузки cirros. Переместить ресурс на labnode-1:

```
sudo pcs resource move cirros labnode-1
```

Посмотреть на результат:

```
sudo pcs status
sudo virsh list --all
```

При подключении к ресурсу, используя remmina, можно увидеть, что гостевая ОС не загружается с нуля, а уже включена. Ресурс был полностью перенесен на другой узел (включая оперативную память), а не просто отключён на первом и включен на втором.