ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ

Федеральное государственное образовательное бюджетное учреждение высшего профессионального образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ им. проф. М. А. БОНЧ-БРУЕВИЧА»

НОЦ “Программно-определяемые системы”

Швидкий А.А., Власов Д.В., Васюткин А.В.

Основы построения гиперконвергентных систем

Лабораторный практикум

Санкт-Петербург
2020

Содержание

Лабораторная работа 1.
Работа с оболочкой в CentOS.

Цель

Получение базовых навыков при работе с оболочкой в операционной системе CentOS 7.

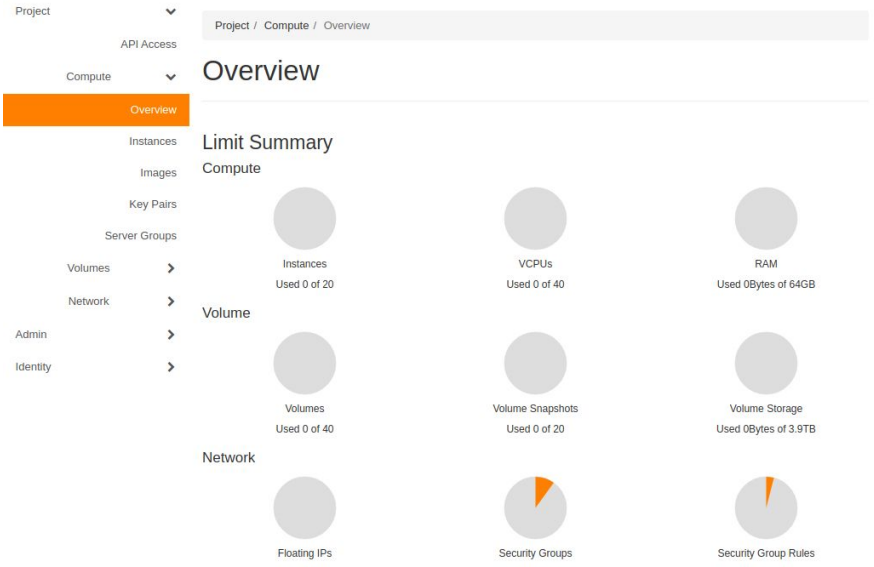
Задачи

1) Подключиться к виртуальной машине в RECSDS 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.

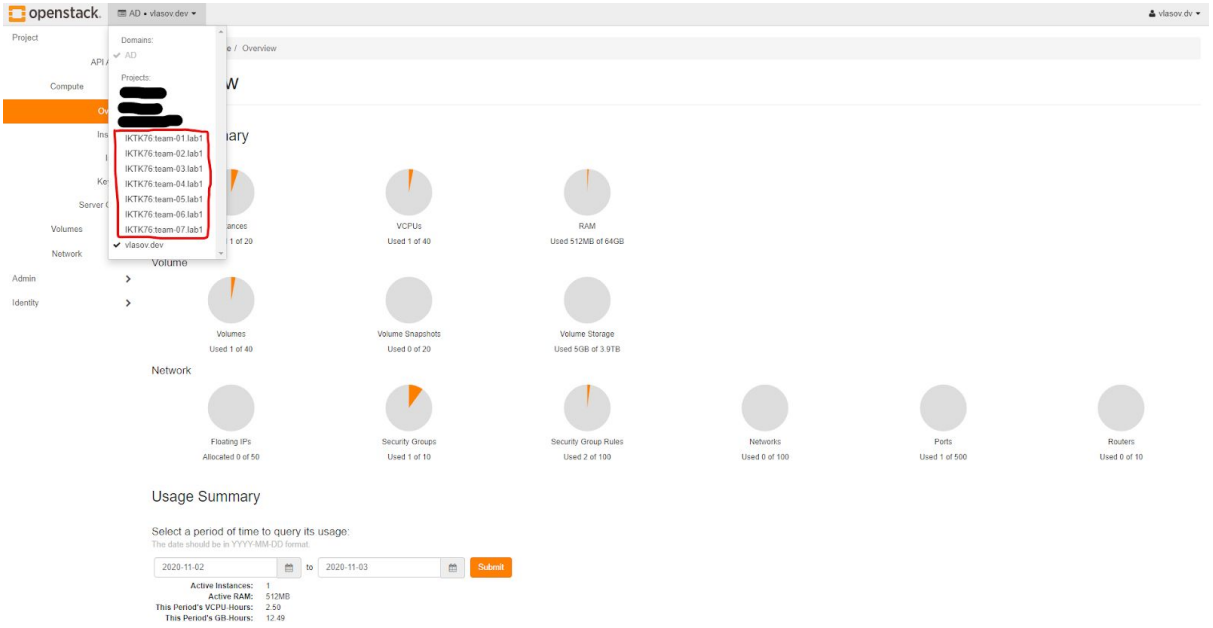
Задание 1. Подключение к RECSDS Dashboard и его использование.

Получить доступ к Dashboard можно из браузера. Для этого перейдите по адресу: `http://cloud.resds.ru/`.

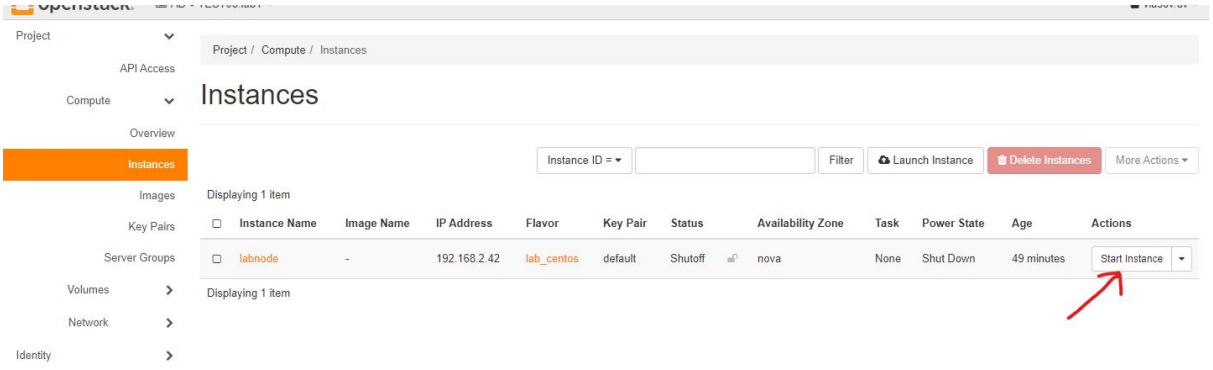
Выберете AD в поле Domain, и введите свой User Name и Password.
У вас появится общая информация по вашему проекту.



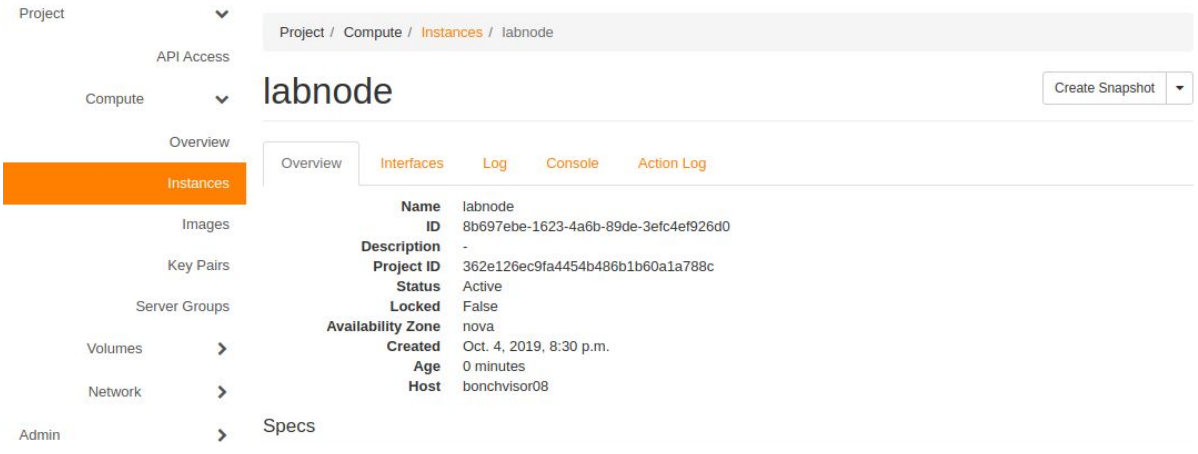
Выбрать нужный проект можно в верхней панели. Выберите **[GROUP]:[team].lab1**.



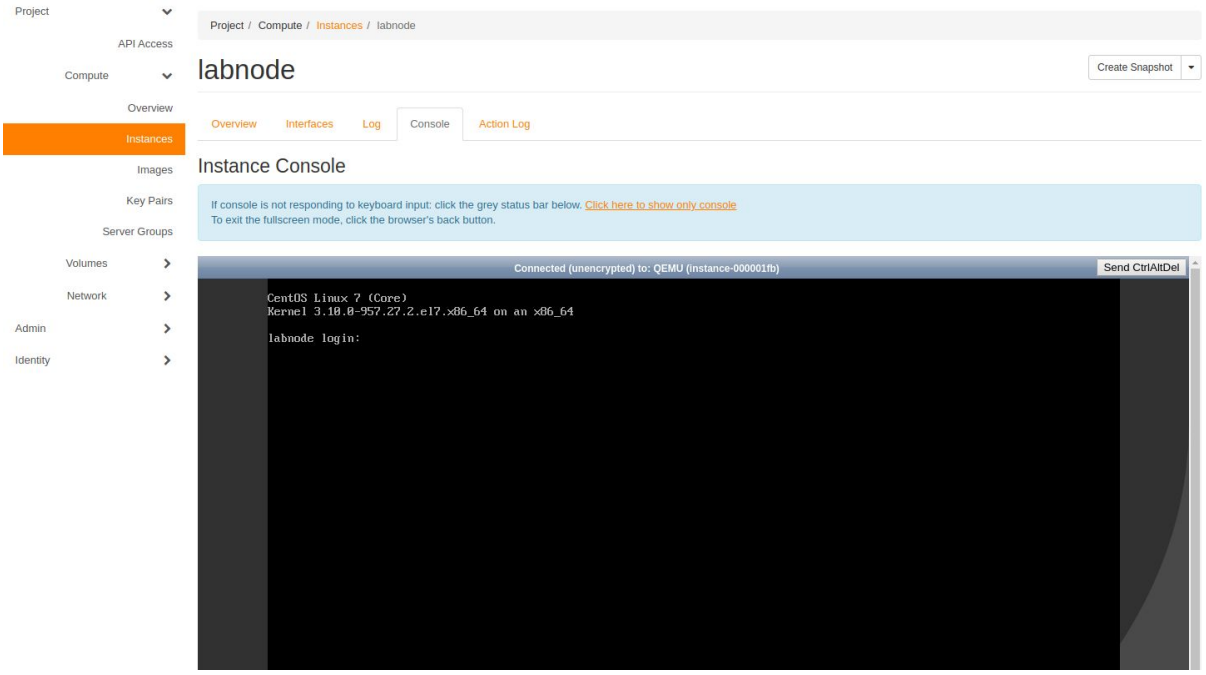
Перейдите во вкладку Instances. Включите инстанс, если выключен.



Вы должны увидеть здесь ваши виртуальные машины. В данной работе нам понадобится только одна. Нажмите на имя виртуальной машины. У вас должна открыться более подробная информация о ней.



Для того, чтобы получить доступ к ВМ, и перейдите во вкладку Console. Здесь вы должны увидеть консоль CentOS 7 (если не работает клавиатура в ВМ, нажмите на синюю рамку). Для того, чтобы открыть консоль в отдельной вкладке, нажмите Click here to show only console.



Залогиньтесь в системе (в CentOS 7 логин - **labuser,** пароль - **labpass1!**).

Задание 2. Работа с файловой системой.

1. Создать директорию `task01`. Перейдите в неё.

```
mkdir task01
cd task01
```

2. Создать в домашнем каталоге текстовый файл `user` при помощи текстового редактора vim, и заполнить его произвольными символами *(*`a` *начало редактирования, ESC выход из режима редактирования,* `:wq` *выход с сохранением* `:q!` *выход без сохранения)*.

```
vi user
```

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

```
cp user root
```

4. Посмотреть права доступа на файлы.

```
ls -l
```

5. Задать владельца `root` и группу `root` на файл `root`. (sudo позволяет выполнять команды от root; введите пароль в диалоге, 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 rwx.

```
chmod 600 lock
```

9. Вывести содержимое файла `root` в терминал.

```
cat root
```

10. Отредактировать файл root и вывести его в консоль.

```
sudo vi root
cat root
```

11. Удалить каталог `task01`.

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

Задание 3. Установка пакетов.

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

Для установки/удаления пакетов существуют команды:

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

Давайте, в качестве примера, установим программу для скачивания файлов по web-протоколам - `wget`.

```
sudo yum install wget
```

Проверьте версию установленного пакета:

```
wget --version
```

Задание 4. Работа с переменными окружения.

Переменные окружения - именованные переменные, содержащие текстовую информацию, которую могут использовать запускаемые программы. Такие переменные могут содержать общие настройки системы, параметры графической или командной оболочки, данные о предпочтениях пользователя и многое другое. Значением такой переменной может быть, например, место размещения исполняемых файлов в системе, имя предпочитаемого текстового редактора и т.п.

Для создания переменной достаточно использовать конструкцию `varname=value`. В дальнейшем переменную можно использовать в качестве аргумента другой программы, или же сохранить там путь до какой-либо программы, и запускать её по имени переменной.

Например, давайте запишем путь до программы `echo` в переменную `program`. В переменную `argument` Мы запишем строку, которую хотим передать в качестве аргумента к `echo`. И после запустим `$program $argument`.

```
program=$(which echo)
echo $program
argument=”Test string”
echo $argument
$program $argument
```

Задание 5. 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 подключиться к пользователю, можно использовать команду su. При этом нужно будет ввести пароль.

```
sudo - newuser
```

Вы увидите, что 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 пользователь, наберите команду `groups`. В вашем случае это должны быть группы `newuser` и `groups`.

```
groups
```

Для выхода из оболочки используется команда `exit`. При этом вы выйдете в оболочку, запущенную пользователем `labuser`. Именно из неё вы запускали ту, в которой находитесь сейчас.

```
exit
```

Давайте теперь сменим оболочку, запускаемую по умолчанию при входу пользователя newuser. Чтобы узнать список доступных в системе оболочек, наберите следующую команду:

```
cat /etc/shells
```

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

```
echo $0
```

Чтобы просто запустить оболочку, достаточно просто набрать путь к ней. Запустите оболочку `sh`, и выведите на экран переменную `$0`. После выйдете 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`. Посмотрите, какая оболочка сейчас является оболочкой по умолчанию для пользователя `usermod`, смените её на `/bin/sh`, и проверьте изменения.

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

Подключитесь к пользователю, и проверьте, какая оболочка используется. После выйдете, и удалите пользователя.

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

Задание 6. Работа с текстом в bash.

Для практики воспользуемся 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. Загрузим необходимый файл, воспользовавшись `s3cmd`:

```
s3cmd get s3://lab1/clients.txt ~/
```

Сначала просто выведем 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`. Давайте выведем информацию о клиенте под номером 24 (`grep` - регистрозависимый, так что обратите внимание, что `Client24` начинается с заглавной).

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

Как можно заметить, вывелась одна строка. Если было несколько строк, содержащих подстроку `Client24`, то и в результате вывелось несколько строк. Именно так можно вывести, например, всех клиентов, номер которых начинается на 2.

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

Помимо обычных строк `grep` поддерживает также регулярные выражения. При этом регулярные выражения должны экранироваться символом `\`. С помощью регулярных выражений можно задать абсолютно любой паттерн. Попробуйте относительно простой - выведите всех клиентов, имя которых заканчивается на 3. Точка в выражении означает любой символ.

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

`grep` также умеет работать с пайплайном. С помощью пайплайна можно передавать вывод от одной программы другой. Например, можно вывести текст через `cat`, а затем обработать вывод с помощью `grep`.

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

Пайплайнов может быть несколько. Так можно, например, воспользовавшись программой AWK вывести только имя клиента и его внешний адрес - то есть первый и четвертый столбцы. (Обратите 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`. Попробуйте 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)}’
```

В завершение, рассмотрим еще один пример. Если, например, необходимо вывести MAC адрес, чтобы он был разделён не двоеточиями, а дефисами, тогда можно воспользоваться `sed`. Нужно лишь задать регулярное выражение. (Набирать в одну строку)

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

Лабораторная работа 2.
Работа с дисковой подсистемой CentOS.

Цель

Получение базовых навыков при работе с дисковой подсистемой в операционной системе CentOS 7.

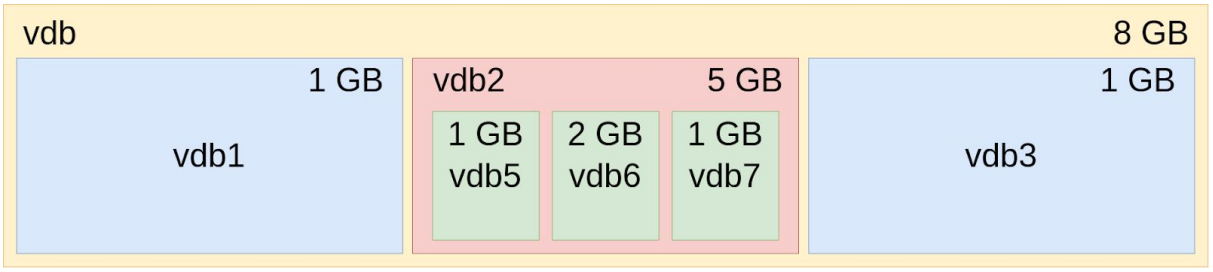
Задачи

1) Разметить диск как DOS (MBR), создать на его разделах файловую систему.
2) Создать LVM и провести с ним все операции.
3) Примонтировать логический том, выполнить все описанные операции на нем.
4) Использовать fstab.

Задание 1. Создание раделов с использованием fdisk на MBR.

Переключитесь на проект [**GROUP]:[team].lab1**.
Подключитесь к **labnode**. Напомню, логин - **labuser**, пароль - **labpass1!**

Давайте сделаем на диске следующую разметку:



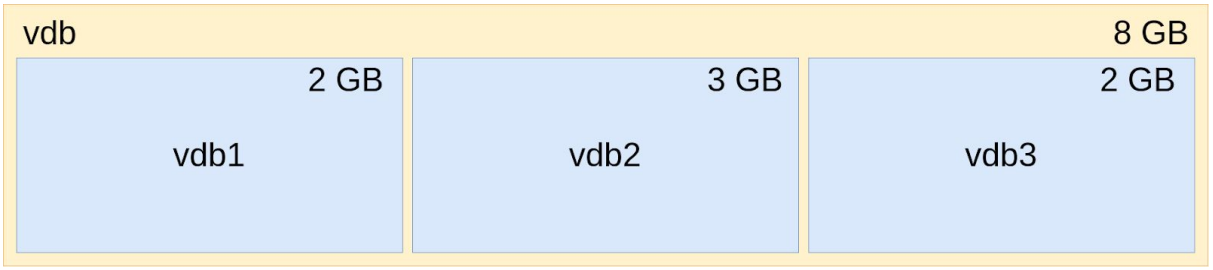
Для этого запускаем fdisk в интерактивном режиме, в качестве аргумента передавая путь к блочному устройству.

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

1. Создаем разметку DOS (MBR), с помощью команды `o`.
2. Создаем primary раздел. Нажимаем `n`, для создания нового раздела. Нажимаем `p`, указывая, что нам нужен именно primary. Номер раздела оставляем по умолчанию. Первый сектор раздела тоже. Последний сектор указываем +1G. В итоге должен получиться primary раздел на 1 ГБ. Можем проверить, что раздел добавиться в таблицу разделов, с помощью команды `p`.
3. Создаем extended раздел на 5 ГБ. Делаем все то же самое, но выбираем вместо primary, extended, набрав `e`, и последний сектор указываем +5G от первого.
4. Четвертый шаг полностью совпадает со вторым. Снова создаем primary раздел на 1 ГБ.

5. Создаем два раздела по 1 ГБ и один на 2 ГБ , но выбираем logical вместо primary и extended, нажимая `l`.
6. Проверяем получившуюся таблицу разделов, с помощью `p`. После записываем её на диск, нажав `w`.

Задание 2. Создание раделов с использованием fdisk на GPT.



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

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

1. Выбираем таблицу разметки GPT. Делаем это, нажав `g`.
2. Создадим 3 раздела, согласно схеме. Делаем это снова с помощью команды `n`, как и в случае с MBR.

Задание 3. Создание файловой системы.

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

1. На vdb1 пусть будет ext4.
2. На vdb2 - xfs.
3. На vdb3 - btrfs.

Делается это так.

```
sudo mkfs.ext4 /dev/vdb1
sudo mkfs.xfs /dev/vdb2
sudo mkfs.btrfs /dev/vdb3
```

Проверьте, что файловые системы были созданы.

```
sudo lsblk -f
```

Задание 4. Работа с LVM.

В этом задании используется разметка из задания 5.
Для выполнения этого задания вам потребуется пакет lvm2. Установите его с помощью yum.

```
sudo yum install lvm2
```

Давайте вначале изменим системный id раздела. Он влияет на то, какая метка файловой системы будет отображаться в fdisk в столбце Type.
Запускаем fdisk в интерактивном режиме, в качестве аргумента передавая путь к блочному устройству.

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

1. Нажимаем `t`, выбираем номер раздела, метку которого хотим поменять. Далее нажимаем `L`, чтобы просмотреть все доступные метки. Ищем там Linux LVM (метка 31). Набираем id, под которым стоит нужная нам метка.
2. Проделываем эту операцию со всеми разделами на диске.
3. Проверяем, после записываем, нажимая `w`.
4. Сделаем теперь все три раздела физическими томами (В процессе будет сообщение, что файловая система на томе будет уничтожена. Соглашаемся, набрав y).

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

Проверить успешность можно с помощью команды `sudo pvdisplay`.

5. На этих физических томах создаём группу томов, которая будет называться, скажем, vg1:

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

Проверяем с помощью `sudo vgdisplay`.

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
```

Проверяем с помощью `sudo lvdisplay`.

7. Теперь у нас есть 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. Давайте удалим из группы том `/dev/vdb3`.

Чтобы убрать из работающей группы томов раздел, сначала перенесем все данные с него на другие разделы:

```
sudo pvmove /dev/vdb3
```

Затем удалим его из группы томов:

```
sudo vgreduce vg1 /dev/vdb3
```

И, наконец, удалим физический том:

```
sudo pvremove /dev/vdb3
```

Вообще-то, 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. Давайте вернем `/dev/vdb3` обратно.

Чтобы добавить новый его в группу томов, создадим физический том:

```
sudo pvcreate /dev/vdb3
```

И добавим его в нашу группу:

```
sudo vgextend vg1 /dev/vdb3
```

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
```

А затем файловую систему на нём:

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

Задание 5. Монтирование разделов.

Давайте удалим существующие логические LVM разделы, и создадим новый.

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

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

Теперь, в каталоге пользователя, создадим каталог `media`.

```
mkdir ~/media
```

И, наконец, примонтируем раздел.

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

Давайте теперь запишем тестовый файл test.

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

Если проблемы с доступом к записи, смените владельца каталога,. После выполните команду заново.

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

Отмонтируйте раздел.

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

Зайдите внутрь созданного каталога, и удостоверьтесь, что файла test там нет. Он остался на разделе.
Если мы не хотим каждый раз после перезагрузки монтировать раздел заново, то нужно добавить автомонтирование в конфигурационный файл `/etc/fstab`.

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

Добавляем туда следующую строку.

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

Перезагружаем систему с помощью команды `reboot`. После загрузки заходим в каталог `~/media`, и видим там наш файл `test`.

Лабораторная работа 3.

Работа с сетью в CentOS. Виртуальный коммутатор Linux Bridge.

Цель

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

Задачи

1) Задать статический IP адрес.
2) Настроить bond0 интерфейс
3) Настроить br0
4) Настроить VxLAN
5) Посмотреть на основные команды netstat

Задание 1. Установка статического IP адреса физическому интерфейсу

Переключитесь на проект [**GROUP]:[team].lab2**.
Подключитесь к **labnode-1**. Напомню, логин - **labuser**, пароль - **labpass1!**

Воспользуемся утилитой ip. Для того, чтобы увидеть существующие в системе интерфейсы, наберите команду:

```
ip a
```

Задайте интерфейсу `eth1` IP адрес:

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

Измените состояние на **up**. Посмотрите изменения:

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

Подключитесь к **labnode-2**. Отключите Network Manager. Установите интерфейсу `eth1` IP адрес:

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

Измените состояние на **up**:

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

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

```
ping -c 4 10.0.12.30
```

После перезагрузки сервера, или сервиса сети все изменения, что вы внесли, отменяться. Перезагрузите **оба** сервера, и посмотрите на интерфейсы:

```
reboot
ip a
```

Для того, чтобы изменения оставались в силе, нужно настроить интерфейс через конфигурационный файл. Тогда настройки будут загружаться при старте системы.

Задание 2. Настройка статического адреса через конфиг

Создайте конфигурацию интерфейса `eth1` на **labnode-1**:

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

И приведите его к следующему виду:

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

Опишу 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 получать по dchp адрес, маску и другие параметры подключения. Соответственно, может быть также 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 - префикса, другой способ задания маски сети. Для префикса *24* маска будет *255.255.255.0*
9. GATEWAY - шлюз

Скопируйте файл `ifcfg-eth1` с именем `ifcfg-eth2`:

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

Зайдите в него через vi:

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

И измените его следующим образом:

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

Перезагрузите сеть и посмотрите интерфейсы:

```
sudo systemctl restart network
ip a
```

Сделайте то же самое с **labnode-2**. Создайте конфигурацию интерфейса `eth1`:

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

И приведите его к следующему виду:

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

Скопируйте файл `ifcfg-eth1` с именем `ifcfg-eth2`:

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

Зайдите в него через vi:

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

И измените его следующим образом:

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

Перезагрузите сеть и посмотрите интерфейсы:

```
sudo systemctl restart network
ip a
```

Проверьте 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 bond интерфейса.

Что такое "объединение" (bonding) сетевых интерфейсов? Под словом объединение будем подразумевать - портовый транкинг (автоматическое распределение каналов по требованию). В дальнейшем будет использоваться слово - "объединение" потому, что происходит практически объединение в единое целое. Объединение позволяет совокупно собрать несколько портов в одну группу, эффективно объединяя пропускную способность в одном направлении. Объединение также позволяет создавать мульти-гигабитные каналы для транспортировки трафика через высокопропускные районы вашей сети. Например, вы можете объединить два порта по 100 мегабит в 200 мегабитный магистральный порт. Это эквивалентно одному интерфейсу с пропускной способностью 200 мегабит.

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. Зайдите в конфиг `eth1`:

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

И приведите его к следующей конфигурации:

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

То же самое сделайте и с `eth2`:

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

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

Перезагрузите сеть.

```
sudo systemctl restart network
```

Посмотрите, что получилось:

```
ip a
```

Теперь проделаем то-же 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
```

Перезагрузите сеть:

```
sudo systemctl restart network
```

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

```
ping -c 4 10.0.12.30
```

Теперь на **labnode-1** отключите `eth1` и посмотрите его состояние:

```
sudo /etc/sysconfig/network-scripts/ifdown eth1
ip a
```

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

```
ping 10.0.12.20
```

Все должно работать. Завершите пинг сочетаниием `^C`. Перезагрузите сеть на **labnode-1**.

Задание 4. Настройка bridge интерфейса.

На **labnode-1** создайте 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
```

Здесь мы включим Spanning Tree Protocol (STP), чтобы избежать петель коммутации.

В конфиг bond0 также нужно добавить параметр `BRIDGE=br0`. Также удалите 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
```

Перезагрузите сеть:

```
sudo systemctl restart network
```

Можете проверить результат:

```
ip a
(labnode-2) ping -c 4 10.0.12.20
```

Мост может подняться не сразу. Если что, немного подождите.

Задание 5. Создание VxLAN интерфейсов.

На **labnode-1** удалите конфигурацию моста `br0`:

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

И приведите `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 перезагрузите сервер:

```
sudo reboot
```

После загрузки сервера проверьте работу сети:

```
(labnode-2) ping -c 4 10.0.12.20
```

На **labnode-1** добавьте интерфейс `vxlan10`:

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

Настройте коммутацию Linux Bridge:

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

Назначьте `vxlan10` IP адрес и переведите 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*. Откройте 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**. Добавьте интерфейс `vxlan10`:

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

Настройте коммутацию Linux Bridge:

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

Назначьте `vxlan10` IP адрес и переведите его в состояние **up**:

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

Откройте порт 8472/udp в фаерволе:

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

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

```
ping -c 4 192.168.1.30
```

Давайте посмотрим на arp таблицу. Там вы увидите соответствие mac адресов с ip адресами.

```
sudo arp
```

Также убедимся в том, что появилось приложение, которое слушает порт 8472.

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

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

```
sudo reboot
```

На **labnode-2** добавьте интерфейс `vxlan20` и настройте 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
```

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

```
ping 192.168.1.30
```

Нажмите `^C` для того, чтобы прекратить пинг.

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!

Задание 1. Установка QEMU.

Переключитесь на проект [**GROUP]:[team].lab3**. Включите **labnode-1** и **virt_viewer**.

На **labnode-1**:

1. Устанавливаем пакеты:

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

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

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

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. Построим пустой образ диска с помощью утилиты qemu-img, которая должна быть уже у вас установлена, если вы правильно сделали первое задание.

1. Проверим какие типы образов поддерживаются в нашей версии `qemu-img`:

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

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

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

3. Проверим что файл был создан:

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

4. Посмотрим 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 вам необходимо преобразовать его вначале в образ raw при помощи команды преобразования qemu-img.

1. Конвертируем образ диска из формата qcow2 в raw:

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

2. Добавляем дополнительно 5 ГБ к образу:

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

3. Проверим новый текущий размер образа:

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

4. Конвертируем образ диска обратно из raw в qcow2:

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

Задание 4. Загрузка образа Cirros.

Нам потребуется утилита wget для загрузки образов с общедоступных репозиториев. Загрузим необходимый образ, воспользовавшись `s3cmd`:

```
s3cmd get s3://lab3/cirros.img /tmp/
```

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

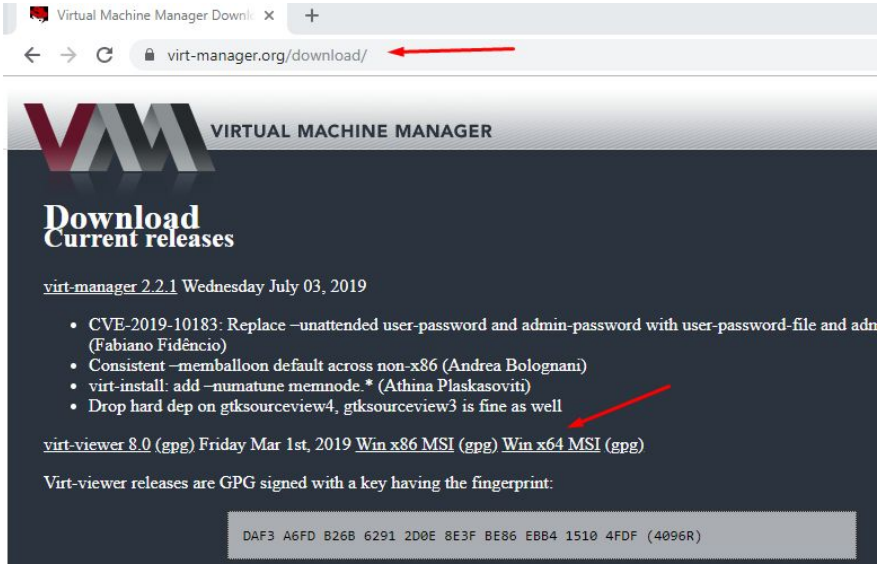
1. Для того, чтобы подключиться к нашей виртуальной машине по протоколу удаленного рабочего стола Spice, нам нужно открыть порт 5900.

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

2. Посмотрите ip адрес вашего сервера (напомню, `ip a`)
3. Запускаем систему при помощи qemu-system:

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

4. Подключаемся из виртуальной машины Windows 10 к виртуальной машине. Для этого скачайте программу Virt Viewer. Программу можно найти на сайте virt-manager.org/download/.



Установите и откройте программу (название - Remote Viewer). Подключаться по адресу `spice://10.0.12.21:5900`. (Пользователь - **Admin**, пароль - **labpass1!**)

5. Залогиньтесь в Cirros. Дефолтные логин и пароль написаны в консоли(**cirros**/**gocubsgo**) ОС. Наберите команду `uname -a`. Посмотрите на версию ядра ОС. Выключите 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!

Задание 1. Установка Libvirt и Virsh.

Необходимо установить всего несколько пакетов, которые не входят в базовую комплектацию системы. В проекте **[GROUP]:[team].lab3,** на **labnode-1** выполните следующую команду:

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

Задание 2. Настройка моста.

Установите пакет bridge-utils:

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

Выведите на экран имеющиеся интерфейсы:

```
ip -c a
```

Откройте файл `/etc/sysconfig/network-scripts/ifcfg-br0`:

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

И добавьте в него следующее содержимое:

```
TYPE=Bridge
DEVICE=br0
BOOTPROTO=dhcp
ONBOOT=yes
```

А в файл `/etc/sysconfig/network-scripts/ifcfg-eth0` добавьте параметр `BRIDGE`, уберите `BOOTPROTO` и `ONBOOT`:

```
~~BOOTPROTO=dhcp~~
~~ONBOOT=yes~~
BRIDGE=br0
```

Перезагрузите сервер:

```
sudo reboot
```

Задание 3. Создание виртуальной машины.

Переместите образ *cirros* в `/var/lib/libvirt/images/`

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

Следующая команда создаёт 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
```

* 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.
* Если вы ошиблись при наборе команды, можно не набирать ее заново, а нажать стрелку вверх, исправить ее, и снова нажать **Enter**

Подробнее о параметрах:

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	Сетевой интерфейс, который будет подключен к виртуальной машине
graphics	Определяет графическую конфигурацию дисплея.
cdrom	CD ROM устройство

6. Подключаемся из виртуальной машины Windows 10 (Пользователь - **Admin**, пароль - **labpass1!**) к виртуальной машине через программу Virt Viewer. Программу можно найти на сайте spice-space.org.(уже была скачана ранее)

Открываем её (название - Remote Viewer). Подключаться по адресу `spice://10.0.12.21:5900`.

Вернитесь в консоль `labnode-1`. Проверьте состояние вашей 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.

Давайте теперь посмотрим, что нам предоставляет virsh. Мы уже смотрели список всех ВМ. Теперь давайте посмотрим, как можно подключиться к нашей ВМ по протоколу удаленного доступа:

```
sudo virsh domdisplay cirros
```

Получить информацию о конкретной ВМ можно так:

```
sudo virsh dominfo cirros
```

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
```

А удаление ВМ из автозапуска:

```
sudo virsh autostart --disable cirros
```

Кроме того нам может потребоваться отредактировать XML конфигурацию нашей ВМ:

```
sudo virsh edit cirros
```

** Чтобы выйти из редактора без сохранения, наберите :q!*

Для того, чтобы выгрузить XML в файл, нужно использовать команду:

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

Удаление ВМ:

```
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!

Задание 1. Настройка nfs клиента

В проекте [**GROUP]:[team].lab3**, на **labnode-1**, **labnode-2** и **labnode-3** зайдите в файл `/etc/fstab`:

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

И раскомментируйте 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 перемонтируйте разделы, введя:

```
sudo mount -a
```

Задание 2. Установка Pacemaker и Corosync

Установка очень проста. На всех узлах выполняем команду:

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

Далее поднимаем pcs. Тоже, на всех узлах

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

Откройте 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
```

Отредактируйте файл `/etc/hosts`, добавив в него строки (на всех узлах):

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

```
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
```

Создаем пользователя hacluster. На всех узлах.

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

И, с помощью pcs создаем 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
```

Отключите fencing (в рамках работы он не рассматривается)

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

Включите автозапуск сервисов на всех трех машинах:

```
sudo systemctl enable pacemaker corosync
sudo systemctl status pacemaker corosync
```

Просмотрите информацию о кластере и кворуме:

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

Задание 3. Настройка моста.

На **labnode-1** вы уже создали мост. Сделайте то-же на **labnode-2** и **labnode-3.** Откройте файл:

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

И добавьте в него следующее содержимое:

```
TYPE=Bridge
DEVICE=br0
BOOTPROTO=dhcp
ONBOOT=yes
```

А в файл `/etc/sysconfig/network-scripts/ifcfg-eth0` добавьте параметр `BRIDGE` и уберите `BOOTPROTO` и `ONBOOT`:

```
DEVICE=eth0
#HWADDR=...(оставить как было)
ONBOOT=yes
TYPE=Ethernet
USERCTL=no
BRIDGE=br0
```

Перезагрузите сеть:

```
sudo systemctl restart network
```

Задание 4. Создание ресурса

Для начала, нам нужно отключить Selinux. Для этого заходим в файл `/etc/selinux/config`:

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

И заменяем значение параметра `SELINUX` с enforcing на permissive:

```
SELINUX = permissive
```

После чего перезагружаем ВМ:

```
sudo reboot
```

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

В предыдущих заданиях мы сделали дамп виртуальной машины cirros на **labnode-1**.

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

Зайдите в него через vi:

```
sudo vi cirros.xml
```

И измените в разделе <disk/> путь до диска с /var/lib/libvirt/images/ на /media/nfs_share/.

```
<source file='/var/lib/libvirt/images/cirros-0.4.0-x86_64-disk.img'/>
<source file='/media/nfs_share/cirros.img'/>
```

* вы можете воспользоваться поиском по файлу, набрав `/`, а затем то, что ищете. Искать нужно `<disk`. То есть, наберите `/<disk`.

Скопируйте 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 переместите файл в `/etc/pacemaker/`

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

Теперь добавьте сам ресурс:

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

Просмотрите список добавленных ресурсов

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

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

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

Проверьте, что ресурс успешно запустился. Для этого из **virt_viewer** (Пользователь - **Admin**, пароль - **labpass1!**) подключитесь к нему через программу Remote Viewer.
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].lab3

Задание 1. Настройка динамической миграции

Вы уже открыли порты в фаерволе, теперь нужно настроить libvirt.
Перейдите в файл `/etc/libvirt/libvirtd.conf`

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

Добавьте туда три параметра

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

Сохраните файл. Следующим действием в файле `/etc/sysconfig/libvirtd`

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

Добавьте параметр:

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

Перезагрузите libvirt.

```
sudo systemctl restart libvirtd
```

Проделайте эти операции на всех узлах.

Задание 2. Миграция ресурса

Давайте переместим ресурс на **labnode-2**:

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

На labnode-2 посмотрите статус кластера, и проверьте список запущенных гостевых машин:

```
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, используйте clear:

```
sudo pcs resource clear cirros
```

Из Remote Viewer Проверьте доступность ВМ на labnode-2. (`spice://10.0.12.22:5900`).
Дождитесь загрузки cirros.
Переместите ресурс на labnode-1:

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

Посмотрите на результат:

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

Теперь, если вы подключитесь к ресурсу, используя Remote Viewer, вы увидите, 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.