

Мой дневник по практике

1. Подготовительная работа для создания ВМ.

1.1 Проверка поддержки виртуализации, если виртуализация есть, то должен быть такой вывод.

```
$ kvm-ok  
INFO: /dev/kvm exists  
KVM acceleration can be used
```

1.2. Установка необходимых пакетов: [bridge-utils](#) (создание моста), [libvirt-daemon-system](#) (создание ВМ), [qemu и qemu-kvm](#) (работа гипервизора).

```
$ sudo apt install qemu qemu-kvm libvirt-daemon-system
```

1.3. Активация **libvirtd** для дальнейшей работы.

```
$ sudo systemctl enable --now libvirtd  
$ sudo systemctl status libvirtd  
● libvirtd.service - Virtualization daemon  
  Loaded: loaded (/lib/systemd/system/libvirtd.service; enabled; vendor preset: enabled)  
  Active: active (running) since Mon 2022-11-21 16:36:04 MSK; 5min ago
```

1.4. Во всяких [гайдах](#) говорится, что есть три способа настройки доступа виртуальных машин в сеть: через одну из ВМ (самый трудно настраиваемый и еще доп ВМ нужна), через NAT (уже реализован по дефолту **virbr0**), через Bridge (самый обычный и удобный, его и реализуем, не зря ведь **bridge-utils** качали).

Для этого создаем сетевой интерфейс **br0** типа bridge.

```
$ sudo ip link add br0 type bridge  
$ sudo ip link show type bridge  
5: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN mode DEFAULT group default qlen 1000  
    link/ether 52:54:00:1f:bc:e8 brd ff:ff:ff:ff:ff:ff  
6: br0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000  
    link/ether 62:94:ae:7f:a2:67 brd ff:ff:ff:ff:ff:ff
```

Потом все удаляем так как [оказывается](#), что нельзя проводить мост к wifi сети, только к ethernet, потому что Wi-Fi требует дополнительных MAC-адресов для связи, из-за чего принято решение пользоваться NAT :-)

1.5. Установка софта для обновления времени сервера [chrony](#), для синхронизации времени виртуалок.

```
$ sudo timedatectl set-timezone Europe/Moscow  
$ sudo apt install chrony  
$ sudo systemctl enable --now chrony
```

```
$ sudo systemctl status chrony
● chrony.service - chrony, an NTP client/server
  Loaded: loaded (/lib/systemd/system/chrony.service; enabled; vendor preset: enabled)
  Active: active (running) since Mon 2022-11-21 20:53:13 MSK; 20s ago
```

2. Установка образа Centos и создание 3 ВМ с помощью KVM.

2.1. Создание диска с помощью [qemu-img](#) формата [qcow2](#) для монтирования в ВМ (**preallocation=metadata** – выделяет пространство, необходимое для метаданных, но не выделяет место для данных. Это самый быстрый способ представления, но самый медленный для гостевой записи). Папки создал сам, кроме **/mnt**.

```
$ sudo qemu-img create -f qcow2 -o preallocation=metadata /mnt/kvm/disk/vm1.qcow2 6G
Formatting '/mnt/kvm/disk/vm1.qcow2', fmt=qcow2 cluster_size=65536 extended_l2=off preallocation=metadata
compression_type=zlib size=6442450944 lazy_refcounts=off refcount_bits=16
```

2.2. Установка утилит [virt-install](#) (создание ВМ) и [osinfo-query](#) (получение информации о поддержке гипервизорами различных операционных систем) для создания гостевых машин в среде kvm из командной строки.

```
$ sudo apt install virtinst libosinfo-bin
```

2.3. Просмотр доступных для KVM образов Centos с помощью команды [osinfo-query os](#).

```
$ osinfo-query os | grep centos
centos-stream8      | CentOS Stream 8      | 8      | http://centos.org/centos-stream/8
centos-stream9      | CentOS Stream 9      | 9      | http://centos.org/centos-stream/9
centos5.0           | CentOS 5.0           | 5.0    | http://centos.org/centos/5.0
centos5.1           | CentOS 5.1           | 5.1    | http://centos.org/centos/5.1
centos5.10          | CentOS 5.10          | 5.10   | http://centos.org/centos/5.10
centos5.11          | CentOS 5.11          | 5.11   | http://centos.org/centos/5.11
centos5.2           | CentOS 5.2           | 5.2    | http://centos.org/centos/5.2
centos5.3           | CentOS 5.3           | 5.3    | http://centos.org/centos/5.3
centos5.4           | CentOS 5.4           | 5.4    | http://centos.org/centos/5.4
centos5.5           | CentOS 5.5           | 5.5    | http://centos.org/centos/5.5
centos5.6           | CentOS 5.6           | 5.6    | http://centos.org/centos/5.6
centos5.7           | CentOS 5.7           | 5.7    | http://centos.org/centos/5.7
centos5.8           | CentOS 5.8           | 5.8    | http://centos.org/centos/5.8
centos5.9           | CentOS 5.9           | 5.9    | http://centos.org/centos/5.9
centos6.0           | CentOS 6.0           | 6.0    | http://centos.org/centos/6.0
centos6.1           | CentOS 6.1           | 6.1    | http://centos.org/centos/6.1
centos6.10          | CentOS 6.10          | 6.10   | http://centos.org/centos/6.10
centos6.2           | CentOS 6.2           | 6.2    | http://centos.org/centos/6.2
centos6.3           | CentOS 6.3           | 6.3    | http://centos.org/centos/6.3
centos6.4           | CentOS 6.4           | 6.4    | http://centos.org/centos/6.4
centos6.5           | CentOS 6.5           | 6.5    | http://centos.org/centos/6.5
centos6.6           | CentOS 6.6           | 6.6    | http://centos.org/centos/6.6
centos6.7           | CentOS 6.7           | 6.7    | http://centos.org/centos/6.7
centos6.8           | CentOS 6.8           | 6.8    | http://centos.org/centos/6.8
centos6.9           | CentOS 6.9           | 6.9    | http://centos.org/centos/6.9
centos7.0           | CentOS 7              | 7      | http://centos.org/centos/7.0
centos8             | CentOS 8              | 8      | http://centos.org/centos/8
```

2.4. Установка дистрибутива операционной системы с сайта http://mirror.yandex.ru/centos/7.9.2009/isos/x86_64/ В моем случае загружается образ CentOS-7-x86_64-Minimal-2009.iso. Оказалось, что **Centos** крутая штука, так как ворует все идеи у коммерческой **Red Hat Enterprise Linux**, но при этом абсолютно бесплатная.

2.5. Создание ВМ с CentOS7.0 с помощью **virt-install** ([сайт](#) с документацией параметров).

```
$ sudo virt-install --virt-type kvm --name VM1 --ram 4096 --arch=x86_64 --disk /mnt/kvm/disk/vm1.qcow2,size=6,format=qcow2 --network network=default --os-type=linux --os-variant=centos7.0 --location=/mnt/kvm/iso/CentOS-7-x86_64-Minimal-2009.iso --graphics none --console pty,target_type=serial --extra-args 'console=ttyS0,115200n8 serial'
```

- **--virt-type kvm** — тип гипервизора
- **--name VM1** — имя виртуалки
- **--ram 4096** — оперативка
- **--arch=x86_64** - имитируемая архитектура процессора
- **--disk /mnt/kvm/disk/vm1.qcow2,size=6,format=qcow2** — пространство хранения данных, **size** — размер, **format** — формат тома.
- **--network network=default** — сетевой интерфейс (**default=virbr0**)
- **--os-type=linux** — тип операционной системы
- **--os-variant=centos7.0** — конкретный вариант ОС
- **--location=/mnt/kvm/iso/CentOS-7-x86_64-Minimal-2009.iso** — путь до установочного образа.
- **--graphics none** — настройки экрана ВМ (**none** — без графического интерфейса)
- **--console pty,target_type=serial** — тип подключения к гостевой консоли (**pty** - обеспечивают работу терминала в оконном интерфейсе, **target_type=serial** — дефолтная консоль, не открывается новое окно)
- **--extra-args 'console=ttyS0,115200n8 serial'** — дополнительные аргументы, в моем случае параметры консоли в которую будет выводится информация иначе не видать меню загрузки.

2.6. Настройка с помощью меню установки **Centos** без графического интерфейса (в первый раз не понял, что за фигня и как этим пользоваться, оказывается циферками). Все загрузится нормально если на месте восклицательных знаков будут крестики.

```

Starting installer, one moment...
anaconda 21.48.22.159-1 for CentOS 7 started.
* installation log files are stored in /tmp during the installation
* shell is available on TTY2
* when reporting a bug add logs from /tmp as separate text/plain attachments
12:38:59 Not asking for VNC because we don't have a network
=====
=====
Installation

1) [x] Language settings           2) [!] Time settings
      (English (United States))   (Timezone is not set.)
3) [!] Installation source        4) [!] Software selection
      (Processing...)            (Processing...)
5) [!] Installation Destination   6) [x] Kdump
      (No disks selected)       (Kdump is enabled)
7) [ ] Network configuration     8) [!] Root password
      (Not connected)          (Password is not set.)
9) [!] User creation             Please make your choice from above ['q' to quit | 'b' to begin installation | 'r' to refresh]:

```

Настройка сети такая (почему-то скрин поплыл, но там **DNS** и **Gateway** одинаковые — **192.168.122.1**):

```

Network configuration

Wired (eth0) connected
IPv4 Address: 192.168.122.200 Netmask: 255.255.255.0 Gateway:
192.168.122.1

Host name: localhost.localdomain
Current host name: localhost

```

Заполненное меню загрузки такое, пользователя не создаем:

```

Installation

1) [x] Language settings           2) [x] Time settings
      (Russian (Russia))         (Europe/Moscow timezone)
3) [x] Installation source        4) [x] Software selection
      (Local media)              (Minimal Install)
5) [x] Installation Destination   6) [x] Kdump
      (Automatic partitioning    (Kdump is enabled)
       selected)
7) [x] Network configuration     8) [x] Root password
      (Wired (eth0) connected)   (Password is set.)
9) [ ] User creation             (No user will be created)

```

Готовая ВМ:

```

localhost логин: root
Пароль:
$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
  link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
  inet 127.0.0.1/8 scope host lo
    valid_lft forever preferred_lft forever
  inet6 ::1/128 scope host
    valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
  link/ether 52:54:00:e8:75:87 brd ff:ff:ff:ff:ff:ff
  inet 192.168.122.200/24 brd 192.168.122.255 scope global noprefixroute eth0
    valid_lft forever preferred_lft forever
  inet6 fe80::a092:eab0:9da0:9cee/64 scope link noprefixroute
    valid_lft forever preferred_lft forever

```

2.7. Полезные команды для работы с ВМ:

- **virsh list --all** — просмотр всех ВМ
- **virsh start vm** — запустить неактивную ВМ
- **virsh shutdown vm** — выключить ВМ через гостевую ОС
- **virsh destroy vm** — выключить ВМ принудительно (выдернуть из розетки)
- **virsh domblklist vm** — информация о дисках ВМ
- **virsh dominfo vm** — информация о ВМ
- **virsh nodeinfo** — информация о хостовой машине
- **virsh undefine --domain vm** — удалить всю информацию о ВМ, например чтобы переиспользовать диск который ей был занят (я то 3 раза ее пересоздал)
- **virsh console vm** — подключение к консоли ВМ
- **virsh dumpxml vm** — информация о настройках вм
- **virsh snapshot-list --domain vm** — просмотр снапшотов ВМ
- **virsh snapshot-create-as vm vm-snapshot** — создание снапшота ВМ
- **virsh snapshot-info --domain vm --snapshotname vm-snapshot** — информация о снапшоте ВМ
- **virsh snapshot-create-as --domain vm --name vm-snapshot** — создание снапшота ВМ
 - **--disk-only** — создание снапшота только диска (без состояния оперативки)
- **virsh snapshot-revert --domain vm --snapshotname vm-snapshot** — откатить ВМ к снапшоту
 - **--running** — запустить ВМ
- **virsh snapshot-delete --domain vm --snapshotname vm-snapshot** — удалить снапшот ВМ
- **virsh net-create path_to_net_conf** — создать сеть для ВМ
- **virsh net-list** — список сетей
- **virsh net-autostart net** — настройка автостарта сети
- **virsh attach-interface vm --type bridge --source interface --model virtio --config --live --persistent** — присоединение сетевого интерфейса к ВМ
 - **--source** — имя сетевого интерфейса на хосте
 - **--model** — тип интерфейса
 - **--config** — через файл конфига
 - **--live** — ВМ не выключать
 - **--persistent** — подключить навсегда

- **virsh detach-interface vm3 --mac '52:54:00:3b:6e:3f' --type bridge --persistent**
— отсоединение сетевого интерфейса от ВМ
- **virsh net-info net** — информация о сети
- **qemu-img info disk1.qcow2** — информация об образе диска
- **qemu-img resize /mnt/kvm/disk/vmserver01-disk1.qcow2 +1G** — увеличение размера диска
- **qemu-img info /mnt/kvm/disk/vmserver01-disk1.qcow2** — информация о диске
- **Ctrl +]** - выход из гостевой консоли
- **virt-install --parameter=?** - информация о параметре (лучше сайтом пользоваться с команды пользы 0)
- **nmcli connection show** — показывает все сетевые интерфейсы кроме loopback
- **nmcli connection delete interface** — удаляет сетевой интерфейс (если вы выяснили, что мост в wifi не прокидывается)
- **sudo mv src dst** — перенести файл, если у вас почему-то нет прав на доступа к папке **/mnt** через графический интерфейс Ubuntu

Остальные 2 ВМ поднимаются по аналогичной схеме только с другими ip адресами, требованиями к оперативке и размеру диска.

3. Установка и настройка RabbitMQ на Control узел.

3.1. Зачем он вообще нужен? Оказывается **RabbitMQ** или какой-нибудь другой брокер сообщений (**Apache Qpid** или **ZeroMQ**) используется для координации операций и обмена информацией между сервисами **OpenStack**, такими как **Glance**, **Cinder**, **Nova**, **Neutron**, **Heat** и **Ceilometer** по протоколу AMQP (Advanced Message Queuing Protocol).

3.2. [Установка RabbitMQ](#) на ВМ с помощью yum (-y — автоответ yes, если не знали).

Сноска от автора (дочитайте до конца!), пока я пытался установить RabbitMQ по гайдам, я повидал многое, но после долгих поисков мне все таки удалось найти нормальное руководство, при помощи которого **RabbitMQ** все таки удалось установить. Но я же проделал такую большую работу качая **RabbitMQ** по другим гайдам, так что было принято решение сохранить мои первые попытки в описание процесса установки, нормальный порядок установки находится во [тут](#).

ВНИМАНИЕ!!!!! Я публично извиняюсь перед Андреем Маркеловым (автор книги) за эти слова, и снимаю перед ним шляпу, так как его способ установки основных утилит для развертывания **openstack** является самым удобным в интернете, достаточно ввести 3 следующие команды после запуска ВМ и все остальные пакеты (сервисы **Openstack** и **RabbitMQ**) будут загружаться через обычный **yum** (таким образом настройка **RabbitMQ** начинается [здесь](#)):

```
$ sudo yum -y update  
$ sudo yum -y install epel-release  
$ sudo yum install -y https://www.rdoproject.org/repos/rdo-release.rpm
```

3.3. Установка **epel-release** (открытое бесплатное хранилище пакетов от Fedora, видимо нужно для RabbitMQ) и обновление компонентов системы.

```
$ sudo yum -y install epel-release  
$ sudo yum -y update
```

3.4. Установка **Erlang** - язык, на котором написан **RabbitMQ**. Для этого надо установить **wget**, так как он не предустановлен, после чего загрузить **rpm** репозиторий с помощью wget, потом добавить его с помощью **rpm -Uvh** (**U** - апгрейд пакета, **vh** - для статус бара и дополнительной информации), и в конце установить необходимые пакеты (**erlang**, **socat**, **logrotate**) с помощью yum, на сколько я понял без них **RabbitMQ** не заработает, как минимум без **erlang**.

```
$ sudo yum install wget  
$ sudo wget http://packages.erlang-solutions.com/erlang-solutions-1.0-1.noarch.rpm  
$ sudo rpm -Uvh erlang-solutions-1.0-1.noarch.rpm  
$ sudo yum -y install erlang socat logrotate
```

3.5. Установка **rabbitmq-server**. Для этого устанавливаем rpm пакет с **RabbitMQ** с помощью **wget**, потом добавляем ключ подписи с помощью **rpm** (без него вылезет ошибка), после чего устанавливаем **rabbitmq-server** с помощью **rpm**, запускаем его и настраиваем автозапуск.

```
$ sudo wget https://github.com/rabbitmq/rabbitmq-server/releases/download/v3.8.8/rabbitmq-server-3.8.8-  
1.el6.noarch.rpm  
$ sudo rpm --import https://www.rabbitmq.com/rabbitmq-signing-key-public.asc  
$ sudo rpm -Uvh rabbitmq-server-3.8.8-1.el6.noarch.rpm  
$ sudo systemctl start rabbitmq-server  
$ sudo systemctl enable rabbitmq-server
```

Понимаем, что **rabbitmq-server** не запускается, а точнее находится в состоянии activating пока не словит timeout, и решаем эту проблему путем настройки **firewall-cmd** и **SELinux**.

```
$ firewall-cmd --state  
running
```

```
$ sestatus  
SELinux status: enabled
```

Для этого освобождаем следующие [порты](#) (где-то говорится, что достаточно сделать публичным только базовый порт **RabbitMQ**, но на всякий и другие тоже опубликуем, хотя наверное можно просто вырубить фаерволл):

- **4369** — порт сервиса **epmd** (Erlang Port Mapper Daemon), служба обнаружения одноранговых узлов, используемая узлами RabbitMQ и инструментами CLI
- **25672** — порт сервера распространения **Erlang**, используется для связи между узлами и инструментами CLI
- **5671** — порт сервиса [**amqp**](#) (Advanced Message Queuing Protocol), тот самый протокол обмена сообщениями, который использует **RabbitMQ**
- **5672** — порт сервиса [**amqps**](#) (amqp protocol over TLS/SSL), из расшифровки и так понятно (как говорится на официальном сайте, **RabbitMQ** не работает если на машине нет OpenSSL 1.1, конфликты с Erlang 24 версии, но может быть все работает, я не проверял)
- **15672** — порт клиентов [**HTTP API**](#), [**пользовательского интерфейса управления**](#) и [**rabbitmqadmin**](#) без TLS/SSL, почему-то с TLS/SSL не публикуется
- **61613** — порт сервиса [**stomp**](#) (Simple Text Oriented Messaging Protocol), альтернатива AMQP для специфических случаев
- **61614** - порт сервиса [**stomps**](#) (stomp protocol over TLS/SSL), аналогично [**amqp**](#) и [**amqps**](#)
- **1883** — порт сервиса [**mqtt**](#) (Message Queue Telemetry Transport) альтернатива AMQP для специфических случаев
- **8883** — порт сервиса [**mqtts**](#) (mqtt protocol over TLS/SSL), аналогично [**amqp**](#) и [**amqps**](#)

```
$ sudo firewall-cmd --zone=public --permanent --add-port=4369/tcp
$ sudo firewall-cmd --zone=public --permanent --add-port=25672/tcp
$ sudo firewall-cmd --zone=public --permanent --add-port=5671-5672/tcp
$ sudo firewall-cmd --zone=public --permanent --add-port=15672/tcp
$ sudo firewall-cmd --zone=public --permanent --add-port=61613-61614/tcp
$ sudo firewall-cmd --zone=public --permanent --add-port=1883/tcp
$ sudo firewall-cmd --zone=public --permanent --add-port=8883/tcp
```

- **--zone=public** — публичный доступ к порту
- **--permanent** — останется после перезагрузки машины

Перезагрузка фаервола, порты добавились:

```
$ sudo firewall-cmd --reload
success
$ sudo firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: eth0
  sources:
    services: dhcpcv6-client ssh
    ports: 4369/tcp 25672/tcp 5672/tcp 15672/tcp 61613-61614/tcp 1883/tcp
            8883/tcp
    protocols:
    masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

Включаем **nis** (Network Information Service - распространяет карты имен, паролей и другую важную информацию компьютерам своего домена) в **SELinux**. Совершенно без понятия зачем, видимо как-то связано с работой **Erlang**, если это не сделать, то порт для **RabbitMQ** не выделится.

```
$ sudo setsebool -P nis_enabled 1
[ 3920.316824] SELinux: Converting 2243 SID table entries...
```

После чего пробуем различные фиксы из интернета (около 6), обращаемся к гадалке, от отчаяния пробуем оригинальный гайд по установке с официального сайта, и в конце концов понимаем, что ничего не работает. **RabbitMQ** мучает людей...

3.3. Итак вот оно — описание установки при помощи другого руководства. Первым делом меняем **hostname**, потому что **RabbitMQ** может не запустится, если **hostname** поменяется в будущем.

```
$ sudo hostnamectl set-hostname controller.test.local --static
$ hostname
controller.test.local
```

3.4. Далее как по [старым гайдам](#) устанавливаем **wget** и **epel-release**.

```
$ sudo yum -y install epel-release wget
$ sudo yum repolist
Загружены модули: fastestmirror
Loading mirror speeds from cached hostfile
epel/x86_64/metalink | 20 kB  00:00
* base: mirror.awanti.com
* epel: mirror.yandex.ru
* extras: mirror.awanti.com
* updates: mirror.awanti.com
epel | 4.7 kB  00:00
(1/3): epel/x86_64/group_gz | 98 kB  00:00
(2/3): epel/x86_64/updateinfo | 1.0 MB  00:00
(3/3): epel/x86_64/primary_db | 7.0 MB  00:00
Идентификатор репозитория репозиторий                                         состояние
base/7/x86_64          CentOS-7 - Base                                10 072
epel/x86_64             Extra Packages for Enterprise Linux 7 - x86_13 739
extras/7/x86_64         CentOS-7 - Extras                               515
updates/7/x86_64        CentOS-7 - Updates                             4 385
repolist: 28 711
```

3.5. Потом как по [старым гайдам](#) устанавливаем **Erlang**, но на этот раз без **socat** и **logrotate** (видимо не так они и нужны), а также понимаем, что в прошлый раз скачали старую версию (**erlang-solutions-1.0-1**), из-за чего могли и возникнуть трудности.

```
$ sudo wget https://packages.erlang-solutions.com/erlang-solutions-2.0-1.noarch.rpm  
$ sudo rpm -Uvh erlang-solutions-2.0-1.noarch.rpm  
$ sudo yum -y install erlang  
$ sudo erl  
Erlang/OTP 24 [erts-12.3.2.1] [source] [64-bit] [smp:2:2] [ds:2:2:10] [async-threads:1]
```

3.6. На этом шаге уже существенные отличия, а именно нужно создать и заполнить файл **/etc/yum.repos.d/rabbitmq.repo** (конфигурация репозитория), возможно в старых гайдах это производилось автоматически. **Nano** на **Centos** не предустановлено так что пользуемся **vi**.

```
$ sudo cat /etc/yum.repos.d/rabbitmq.repo  
[rabbitmq_rabbitmq-server]  
name=rabbitmq_rabbitmq-server  
baseurl=https://packagecloud.io/rabbitmq/rabbitmq-server/el/\$releasever/\$basearch  
repo_gpgcheck=1  
gpgcheck=0  
enabled=1  
gpgkey=https://packagecloud.io/rabbitmq/rabbitmq-server/gpgkey  
sslverify=1  
sslcacert=/etc/pki/tls/certs/ca-bundle.crt  
metadata_expire=300
```

Обновляем кэш репозиториев ут и уже можно наблюдать как все красиво:

```
$ sudo yum clean all  
$ sudo yum makecache  
$ sudo yum repolist  
Загружены модули: fastestmirror  
Loading mirror speeds from cached hostfile  
* base: mirror.docker.ru  
* epel: mirror.nsc.liu.se  
* extras: mirror.awanti.com  
* updates: mirror.docker.ru  
Идентификатор репозитория      репозиторий          состояние  
base/7/x86_64                    CentOS-7 - Base        10 072  
epel/7/x86_64                    Extra Packages for Enterprise Linux 13 739  
erlang-solutions/7/x86_64        Centos 7 - x86_64 - Erlang Solutions 6 179  
extras/7/x86_64                  CentOS-7 - Extras      515  
rabbitmq_rabbitmq-server/7/x86_64 rabbitmq_rabbitmq-server    98  
updates/7/x86_64                 CentOS-7 - Updates     4 385  
repolist: 34 988
```

3.7. Момент истины, загружаем **rabbitmq-server** и запускаем его. Загрузилось нормально ухх, включалось долго (чуть инфаркт не схватил), и..... (драматическая пауза) да Высшие Силы сжалились надо мной и все запустилось, ни фаерволл ни **SELinux** настраивать (пока) не пришлось.

```
$ sudo yum -y install rabbitmq-server  
$ sudo systemctl enable --now rabbitmq-server  
$ sudo systemctl status rabbitmq-server  
● rabbitmq-server.service - RabbitMQ broker  
  Loaded: loaded (/usr/lib/systemd/system/rabbitmq-server.service; enabled; vendor preset: disabled)  
  Active: active (running) since Ср 2022-11-23 16:47:36 MSK; 18s ago
```

Если что вот так выглядело описание установки из книги (если вы сюда телепортировались, то сначала телепортируйтесь [сюда](#)):

```
[root@controller ~]# yum -y install rabbitmq-server  
[root@controller ~]# systemctl start rabbitmq-server.service  
[root@controller ~]# systemctl enable rabbitmq-server.service
```

3.8. Постоянная рубрика «Удобные команды» на этот раз **RabbitMQ** (правда они вряд ли понадобятся, напрямую с **RabbitMQ** работает только **Openstack**, а нет понадобятся для настройки пользователя):

- **rabbitmqctl delete_user user** — удалить пользователя
- **rabbitmqctl change_password user strongpassword** — сменить пароль пользователя
- **rabbitmqctl add_vhost /my_vhost** — добавить [virtualhost](#)
- **rabbitmqctl list_vhosts** — список virtualhosts
- **rabbitmqctl delete_vhost /my_vhost** — удалить virtualhost
- **rabbitmqctl set_permissions -p /my_vhost user ".*" ".*" ":"** - дать разрешение пользователю внутри virtualhost
- **rabbitmqctl list_permissions -p /my_vhost** — список разрешений внутри virtualhost
- **rabbitmqctl list_user_permissions user** — список всех разрешений пользователя
- **rabbitmqctl clear_permissions -p /my_vhost user** — удалить разрешение пользователя внутри virtualhost
- **rabbitmqctl status** — статус сервиса rabbitmq-server
- **rabbitmqctl create_user user strongpassword** — создать пользователя
- **rabbitmqctl list_users** — список пользователей
- **rabbitmqctl set_user_tags user tag** — присвоит тэг пользователю

3.9. Оказывается я пропустил некоторые настройки из начала книги, а именно обновление всех установленных пакетов:

```
$ sudo yum -y update
```

Отключение network manager и редактирование файла **ifcfg-eth0** (поправить параметры **NM_CONTROLLED** и **ONBOOT**, а также проверить настройку ip-адреса и тд) в папке **/etc/sysconfig/network-scripts/**, так как **NetworkManager** будет мешать работе **Openstack Neutron**, судя по [документации RedHat](#):

```
$ systemctl stop NetworkManager.service  
$ systemctl disable NetworkManager.service
```

```
$ sudo cat /etc/sysconfig/network-scripts/ifcfg-eth0
TYPE="Ethernet"
PROXY_METHOD="none"
BROWSER_ONLY="no"
BOOTPROTO="none"
DEFROUTE="yes"
IPV4_FAILURE_FATAL="no"
IPV6INIT="yes"
IPV6_AUTOCONF="yes"
IPV6_DEFROUTE="yes"
IPV6_FAILURE_FATAL="no"
IPV6_ADDR_GEN_MODE="stable-privacy"
NAME="eth0"
UUID="d456563f-95f2-47f1-a0f7-2236248f3132"
DEVICE="eth0"
ONBOOT="yes"
IPADDR="192.168.122.200"
PREFIX="24"
GATEWAY="192.168.122.1"
DNS1="192.168.122.1"
DOMAIN="test.local"
NM_CONTROLLED=no
```

```
$ sudo systemctl start network.service
$ sudo systemctl enable network.service
```

Отключение фаервола, а я то мучался ([firewalld не используется даже когда хочется](#), вместо него используется **iptables**, будем их настраивать во время работы с сервисом **Neutron**):

```
$ systemctl stop firewalld.service
$ systemctl disable firewalld.service
```

Установка пакета **openstack-selinux** (`sudo yum -y install openstack-selinux`) или отключение **SELinux** (я решил отключить (изменить поле **SELINUX** в файле `/etc/sysconfig/selinux`, потом перезагрузиться), так как не особо разбираюсь в данной технологии, но может быть изучу, года наконец разверну кластер):

```
$ sudo cat /etc/sysconfig/selinux
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - No SELinux policy is loaded.
SELINUX=disabled
# SELINUXTYPE= can take one of three values:
#       targeted - Targeted processes are protected,
#       minimum - Modification of targeted policy. Only selected processes are protected.
#       mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

Проверка после перезагрузки:

```
$ sudo getenforce
Disabled
```

Установка **chrony**, как на хосте, у меня он уже был предустановлен и настроен (вроде бы устанавливал только на хост машину):

```
$ yum -y install chrony  
$ systemctl start chronyd  
$ systemctl enable chronyd
```

Установка **crudini** для редактирования конфигурационных файлов:

```
$ yum -y install crudini
```

Пример команды для редактирования конфига (не буду же я одну команду выносить в рубрику):

```
$ sudo crudini --set /путь/к/конфигурационному/файлу СЕКЦИЯ параметр значение
```

НО это все равно не меняет того, что способ установки **RabbitMQ** представленный в книге неполный, хотя и была показана установка репозитория **epel-release** (выполните две команды ниже и [назад](#)).

```
# yum -y install epel-release
```

Подключаем репозиторий дистрибутива RDO:

```
# yum install -y https://www.rdoproject.org/repos/rdo-release.rpm
```

3.10. Ну теперь можно заняться настройкой **RabbitMQ**. Для начала настроим аутентификацию, в руководствах говорится что есть два способа аутентификации: с использованием имени и пароля **SASL-аутентификации** (Simple Authentication and Security Layer), обеспечиваемой фреймворком **Erlang**, и при помощи сертификатов и SSL. Я решил (посмотрел в руководствах), что буду реализовывать первый способ + все сервисы будут работать под одним пользователем **RabbitMQ** (да я знаю, что так не очень безопасно, но я и фаерволл уже вырубил). Что касается **SASL-аутентификации** в **RabbitMQ**, то есть два метода: **PLAIN** и **AMQPLAIN** и дефолтный пользователь **guest**.

Создаем пользователя **openstack** (пароль - **openstack**) для настройки сервисов **Openstack** и добавляем ему права на настройку, чтение и запись:

```
$ sudo rabbitmqctl add_user openstack openstack  
$ sudo rabbitmqctl set_permissions openstack ".*" ".*" ".*"  
$ sudo rabbitmqctl list_users  
Listing users ...  
user    tags  
openstack  []  
guest   [administrator]
```

Для дальнейшего удобства активируем графическую консоль **RabbitMQ**, она будет работать на порту **15672**:

```
$ sudo /usr/lib/rabbitmq/bin/rabbitmq-plugins enable rabbitmq_management
$ sudo systemctl restart rabbitmq-server.service
$ sudo rabbitmqctl status
Listeners
Interface: [::], port: 15672, protocol: http, purpose: HTTP API
Interface: [::], port: 25672, protocol: clustering, purpose: inter-node and CLI tool communication
Interface: [::], port: 5672, protocol: amqp, purpose: AMQP 0-9-1 and AMQP 1.0
```

Для того чтобы получить доступ к в браузере обращаемся к **192.168.122.200:15672** (к моему большому удивлению проброс портов осуществлять не надо было). Для регистрации необходимо присвоить один из тэгов (**management**, **policymaker**, **monitoring**, **administrator**) созданному пользователю **openstack**, так как за пользователя **guest** зайти будет нельзя, из-за того что он может коннектиться **только через loopback интерфейс**. Я пока не знаю какой тег больше подойдет для пользователя **openstack**, когда узнаю нужный подчеркну выше (в этот web-интерфейс я больше ни разу не зашел). А пока по-быстрому создал пользователя **test** для проверки работоспособности web-интерфейса.

```
$ sudo rabbitmqctl add_user test test
$ sudo rabbitmqctl set_user_tags test administrator
$ sudo rabbitmqctl set_permissions -p / test ".*" ".*" ".*"
$ sudo rabbitmqctl list_users
Listing users ...
user    tags
test   [administrator]
openstack []
guest   [administrator]
```

Все нормально работает, главное выставить тэг:



4. Установка и настройка БД MariaDB и PyMySQL.

4.1. Итак мы наконец закончили с конфигурацией **RabbitMQ**, теперь можно приступить к конфигурирования БД для использования сервисами **Openstack**. Качаем **MariaDB** и клиентскую библиотеку **PyMySQL** с помощью одной команды!

```
$ sudo yum -y install mariadb-server python2-PyMySQL
```

4.2. Создаем файл **/etc/my.cnf.d/openstack.cnf** с конфигом **mysqld** и запускаем **MariaDB**:

```
$ sudo cat /etc/my.cnf.d/openstack.cnf
[mysqld]
bind-address = 192.168.122.200
default-storage-engine = innodb
innodb_file_per_table = on
max_connections = 4096
collation-server = utf8_general_ci
character-set-server = utf8
```

- bind-address = 192.168.122.200 — ip-адрес контроллера БД
- default-storage-engine = innodb - механизм хранения для таблиц по умолчанию, в данном случае innodb (данные хранятся в больших совместно используемых файлах).
- innodb file per table = on — для каждой таблицы будет создаваться новый файл
- max connections = 4096 — максимальное количество подключений
- collation-server = utf8_general_ci — тип сервера сравнения и сортировки строк, utf8_general_ci — дефолт для utf8.
- character-set-server = utf8 — тип набора символов на сервере

```
$ sudo systemctl enable mariadb.service
$ sudo systemctl start mariadb.service
$ sudo systemctl status mariadb.service
● mariadb.service - MariaDB database server
  Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled; vendor preset: disabled)
  Active: active (running) since Чт 2022-11-24 23:05:34 MSK; 7s ago
```

4.3. Запускаем супер-пупер мега скрипт [mysql_secure_installation](#), который поможет безопасно установить **MariaDB** (что в принципе видно из названия), во время скрипта можно:

- Установить пароль для учетных записей **root**.
- Удалить корневые учетные записи, доступные из-за пределов локального хоста.
- Удалить учетные записи анонимных пользователей.
- Удалить тестовую базу данных, к которой по умолчанию могут обращаться анонимные пользователи.

Я везде отвечал **yes** (возможно это была критическая ошибка (нет)).

```
$ sudo mysql_secure_installation
Enter current password for root (enter for none):
OK, successfully used password, moving on...

Set root password? [Y/n] y
New password:
Re-enter new password:
Password updated successfully!
Reloading privilege tables..
... Success!

Remove anonymous users? [Y/n] y
... Success!

Disallow root login remotely? [Y/n] y
... Success!

Remove test database and access to it? [Y/n] y
- Dropping test database...
... Success!
- Removing privileges on test database...
... Success!

Reload privilege tables now? [Y/n] y
... Success!

Cleaning up...

All done! If you've completed all of the above steps, your MariaDB
installation should now be secure.

Thanks for using MariaDB!
```

Проверить работу сервиса можно так:

```
$ sudo mysqladmin -uroot -popenstack status
Uptime: 754 Threads: 1 Questions: 26 Slow queries: 0 Opens: 1 Flush tables: 2 Open tables: 27 Queries per
second avg: 0.034
```

5. Установка и настройка сервиса идентификации Keystone

5.1. Теперь можно и потихоньку начать настраивать сервисы **Openstack**, все начинается в **Keystone**. Что такое **Keystone** — это сервис идентификации **Openstack**, представляет собой централизованный каталог пользователей и сервисов, к которым они имеют доступ. Если хотите еще одно синонимичное описание то вот: **Keystone** — единная система аутентификации и авторизации облачной операционной системы. Сервис поддерживает следующие типы аутентификации:

- Аутентификация по токенам
- Аутентификация при помощи пары имя пользователя/пароль
- **AWS**-совместимая аутентификация (Amazon Web Services)

Также **Keystone** хранит в себе список всех доступных сервисов в **Openstack** и реквизитов для обращения к их API (без привязки к пользователю), из чего следует,

что и поднимать его нужно первым, ведь потом каждый последующий сервис надо в нем регистрировать.

Что такое **Сервис**, **Пользователь** и **Токен** и так понятно, а вот **Проект** что-то новое — это объединение **Ресурсов** (виртуальные машины, образы и т. д.), придуман чтобы контролировать права пользователей, чтобы они могли видеть и работать с ВМ только внутри одного проекта (изначально думал, что это типа **Namespace**, но есть еще **Домен**). Также в **Openstack** как и в **Kubernetes** **Пользователи** напрямую не имеют доступ к **Ресурсам Проектов**, а получают его через так называемые **Роли** (понятно для чего так придумано, чтобы можно было создавать кастомные права доступа и быстро их назначать пользователям). Думали все, а нет, еще есть **Домены** — объединения **Проектов**, уже полные аналоги **Namespace**. Ладно по теории вроде бы все теперь качаем.

5.2. Качаем пакеты **openstack-keystone** (сам **Keystone**) [python-openstack-client](#) (удобный CLI для работы с **Openstack**) [httpd](#) (http сервер **Apache**) [mod_wsgi](#) (модуль http сервера для связи сервера и программой на python) с помощью **yum**. Проблема подкралась незаметно — пакеты **openstack-keystone** и **pythonopenstackclient** не скачались, но хоть фиксится просто во первых в файле **/etc/yum.repos.d/epel.repo** дизайблым все репозитории (**enabled=0**), а во вторых я дефис забыл в названии пакета **python-openstackclient**.

```
$ sudo yum -y install openstack-keystone python-openstack-client httpd mod_wsgi
```

```
$ sudo vi /etc/yum.repos.d/epel.repo
[epel]
name=Extra Packages for Enterprise Linux 7 - $basearch
# It is much more secure to use the metalink, but if you wish to use a local mirror
# place its address here.
#baseurl=http://download.example/pub/epel/7/$basearch
metalink=https://mirrors.fedoraproject.org/metalink?repo=epel-7&arch=$basearch&infra=$infra&content=$contentdir
failovermethod=priority
enabled=0
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-7
```

5.3. Как уже говорилось, для каждого сервиса **Openstack** необходима отдельная БД, которую надо создавать руками, если происходит установка каждого сервера отдельно, так что создаем **keystone** БД, также надо выдать привилегии, чтобы сервис мог нормально работать с БД.

```
$ sudo mysql -u root -p
Enter password:
MariaDB [(none)]> CREATE DATABASE keystone;
Query OK, 1 row affected (0.00 sec)
MariaDB [(none)]> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost' IDENTIFIED BY 'openstack';
Query OK, 0 rows affected (0.01 sec)
MariaDB [(none)]> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' IDENTIFIED BY 'openstack';
Query OK, 0 rows affected (0.00 sec)
MariaDB [(none)]> exit
```

5.4. В конфиг **Keystone** (**/etc/keystone/keystone.conf**) пишем путь к БД, можно с помощью **crudini**, можно с **vi** и тд.

```
$ sudo crudini --set /etc/keystone/keystone.conf database connection mysql+pymysql://keystone:openstack@controller.test.local/keystone
$ sudo cat /etc/keystone/keystone.conf
...
[database]
connection = mysql+pymysql://keystone:openstack@controller.test.local/keystone
```

5.5. Задаем формат и провайдер токенов генерируемых сервисом **Keystone**. Формат токенов на выбор:

- **UUID** - строка из 32 символов, которую удобно использовать в вызовах **OpenStack** API, например применяя команду **curl**. Круто — небольшой размер, не круто — токен не содержит информации, достаточной для того, чтобы произвести локальную авторизацию (сервисы **OpenStack** каждый раз должны отправлять токен сервису **Keystone**, для того чтобы получить информацию, какие операции разрешены с этим токеном).
- **PKI** - содержат всю необходимую для локальной авторизации информацию и, кроме того, содержат в себе цифровую подпись и информацию об устаревании, а значит сервисы **OpenStack** могут локально кэшировать эти токены. Круто — нет теперь дудоса токенами, не круто — токены большие (могут быть больше 8 Кб), и некоторые сервисы не поддерживают HTTP-заголовки такого размера, и еще как с **curl** работать, когда у тебя заголовок на 8 Кб.
- **PKIz** — пытались пофиксить проблемы **PKI**, но ничего не вышло, и они ушли в небытие.
- **Fernet** - небольшого размера (до 255 символов), но содержат достаточно информации для локальной авторизации. Их не требуется синхронизировать между регионами, для них не нужна база данных (токены без сохранения состояния), и процесс генерации их быстрее, чем в первых двух реализациях. Дополнительным плюсом будет отсутствие необходимости настройки memcached (можно не кэшировать, но по итогу все равно надо, если хочешь быть крутым). **Их и используем.**

Выставляем провайдера токенов в файле **/etc/keystone/keystone.conf** сами знаете как.

```
$ sudo crudini --set /etc/keystone/keystone.conf token provider fernet
$ sudo cat /etc/keystone/keystone.conf
...
[token]
provider = fernet
```

5.6. Инициализируем базу данных (скрипт **keystone-manage db_sync**) репозитории ключей **Fernet** (2 команды - «**keystone-manage fernet_setup**» и «**keystone-manage credential_setup**»).

```
$ sudo -s /bin/sh -c "keystone-manage db_sync" keystone
...
Database is synced successfully.
$ sudo keystone-manage fernet_setup --keystone-user keystone --keystone-group keystone
$ sudo keystone-manage credential_setup --keystoneuser keystone --keystone-group keystone
```

5.7. Конфигурируем доменное имя сервера **Apache httpd** - сервис идентификации для сетевого взаимодействия, путем редактирования файла **/etc/httpd/conf/httpd.conf** (без использования **cruduini**).

```
$ sudo vi /etc/httpd/conf/httpd.conf
...
ServerName controller.test.local
```

Также создаем и редактируем файл **/etc/httpd/conf.d/wsgi-keystone.conf** (это конфигурация модуля **http-сервера** для связи сервера и программой на **python**).

Ну что ж вот [описание полей](#):

- **Listen** — на каком порту работает
- **VirtualHost** — имя **virtualhost** (любое на таком-то порту)
- **WSGIaemonProcess** — создать отдельные процессы демона, которым будет делегирован запуск приложений WSGI.
- **WSGIProcessGroup** — в какой группе процессов будет выполняться приложение WSGI.
- **WSGIScriptAlias** — помечает каталог как содержащий сценарии WSGI
- **WSGIApplicationGroup** — какой группе приложений принадлежит приложение WSGI.
- **WSGIPassAuthorization** - передаются ли заголовки авторизации HTTP в приложение WSGI.
- **LimitRequestBody** - максимальное количество байт, которое может быть в запросе.
- **IfVersion** — если версия **Apache** больше/меньше
- **ErrorLogFormat** — формат лога ошибки
- **ErrorLog** — файл сохранения логов
- **CustomLog** — файл сохранения логов доп хоста
- **Directory** — местонахождение проекта
- **Require all granted** — нет IP-адресов, заблокированных для доступа к сервису (новая версия)
- **Order allow,deny** — [настройка](#) системы контроля доступа
- **Allow from all** — нет IP-адресов, заблокированных для доступа к сервису (старая версия)
- **Alias** — сопоставление URL-адреса с путем файловой системы
- **Location** — внутри настройки управления доступа к URL

- **SetHandler** — просмотр определенного хендлера
- **Options** — разрешенные особенности сервера

```
$ sudo vi /etc/httpd/conf.d/wsgi-keystone.conf
Listen 5000
Listen 35357

<VirtualHost *:5000>
    WSGIDaemonProcess keystone-public processes=5 threads=1 user=key
stone group=keystone display-name=%{GROUP}
    WSGIProcessGroup keystone-public
    WSGIScriptAlias / /usr/bin/keystone-wsgi-public
    WSGIApplicationGroup %{GLOBAL}
    WSGIPassAuthorization On
    LimitRequestBody 114688
    <IfVersion >= 2.4>
        ErrorLogFormat "%{cu}t %M"
    </IfVersion>
    ErrorLog /var/log/httpd/keystone.log
    CustomLog /var/log/httpd/keystone_access.log combined

    <Directory /usr/bin>
        <IfVersion >= 2.4>
            Require all granted
        </IfVersion>
        <IfVersion < 2.4>
            Order allow,deny
            Allow from all
        </IfVersion>
    </Directory>
</VirtualHost>

<VirtualHost *:35357>
    WSGIDaemonProcess keystone-admin processes=5 threads=1 user=key
stone group=keystone display-name=%{GROUP}
    WSGIProcessGroup keystone-admin
    WSGIScriptAlias / /usr/bin/keystone-wsgi-admin
    WSGIApplicationGroup %{GLOBAL}
    WSGIPassAuthorization On
    LimitRequestBody 114688
    <IfVersion >= 2.4>
        ErrorLogFormat "%{cu}t %M"
    </IfVersion>
    ErrorLog /var/log/httpd/keystone.log
    CustomLog /var/log/httpd/keystone_access.log combined

    <Directory /usr/bin>
        <IfVersion >= 2.4>
            Require all granted
        </IfVersion>
        <IfVersion < 2.4>
            Order allow,deny
            Allow from all
        </IfVersion>
    </Directory>
</VirtualHost>

Alias /identity /usr/bin/keystone-wsgi-public
<Location /identity>
    SetHandler wsgi-script
    Options +ExecCGI

    WSGIProcessGroup keystone-public
    WSGIApplicationGroup %{GLOBAL}
    WSGIPassAuthorization On
</Location>

Alias /identity_admin /usr/bin/keystone-wsgi-admin
<Location /identity_admin>
    SetHandler wsgi-script
    Options +ExecCGI

    WSGIProcessGroup keystone-admin
    WSGIApplicationGroup %{GLOBAL}
    WSGIPassAuthorization On
</Location>
```

Наконец-то запускаем веб-сервер:

```
$ sudo systemctl enable httpd.service
$ sudo systemctl start httpd.service
$ sudo systemctl status httpd.service
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset: disabled)
   Active: active (running) since Сб 2022-11-26 16:53:56 MSK; 5s ago
```

5.8. Теперь инициализируем **Keystone**. Опять же есть два варианта инициализации:

- Использовать команду «**keystone-manage bootstrap**», которая выполнит инициализацию за нас (рекомендовано разработчиками, после того как попробовал второй способ использовал этот):

Данная команда создаст за нас записи о сервисе Keystone, точки входа (**admin**, **internal** и **public**) в сервис Keystone и регион в котором расположен сервис Keystone, а также **default** домен, проект **admin**, пользователь **admin** и роль **admin** и соединить их:

```
$ sudo keystone-manage bootstrap --bootstrap-password openstack \
--bootstrap-admin-url http://controller.test.local:35357/v3/ \
--bootstrap-internal-url http://controller.test.local:5000/v3/ \
--bootstrap-public-url http://controller.test.local:5000/v3/ \
--bootstrap-region-id RegionOne
```

Измененный скрипт (создаем сами, **PS1** — для красивого названия в консоли):

```
$ sudo vi keystonerc_admin
unset OS_USERNAME OS_PASSWORD OS_PROJECT_NAME OS_USER_DOMAIN_NAME OS_PROJECT_DOMAIN_NAME OS_AUTH_URL OS_IDENTITY_API_VERSION PS1
export OS_USERNAME=admin
export OS_PASSWORD=openstack
export OS_PROJECT_NAME=admin
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_DOMAIN_NAME=Default
export OS_AUTH_URL=http://controller.test.local:35357/v3
export OS_IDENTITY_API_VERSION=3
export PS1='[\u@\h \W(Openstack_Admin)]\$ '
```

Созданные ресурсы:

```
$ sudo openstack role list
+-----+-----+
| ID | Name |
+-----+-----+
| 16c00a6cab0e4c5b8c7f12ce558de999 | member |
| 444afaa061f043ceadb4ede468800a4b | admin |
| c831efc6458d4b25a3de990e4092f9fe | reader |
+-----+-----+
$ sudo openstack user list
+-----+-----+
| ID | Name |
+-----+-----+
| 549949769da94550a63fa6667741afb4 | admin |
+-----+-----+
```

```
$ sudo openstack domain list
+-----+-----+-----+
| ID      | Name     | Enabled | Description      |
+-----+-----+-----+
| default | Default | True    | The default domain |
+-----+-----+-----+
$ sudo openstack project list
+-----+-----+
| ID                         | Name   |
+-----+-----+
| 33ad52bb25284ad8be4b8681b8fe0063 | admin  |
+-----+-----+
$ sudo openstack catalog list
+-----+-----+
| Name      | Type     | Endpoints          |
+-----+-----+
| keystone  | identity | RegionOne
|           |           |       admin: http://controller.test.local:35357/v3/
|           |           |       RegionOne
|           |           |       public: http://controller.test.local:5000/v3/
|           |           |       RegionOne
|           |           |       internal: http://controller.test.local:5000/v3/
+-----+-----+
```

- Воспользоваться авторизационным токеном (общий секрет между **Keystone** и другими сервисами, а также вход в админку без пароля), долго, но зато идеально для понимания процессов (мой выбор, который был изменен на первый способ, так как этот способ был [вырезан](#)):

Для начала надо сгенерировать токен с помощью **OpenSSL** и поместить его в файл конфига **Keystone**:

```
$ sudo export ADM_TOKEN=$(openssl rand -hex 10)
$ sudo crudini --set /etc/keystone/keystone.conf DEFAULT admin_token $ADM_TOKEN
$ sudo cat /etc/keystone/keystone.conf
[DEFAULT]
admin_token = 15044d0ebc57046fac08
...
```

Немножко займемся оптимизацией и напишем скрипт для задания значений переменным окружения (**OS_TOKEN** — наш токен, **OS_URL** — URL **Keystone**, **OS_IDENTITY_API_VERSION** — версия API **Keystone**)

```
$ sudo vi keystonerc_adm
unset OS_USERNAME OS_TENANT_NAME OS_PASSWORD OS_AUTH_URL OS_TOKEN OS_URL
export OS_TOKEN=15044d0ebc57046fac08
export OS_URL=http://controller.test.local:35357/v3
export OS_IDENTITY_API_VERSION=3
$ sudo source keystonerc_adm
```

После попытки отправки запроса на сервер, было получено много различных ошибок.

5.9. После того как помучились с инициализацией создадим непrivилегированного пользователя и все что для него надо (admin пользователь был создан автоматически).

Создаем проект **demo**:

```
$ sudo openstack project create --domain default --description "Demo Tenant" demo
+-----+
| Field      | Value
+-----+
| description | Demo Tenant
| domain_id   | default
| enabled     | True
| id          | e420dacf94514ed587d878bf32cabf46
| is_domain   | False
| name        | demo
| options     | {}
| parent_id   | default
| tags         | []
+-----+
```

Создаем пользователя **demo** с собственным **email**:

```
$ sudo openstack user create --domain default --project demo --email user@controller.test.local --password openstack demo
+-----+
| Field           | Value
+-----+
| default_project_id | e420dacf94514ed587d878bf32cabf46
| domain_id       | default
| email           | user@controller.test.local
| enabled          | True
| id              | b1457723dc214b9aa669676043c7c7f2
| name             | demo
| options          | {}
| password_expires_at | None
+-----+
```

Создаем роль **user**:

```
$ sudo openstack role create user
+-----+
| Field      | Value
+-----+
| description | None
| domain_id   | None
| id          | ef4065a71ca94f76b20043ed3ef9d3dd
| name        | user
| options     | {}
+-----+
```

Добавляем роль **user** в проекте **demo** пользователю **demo**:

Role	User	Group	Project
444afaa061f043ceadb4ede468800a4b	549949769da94550a63fa6667741afb4		33ad52bb25284ad8be4b8681b8fe0063
ef4065a71ca94f76b20043ed3ef9d3dd	b1457723dc214b9aa669676043c7c7f2		e420dacf94514ed587d878bf32cabf46
444afaa061f043ceadb4ede468800a4b	549949769da94550a63fa6667741afb4		

Скрипт для входа за пользователя **demo (kestonerc_usr)**:

```
$ sudo vi kestonerc_usr
unset OS_USERNAME OS_PASSWORD OS_PROJECT_NAME OS_USER_DOMAIN_NAME OS_PROJECT_DOMAIN_NAME OS_AUTH_URL OS_IDENTITY_API_VERSION PS1
export OS_USERNAME=demo
export OS_PASSWORD=openstack
export OS_PROJECT_NAME=demo
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_DOMAIN_NAME=Default
export OS_AUTH_URL=http://controller.test.local:5000/v3
export OS_IDENTITY_API_VERSION=3
export PS1='[\u@\h \W(Openstack_Demo)]\$ '
```

5.10. Создаем проект **service** для всех сервисов Openstack:

Field		Value
description		Service Project
domain_id	default	
enabled	True	
id	9bc0bc30645844cd9c966536feb1df8	
is_domain	False	
name	service	
options	{}	
parent_id	default	
tags	[]	

С Keystone пока закончили, теперь можно перечислить полезные команды **openstack cli** (сюда будут добавляться все команды использованные в процессе конфигурирования кластера):

- **openstack user list** — список пользователей
- **openstack role list** — список ролей
- **openstack role assignment list** — список назначения ролей
- **openstack domain list** — список доменов
- **openstack volume list** — список томов
- **openstack stack list** — список стеков
- **openstack stack event list имя_стека** — список событий при создании стека
- **openstack stack resource list имя_стека** — список ресурсов стека
- **openstack volume backup list** — список бэкапов томов

- **openstack server add volume имя_ВМ имя_тома** — подключение тома к ВМ
- **openstack server remove volume имя_ВМ имя_тома** — отключение тома от ВМ
- **openstack service list** — список сервисов
- **openstack server list** — список ВМ
- **openstack metric resource-type list** — список типов ресурсов метрик
- **openstack metric resource list** — список ресурсов метрик
- **openstack metric resource show имя** — список метрик связанных с ресурсом
- **openstack metric measures show --resource-id id_ресурса имя_метрики** — список значений определенной метрики ресурса
- **openstack alarm list** — список триггеров
- **openstack orchestration service list** — список сервисов оркестрации
- **openstack backup list** — список резервных копий
- **openstack volume type list** — список типов томов (зашифрованный или нет)
- **openstack security group list** — список групп безопасности
- **openstack aggregate list** — список агрегаторов узлов
- **openstack availability zone list** — список зон доступности
- **openstack security group rule list имя_группы** — список правил брандмауэра для группы безопасности
- **openstack security group rule create --protocol протокол --dst-port номер_порта имя_группы** — добавление правила брандмауэра в группу безопасности
- **openstack hypervisor list** — список гипервизоров
- **openstack hypervisor stats show** — статистика по гипервизорам
- **openstack usage list** — использование ресурсов объектами
- **openstack project list** — список проектов
- **openstack quota list** — список квот на ресурсы
- **openstack catalog list** — список каталогов
- **openstack network agent list** — список сетевых агентов (dhcp, open vswitch и тд)
- **openstack compute service list** — список служб Nova
- **openstack host list** — список узлов
- **openstack floating ip list** — список плавающих ip
- **openstack floating ip create имя_сети** — создание плавающего ip из сети

- **openstack server add floating ip имя_ВМ ip_адрес** — присоединение плавающего Ip к ВМ
- **openstack hypervisor show имя_узла** — информация о гипервизоре узла
- **openstack endpoint list** — список конечных точек
- **openstack image list** — список образов
- **openstack image save имя_образа > путь_куда_сохранять** — скачать образ на локальную машину
- **openstack image create имя_ВМ имя_образа** — создать образ из ВМ в Openstack
- **openstack volume snapshot create --volume имя_тома имя_снимка** — создать снимок тома
- **openstack volume backup create имя_тома** — создать бэкап тома
- **openstack volume backup restore id_бэкапа имя_тома** — откатить том к бэкапу
- **openstack console url show имя_вм** — показать URL по которому можно подключится к ВМ через noVNC
- **openstack тип show имя_объекта** — информация о конкретном объекте
- **openstack тип delete имя_объекта** — удаление объекта
- **openstack service create --name имя_сервиса --description "описание" тип_сервиса** — создание сервиса
- **openstack endpoint create тип_точки тип_интерфейса url --region регион** — создание конечной точки
- **openstack domain create --description "описание" имя_домена** — создание домена
- **openstack project create --domain имя_домена --description "описание" имя_проекта** — создание проекта
- **openstack user create --domain имя_домена --email почта_пользователя --password пароль_пользователя имя_пользователя** — создание пользователя
- **openstack role create имя_роли** — создание роли
- **openstack role add --project имя_проект --user имя_пользователя имя_роли** — создание назначения роли (привязка роли к пользователю)
- **openstack тип set --имя_поля значение_поля имя_объекта** — изменение полей объекта
- **openstack тип unset --имя_поля значение_поля имя_объекта** — удаление поля объекта

- **openstack console log show имя_ВМ** — показать логи ВМ
- **openstack --debug** — полезный флаг для просмотра вызовов к Openstack API
- **openstack token issue** — запрос токена для аутентификации
- **openstack keypair create имя_пары > файл_сохранения_пары** — создание пары ssh ключей
- **source kestonerc_adm** — запуск скриптов (в моем случае задача глобальный переменных для авторизации под ролью **admin**)
- **env** — просмотр переменных окружения
- **/etc/имя_сервиса/имя_сервиса(или)службы.conf** — файлы конфигов
- **/var/log/имя_сервиса/имя_сервиса-имя_службы.log** — файлы логов (некоторые тут **/var/log/httpd/***)

6. Установка и настройка сервиса хранения образов Glance

Перед началом хочется отметить, что обращения к **Openstack** API начали проходить достаточно медленно, после перезагрузки машины, возможно я дал ВМ слишком мало системных ресурсов, а может так и должно быть, вобщем пугаться долгих ответов сервера не надо, ниче не умерло.

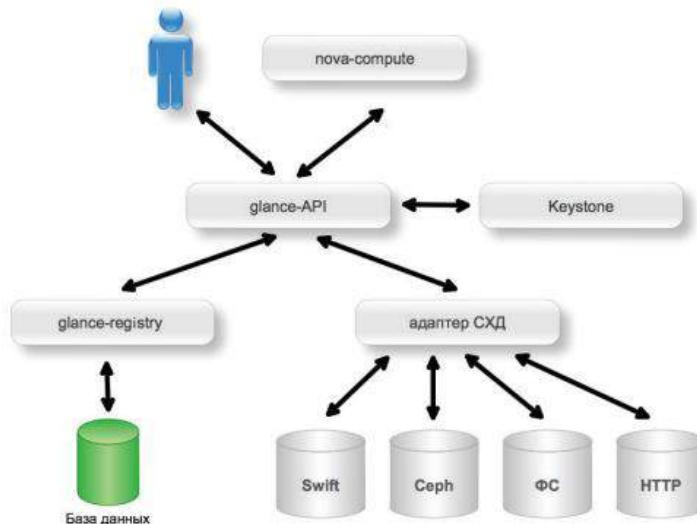
6.1. Что ж для чего нам нужен **Glance**? **Glance** - ведет каталог, регистрация рует и доставляет образы виртуальных машин (как известно образ представляет собой шаблон для ВМ, может быть просто ОС, а может и содержать предустановленные пакеты). Но **Glance** самостоятельно не хранит образы, а использует для этого систему хранения данных (**Swift**, **Ceph** или просто **локально** на узле), информация же о размере, формате, имени образа и т. д. хранится в БД. Поддерживаются следующие форматы образов, еще есть образы контейнеров, но это уже другая история:

- **vhd** (Virtual Hard Disk) — виртуальный жесткий диск от Microsoft
- **vmdk** (Virtual Machine Disk) — диск виртуальной машины от VMware
- **vdi** (Virtual Disk Image) — образ виртуального диска от VirtualBox(Oracle)
- **iso** (International Organization for Standardization) — образ оптического диска от International Organization for Standardization
- **qcow2** (QEMU Copy On Write 2) — образ QEMU
- **ami** — образ Amazon Machine
- **ari** — образ Amazon Ramdisk
- **aki** — образ Amazon Kernel
- **vhdx** — улучшенная версия vhd
- **raw** — диск неструктурированного формата RAW

Стоит отметить, что данный сервис состоит из двух служб:

- **glance-api** – предоставляет доступ к REST API сервиса образов для поиска, хранения и получения образов;
- **glance-registry** – хранит, обрабатывает и предоставляет информацию. Непосредственно пользователи не взаимодействуют с этим сервисом, только сервисы **Glance**.

Вот красавая картинка компонентов **Glance**:



Что происходит когда мы отправляем запрос на создание ВМ — **Nova** (еще не созданный сервис по управлению виртуальными машинами и сетью) отправляет GET-запрос по адресу <http://путь-к-сервису-glance/images/идентификатор-образа>. В случае если образ найден, то glance-api возвращает URL, ссылающийся на образ. **Nova** передает ссылку драйверу гипервизора, который напрямую скачивает образ и запускает ВМ.

6.2. Теперь начинается практика, первым делом добавляем все необходимые компоненты в наш **Keystone**:

Создаем пользователя **glance**:

```
$ sudo openstack user create --domain default --password openstack glance
+-----+
| Field      | Value   |
+-----+
| domain_id  | default |
| enabled     | True    |
| id          | cc9df70e640e4c3cb8b3d85205162470 |
| name        | glance  |
| options     | {}      |
| password_expires_at | None   |
+-----+
```

Присваиваем пользователю **glance** роль **admin** в проекте **service**:

```
$ sudo openstack role add --project service --user glance admin
```

Создаем сервис **glance**:

```
$ sudo openstack service create --name glance --description "OpenStack Image service" image
+-----+
| Field      | Value
+-----+
| description | OpenStack Image service
| enabled     | True
| id          | 5df89ee12c9c4518b5f82cb1d018b3ca
| name        | glance
| type        | image
+-----+
```

Создадим точки входа для сервиса **glance**:

```
$ sudo openstack endpoint create --region RegionOne image public http://controller.test.local:9292
+-----+
| Field      | Value
+-----+
| enabled    | True
| id         | 4a25921e5925468e8ac9e649f2cfa398
| interface   | public
| region     | RegionOne
| region_id  | RegionOne
| service_id | 5df89ee12c9c4518b5f82cb1d018b3ca
| service_name| glance
| service_type| image
| url        | http://controller.test.local:9292
+-----+
$ sudo openstack endpoint create --region RegionOne image internal http://controller.test.local:9292
+-----+
| Field      | Value
+-----+
| enabled    | True
| id         | 4b8c00a3ef2e414d91472e6acc0f95f0
| interface   | internal
| region     | RegionOne
| region_id  | RegionOne
| service_id | 5df89ee12c9c4518b5f82cb1d018b3ca
| service_name| glance
| service_type| image
| url        | http://controller.test.local:9292
+-----+
$ sudo openstack endpoint create --region RegionOne image admin http://controller.test.local:9292
+-----+
| Field      | Value
+-----+
| enabled    | True
| id         | 26cb58f242404d18b4a72695c6daa95d
| interface   | admin
| region     | RegionOne
| region_id  | RegionOne
| service_id | 5df89ee12c9c4518b5f82cb1d018b3ca
| service_name| glance
| service_type| image
| url        | http://controller.test.local:9292
+-----+
```

6.3. Теперь устанавливаем сервис **Glance** с помощью **yum**:

```
$ sudo yum -y install openstack-glance
```

6.4. Создаем БД **glance** и выдаем на нее права:

```
$ sudo mysql -u root -p
Enter password:
MariaDB [(none)]> CREATE DATABASE glance;
Query OK, 1 row affected (0.00 sec)
MariaDB [(none)]> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost' IDENTIFIED BY 'glance';
Query OK, 0 rows affected (0.00 sec)
MariaDB [(none)]> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'%' IDENTIFIED BY 'glance';
Query OK, 0 rows affected (0.00 sec)
```

6.5. Прописываем строку подключения к базе данных в файлах конфигов служб **glance-api** (*/etc/glance/glance-api.conf*) и **glance-registry** (*/etc/glance/glance-registry.conf*):

```
$ sudo crudini --set /etc/glance/glance-registry.conf database connection mysql+pymysql://glance:glance@controller.test.local/glance
$ sudo cat /etc/glance/glance-registry.conf
...
[database]
connection = mysql+pymysql://glance:glance@controller.test.local/glance
$ sudo crudini --set /etc/glance/glance-api.conf database connection mysql+pymysql://glance:glance@controller.test.local/glance
$ sudo cat /etc/glance/glance-api.conf
...
[database]
connection = mysql+pymysql://glance:glance@controller.test.local/glance
```

6.6. Завершаем настройку БД **glance** с помощью скрипта (**db_sync**):

```
$ sudo -s /bin/sh -c "glance-manage db_sync" glance
...
Database is synced successfully.
```

6.7. Все что касается БД **glance** настроено, теперь настроим сам службу **glance-api** через конфиг (*/etc/glance/glance-api.conf*):

Настраиваем тип сервис аутентификации:

```
$ sudo crudini --set /etc/glance/glance-api.conf paste_deploy flavor keystone
```

Настраиваем разрешенный ip отправителя запросов к сервису **Glance**:

```
$ sudo crudini --set /etc/cinder/cinder.conf keystone_auth_token www_authenticate_uri http://controller.test.local:5000
```

Настраиваем URL аутентификации:

```
$ sudo crudini --set /etc/glance/glance-api.conf keystone_auth_token auth_url http://controller.test.local:35357
```

Настраиваем тип авторизации:

```
$ sudo crudini --set /etc/glance/glance-api.conf keystone_auth_token auth_type password
```

Настраиваем имя домена в котором находится проект **glance**:

```
$ sudo crudini --set /etc/glance/glance-api.conf keystone_auth_token project_domain_name default
```

Настраиваем имя домена в котором находится пользователь **glance**:

```
$ sudo crudini --set /etc/glance/glance-api.conf keystone_auth_token user_domain_name default
```

Настраиваем имя проекта в котором находится пользователь **glance**:

```
$ sudo crudini --set /etc/glance/glance-api.conf keystone_auth_token project_name service
```

Настраиваем имя пользователя:

```
$ sudo crudini --set /etc/glance/glance-api.conf keystone_auth_token username glance
```

Настраиваем пароль пользователя **glance**:

```
$ sudo crudini --set /etc/glance/glance-api.conf keystone_auth_token password openstack
```

Настраиваем тип хранения образов (локальная файловая система):

```
$ sudo crudini --set /etc/glance/glance-api.conf glance_store default_store file
```

Настраиваем путь к папке, где будут хранится образы:

```
$ sudo crudini --set /etc/glance/glance-api.conf glance_store filesystem_store_datadir /var/lib/glance/images/
```

Также стоит отметить, что можно добавить несколько хранилищ образов с приоритетом (больше цифра - больше приоритет сохранения образа) по использованию, но я так не делал:

```
$ sudo crudini --set /etc/glance/glance-api.conf glance_store file system_store_datadirs /var/lib/glance/images/ mountA/:10  
$ sudo crudini --set /etc/glance/glance-api.conf glance_store file system_store_datadirs /var/lib/glance/images/ mountB/:20
```

Поля файла **/etc/glance/glance-api.conf** после исполнения команд (легко найти через **vi**, потом /текст, который надо найти):

```
$sudo cat /etc/glance/glance-api.conf
...
[glance_store]
default_store = file
filesystem_store_datadir = /var/lib/glance/images/
...
[keystone_auth_token]
www_authenticate_uri = http://controller.test.local:5000
auth_url = http://controller.test.local:35357
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = glance
password = openstack
...
[paste_deploy]
flavor = keystone
```

6.8. Теперь настроим сам службу **glance-registry** через конфиг (**/etc/glance/glance-registry.conf**), те же самые названия полей, кроме секции **glance_store**:

```
$ sudo crudini --set /etc/cinder/cinder.conf paste_deploy flavor keystone
$ sudo crudini --set /etc/cinder/cinder.conf keystone_auth_token project_name service
$ sudo crudini --set /etc/cinder/cinder.conf keystone_auth_token user_domain_name default
$ sudo crudini --set /etc/cinder/cinder.conf keystone_auth_token project_domain_name default
$ sudo crudini --set /etc/cinder/cinder.conf keystone_auth_token auth_type password
$ sudo crudini --set /etc/cinder/cinder.conf keystone_auth_token username glance
$ sudo crudini --set /etc/cinder/cinder.conf keystone_auth_token password openstack
$ sudo crudini --set /etc/cinder/cinder.conf keystone_auth_token www_authenticate_uri http://controller.test.local:5000
$ sudo crudini --set /etc/cinder/cinder.conf keystone_auth_token auth_url http://controller.te st.local:35357
```

```
$sudo cat /etc/glance/glance-registry.conf
...
[keystone_auth_token]
www_authenticate_uri = http://controller.test.local:5000
auth_url = http://controller.test.local:35357
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = glance
password = openstack
...
[paste_deploy]
flavor = keystone
```

6.9. Настраиваем параметры подключения к сервису **RabbitMQ** для службы **glance-api** (**/etc/glance/glance-api.conf**):

Настраиваем пароль пользователя **RabbitMQ**:

```
$ sudo crudini --set /etc/glance/glance-api.conf DEFAULT rabbit_password openstack
```

Настраиваем имя пользователя **RabbitMQ**:

```
$ sudo crudini --set /etc/glance/glance-api.conf DEFAULT rabbit_userid openstack
```

Настраиваем имя хоста **RabbitMQ**:

```
$ sudo crudini --set /etc/glance/glance-api.conf DEFAULT rabbit_host controller.test.local
```

```
$ sudo cat /etc/glance/glance-api.conf  
[DEFAULT]  
rabbit_password = openstack  
rabbit_userid = openstack  
rabbit_host = controller.test.local
```

6.10. Запускаем сконфигурированные службы **glance-api** и **glance-registry**:

```
$ sudo systemctl start openstack-glance-registry  
$ sudo systemctl enable openstack-glance-registry  
$ sudo systemctl status openstack-glance-registry  
● openstack-glance-registry.service - OpenStack Image Service (code-named Glance) Registry server  
  Loaded: loaded (/usr/lib/systemd/system/openstack-glance-registry.service; enabled; vendor preset: disabled)  
  Active: active (running) since Вс 2022-11-27 20:39:26 MSK; 22s ago
```

Папку для хранения образов нужно создать, а то **glance-api** не запустится (словил прикол, что у **glance-api** не было разрешений на файл, в котором хранятся логи - **/var/log/glance/api.log**). Решение прикола с разрешениями:

```
$ sudo chmod 777 /var/lib/glance/images/  
$ sudo chmod 777 /var/log/glance/api.log
```

```
$ sudo mkdir /var/lib/glance/images  
$ sudo systemctl start openstack-glance-api  
$ sudo systemctl enable openstack-glance-api  
$ sudo systemctl status openstack-glance-api  
● openstack-glance-api.service - OpenStack Image Service (code-named Glance) API server  
  Loaded: loaded (/usr/lib/systemd/system/openstack-glance-api.service; enabled; vendor preset: disabled)  
  Active: active (running) since Вс 2022-11-27 20:42:49 MSK; 98ms ago
```

6.11. Итак мы все настроили и запустили, теперь можно затестить

Для начала скачаем образ **CirrOS** – минималистская операционная система с помощью **wget**:

```
$ sudo wget -P /tmp http://download.cirros-cloud.net/0.4.0/cirros-0.4.0-x86_64-disk.img
```

Необходимо также установить **qemu-img**, а потом посмотреть информацию о скачанном образе:

```
$ sudo yum -y install qemu-img
$ sudo qemu-img info /tmp/cirros-0.4.0-x86_64-disk.img
image: /tmp/cirros-0.4.0-x86_64-disk.img
file format: qcow2
virtual size: 44M (46137344 bytes)
disk size: 12M
cluster_size: 65536
Format specific information:
    compat: 1.1
    lazy refcounts: false
    refcount bits: 16
    corrupt: false
```

Какую информацию qemu-img:

- **file format** – формат диска
- **virtual size** – размер диска виртуальной машины
- **disk size** – действительный размер файла
- **cluster_size** – размер блока (кластера) qcow
- **format specific information** – специфичная для формата информация (**compat** — версия QEMU, **lazy refcounts** — если on, то отключен ввод вывод метаданных, **refcount bits** — размер refcount table (но я не уверен), **corrupt** — искаженные данные или нет)

Далее надо скачать какую-нибудь утилиту автоматизированного создания образов, я выбрал **Oz**, которая использует заранее подготовленные файлы ответов неинтерактивной установки операционной системы. Если во время установки Keystone, были ошибки как и у меня и поэтому пришлось отключить epel repo, то его (и только его) надо опять [включить](#).

```
$ sudo yum -y install oz
```

Проверяем, что сеть libvird (перед этим libvird надо врубить), используемая по умолчанию с помощью команды **virsh net-list**, если нет, то определяем ее и задаем автозапуск сети:

Врубаем **libvird**:

```
$ sudo systemctl start libvird
$ sudo systemctl enable libvird
$ sudo systemctl status libvird
● libvird.service - Virtualization daemon
   Loaded: loaded (/usr/lib/systemd/system/libvird.service; enabled; vendor preset: enabled)
     Active: active (running) since Вс 2022-11-27 21:30:34 MSK; 18s ago
```

У меня после запуска **libvird** дефолтная сеть задалась автоматически:

\$ sudo virsh net-list	Имя	Статус	Автозапуск	Persistent
	default	активен	yes	yes

Если у вас команда **virsh net-list** вывела что-то другое, то выполните две команды **net-define** (инициализация сети) и **net-autostart** (автостарт сети):

```
$ sudo virsh net-define /usr/share/libvirt/networks/default.xml
Network default defined from /usr/share/libvirt/networks/default.xml
$ sudo virsh net-autostart default
```

6.12. Теперь, после того как скачали **oz** и настроили дефолтную сеть, можно начать настраивать **oz**.

Сначала зададим в качестве типа образа формат **qcow2** в файле конфигурации **oz** (**/etc/oz/ oz.cfg**):

```
$ sudo crudini --set /etc/oz/oz.cfg libvirt image_type qcow2
$ sudo cat /etc/oz/oz.cfg
...
[libvirt]
...
image_type = qcow2
```

Теперь выбираем TDL шаблон для создания образа, примеры TDL шаблонов находятся в папке **/usr/share/doc/oz-0.15.0/examples**. Я списал самый простой TDL шаблон, а именно тот в котором определяются только путь к дистрибутиву (**<install>**) и пароль пользователя root (**<rootpw>**):

```
$ sudo cat my_template.tdl
<template>
<name>CentOS-7</name>
<os>
<name>CentOS-7</name>
<version>1</version>
<arch>x86_64</arch>
<rootpw>openstack</rootpw>
<install type='url'>
<url>http://centos-mirror.rbc.ru/pub/centos/7/os/x86_64/</url>
</install>
</os>
<disk>
<size>1</size>
</disk>
</template>
```

Полезные секции кроме **<install>** и **<rootpw>**:

- **<packages>** - установка пакетов
- **<repositories>** - добавление репозиториев
- **<files>** - создание файлов
- **<commands>** - выполнение команд

После создание TDL шаблона можно создать образ с помощью oz-install (**-t** — через сколько секунд инсталлятор должен прервать установку, **-d** — показывает уровень сообщений об ошибках (**2** — норм, **3** — подробно)). Создается очень долго:

```
$ sudo oz-install -d 3 -t 4000 my_template.tdl
```

Образ должен появится в папке **/var/lib/libvirt/images**, но он весит 11GB, так что я его так и не скачал (диск на виртуалке 11 GB), дальше если хотите подготовить образ к использованию в **Openstack** надо с помощью **virt-sysprep** (надо скачать) убрать специфичную для конкретного экземпляра машины информацию, потом уменьшить размер образа (превратить в тонкий диск) при помощи **virt-sparsify**. Если надо отредактировать уже готовый образ то можно скачать **questfish**:

```
$ sudo virt-sysprep -a /var/lib/libvirt/images/centos7.0-1.qcow2  
$ sudo virt-sparsify /var/lib/libvirt/images/centos7.0-1.qcow2 /var/  
$ sudo guestfish -a /var/lib/libvirt/images/centos7.0-1.qcow2
```

6.13. Наконец-то после того как мы поработали с образами сами, можно поработать с образами с помощью **Openstack**. Для начала надо добавить в оба рабочих файла **keystonerc_adm** и **keystonerc_usr** переменную среды, определяющую версию **Glance API**, с которой мы будем работать (**OS_IMAGE_API_VERSION=2**):

```
$ sudo cat keystonerc_usr
unset OS_USERNAME OS_PASSWORD OS_PROJECT_NAME OS_USER_DOMAIN_NAME OS_PROJECT_DOMAIN_NAME OS_AUTH_URL OS_IDENTITY_API_VERSION PSI OS_IMAGE_API_VERSION
export OS_USERNAME=demo
export OS_PASSWORD=openstack
export OS_PROJECT_NAME=demo
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_DOMAIN_NAME=Default
export OS_AUTH_URL=http://controller.test.local:5000/v3
export OS_IDENTITY_API_VERSION=3
export PSI='[\u0@h|W(Openstack_Demo)]\$ '
export OS_IMAGE_API_VERSION=2
```

```
$ sudo cat keystonerc_adm
unset OS_USERNAME OS_PASSWORD OS_PROJECT_NAME OS_USER_DOMAIN_NAME OS_PROJECT_DOMAIN_NAME OS_AUTH_URL OS_IDENTITY_API_VERSION PS1 OS_IMAGE_API_VERSION
export OS_USERNAME=admin
export OS_PASSWORD=openstack
export OS_PROJECT_NAME=admin
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_DOMAIN_NAME=Default
export OS_AUTH_URL=http://controller.test.local:35357/v3
export OS_IDENTITY_API_VERSION=3
export PS1='[\u001b[34mOpenStack Admin]\u001b[0m] \$ '
export OS_IMAGE_API_VERSION=3
```

Загружаем наш скачанный образ Cirros в Glance.

Field	Value
checksum	443b7623e27ecf03dc9e01ee93f67afe
container_format	bare
created_at	2022-11-28T13:59:35Z
disk_format	qcow2
file	/v2/images/57c57075-fd8c-4475-bd3d-9d1c745849eb/file
id	57c57075-fd8c-4475-bd3d-9d1c745849eb
min_disk	0
min_ram	0
name	cirros-0.4.0-x86_64
owner	33ad52bb25284ad8be4b8681b8fe0063
properties	os_hash_algo='sha512', os_hash_value='огромная строка', os_hidden='False'
protected	False
schema	/v2/schemas/image
size	12716032
status	active
tags	
updated_at	2022-11-28T13:59:35Z
virtual_size	None
visibility	public

Используемые параметры:

- file** — путь до файла с образом
- disk-format** — формат образа диска
- container-format** — формат контейнера (bare — не контейнер)
- public** — образ доступен всем

Для вас с прошлой строчки прошла одна секунда, а для меня - два дня, так как команда на скрине не выполнялась из-за того, что **Glance** не мог авторизоваться в **Keystone** (я так думал), в результате оказалось, что **Glance** не мог отправить запрос на URL <http://controller.test.local:35357/v3/>, так как это доменное имя не было прописано в **/etc/hosts** (но ведь запросы к **Keystone** шли нормально!!!), чтобы все работало надо каждое имя узла добавлять в **/etc/hosts** (зато я поработал с файлами логов **keystone** и **glance**, а также флагом **--debug**):

```
$ sudo cat /etc/hosts
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.122.200 controller.test.local controller
```

После того как образ был загружен, он должен появится в этой папке **/var/lib/glance/images/**:

```
$ sudo ls -l /var/lib/glance/images/
итого 12420
-rw-r----- 1 glance glance 12716032 ноя 28 16:59 57c57075-fd8c-4475-bd3d-9d1c745849eb
```

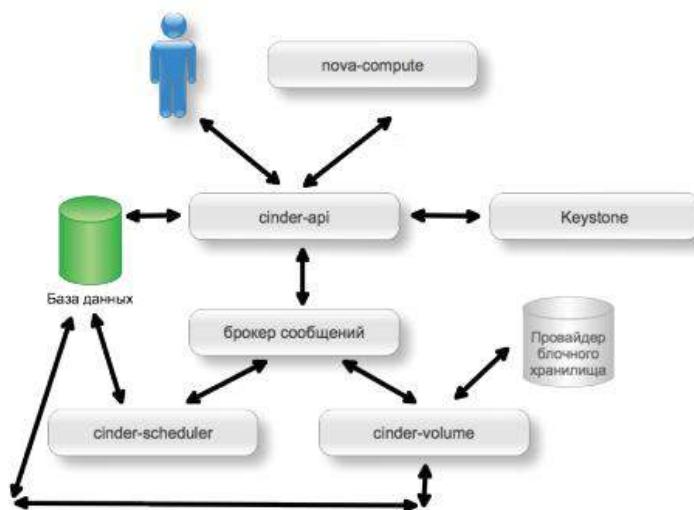
На этом все, но можно добавить, что образы популярных ОС можно найти тут:

- Ubuntu: <http://cloud-images.ubuntu.com/>
- Fedora: <https://cloud.fedoraproject.org/>
- Debian: <http://cdimage.debian.org/cdimage/openstack/>
- CentOS: <http://cloud.centos.org/centos/7/>

7. Установка и настройка сервиса блочного хранилища Cinder

7.1. **Cinder** придуман для того чтобы хранить модифицированные в ходе работы тома, созданных образов, зачем? Ну, например, когда мы создали ВМ с помощью сервиса **Nova** (потом установим) мы с ней работаем (что-то устанавливаем, изменяем и тд), а что если узел на котором была создана ВМ выйдет из строя, и нам нужно ее срочно восстановить, **Glance** у нас хранит только базовые образы и никак не поможет, вот для из-за таких ситуаций и придуман **Cinder**, который записывает все изменения тома на узлах, где установлен **cinder-volume**. Также в **Cinder** можно не только хранить тома, но и создавать их снимки в режиме «только для чтения», с помощью которых создавать новые тома, доступные на запись.

Архитектура Cinder:



Службы Cinder:

- **cinder-api** – точка входа для запросов в сервис по протоколу HTTP. Приняв запрос, сервис проверяет полномочия на выполнение запроса и переправляет запрос брокеру сообщений для доставки другим службам
- **cinder-scheduler** – сервис-планировщик принимает запросы от брокера сообщений и определяет, какой узел с сервисом **openstack-cinder-volume** должен обработать запрос
- **cinder-volume** – сервис отвечает за взаимодействие с бэкэндом – блочным устройством. Получает запросы от планировщика и транслирует непосредственно в хранилище. **Cinder** позволяет одновременно использовать несколько бэкендов. При этом для каждого из них запускается свой **openstack-cinder-volume**. И при помощи параметров **CapacityFilter** и **CapacityWeigher** можно управлять тем, какой бэкенд выберет планировщик
- **cinder-backup** – сервис отвечает за создание резервных копий томов в объектное хранилище

7.2. К этому пункту свободное место на control узле равнялось 1GB, так что надо увеличивать размер:

Первым делом вырубаем **VM2** и увеличиваем размер диска **qcow2** на 3GB:

```
$ sudo virsh shutdown VM2
Domain 'VM2' is being shutdown
$ sudo qemu-img resize /mnt/kvm/disk/vm2.qcow2 +3G
$ sudo qemu-img info /mnt/kvm/disk/vm2.qcow2
image: /mnt/kvm/disk/vm2.qcow2
file format: qcow2
virtual size: 9 GiB (9663676416 bytes)
disk size: 3.67 GiB
cluster_size: 65536
Format specific information:
  compat: 1.1
  compression type: zlib
  lazy refcounts: false
  refcount bits: 16
  corrupt: false
  extended l2: false
```

Потом запускаем **VM2**:

```
$ sudo virsh start VM2
Domain 'VM2' started
$ sudo virsh console VM2
```

Теперь с помощью **lsblk** проверяем, что появился диск **/dev/vda2**, а с помощью **pvs**, что на нем стоит система:

```
$ sudo lsblk
NAME      MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sr0        11:0    1 1024M  0 rom
vda       252:0    0   9G  0 disk
└─vda1    252:1    0   1G  0 part /boot
└─vda2    252:2    0   5G  0 part
  ├─centos-root 253:0    0 4,4G  0 lvm /
  └─centos-swap 253:1    0 616M  0 lvm [SWAP]
$ sudo pvs
PV          VG     Fmt Attr PSize  PFree
/dev/vda2  centos lvm2 a-- <5,00g    0
```

Скорее всего увеличить размер партиции можно и другим способом, но я побоялся экспериментировать и скачал утилиту **growpart**:

```
$ sudo yum -y install cloud-utils-growpart
$ sudo growpart /dev/vda 2
CHANGED: partition=2 start=2099200 old: size=10483712 end=12582912 new: size=16775135 end=18874335
```

После этого увеличиваем размер раздела **/dev/vda2**:

```
$ sudo pvresize /dev/vda2
Physical volume "/dev/vda2" changed
1 physical volume(s) resized or updated / 0 physical volume(s) not resized
$ sudo vgs
VG #PV #LV #SN Attr  VSize  VFree
centos 1 2 0 wz--n- <8,00g 3,00g
$ sudo df -h
Файловая система      Размер Использовано  Дост Использовано% Смонтировано в
devtmpfs           1,9G      0  1,9G      0% /dev
tmpfs             1,9G      0  1,9G      0% /dev/shm
tmpfs             1,9G      8,6M 1,9G      1% /run
tmpfs             1,9G      0  1,9G      0% /sys/fs/cgroup
/dev/mapper/centos-root 4,4G      3,4G 1,1G    77% /
/dev/vda1          1014M    195M 820M    20% /boot
tmpfs            379M      0  379M      0% /run/user/0
```

И наконец увеличиваем раздел **/dev/mapper/centos-root**:

```
$ sudo lvextend -r -l +100%FREE /dev/mapper/centos-root
Size of logical volume centos/root changed from 4,39 GiB (1125 extents) to 7,39 GiB (1893 extents).
Logical volume centos/root successfully resized.
meta-data=/dev/mapper/centos-root isize=512    agcount=4, agsize=288000 blks
          =                     sectsz=512  attr=2, projid32bit=1
          =                     crc=1    finobt=0 spinodes=0
data      =                     bsize=4096   blocks=1152000, imaxpct=25
          =                     sunit=0   swidth=0 blks
naming    =version 2           bsize=4096   ascii-ci=0 ftype=1
log       =internal            bsize=4096   blocks=2560, version=2
          =                     sectsz=512  sunit=0 blks, lazy-count=1
realtime  =none               extsz=4096   blocks=0, rtextents=0
data blocks changed from 1152000 to 1938432
$ sudo df -h
Файловая система      Размер Использовано Дост Использовано% Смонтировано в
devtmpfs             1,9G     0  1,9G      0% /dev
tmpfs                1,9G     0  1,9G      0% /dev/shm
tmpfs                1,9G     8,6M 1,9G      1% /run
tmpfs                1,9G     0  1,9G      0% /sys/fs/cgroup
/dev/mapper/centos-root 7,4G   3,4G 4,1G     46% /
/dev/vda1              1014M  195M 820M     20% /boot
tmpfs                379M     0  379M      0% /run/user/0
```

7.3. Теперь можно начать настройку окружения для сервиса **Cinder**, для этого надо добавить к виртуальной машине еще один диск в качестве блочного устройства, созданного с помощью [Linux LVM](#) (использует протокол [iSCSI](#)), на который будут писаться все изменения томов ВМ параллельно их записи на локальную систему рабочего узла.

Создадим новый диск и прикрепим его к запущенной **VM2**:

```
$ sudo qemu-img create -f qcow2 -o preallocation=metadata /mnt/kvm/disk/lvm.qcow2 2G
Formatting '/mnt/kvm/disk/lvm.qcow2', fmt=qcow2 cluster_size=65536 extended_l2=off preallocation=metadata
compression_type=zlib size=2147483648 lazy_refcounts=off refcount_bits=16
$ sudo virsh attach-disk VM2 /mnt/kvm/disk/lvm.qcow2 vdb --persistent
Disk attached successfully
```

В **VM2** проверяем присоединенный диск:

```
$ sudo lsblk
NAME      MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sr0        11:0    1 1024M  0 rom
vda       252:0    0   9G  0 disk
└─vda1    252:1    0   1G  0 part /boot
└─vda2    252:2    0   8G  0 part
  └─centos-root 253:0    0  7,4G  0 lvm /
  └─centos-swap 253:1    0  616M  0 lvm [SWAP]
vdb       252:16   0   2G  0 disk
```

Теперь создаем **LVM-группу** на присоединенном диске:

```
$ sudo vgcreate cinder-volumes /dev/vdb
Physical volume "/dev/vdb" successfully created.
Volume group "cinder-volumes" successfully created
```

```
$ sudo vgdisplay
--- Volume group ---
VG Name          cinder-volumes
System ID
Format          lvm2
Metadata Areas   1
Metadata Sequence No  1
VG Access        read/write
VG Status        resizable
MAX LV
Cur LV
Open LV
Max PV
Cur PV
Act PV
VG Size         <2,00 GiB
PE Size          4,00 MiB
Total PE        511
Alloc PE / Size 0 / 0
Free PE / Size  511 / <2,00 GiB
VG UUID          LHM4mQ-SSt7-fSwu-Ikk0-of4t-GTqv-42dhqK
```

7.3. Теперь качаем пакет openstack-cinder и редактируем его конфигурационный файл **/etc/cinder/cinder.conf**.

```
$ sudo yum -y install openstack-cinder
```

Настраиваем подключение к БД:

```
$ sudo crudini --set /etc/cinder/cinder.conf database connection mysql+pymysql://cinder:cinder@controller.test.local/cinder
```

Настраиваем URL, имя и пароль RabbitMQ:

```
$ sudo crudini --set /etc/cinder/cinder.conf DEFAULT transport_url rabbit://openstack:openstack@controller.test.local
```

Как раньше настраиваем реквизиты пользователя и проекта в Keystone:

```
$ sudo crudini --set /etc/cinder/cinder.conf DEFAULT auth_strategy keystone
$ sudo crudini --set /etc/cinder/cinder.conf keystone_authtoken project_name service
$ sudo crudini --set /etc/cinder/cinder.conf keystone_authtoken user_domain_name default
$ sudo crudini --set /etc/cinder/cinder.conf keystone_authtoken project_domain_name default
$ sudo crudini --set /etc/cinder/cinder.conf keystone_authtoken auth_type password
$ sudo crudini --set /etc/cinder/cinder.conf keystone_authtoken username cinder
$ sudo crudini --set /etc/cinder/cinder.conf keystone_authtoken password openstack
$ sudo crudini --set /etc/cinder/cinder.conf keystone_authtoken www_authenticate_uri http://controller.test.local:5000
$ sudo crudini --set /etc/cinder/cinder.conf keystone_authtoken auth_url http://controller.test.local:35357
```

Настройка имени lvm-группы:

```
$ sudo crudini --set /etc/cinder/cinder.conf lvm volume_group cinder-volumes
```

Настраиваем бэкенд хранения **Cinder**, а также драйвер (отвечает за хранение данных при помощи локального менеджера логических томов и протокола транспорта iSCSI) для него:

```
$ sudo crudini --set /etc/cinder/cinder.conf DEFAULT enabled_backends lvm  
$ sudo crudini --set /etc/cinder/cinder.conf lvm volume_driver cinder.volume.drivers.lvm.LVMVolumeDriver
```

Настраиваем используемый протокол транспорта и возможность использовать команду **cinder-rtstool** для управления томами (**lioadm** — поддержка [LIO iSCSI](#)):

```
$ sudo crudini --set /etc/cinder/cinder.conf lvm iscsi_protocol iscsi  
$ sudo crudini --set /etc/cinder/cinder.conf lvm iscsi_helper lioadm
```

Настройка URL **Glance API**:

```
$ sudo crudini --set /etc/cinder/cinder.conf DEFAULT glance_api_servers http://controller.test.local:9292
```

Настройка пути к папке, где будут хранится [lock-файлы](#) (файлы, создаваемые сервисами для межпроцессорной блокировки):

```
$ sudo crudini --set /etc/cinder/cinder.conf oslo_concurrency lock_path /var/lib/cinder/tmp
```

7.4. Теперь аналогично прошлым настройка создаем БД **cinder**:

```
$ sudo mysql -u root -p  
Enter password:  
MariaDB [(none)]> CREATE DATABASE cinder;  
Query OK, 1 row affected (0.01 sec)  
MariaDB [(none)]> GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'localhost' IDENTIFIED BY 'cinder';  
Query OK, 0 rows affected (0.00 sec)  
MariaDB [(none)]> GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'%' IDENTIFIED BY 'cinder';  
Query OK, 0 rows affected (0.00 sec)
```

Во время выполнения скрипта синхронизации вылезают сообщения об устаревших полях, полностью их игнорируем, БД и так поднимется:

```
$ sudo -s /bin/sh -c "cinder-manage db sync" cinder  
Deprecated: Option "logdir" from group "DEFAULT" is deprecated. Use option "log-dir" from group "DEFAULT".
```

7.5. Теперь регистрируем сервис **Cinder** в **Keystone**:

Создаем пользователя **cinder** и предоставляем ему роль:

```
$ sudo source keystonerc_admin  
$ sudo openstack user create --domain default --password openstack cinder  
+-----+  
| Field | Value |  
+-----+  
| domain_id | default |  
| enabled | True |  
| id | 964cdcbe60dc49baa48d23930d7bb7ee |  
| name | cinder |  
| options | {} |  
| password_expires_at | None |  
+-----+  
$ sudo openstack role add --project service --user cinder admin
```

Создаем сервисы **cinderv2** и **cinderv3**. Раньше надо было иметь две версии **cinder** для полной совместимости с остальными сервисами, но вроде как это уже **не необходимо** и можно ограничиться только сервисом **cinderv3** (я боюсь и разверну 2 сервиса):

```
$ sudo openstack service create --name cinderv2 --description "OpenStack Block Storage" volumev2
+-----+
| Field      | Value
+-----+
| description | OpenStack Block Storage
| enabled     | True
| id          | a4e6400475f24e9f89fa5672ae7a0050
| name        | cinderv2
| type        | volumev2
+-----+
$ sudo openstack service create --name cinderv3 --description "OpenStack Block Storage" volumev3
+-----+
| Field      | Value
+-----+
| description | OpenStack Block Storage
| enabled     | True
| id          | bcb5593a5c094c25b7d33d2328de5e22
| name        | cinderv3
| type        | volumev3
+-----+
```

Соответственно раз сервиса два, то и точек входа в два раза больше:

```
$ sudo openstack endpoint create --region RegionOne volumev2 public http://controller.test.local:8776/v2/%(project_id)s
+-----+
| Field      | Value
+-----+
| enabled    | True
| id         | 67325da7f260442aa3b9e0d822d19be1
| interface   | public
| region     | RegionOne
| region_id  | RegionOne
| service_id | a4e6400475f24e9f89fa5672ae7a0050
| service_name| cinderv2
| service_type| volumev2
| url        | http://controller.test.local:8776/v2/%(project_id)s
+-----+
```

```

$ sudo openstack endpoint create --region RegionOne volumev2 internal
http://controller.test.local:8776/v2/\%(project_id\%)s
+-----+
| Field      | Value
+-----+
| enabled    | True
| id         | 5dfb47b6f8114274b0c372d6b9bdd0ef
| interface   | internal
| region     | RegionOne
| region_id  | RegionOne
| service_id | a4e6400475f24e9f89fa5672ae7a0050
| service_name| cinderv2
| service_type| volumev2
| url        | http://controller.test.local:8776/v2/\%(project_id\%)s
+-----+
$ sudo openstack endpoint create --region RegionOne volumev2 admin
http://controller.test.local:8776/v2/\%(project_id\%)s
+-----+
| Field      | Value
+-----+
| enabled    | True
| id         | c220d65624ef4bb59f0690d76e9993a7
| interface   | admin
| region     | RegionOne
| region_id  | RegionOne
| service_id | a4e6400475f24e9f89fa5672ae7a0050
| service_name| cinderv2
| service_type| volumev2
| url        | http://controller.test.local:8776/v2/\%(project_id\%)s
+-----+
$ sudo openstack endpoint create --region RegionOne volumev3 public
http://controller.test.local:8776/v3/\%(project_id\%)s
+-----+
| Field      | Value
+-----+
| enabled    | True
| id         | 3d9496ae33de44e8975d7cf8db581b39
| interface   | public
| region     | RegionOne
| region_id  | RegionOne
| service_id | bcb5593a5c094c25b7d33d2328de5e22
| service_name| cinderv3
| service_type| volumev3
| url        | http://controller:8776/v3/\%(project_id\%)s
+-----+
$ sudo openstack endpoint create --region RegionOne volumev3 internal
http://controller.test.local:8776/v3/\%(project_id\%)s
+-----+
| Field      | Value
+-----+
| enabled    | True
| id         | 61aa4e69aa5404b9e7585dac3d7653e
| interface   | internal
| region     | RegionOne
| region_id  | RegionOne
| service_id | bcb5593a5c094c25b7d33d2328de5e22
| service_name| cinderv3
| service_type| volumev3
| url        | http://controller:8776/v3/\%(project_id\%)s
+-----+
$ sudo openstack endpoint create --region RegionOne volumev3 admin
http://controller.test.local:8776/v3/\%(project_id\%)s
+-----+
| Field      | Value
+-----+
| enabled    | True
| id         | 9805a989caad40f19d686d884cd825f4
| interface   | admin
| region     | RegionOne
| region_id  | RegionOne
| service_id | bcb5593a5c094c25b7d33d2328de5e22
| service_name| cinderv3
| service_type| volumev3
| url        | http://controller:8776/v3/\%(project_id\%)s
+-----+

```

7.6. Теперь устанавливаем и настраиваем сервис **iSCSI**, по идее следующие действия надо производить на других узлах для большей доступности и отказоустойчивости, поэтому некоторые команды могут повторяться.

Устанавливаем [targetcli](#) (оболочка для управления LIO iSCSI):

```
$ sudo yum -y install targetcli
```

Задаем ip машины на которой будет находиться **cinder-volume**:

```
$ sudo crudini --set /etc/cinder/cinder.conf DEFAULT my_ip 192.168.122.200
```

Задаем бэкенд хранения **Cinder**:

```
$ sudo crudini --set /etc/cinder/cinder.conf DEFAULT enabled_backends lvm
```

По итогу всех махинаций наш конфиг (**/etc/cinder/cinder.conf**) выглядит так:

```
$ sudo vi /etc/cinder/cinder.conf
[DEFAULT]
transport_url = rabbit://openstack:openstack@controller.test.local
auth_strategy = keystone
enabled_backends = lvm
glance_api_servers = http://controller.test.local:9292
my_ip = 192.168.122.200
...
[database]
connection = mysql+pymysql://cinder:cinder@controller.test.local/cinder
...
[keystone_auth_token]
project_name = service
user_domain_name = default
project_domain_name = default
auth_type = password
username = cinder
password = openstack
www_authenticate_uri = http://controller.test.local:5000
auth_url = http://controller.test.local:35357
...
[lvm]
volume_group = cinder-volumes
volume_driver = cinder.volume.drivers.lvm.LVMVolumeDriver
iscsi_protocol = iscsi
iscsi_helper = lioadm
...
[oslo_concurrency]
lock_path = /var/lib/cinder/tmp
```

7.7. Теперь запускаем все что можно:

```
$ sudo systemctl enable target.service
Created symlink from /etc/systemd/system/multiuser.target.wants/target.service
to /usr/lib/systemd/system/target.service.
$ sudo systemctl start target.service
[ 6348.118002] Rounding down aligned max_sectors from 4294967295 to 4294967288
$ sudo systemctl status target.service
● target.service - Restore LIO kernel target configuration
  Loaded: loaded (/usr/lib/systemd/system/target.service; enabled; vendor
  preset: disabled)
  Active: active (exited) since Cp 2022-11-30 16:36:06 MSK; 9s ago
```

```
$ sudo systemctl enable openstack-cinder-api
Created symlink from /etc/systemd/system/multi-user.target.wants/openstack-
cinder-api.service to /usr/lib/systemd/system/openstack-cinder-api.service.
$ sudo systemctl start openstack-cinder-api
$ sudo systemctl status openstack-cinder-api
● openstack-cinder-api.service - OpenStack Cinder API Server
    Loaded: loaded (/usr/lib/systemd/system/openstack-cinder-api.service;
   enabled; vendor preset: disabled)
    Active: active (running) since Cp 2022-11-30 16:38:33 MSK; 5s ago
```

```
$ sudo systemctl enable openstack-cinder-scheduler
Created symlink from /etc/systemd/system/multi-user.target.wants/openstack-
cinder-scheduler.service to /usr/lib/systemd/system/openstack-cinder-
scheduler.service.
$ sudo systemctl start openstack-cinder-scheduler
$ sudo systemctl status openstack-cinder-scheduler
● openstack-cinder-scheduler.service - OpenStack Cinder Scheduler Server
    Loaded: loaded (/usr/lib/systemd/system/openstack-cinder-scheduler.service;
   enabled; vendor preset: disabled)
    Active: active (running) since Cp 2022-11-30 16:40:28 MSK; 5s ago
```

```
$ sudo systemctl enable openstack-cinder-volume
Created symlink from /etc/systemd/system/multi-user.target.wants/openstack-
cinder-volume.service to /usr/lib/systemd/system/openstack-cinder-
volume.service.
$ sudo systemctl start openstack-cinder-volume
$ sudo systemctl status openstack-cinder-volume
● openstack-cinder-volume.service - OpenStack Cinder Volume Server
    Loaded: loaded (/usr/lib/systemd/system/openstack-cinder-volume.service;
   enabled; vendor preset: disabled)
    Active: active (running) since Cp 2022-11-30 16:41:42 MSK; 4s ago
```

```
$ sudo systemctl enable openstack-cinder-backup
Created symlink from /etc/systemd/system/multi-user.target.wants/openstack-
cinder-backup.service to /usr/lib/systemd/system/openstack-cinder-
backup.service.
$ sudo systemctl start openstack-cinder-backup
$ sudo systemctl status openstack-cinder-backup
● openstack-cinder-backup.service - OpenStack Cinder Backup Server
    Loaded: loaded (/usr/lib/systemd/system/openstack-cinder-backup.service;
   enabled; vendor preset: disabled)
    Active: active (running) since Cp 2022-11-30 16:43:33 MSK; 3s ago
```

Также все эти службы можно было запустить одной командой с помощью пакета [openstack-utils](#), который надо скачать, так как с помощью него также можно удобно селдить за статусами всех сервисов кластера:

```
$ sudo yum -y install openstack-utils
```

Запуск всех служб **Cinder** одной командой:

```
$ sudo openstack-service start cinder
```

Просмотр статуса всех сервисов одной командой (если что **keystone** и должен быть в **inactive**, мы ведь его напрямую не поднимали, а через **RabbitMQ**):

```
$ sudo openstack-status
== Glance services ==
openstack-glance-api:           active
openstack-glance-registry:       active
== Keystone service ==
openstack-keystone:             inactive (disabled on boot)
== Cinder services ==
openstack-cinder-api:           active
openstack-cinder-scheduler:     active
openstack-cinder-volume:        active
openstack-cinder-backup:        active
== Support services ==
libvirtd:                      active
dbus:                           active
target:                         active
rabbitmq-server:                active
== Keystone users ==
+-----+
| ID          | Name |
+-----+
| 549949769da94550a63fa6667741afb4 | admin |
| b1457723dc214b9aa669676043c7c7f2 | demo  |
| cc9df70e640e4c3cb8b3d85205162470 | glance |
| 964cdcbcbe60dc49baa48d23930d7bb7ee | cinder |
+-----+
== Glance images ==
+-----+
| ID          | Name      |
+-----+
| 57c57075-fd8c-4475-bd3d-9d1c745849eb | cirros-0.4.0-x86_64 |
+-----+
```

Также можно посмотреть состояние служб **Cinder** (**cinder-backup** по идеи должен был быть остановлен, так как ему не задана конечная точка бэкенда хранения этих самых бэкапов (то самый сервис **Swift** из будущего), он почему-то работает, но в логах пишет, что могут быть проблемы):

```
$ sudo cinder service-list
+-----+
| Binary | Host           | Zone | Status | State | Updated_at            | Cluster | Disabled Reason | Backend State |
+-----+
| cinder-backup | controller.test.local | nova | enabled | up    | 2022-11-30T13:53:37.000000 | -      | -                  | -          |
| cinder-scheduler | controller.test.local | nova | enabled | up    | 2022-11-30T13:53:42.000000 | -      | -                  | -          |
| cinder-volume   | controller.test.local@lvm | nova | enabled | up    | 2022-11-30T13:53:41.000000 | -      | -                  | up         |
+-----+
```

Содержимое файла логов **/var/log/cinder/backup.log** **cinder-backup**:

```
$ sudo cat /var/log/cinder/backup.log
2022-11-30 16:43:36.755 8716 WARNING cinder.backup.drivers.swift [-] We will use
endpoints from keystone. It is possible we could have problems because of it.
```

На этом настройка закончена, теперь тестируем функционал. Для этого попробуем создать том (**volume**) **testvol1**:

Field	Value
attachments	[]
availability_zone	nova
bootable	false
consistencygroup_id	None
created_at	2022-11-30T14:22:10.000000
description	None
encrypted	False
id	1c603790-ba71-476e-b46c-43ec0ebabb2d
migration_status	None
multiattach	False
name	testvol1
properties	
replication_status	None
size	1
snapshot_id	None
source_volid	None
status	creating
type	__DEFAULT__
updated_at	None
user_id	549949769da94550a63fa6667741afb4

Пока с помощью **openstack cli** посмотреть на созданный том (**openstack volume list**) нельзя, так как мы еще не развернули сервис nova, но можно посмотреть с помощью **cinder cli**:

\$ sudo cinder list
+-----+-----+-----+-----+-----+-----+-----+
ID Status Name Size Volume Type Bootable Attached to
+-----+-----+-----+-----+-----+-----+-----+
1c603790-ba71-476e-b46c-43ec0ebabb2d available testvol1 1 __DEFAULT__ false
+-----+-----+-----+-----+-----+-----+-----+

Зря я сделал диск размером всего в 2GB:

\$ sudo lvs
LV
root
swap
cinder-volumes-pool
volume-1c603790-ba71-476e-b46c-43ec0ebabb2d

VG	Attr	LSize	Pool	Origin	Data%	Meta%	Move	Log	Cpy%	Sync	Convert
centos	-wi-a0---	7,39g									
centos	-wi-a0----	616,00m									
cinder-volumes	twi-aotz--	1,90g			0,00	11,04					
cinder-volumes	Vwi-a-tz--	1,00g	cinder-volumes-pool		0,00						

7.8. По идее на этом можно заканчивать настройку сервиса **Cinder**, но надо кое-что уточнить, а именно, что в реальной жизни бэкеном **cinder-volume** и **cinder-backup** должен быть сервис **Swift** или **Ceph**, которые нужно разворачивать на других узлах. Мой ноут с 8GB оперативки и в перспективе двумя рабочими узлами такое точно не потянет, но все-таки было принято решение сделать бэкеном **cinder-backup NFS-сервер** расположенный на одном из рабочих узлов в качестве эксперимента.

Итак для начала нужно этот рабочий узел поднять (когда я в первом пункте сказал, что поднял 3 ВМ, я соврал):

```
$ sudo qemu-img create -f qcow2 -o preallocation=metadata /mnt/kvm/disk/vm3.qcow2 6G
[sudo] пароль для artem:
Formatting '/mnt/kvm/disk/vm3.qcow2', fmt=qcow2 cluster_size=65536 extended_l2=off preallocation=metadata
compression_type=zlib size=6442450944 lazy_refcounts=off refcount_bits=16
$ sudo virt-install --virt-type kvm --name VM3 --ram 2048 --arch=x86_64 --disk
/mnt/kvm/disk/vm3.qcow2,size=6,format=qcow2 --network network=default --os-type=linux --os-
variant=centos7.0 --location=/mnt/kvm/iso/CentOS-7-x86_64-Minimal-2009.iso --graphics none --console
pty,target_type=serial --extra-args 'console=ttyS0,115200n8 serial'
```

```
Network configuration
Wired (eth0) connected
IPv4 Address: 192.168.122.210 Netmask: 255.255.255.0 Gateway: 192.168.122.1
DNS: 192.168.122.1
```

Теперь наконец-то понятная настройка ВМ для работы с **Openstack** (если что файл **/etc/hosts** должен выглядеть да всех ВМ одинаково, так что редактируем его на каждой ВМ):

```
$ sudo hostnamectl set-hostname compute.test.local
$ sudo hostname
compute.test.local
$ sudo vi /etc/hosts
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.122.200 controller.test.local controller
192.168.122.210 compute.test.local compute
```

```
$ sudo yum -y update
$ sudo yum -y install epel-release
$ sudo yum install -y https://www.rdoproject.org/repos/rdo-release.rpm
$ sudo systemctl stop NetworkManager.service
$ sudo systemctl disable NetworkManager.service
$ sudo systemctl start network.service
$ sudo systemctl enable network.service
$ sudo systemctl stop firewalld.service
$ sudo systemctl disable firewalld.service
$ sudo vi /etc/sysconfig/selinux
...
SELINUX=disabled
$ sudo yum -y install chrony
$ sudo systemctl start chronyd
$ sudo systemctl enable chrony
$ sudo yum -y install crudini
$ sudo reboot
```

Теперь настраиваем все что касается [**nfs сервера**](#):

Скачиваем и запускаем **nfs-server**:

```
$ sudo yum -y install nfs-utils
$ sudo systemctl enable nfs-server
$ sudo systemctl start nfs-server
```

Создаем папку в которая будет общая для всех **nfs-клиентов**:

```
$ sudo mkdir -p /backup/nfs
$ sudo chmod -R 777 /backup/nfs
```

Редактируем файл **/etc/exports**, указывая каким ip-адресам будет доступна папка, после чего применяем настройка и перезапускаем **nfs-сервер**:

```
$ sudo vi /etc/exports  
/backup/nfs 192.168.122.0/24(rw,sync,no_root_squash,no_all_squash)  
$ sudo exportfs -a  
$ sudo systemctl restart nfs-server
```

- **rw** – права на запись в каталоге (**ro** – доступ только на чтение)
- **sync** – синхронный режим доступа (**async** – подразумевает, что не нужно ждать подтверждения записи на диск)
- **no_root_squash** – разрешает root пользователю с клиента получать доступ к NFS каталогу (обычно не рекомендуется)
- **no_all_squash** — включение авторизации пользователя (**all_squash** – доступ к ресурсам от анонимного пользователя)

Теперь попробуем настроить **NFS** бэкенд для **cinder-backup** на **controllet.test.local** (оказалось, что не обязательно настраивать **nfs-клиента** самостоятельно, монтировать папки и тд, достаточно просто добавить параметры в конфиг **Cinder** **/etc/cinder/cinder.conf**, и все смонтируется автоматически после перезагрузки сервиса **Cinder**):

Настраиваем драйвер для **cinder-backup** и папку **nfs-сервера**, которую надо смонтировать:

```
$ sudo crudini --set /etc/cinder/cinder.conf DEFAULT backup_driver  
cinder.backup.drivers.nfs.NFSBackupDriver  
$ sudo crudini --set /etc/cinder/cinder.conf DEFAULT backup_share  
192.168.122.210:/backup/nfs
```

```
$ sudo openstack restart cinder  
$ sudo df -h  
Файловая система Размер Использовано Дост Использовано% Смонтировано в  
devtmpfs 1,9G 0 1,9G 0% /dev  
tmpfs 1,9G 0 1,9G 0% /dev/shm  
tmpfs 1,9G 8,7M 1,9G 1% /run  
tmpfs 1,9G 0 1,9G 0% /sys/fs/cgroup  
/dev/mapper/centos-root 7,4G 3,5G 4,0G 47% /  
/dev/vda1 1014M 195M 820M 20% /boot  
tmpfs 379M 0 379M 0% /run/user/0  
192.168.122.210:/backup/nfs 4,4G 1,6G 2,9G 35% /var/lib/cinde  
r/backup_mount/1c3f5076e509ece9739766c569246615
```

Все, поздравляю **Swift** можно не настраивать как и **Ceph**, но сами понимаете, что если мощность позволяют надо обязательно ставить или **Swift** или **Ceph**, желательно на трех отдельных узлах. Пока нет желания про них что-то расписывать, может только в самом конце, когда все будет настроено, я разверну **Swift** или **Ceph** на рабочих узлах.

8. Установка и настройка сервиса управления виртуальными машинами и сетью Nova

8.1. Сервис **Nova** у нас отвечает за управление запущенными экземплярами виртуальных машин, а также сетью (раньше да, теперь за сеть отвечает **Neutron**) как

видно из названия раздела. Состоит из следующих служб (помимо **RabbitMQ** (брокера сообщений) и собственной БД):

- **openstack-nova-api** – как и подобные службы других рассмотренных сервисов, отвечает за обработку пользовательских вызовов API.
- **openstack-nova-scheduler** – сервис-планировщик. Получает из очереди запросы на запуск виртуальных машин и выбирает узел для их запуска. Выбор осуществляется, согласно весам узлов после применения фильтров (необходимый объем оперативной памяти, определенная зона доступности и т. д.). Вес рассчитывается каждый раз при запуске или миграции виртуальной машины.
- **openstack-nova-conductor** – служба выступает в качестве посредника между базой данных и nova-compute, позволяя осуществлять горизонтальное масштабирование (нельзя развертывать на тех же узлах, что и nova-compute).
- **openstack-nova-novncproxy** – выступает в роли [VNCпрокси](#) (такую опцию можно было выбрать во время запуска виртуалок при помощи `virsh`) и позволяет подключаться к консоли виртуальных машин при помощи браузера.
- **openstack-nova-consoleauth** – отвечает за авторизацию для сервиса **openstack-nova-novncproxy** (вроде как больше не поддерживается).
- **openstack-nova-placement-api** – сервис отвечает за отслеживание списка ресурсов и их использование (раньше был частью nova, потом отпочковался и стал называться **openstack-placement-api**, а по-благородному **Placement**).
- **openstack-nova-compute** – демон, управляющий виртуальными машинами через API гипервизора. Как правило, запускается на узлах, где располагается сам гипервизор.

Все сервисы кроме **nova-compute** располагаются на **controller** узле, а **nova-compute** на рабочих узлах **compute** и **compute-opt** (пока не создан).

8.2. Приступаем к установке сервиса **Nova**:

Как всегда базовые начальные шаги (и хорошо, что они у всех сервисов очень похожие):

Устанавливаем все службы **Nova**, кроме **openstack-nova-compute** (Placement раз уж он был когда-то частью **Nova** тоже скачаем и потом настроим, также возможно не надо качать **openstack-nova-console**, так как **openstack-nova-consoleauth** больше не поддерживается, но это не точно):

```
$ sudo yum -y install openstack-nova-api openstack-nova-conductor openstack-nova-novncproxy openstack-nova-scheduler openstack-nova-console openstack-placement-api
```

Создаем три бд (**nova** — содержит всю информацию о ВМ, которые удалось запланировать, **nova-api** — содержит информацию о местонахождении и временном (еще только создаются) местонахождении ВМ и **nova-cell0** — содержит ВМ, которые не удалось запланировать) и выдаем всем привилегии:

```

$ sudo mysql -u root -p
Enter password:
MariaDB [(none)]> CREATE DATABASE nova_api;
Query OK, 1 row affected (0.00 sec)
MariaDB [(none)]> CREATE DATABASE nova;
Query OK, 1 row affected (0.00 sec)
MariaDB [(none)]> CREATE DATABASE nova_cell0;
Query OK, 1 row affected (0.00 sec)
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'localhost'
IDENTIFIED BY 'nova';
Query OK, 0 rows affected (0.01 sec)
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'%' IDENTIFIED BY
'nova';
Query OK, 0 rows affected (0.00 sec)
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' IDENTIFIED BY
'nova';
Query OK, 0 rows affected (0.00 sec)
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%' IDENTIFIED BY
'nova';
Query OK, 0 rows affected (0.00 sec)
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_cell0.* TO 'nova'@'localhost'
IDENTIFIED BY 'nova';
Query OK, 0 rows affected (0.00 sec)
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_cell0.* TO 'nova'@'%' IDENTIFIED BY
'nova';
Query OK, 0 rows affected (0.00 sec)

```

Регистрируем сервис Nova в Keystone:

```

$ sudo source keystonerc_admin
$ sudo openstack user create --domain default --password openstack nova
+-----+
| Field      | Value
+-----+
| domain_id  | default
| enabled     | True
| id          | 9c8728a01d93496c8f0656300acf1993
| name        | nova
| options     | {}
| password_expires_at | None
+-----+
$ sudo openstack role add --project service --user nova admin
$ sudo openstack service create --name nova --description "OpenStack Compute
Service" compute
+-----+
| Field      | Value
+-----+
| description | OpenStack Compute Service
| enabled     | True
| id          | 9c1656e2cd424173b56a3ceedb7c4653
| name        | nova
| type        | compute
+-----+

```

```
$ sudo openstack endpoint create --region RegionOne compute public
http://controller.test.local:8774/v2.1
+-----+-----+
| Field | Value |
+-----+-----+
| enabled | True
| id | 818cf96645e8479a8043b760f637c84a
| interface | public
| region | RegionOne
| region_id | RegionOne
| service_id | 9c1656e2cd424173b56a3ceedb7c4653
| service_name | nova
| service_type | compute
| url | http://controller.test.local:8774/v2.1
+-----+
$ sudo openstack endpoint create --region RegionOne compute internal
http://controller.test.local:8774/v2.1
+-----+-----+
| Field | Value |
+-----+-----+
| enabled | True
| id | 43a10c9b98224a12944c95de19880bf1
| interface | internal
| region | RegionOne
| region_id | RegionOne
| service_id | 9c1656e2cd424173b56a3ceedb7c4653
| service_name | nova
| service_type | compute
| url | http://controller.test.local:8774/v2.1
+-----+
$ sudo openstack endpoint create --region RegionOne compute admin
http://controller.test.local:8774/v2.1
+-----+-----+
| Field | Value |
+-----+-----+
| enabled | True
| id | a09596b6b9524901b937de50af66dd24
| interface | admin
| region | RegionOne
| region_id | RegionOne
| service_id | 9c1656e2cd424173b56a3ceedb7c4653
| service_name | nova
| service_type | compute
| url | http://controller.test.local:8774/v2.1
+-----+
```

Теперь регистрируем сервис **Placement** в Keystone:

```

$ sudo source keystonerc_adm
$ sudo openstack user create --domain default --password openstack placement
+-----+-----+
| Field | Value |
+-----+-----+
| domain_id | default |
| enabled | True |
| id | 891af44bec9d44389366687d0e217ed9 |
| name | placement |
| options | {} |
| password_expires_at | None |
+-----+
$ sudo openstack role add --project service --user placement admin
$ sudo openstack service create --name placement --description "OpenStack Placement API" placement
+-----+-----+
| Field | Value |
+-----+-----+
| description | OpenStack Placement API |
| enabled | True |
| id | e5d5c3305169478ba07087c4d60a80a9 |
| name | placement |
| type | placement |
+-----+

```

```

$ sudo openstack endpoint create --region RegionOne placement public
http://controller.test.local:8778
+-----+-----+
| Field | Value |
+-----+-----+
| enabled | True |
| id | 6840b7cd76bc4973b4813797d342b9c8 |
| interface | public |
| region | RegionOne |
| region_id | RegionOne |
| service_id | e5d5c3305169478ba07087c4d60a80a9 |
| service_name | placement |
| service_type | placement |
| url | http://controller.test.local:8778 |
+-----+
$ sudo openstack endpoint create --region RegionOne placement internal
http://controller.test.local:8778
+-----+-----+
| Field | Value |
+-----+-----+
| enabled | True |
| id | ea131937f7f44ee9951d21f992e96c31 |
| interface | internal |
| region | RegionOne |
| region_id | RegionOne |
| service_id | e5d5c3305169478ba07087c4d60a80a9 |
| service_name | placement |
| service_type | placement |
| url | http://controller.test.local:8778 |
+-----+
$ sudo openstack endpoint create --region RegionOne placement admin
http://controller.test.local:8778
+-----+-----+
| Field | Value |
+-----+-----+
| enabled | True |
| id | 2fad1612ab4d4486920e79c54002a2e1 |
| interface | admin |
| region | RegionOne |
| region_id | RegionOne |
| service_id | e5d5c3305169478ba07087c4d60a80a9 |
| service_name | placement |
| service_type | placement |
| url | http://controller.test.local:8778 |
+-----+

```

Далее конфигурируем файл **/etc/nova/nova.conf** (все параметры (я напишу какие не надо) ниже также надо будет добавить в аналогичные файлы на рабочих узлах):

Базовая конфигурация (**enabled_apis** — разрешенные API):

```
$ sudo crudini --set /etc/nova/nova.conf DEFAULT transport_url rabbit://openstack:openstack@controller.test.local
$ sudo crudini --set /etc/nova/nova.conf DEFAULT enabled_apis osapi_compute,metadata
$ sudo crudini --set /etc/nova/nova.conf api auth_strategy keystone
$ sudo crudini --set /etc/nova/nova.conf keystone_authtoken project_name service
$ sudo crudini --set /etc/nova/nova.conf keystone_authtoken user_domain_name default
$ sudo crudini --set /etc/nova/nova.conf keystone_authtoken project_domain_name default
$ sudo crudini --set /etc/nova/nova.conf keystone_authtoken auth_type password
$ sudo crudini --set /etc/nova/nova.conf keystone_authtoken username nova
$ sudo crudini --set /etc/nova/nova.conf keystone_authtoken password openstack
$ sudo crudini --set /etc/nova/nova.conf keystone_authtoken www_authenticate_uri http://controller.test.local:5000
$ sudo crudini --set /etc/nova/nova.conf keystone_authtoken auth_url http://controller.test.local:35357
```

Теперь на будущее включаем поддержку сервиса Neutron (**firewall_driver** — должно быть такое значение, если используется **Neutron** вместо **nova-network**):

```
$ sudo crudini --set /etc/nova/nova.conf DEFAULT use_neutron True
$ sudo crudini --set /etc/nova/nova.conf DEFAULT firewall_driver nova.virt.firewall.NoopFirewallDriver
```

Настраиваем URL для использования сервиса **Glance**:

```
$ sudo crudini --set /etc/nova/nova.conf glance api_servers http://controller.test.local:9292
```

Указываем имя региона для сервиса **Cinder**:

```
$ sudo crudini --set /etc/nova/nova.conf cinder os_region_name RegionOne
```

Указываем путь к файлам блокировок:

```
$ sudo crudini --set /etc/nova/nova.conf oslo_concurrency lock_path /var/lib/nova/tmp
```

Настраиваем параметры для подключения к сервису **Placement**:

```
$ sudo crudini --set /etc/nova/nova.conf placement region_name RegionOne
$ sudo crudini --set /etc/nova/nova.conf placement auth_url http://controller.test.local:35357/v3
$ sudo crudini --set /etc/nova/nova.conf placement auth_type password
$ sudo crudini --set /etc/nova/nova.conf placement project_domain_name default
$ sudo crudini --set /etc/nova/nova.conf placement user_domain_name default
$ sudo crudini --set /etc/nova/nova.conf placement project_name service
$ sudo crudini --set /etc/nova/nova.conf placement username placement
$ sudo crudini --set /etc/nova/nova.conf placement password openstack
```

```
$ sudo vi /etc/nova/nova.conf
[DEFAULT]
transport_url = rabbit://openstack:openstack@controller.test.local
enabled_apis = osapi_compute,metadata
use_neutron = True
firewall_driver = nova.virt.firewall.NoopFirewallDriver
...
[api]
auth_strategy = keystone
...
[keystone_authtoken]
project_name = service
user_domain_name = default
project_domain_name = default
auth_type = password
username = nova
password = openstack
www_authenticate_uri = http://controller.test.local:5000
auth_url = http://controller.test.local:35357
...
[glance]
api_servers = http://controller.test.local:9292
...
[cinder]
os_region_name = RegionOne
...
[placement]
project_name = service
user_domain_name = default
project_domain_name = default
auth_type = password
username = placement
password = openstack
auth_url = http://controller.test.local:35357/v3
region_name = RegionOne
...
[oslo_concurrency]
lock_path = /var/lib/nova/tmp
```

На этом общие параметры заканчиваются и начинаются специфичные для **controller.test.local**:

Настраиваем пути к созданным БД (почему только к 2 из 3 я без понятия):

```
$ sudo crudini --set /etc/nova/nova.conf api_database connection
mysql+pymysql://nova:nova@controller.test.local/nova_api
$ sudo crudini --set /etc/nova/nova.conf database connection
mysql+pymysql://nova:nova@controller.test.local/nova
```

Указываем ip **controller.test.local**:

```
$ sudo crudini --set /etc/nova/nova.conf DEFAULT my_ip 192.168.122.200
```

Указываем ip адрес на котором будет работать **VNC-сервер**:

```
$ sudo crudini --set /etc/nova/nova.conf vnc server_listen
192.168.122.200
$ sudo crudini --set /etc/nova/nova.conf vnc server_proxyclient_address
192.168.122.200
$ sudo crudini --set /etc/nova/nova.conf vnc enabled true
```

```
$ sudo vi /etc/nova/nova.conf
[DEFAULT]
...
my_ip = 192.168.122.200
...
[api_database]
connection = mysql+pymysql://nova:nova@controller.test.local/nova_api
...
[database]
connection = mysql+pymysql://nova:nova@controller.test.local/nova
...
[vnc]
server_listen = 192.168.122.200
server_proxyclient_address = 192.168.122.200
enabled = true
```

Теперь можно инициализировать базу данных **nova-api**:

```
$ sudo -s /bin/sh -c "nova-manage api_db sync" nova
```

Также надо зарегистрировать базу данных **cell0** в БД **nova** и создать ячейку **cell1**:

```
$ sudo -s /bin/sh -c "nova-manage cell_v2 map_cell0" nova
$ sudo -s /bin/sh -c "nova-manage cell_v2 create_cell --name=cell1 --verbose" nova
1d26f0d0-e23b-4e62-87b4-95945eb34ef2
```

Теперь заполняем (очень долго) БД **nova** (я уже сошел с нити понимания конфигурации):

Name	UUID	Transport URL	Database Connection	Disabled
cell0	00900000-0000-0000-0000-000000000000	none:/	mysql+pymysql://nova:****@controller.t est.local/nova_cell0	False
cell1	1d26f0d0-e23b-4e62-87b4-95945eb34ef2	rabbit://openstack:****@controller.tes t.local	mysql+pymysql://nova:****@controller.t est.local/nova	False

8.3. Перед тем как запустить все сервисы **Nova** сконфигурируем и запустим сервис **Placement** (оказывается зарегистрировать его в **Keystone** было недостаточно):

Для этого создадим БД **placement** и выдадим разрешения:

```
$ sudo mysql -u root -p
Enter password:
MariaDB [(none)]> CREATE DATABASE placement;
Query OK, 1 row affected (0.00 sec)
MariaDB [(none)]> GRANT ALL PRIVILEGES ON placement.* TO 'placement'@'localhost' IDENTIFIED BY 'placement';
Query OK, 0 rows affected (0.00 sec)
MariaDB [(none)]> GRANT ALL PRIVILEGES ON placement.* TO 'placement'@'%' IDENTIFIED BY 'placement';
Query OK, 0 rows affected (0.00 sec)
```

Так-как в **Keystone** мы **placement** зарегистрировали, пропускаем этот шаг и заполняем файл конфига **/etc/placement/placement.conf**:

```
$ sudo crudini --set /etc/placement/placement.conf placement_database connection
mysql+pymysql://placement:placement@controller/placement
$ sudo crudini --set /etc/placement/placement.conf api auth_strategy keystone
$ sudo crudini --set /etc/placement/placement.conf keystone_authtoken auth_url
http://controller.test.local:35357/v3
$ sudo crudini --set /etc/placement/placement.conf keystone_authtoken auth_type password
$ sudo crudini --set /etc/placement/placement.conf keystone_authtoken project_domain_name default
$ sudo crudini --set /etc/placement/placement.conf keystone_authtoken user_domain_name default
$ sudo crudini --set /etc/placement/placement.conf keystone_authtoken project_name service
$ sudo crudini --set /etc/placement/placement.conf keystone_authtoken username placement
$ sudo crudini --set /etc/placement/placement.conf keystone_authtoken password openstack
```

```
$ sudo vi /etc/placement/placement.conf
...
[placement_database]
connection = mysql+pymysql://placement:placement@controller/placement
...
[api]
auth_strategy = keystone
...
[keystone_authtoken]
auth_url = http://controller.test.local:35357/v3
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = placement
password = openstack
```

Синхронизируем БД **placement**:

```
$ sudo -s /bin/sh -c "placement-manage db sync" placement
```

8.4. Запускаем все службы **Nova** и перезапускаем **httpd** (на счет **openstack-nova-console** пока не знаю, поэтому не запускаю, так как в официальном гайде **Openstack** (не из книги) **openstack-nova-console** даже не скачивается):

```
$ sudo systemctl enable openstack-nova-api.service
Created symlink from /etc/systemd/system/multi-user.target.wants/openstack-nova-
api.service to /usr/lib/systemd/system/openstack-nova-api.service.
$ sudo systemctl enable openstack-nova-scheduler.service
Created symlink from /etc/systemd/system/multi-user.target.wants/openstack-nova-
scheduler.service to /usr/lib/systemd/system/openstack-nova-scheduler.service.
$ sudo systemctl enable openstack-nova-conductor.service
Created symlink from /etc/systemd/system/multi-user.target.wants/openstack-nova-
conductor.service to /usr/lib/systemd/system/openstack-nova-conductor.service.
$ sudo systemctl enable openstack-nova-novncproxy.service
Created symlink from /etc/systemd/system/multi-user.target.wants/openstack-nova-
novncproxy.service to /usr/lib/systemd/system/openstack-nova-novncproxy.service.
$ sudo systemctl start openstack-nova-api.service
$ sudo systemctl start openstack-nova-scheduler.service
$ sudo systemctl start openstack-nova-conductor.service
$ sudo systemctl start openstack-nova-novncproxy.service
$ sudo systemctl restart httpd.service
```

```
$ sudo systemctl status openstack-nova-api.service
● openstack-nova-api.service - OpenStack Nova API Server
  Loaded: loaded (/usr/lib/systemd/system/openstack-nova-api.service; enabled; vendor
  preset: disabled)
  Active: active (running) since Пт 2022-12-02 17:58:47 MSK; 1min 29s ago
$ sudo systemctl status openstack-nova-scheduler.service
● openstack-nova-scheduler.service - OpenStack Nova Scheduler Server
  Loaded: loaded (/usr/lib/systemd/system/openstack-nova-scheduler.service; enabled;
  vendor preset: disabled)
  Active: active (running) since Пт 2022-12-02 17:58:19 MSK; 2min 18s ago
$ sudo systemctl status openstack-nova-conductor.service
● openstack-nova-conductor.service - OpenStack Nova Conductor Server
  Loaded: loaded (/usr/lib/systemd/system/openstack-nova-conductor.service; enabled;
  vendor preset: disabled)
  Active: active (running) since Пт 2022-12-02 17:58:06 MSK; 2min 47s ago
$ sudo systemctl status openstack-nova-novncproxy.service
● openstack-nova-novncproxy.service - OpenStack Nova NoVNC Proxy Server
  Loaded: loaded (/usr/lib/systemd/system/openstack-nova-novncproxy.service; enabled;
  vendor preset: disabled)
  Active: active (running) since Пт 2022-12-02 17:57:44 MSK; 4min 47s ago
$ sudo systemctl status httpd.service
● httpd.service - The Apache HTTP Server
  Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset:
  disabled)
  Active: active (running) since Пт 2022-12-02 13:54:25 MSK; 4h 8min ago
```

8.5. На этом настройка сервиса **Nova** на **controller.test.local** закончена и можно переходить к настройке рабочих узлов (все настройки проводятся на **compute.test.local**, **compute-opt.test.local** аналогично, но с другими ip).

Первым делом устанавливаем пакеты **openstack-nova-compute** (служба **Nova** для рабочих узлов) **sysfsutils** (утилита получения информации о ВМ) **libvirt** (утилита виртуализации), если не качается фикс [здесь](#):

```
$ sudo yum -y install openstack-nova-compute sysfsutils libvirt
```

Далее необходимо проверить поддержку аппаратной виртуализации на рабочем узле (**svm** — поддержка от Amd, **vmx** — от Intel):

```
$ sudo grep -E ' svm | vmx' /proc/cpuinfo
flags      : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca
cmov pat pse36 clflush mmx fxsr sse sse2 ss syscall nx pdpe1gb rdtscp lm
constant_tsc arch_perfmon rep_good nopl xtTopology eagerfpu pni pclmulqdq
vmx ssse3 fma cx16 pdcm pcid sse4_1 sse4_2 x2apic movbe popcnt
tsc_deadline_timer aes xsave avx f16c rdrand hypervisor lahf_lm abm
3dnowprefetch invpcid_single ssbd ibrs ibpb stibp tpr_shadow vnmi
flexpriority ept vpid fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms invpcid
mpx rdseed adx smap clflushopt xsaveopt xsaves xgetbv1 arat umip md_clear
spec_ctrl intel_stibp arch_capabilities
flags      : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca
cmov pat pse36 clflush mmx fxsr sse sse2 ss syscall nx pdpe1gb rdtscp lm
constant_tsc arch_perfmon rep_good nopl xtTopology eagerfpu pni pclmulqdq
vmx ssse3 fma cx16 pdcm pcid sse4_1 sse4_2 x2apic movbe popcnt
tsc_deadline_timer aes xsave avx f16c rdrand hypervisor lahf_lm abm
3dnowprefetch invpcid_single ssbd ibrs ibpb stibp tpr_shadow vnmi
flexpriority ept vpid fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms invpcid
mpx rdseed adx smap clflushopt xsaveopt xsaves xgetbv1 arat umip md_clear
spec_ctrl intel_stibp arch_capabilities
```

Если вдруг аппаратная виртуализация не поддерживается (пустой вывод команды выше), то необходимо включить эмуляцию с помощью **QEMU**:

```
$ sudo crudini --set /etc/nova/nova.conf libvirt virt_type qemu
```

Теперь настраиваем конфигурационный файл сервиса **Nova** **/etc/nova/nova.conf**, для начала базовые параметры для всех узлов, о которых упоминалось [ранее](#):

```
$ sudo vi /etc/nova/nova.conf
[DEFAULT]
transport_url = rabbit://openstack:openstack@controller.test.local
enabled_apis = osapi_compute,metadata
use_neutron = True
firewall_driver = nova.virt.firewall.NoopFirewallDriver
...
[api]
auth_strategy = keystone
...
[keystone_auth_token]
project_name = service
user_domain_name = default
project_domain_name = default
auth_type = password
username = nova
password = openstack
www_authenticate_uri = http://controller.test.local:5000
auth_url = http://controller.test.local:35357
...
[glance]
api_servers = http://controller.test.local:9292
...
[cinder]
os_region_name = RegionOne
...
[placement]
project_name = service
user_domain_name = default
project_domain_name = default
auth_type = password
username = placement
password = openstack
auth_url = http://controller.test.local:35357/v3
region_name = RegionOne
...
[oslo_concurrency]
lock_path = /var/lib/nova/tmp
```

Теперь устанавливаем параметры **/etc/nova/nova.conf** специфичные для рабочих узлов:

Указываем адрес вычислительного узла (**compute-opt — 192.168.122.215**):

```
$ sudo crudini --set /etc/nova/nova.conf DEFAULT my_ip 192.168.122.210
```

Включаем поддержку vnc, а также указываем что **VNC-сервер** будет слушать подключения на всех интерфейсах и URL прокси-сервера, где будет доступен браузеру интерфейс виртуальной машины:

```
$ sudo crudini --set /etc/nova/nova.conf vnc vnc_enabled True  
$ sudo crudini --set /etc/nova/nova.conf vnc vncserver_listen 0.0.0.0  
$ sudo crudini --set /etc/nova/nova.conf vnc novncproxy_base_url  
http://controller.test.local:6080/vnc_auto.html
```

Указываем адрес узла-клиента nvcproxy (**compute-opt — 192.168.122.215**):

```
$ sudo crudini --set /etc/nova/nova.conf vnc vncserver_proxyclient_address  
192.168.122.210
```

На этом работа с конфигом заканчивается и можно запускать сервисы **libvirtd** **openstack-nova-compute** (**openstack-nova-compute** не запустится, если **controller** узел не в сети):

```
$ sudo systemctl enable libvirtd.service  
$ sudo systemctl enable openstack-nova-compute.service  
$ sudo systemctl start libvirtd.service  
$ sudo systemctl start openstack-nova-compute.service  
$ sudo systemctl status libvirtd.service  
● libvirtd.service - Virtualization daemon  
  Loaded: loaded (/usr/lib/systemd/system/libvirtd.service; enabled;  
  vendor preset: enabled)  
    Active: active (running) since Вс 2022-12-04 16:56:53 MSK; 3min 4s ago  
$ sudo systemctl status openstack-nova-compute.service  
● openstack-nova-compute.service - OpenStack Nova Compute Server  
  Loaded: loaded (/usr/lib/systemd/system/openstack-nova-compute.service;  
  enabled; vendor preset: disabled)  
    Active: active (running) since Вс 2022-12-04 16:57:13 MSK; 2min 21s ago
```

8.6. Теперь проверяем что мы там настроили и заканчиваем синхронизацию на **controller** узле:

Проверяем видит ли **controller** узел наш **compute** узел:

```
$ sudo openstack compute service list  
+-----+-----+-----+-----+-----+-----+  
| ID | Binary      | Host           | Zone   | Status | State | Updated At   |  
+-----+-----+-----+-----+-----+-----+  
| 1  | nova-conductor | controller.test.local | internal | enabled | up    | 2022-12-04T14:02:36.000000 |  
| 2  | nova-scheduler | controller.test.local | internal | enabled | up    | 2022-12-04T14:02:37.000000 |  
| 5  | nova-compute   | compute.test.local   | nova    | enabled | up    | 2022-12-04T14:02:42.000000 |  
+-----+-----+-----+-----+-----+-----+
```

Если видит, то регистрируем гипервизоры с помощью скрипта (вывод примерно такой, честно скажу, свой вывод для **compute.test.local** я потерял, так что взял вывод после подключения **compute-opt.test.local**):

```
$ sudo -s /bin/sh -c "nova-manage cell_v2 discover_hosts --verbose" nova
Found 2 cell mappings.
Skipping cell0 since it does not contain hosts.
Getting computes from cell 'cell1': 1d26f0d0-e23b-4e62-87b4-95945eb34ef2
Checking host mapping for compute host 'compute-opt.test.local':
2b1ae767-f0fb-496c-be57-e130e5b97e29
Creating host mapping for compute host 'compute-opt.test.local':
2b1ae767-f0fb-496c-be57-e130e5b97e29
Found 1 unmapped computes in cell: 1d26f0d0-e23b-4e62-87b4-95945eb34ef2
```

Теперь можно проверить работу **Placement API** с помощью еще одного скрипта:

```
$ sudo -s /bin/sh -c "nova-status upgrade check" nova
+-----+
| Upgrade Check Results |
+-----+
| Check: Cells v2
| Result: Success
| Details: None
+-----+
| Check: Placement API
| Result: Success
| Details: None
+-----+
| Check: Ironic Flavor Migration
| Result: Success
| Details: None
+-----+
| Check: Cinder API
| Result: Success
| Details: None
+-----+
```

Если скрипт выводит ошибку 403 — то вы такой же как и я (хотя как может быть по другому, если вы шли по этому гайду), так вот ошибка решается добавлением секции **<Directory /usr/bin>** в конфигурационный файл **httpd** для **Placement API** **/etc/httpd/conf.d/00-placement-api.conf** и последующим перезапуском сервиса **httpd**:

```
$ sudo vi /etc/httpd/conf.d/00-placement-api.conf
Listen 8778

<VirtualHost *:8778>
    WSGIProcessGroup placement-api
    WSGIApplicationGroup %{GLOBAL}
    WSGIPassAuthorization On
    WSGIDaemonProcess placement-api processes=3
    threads=1 user=placement group=placement
    WSGIScriptAlias / /usr/bin/placement-api
    <IfVersion >= 2.4>
        ErrorLogFormat "%M"
    </IfVersion>
    ErrorLog /var/log/placement/placement-api.log
    #SSLEngine On
    #SSLCertificateFile ...
    #SSLCertificateKeyFile ...
    <Directory /usr/bin>
        <IfVersion >= 2.4>
            Require all granted
        </IfVersion>
        <IfVersion < 2.4>
            Order allow,deny
            Allow from all
        </IfVersion>
    </Directory>
</VirtualHost>

Alias /placement-api /usr/bin/placement-api
<Location /placement-api>
    SetHandler wsgi-script
    Options +ExecCGI
    WSGIProcessGroup placement-api
    WSGIApplicationGroup %{GLOBAL}
    WSGIPassAuthorization On
</Location>
$ sudo systemctl restart httpd.service
```

На этом настройка сервиса **Nova** закончилась (нашел [крутой современный гайд](#) на установку **Openstack**), теперь экспромтом покажу как настроить **nfs-клиента** на **compute-opt** узле, так как автоматически он его не создаст:

Во-первых вот так по итогу выглядит итоговый **/etc/hosts**, каждого узла нашего проекта:

```
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.122.200 controller.test.local controller
192.168.122.210 compute.test.local compute
192.168.122.215 compute-opt.test.local compute-opt
```

Скачиваем и запускаем **nfs-server**:

```
$ sudo yum -y install nfs-utils
$ sudo systemctl enable nfs-server
$ sudo systemctl start nfs-server
```

Создаем папку, в которую мы смонтируем **nfs-каталог**, и собственно монтируем в нее **nfs-каталог** нашего **nfs-сервера**:

```
$ sudo mkdir /backup
$ sudo mount -t nfs 192.168.122.210:/backup/nfs/ /backup
```

Для того чтобы каталог автоматически монтировался, добавим соответствующую строку в файл **/etc/fstab** и применим конфигурацию:

```
$ sudo vi /etc/fstab
...
192.168.122.210:/backup/nfs/ /backup/ nfs rw,sync,hard,intr 0 0
$ sudo mount -a
```

- **rw** — разрешение на чтение и запись
- **sync** — все изменения сразу сбрасываются на диск без использования кэша
- **hard** — клиент будет пинговать nfs-сервер, если тот станет недоступен
- **intr** — монтирование можно прервать

9. Установка и настройка сетевого сервиса Neutron

9.1. Про сервис Neutron написано много, так что введение будет большим. Для начала **Neutron** — реализует коммутацию (**NetFlow**, **RSPAN**, **SPAN**, **LACP**, **802.1q**, туннелирование **GRE** и **VXLAN**), балансировку нагрузки, брандмауэр и **VPN** в кластере **Openstack** (раз он сетевой сервис). Определим основные абстракции сервиса:

- **сеть** — есть внутренние, виртуальные сети, которых может быть много, и как минимум одна внешняя. Доступ к виртуальным машинам внутри внутренней сети могут получить только машины в этой же сети или узлы, связанные через виртуальные маршрутизаторы. Внешняя сеть представляет собой отображение части реально существующей физической сети для обеспечения сетевой связанности виртуальных машин внутри облака и «внешнего мира».
- **подсеть** — сеть должна иметь ассоциированные с ней подсети. Именно через подсеть задается конкретный диапазон IP-адресов
- **маршрутизатор** — как и в физическом мире, служит для маршрутизации между сетями.
- **группа безопасности** — набор правил брандмауэра, применяющихся к виртуальным машинам в этой группе
- **«плавающий IP-адрес»** (Floating IP) — IP-адрес внешней сети, назначаемый экземпляру виртуальной машины. Он может быть выделен только из существующей внешней сети (по умолчанию каждый проект имеет квоту в 50 адресов)
- **порт** — подключение к подсети. Порт на виртуальном коммутаторе. Включает в себя MAC-адрес и IP-адрес

Соответственно в кластере приходится иметь дело со следующими видами трафика:

- **внешний трафик** — публичная сеть, которой принадлежат «реальные» IP-адреса виртуальных машин, а также демилитаризованная зона
- **внутренний трафик** — сеть для операционных систем и служб (ssh и мониторинг), сеть для сетевой установки узлов под сервисы OpenStack, сеть

для **IPMI**, **DRAC**, **iLO**, консолей коммутаторов и сеть передачи данных для служб **SDS** (**Swift**, **Ceph**, **iSCSI**, **NFS**)

- **трафик виртуальных машин** — сеть связи внешнего трафика с ВМ
- **трафик API и управления** — сеть, предоставляющая наружу публичный API и веб-интерфейс **Horizon**

Для функционирования Neutron нужно минимум три узла:

- **узел управления** — узел с брокером сообщений (**controller.test.local**)
- **сетевой узел** — производительный физический сервер или физический коммутатор
- **вычислительный узел** — некоторые службы Neutron добавляются на каждый рабочий узел (**compute.test.local** и **compute-opt.test.local**)

Почему все используют **Neutron** вместо **nova-network**? Потому что в **Neutron** можно расширять API при помощи подключаемых модулей (в **nova-network** для настройки сети использовались только возможности ОС на которой работает узел), при этом у **Neutron** есть супер модуль **Modular Layer 2 (ML2 – OVS/LB)**, позволяющий использовать сразу несколько технологий 2 уровня (раньше можно было только **LinuxBridge** или только **Open vSwitch**) путем использования специальных драйверов (присутствуют из коробки):

- драйвера агентов, например **LinuxBridge** или **OVS**
- драйвера контроллеров **SDN**, например **OpenDaylight**, **OpenContrail**, **VMware NSX** или **PLUMgrid**
- драйвера аппаратных коммутаторов, например **Cisco Nexus** или **Extreme Networks**

Также можно качать сторонние драйвера или разворачивать сервисы:

- **маршрутизатор**
- **балансировщик нагрузки (LBaaS)**
- **брандмауэр (FaaS)**
- **виртуальные частные сети (VPNaas)**

Наконец на 65 странице можно частично понять смысл проектов **Openstack** — каждый проект может иметь свою сеть, соответственно один узел может иметь несколько адресных пространств которые даже могут пересекаться (несколько ARP таблиц и таблиц маршрутизации, наборов правил брандмауэра, сетевых устройств и т. д.).

Что касается служб **Neutron**:

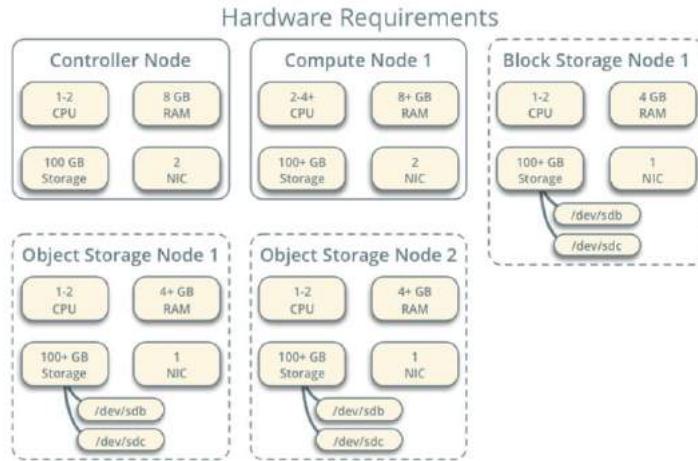
- **neutron-server** — центральный управляющий компонент. Не занимается непосредственно маршрутизацией пакетов. С остальными компонентами взаимодействует через брокер сообщений (**сидит на узле управления**)

- **neutron-openvswitch-agent** – взаимодействует с **neutron-server** через брокер сообщений и отдает команды **OVS** для построения таблицы потоков (**сидит на сетевом и рабочих узлах**)
- **neutron-l3-agent** – обеспечивает маршрутизацию и NAT, используя технологию сетевых пространств имен (**сидит на сетевом узле**)
- **openvswitch** – программный коммутатор, используемый для построения сетей (**сидит на сетевом и рабочих узлах**)
- **neutron-dhcp-agent** – сервис отвечает за запуск и управление процессами **dnsmasq**, легковесного **dhcp-сервера** и сервиса кэширования DNS. Также **neutron-dhcp-agent** отвечает за запуск прокси-процессов сервера предоставления метаданных (каждая сеть, создаваемая агентом, получает собственное пространство имен **qdhcp-UUID_сети**) (**сидит на сетевом узле**)
- **neutron-metadata-agent** – данный сервис позволяет виртуальным машинам запрашивать данные о себе, такие как имя узла, открытый **ssh-ключ** для аутентификации и др (ВМ получают эту информацию во время загрузки скриптом, обращаясь на адрес **http://169.254.169.254**. Агент проксирует соответствующие запросы к **openstack-nova-api** при помощи пространства имен маршрутизатора или **DHCP**) (**сидит на сетевом узле**)
- **neutron-ovs-cleanup** – отвечает во время старта за удаление из базы данных **OVS** неиспользуемых мостов «старых» виртуальных машин (**сидит на сетевом узле**)

Итак что происходит в **Neutron** при создании ВМ через сервис **Nova**:

1. Сервис **Nova** отправляет запрос на сервис **neutron-server**, отвечающий за API
2. **Neutron-server** отправляет запрос агенту DHCP (**neutron-dhcp-agent**) на создание IP-адреса
3. **Neutron-dhcp-agent** обращается к сервису **dnsmasq**, отвечающему за подсеть, в которой создается виртуальная машина
4. **Dnsmasq** возвращает первый свободный IP-адрес из диапазона адресов подсети
5. **Neutron-dhcp-agent** отправляет этот адрес сервису **neutron-server**
6. После того как за виртуальной машиной закреплен IP-адрес, сервис **Neutron** отправляет запрос на **Open vSwitch** для создания конфигурации, включающей IP-адрес в существующую сеть.
7. **Open vSwitch** возвращает обратно параметры конфигурации на сервис **Neutron** при помощи шины сообщений
8. **Neutron** отправляет параметры конфигурации сервису **Nova**
9. Конец

Небольшая сноска перед следующим пунктом, я тут нашел рекомендуемые требования к узлам... (у меня всего 8GB, возможно в скором времени мне это все еще как аукнется, как минимум **Swift** и **Ceph** даже если бы хотел развернуть не смог), в такие моменты внезапно понимаешь, что по идеи эти узлы должны одновременно запущены 24/7 и их не надо останавливать, чтобы перестать ловить фризы всей системы, а ведь впереди еще как минимум 3 сервиса.



9.2. Начинаем установку и настройку сервиса **Neutron** на узле управления (**controller.test.local**):

На всякий случай сделаем снимок состояния виртуалки, а то мне кажется, что грань великих тормозов все ближе:

```
$ sudo virsh snapshot-create-as --domain VM2 --name v2-snapshot
$ sudo virsh snapshot-list VM2
Name          Creation Time           State
v2-snapshot   2022-12-05 20:45:25 +0300  shutoff
```

Устанавливаем пакеты **openstack-neutron** (службы **Neutron**) **openstack-neutron-ml2** (тот самый [волшебный модуль](#)):

```
$ sudo yum -y install openstack-neutron openstack-neutron-ml2
```

Классическое создание БД для сервиса, на это раз как нетрудно понять БД зовется **neutron**:

```
$ sudo mysql -u root -p
Enter password:
MariaDB [(none)]> CREATE DATABASE neutron;
Query OK, 1 row affected (0.00 sec)
MariaDB [(none)]> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'localhost'
IDENTIFIED BY 'neutron';
Query OK, 0 rows affected (0.01 sec)
MariaDB [(none)]> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'%'
IDENTIFIED BY 'neutron';
Query OK, 0 rows affected (0.00 sec)
```

Далее... да вы угадали — регистрируем сервис в **Keystone**:

```

$ sudo source keystonerc_adm
$ sudo openstack user create --domain default --password openstack neutron
+-----+
| Field      | Value
+-----+
| domain_id | default
| enabled    | True
| id         | 95d5cb57dcc34b2e9c74ccaddf9fd2cd
| name       | neutron
| options    | {}
| password_expires_at | None
+-----+
$ sudo openstack role add --project service --user neutron admin
$ sudo openstack service create --name neutron --description "OpenStack
Networking" network
+-----+
| Field      | Value
+-----+
| description | OpenStack Networking
| enabled    | True
| id         | 3f8f40d4d5ca49da8997082156391240
| name       | neutron
| type       | network
+-----+

```

```

$ sudo openstack endpoint create --region RegionOne network public
http://controller.test.local:9696
+-----+
| Field      | Value
+-----+
| enabled    | True
| id         | 612c55853e1c464a814a7820216b626e
| interface  | public
| region     | RegionOne
| region_id  | RegionOne
| service_id | 3f8f40d4d5ca49da8997082156391240
| service_name| neutron
| service_type| network
| url        | http://controller.test.local:9696
+-----+
$ sudo openstack endpoint create --region RegionOne network internal
http://controller.test.local:9696
+-----+
| Field      | Value
+-----+
| enabled    | True
| id         | 448163e58b3f4c338e01354fc3339320
| interface  | internal
| region     | RegionOne
| region_id  | RegionOne
| service_id | 3f8f40d4d5ca49da8997082156391240
| service_name| neutron
| service_type| network
| url        | http://controller.test.local:9696
+-----+
$ sudo openstack endpoint create --region RegionOne network admin
http://controller.test.local:9696
+-----+
| Field      | Value
+-----+
| enabled    | True
| id         | 3b566d97f6a44de085d3ad480eb7cc98
| interface  | admin
| region     | RegionOne
| region_id  | RegionOne
| service_id | 3f8f40d4d5ca49da8997082156391240
| service_name| neutron
| service_type| network
| url        | http://controller.test.local:9696
+-----+

```

Далее идут общие параметры конфигурационных файлов для сетевого, рабочих и управляющего узлов. **Neutron** — король по количеству настроек, вот описание файлов конфигураций, которые мы будем изменять:

- **/etc/neutron/neutron.conf** — основной конфиг **Neutron**
- **/etc/neutron/plugins/ml2/ml2_conf.ini** — конфиг модуля **ML2**
- **/etc/neutron/plugin.ini** — ссылка на конфиг **ML2**, там прикол, что если не будет ссылки то конфиг **ML2** не подключится, так как находится в другой папке.
- **/etc/neutron/l3_agent.ini** — конфиг **L3-агента** (отвечает за маршрутизацию)
- **/etc/neutron/dhcp_agent.ini** — конфиг **DHCP-агента**
- **/etc/neutron/metadata_agent.ini** — конфиг агента, предоставляющего метаданные виртуальным машинам
- **/etc/neutron/plugins/ml2/openvswitch_agent.ini** — конфиг агента **Open vSwitch** для модуля **ML2**

Ну что приступаем к настройке **/etc/neutron/neutron.conf**, первым делом настраиваем базовые вещи (подключение к брокеру сообщений и аутентификацию **Keystone**):

```
$ sudo crudini --set /etc/neutron/neutron.conf DEFAULT
transport_url rabbit://openstack:openstack@controller.test.local
$ sudo crudini --set /etc/neutron/neutron.conf DEFAULT auth_strategy
keystone
$ sudo crudini --set /etc/neutron/neutron.conf keystone_authtoken
project_name service
$ sudo crudini --set /etc/neutron/neutron.conf keystone_authtoken
user_domain_name default
$ sudo crudini --set /etc/neutron/neutron.conf keystone_authtoken
project_domain_name default
$ sudo crudini --set /etc/neutron/neutron.conf keystone_authtoken
auth_type password
$ sudo crudini --set /etc/neutron/neutron.conf keystone_authtoken
username neutron
$ sudo crudini --set /etc/neutron/neutron.conf keystone_authtoken
password openstack
$ sudo crudini --set /etc/neutron/neutron.conf keystone_authtoken
www_authentication_uri http://controller.test.local:5000
$ sudo crudini --set /etc/neutron/neutron.conf keystone_authtoken
auth_url http://controller.test.local:35357
```

Указываем название основного подключаемого модуля второго уровня модели OSI (также есть : **hyperv**, **cisco**, **brocade**, **embrance**, **vmware**, **nec** и др):

```
$ sudo crudini --set /etc/neutron/neutron.conf DEFAULT core_plugin ml2
```

Указываем погружаемые модули (из **router**, **firewall**, **lbaas**, **vpnaas**, **metering**), подгружаем только маршрутизатор:

```
$ sudo crudini --set /etc/neutron/neutron.conf DEFAULT service_plugins router
```

Настройка разрешения пересечения IP внутри проектов:

```
$ sudo crudini --set /etc/neutron/neutron.conf DEFAULT allow_overlapping_ips True
```

Забыл настроить путь к файлам блокировки:

```
$ sudo crudini --set /etc/neutron/neutron.conf oslo_concurrency lock_path /var/lib/neutron/tmp
```

Настройка **/etc/neutron/neutron.conf** закончена (вроде как), теперь настраиваем **/etc/neutron/plugins/ml2/ml2_conf.ini**:

Настраиваем используемые сегменты сетей (выбор из: **flat, GRE, VLAN, VXLAN** и **local**):

```
$ sudo crudini --set /etc/neutron/plugins/ml2/ml2_conf.ini ml2 type_drivers flat,gre,vlan,vxlan
```

Настраиваем технологии используемые для сетей проектов (поддерживающие пересечение IP: **GRE, VLAN, VXLAN** и **flat**):

```
$ sudo crudini --set /etc/neutron/plugins/ml2/ml2_conf.ini ml2 tenant_network_types gre
```

Настраиваем тип драйвера механизма **ML2** (выбор из: **openvswitch, mlnx, arista, cisco, logger, linuxbridge** и **brocade**):

```
$ sudo crudini --set /etc/neutron/plugins/ml2/ml2_conf.ini ml2 mechanism_drivers openvswitch
```

Настраиваем диапазон ID для **GRE-туннелей**:

```
$ sudo crudini --set /etc/neutron/plugins/ml2/ml2_conf.ini ml2_type_gre tunnel_id_ranges 1:1000
```

Включаем группы безопасности **ipset**:

```
$ sudo crudini --set /etc/neutron/plugins/ml2/ml2_conf.ini securitygroup enable_security_group True  
$ sudo crudini --set /etc/neutron/plugins/ml2/ml2_conf.ini securitygroup enable_ipset True
```

```
$ sudo vi /etc/neutron/neutron.conf
[DEFAULT]
transport_url = rabbit://openstack:openstack@controller.test.local
auth_strategy = keystone
service_plugins = router
core_plugin = ml2
allow_overlapping_ips = True
...
[keystone_authtoken]
project_name = service
user_domain_name = default
project_domain_name = default
auth_type = password
username = neutron
password = openstack
www_authentication_uri = http://controller.test.local:5000
auth_url = http://controller.test.local:35357
...
[oslo_concurrency]
lock_path = /var/lib/neutron/tmp
$ sudo vi /etc/neutron/plugins/ml2/ml2_conf.ini
...
[securitygroup]
enable_security_group = True
enable_ipset = True
...
[ml2_type_gre]
tunnel_id_ranges = 1:1000
...
[ml2]
mechanism_drivers = openvswitch
tenant_network_types = gre
type_drivers = flat,gre,vlan,vxlan
```

Выше были настройки общие для всех узлов, теперь пойдут настройки только для управляющего узла:

Настраиваем параметры подключения к БД:

```
$ sudo crudini --set /etc/neutron/neutron.conf database connection
mysql://neutron:neutron@controller.test.local/neutron
```

Настраиваем отправку уведомлений сервису **Nova** при изменении статуса портов и IP-адресов:

```
$ sudo crudini --set /etc/neutron/neutron.conf DEFAULT notify_nova_on_port_status_changes
True
$ sudo crudini --set /etc/neutron/neutron.conf DEFAULT notify_nova_on_port_data_changes
True
```

Настраиваем параметры подключения к **Keystone** сервиса **Nova** (видимо **Neutron** нужно в определенных обстоятельствах нужно выдавать себя за **Nova**):

```
$ sudo crudini --set /etc/neutron/neutron.conf nova auth_url http://controller.test.local:35357
$ sudo crudini --set /etc/neutron/neutron.conf nova auth_type password
$ sudo crudini --set /etc/neutron/neutron.conf nova project_domain_name default
$ sudo crudini --set /etc/neutron/neutron.conf nova user_domain_name default
$ sudo crudini --set /etc/neutron/neutron.conf nova region_name RegionOne
$ sudo crudini --set /etc/neutron/neutron.conf nova project_name service
$ sudo crudini --set /etc/neutron/neutron.conf nova username nova
$ sudo crudini --set /etc/neutron/neutron.conf nova password openstack
```

Переносим все заботы о сети с сервиса **Nova** на **Neutron** в файле **/etc/nova/nova.conf** (еще брандмауэр **Nova** вырубаем), третью команду отдельно выносить не буду, там мы задаем поддержку **Open vSwitch**:

```
$ sudo crudini --set /etc/nova/nova.conf DEFAULT network_api_class nova.network.neutronv2.api.API  
$ sudo crudini --set /etc/nova/nova.conf DEFAULT security_group_api neutron  
$ sudo crudini --set /etc/nova/nova.conf DEFAULT linuxnet_interface_driver nova.network.linux_net.LinuxOVSInterfaceDriver  
$ sudo crudini --set /etc/nova/nova.conf DEFAULT firewall_driver nova.virt.firewall.NoopFirewallDriver
```

Настраиваем параметры аутентификации **Neutron** для **Nova**, если раньше мы писали их в **/etc/neutron/neutron.conf**, то теперь в **/etc/nova/nova.conf**:

```
$ sudo crudini --set /etc/nova/nova.conf neutron url http://controller.test.local:9696  
$ sudo crudini --set /etc/nova/nova.conf neutron auth_url http://controller.test.local:35357  
$ sudo crudini --set /etc/nova/nova.conf neutron auth_type password  
$ sudo crudini --set /etc/nova/nova.conf neutron project_domain_name default  
$ sudo crudini --set /etc/nova/nova.conf neutron user_domain_name default  
$ sudo crudini --set /etc/nova/nova.conf neutron region_name RegionOne  
$ sudo crudini --set /etc/nova/nova.conf neutron project_name service  
$ sudo crudini --set /etc/nova/nova.conf neutron username neutron  
$ sudo crudini --set /etc/nova/nova.conf neutron password openstack
```

Теперь можно синхронизировать БД:

```
$ sudo -s /bin/sh -c "neutron-db-manage --config-file /etc/neutron/neutron.conf --config-file /etc/neutron/plugins/ml2/ml2_conf.ini upgrade head" neutron
```

У меня вылезла ошибка «**No module named MySQLdb**», почему только сейчас известно только разрабам **Openstack** (хотя вывод скрипта огромный, мб он единственный использует это модуль), но фиксится она как нетрудно догадаться скачиванием данного модуля (после снапшота, я теперь готов все что угодно качать):

```
$ sudo yum -y install MySQL-python
```

Теперь создаем ссылку на конфиг ML2 в папке настроек Neutron **/etc/neutron/**:

```
$ sudo ln -s /etc/neutron/plugins/ml2/ml2_conf.ini /etc/neutron/plugin.ini
```

Указываем использование прокси для сервера метаданных и общий секрет, которым будут подписываться запросы к серверу метаданных в конфиге **/etc/nova/nova.conf**:

```
$ sudo crudini --set /etc/nova/nova.conf neutron service_metadata_proxy True  
$ sudo crudini --set /etc/nova/nova.conf neutron metadata_proxy_shared_secret openstack
```

По итогу на узле управления в конфиги добавляются следующие параметры:

```

$ sudo vi /etc/neutron/neutron.conf
[DEFAULT]
...
notify_nova_on_port_status_changes = True
notify_nova_on_port_data_changes = True
...
[database]
connection = mysql://neutron:neutron@controller.test.local/neutron
...
[nova]
auth_url = http://controller.test.local:35357
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = nova
password = openstack
$ sudo vi /etc/nova/nova.conf
[DEFAULT]
...
network_api_class = nova.network.neutronv2.api.API
security_group_api = neutron
linuxnet_interface_driver = nova.network.linux_net.LinuxOVSInterfaceDriver
...
[neutron]
url = http://controller.test.local:9696
auth_url = http://controller.test.local:35357
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = neutron
password = openstack
service_metadata_proxy = True
metadata_proxy_shared_secret = openstack

```

Перезапускаем сервисы **nova** и стартуем сервис **neutron-server**:

```

$ sudo systemctl restart openstack-nova-api.service
$ sudo systemctl restart openstack-nova-scheduler.service
$ sudo systemctl restart openstack-nova-conductor.service
$ sudo systemctl enable neutron-server.service
Created symlink from /etc/systemd/system/multi-user.target.wants/neutron-
server.service to /usr/lib/systemd/system/neutron-server.service.
$ sudo systemctl start neutron-server.service

```

Проверка итога настройки:

```

$ sudo neutron ext-list
neutron CLI is deprecated and will be removed in the future. Use openstack CLI instead.
+-----+-----+
| alias | name |
+-----+-----+
| subnetpool-prefix-ops | Subnet Pool Prefix Operations |
| default-subnetpools | Default Subnetpools |
| availability_zone | Availability Zone |
| network_availability_zone | Network Availability Zone |
...

```

9.3. Теперь я оказался на перепутье ведь надо настраивать сетевой узел как отдельный узел, а ноутбук уже просит пощады, так что мой выбор пал на разворачивание сетевого узла на узле управления, может выйти гениально, а может и нет (но я ведь не зря снимок сделал). Далее я буду писать как будто настраиваю отдельный узел, но делать пометки лишних действий, так как некоторые уже делал при настройке узла управления:

Начнем с установки необходимых пакетов (если как и я устанавливаете на узел управления, то **openstack-neutron** и **openstack-neutron-ml2** качать не надо)

openstack-neutron-openvswitch (модуль openvswitch для openstack) и [openvswitch](#) (программный коммутатор):

```
$ sudo yum -y install openstack-neutron openstack-neutron-ml2 openstack-neutron-openvswitch openvswitch
```

Далее идет установка всех общих параметров конфигов (на узле управления уже установлены) и создание ссылки на конфиг в **/etc/neutron/plugins/ml2/ml2_conf.ini**:

```
$ sudo vi /etc/neutron/neutron.conf
[DEFAULT]
transport_url = rabbit://openstack:openstack@controller.test.local
auth_strategy = keystone
service_plugins = router
core_plugin = ml2
allow_overlapping_ips = True
...
[keystone_authtoken]
project_name = service
user_domain_name = default
project_domain_name = default
auth_type = password
username = neutron
password = openstack
www_authentication_uri = http://controller.test.local:5000
auth_url = http://controller.test.local:35357
...
[oslo_concurrency]
lock_path = /var/lib/neutron/tmp
$ sudo vi /etc/neutron/plugins/ml2/ml2_conf.ini
...
[securitygroup]
enable_security_group = True
enable_ipset = True
...
[ml2_type_gre]
tunnel_id_ranges = 1:1000
...
[ml2]
mechanism_drivers = openvswitch
tenant_network_types = gre
type_drivers = flat,gre,vlan,vxlan
```

```
$ sudo ln -s /etc/neutron/plugins/ml2/ml2_conf.ini /etc/neutron/plugin.ini
```

Дальше уже страшно, так как нужно внести изменения в параметры ядра, а именно включить маршрутизацию пакетов (**net.ipv4.ip_forward**) и выключить фильтрацию пакетов по их исходящему адресу (**net.ipv4.conf.all.rp_filter** и **net.ipv4.conf.default.rp_filter**):

```
$ sudo vi /etc/sysctl.conf
...
net.ipv4.ip_forward = 1
net.ipv4.conf.all.rp_filter = 0
net.ipv4.conf.default.rp_filter = 0
$ sudo sysctl -p
net.ipv4.ip_forward = 1
net.ipv4.conf.all.rp_filter = 0
net.ipv4.conf.default.rp_filter = 0
```

Настраиваем конфиг **ML2 /etc/neutron/plugins/ml2/ml2_conf.ini**, указывая **flat** провайдер для внешней сети:

```
$ sudo crudini --set /etc/neutron/plugins/ml2/ml2_conf.ini ml2_type_flat  
flat_networks external
```

Настраиваем конфиг **Open vSwitch** **/etc/neutron/plugins/ml2/openvswitch_agent.ini**:

Указываем IP-адрес для конечной точки туннеля (ip узла управления):

```
$ sudo crudini --set /etc/neutron/plugins/ml2/openvswitch_agent.ini ovs local_ip  
192.168.122.200
```

Сопоставляем внешнюю сеть созданному мосту **br-ex**:

```
$ sudo crudini --set /etc/neutron/plugins/ml2/openvswitch_agent.ini ovs  
bridge_mappings external:br-ex
```

Настраиваем включение **GRE-туннелирования**:

```
$ sudo crudini --set /etc/neutron/plugins/ml2/openvswitch_agent.ini ovs  
enable_tunneling True  
$ sudo crudini --set /etc/neutron/plugins/ml2/openvswitch_agent.ini agent  
tunnel_types gre
```

Настраиваем конфиг **L3-агента** (маршрутизация) **/etc/neutron/l3_agent.ini**:

Указываем драйвер **Open vSwitch**:

```
$ sudo crudini --set /etc/neutron/l3_agent.ini DEFAULT interface_driver  
neutron.agent.linux.interface.OVSInterfaceDriver
```

Настраиваем использование **namespace**:

```
$ sudo crudini --set /etc/neutron/l3_agent.ini DEFAULT use_namespaces True
```

Указываем имя, используемого моста:

```
$ sudo crudini --set /etc/neutron/l3_agent.ini DEFAULT external_network_bridge  
br-ex
```

Настраиваем удаление неиспользуемых **namespace**:

```
$ sudo crudini --set /etc/neutron/l3_agent.ini DEFAULT router_delete_namespaces  
True
```

Настраиваем конфиг **DHCP-агента** **/etc/neutron/dhcp_agent.ini**:

Указываем драйвер **Open vSwitch**:

```
$ sudo crudini --set /etc/neutron/dhcp_agent.ini DEFAULT interface_driver  
neutron.agent.linux.interface.OVSInterfaceDriver
```

Указываем **Dnsmasq** DHCP-драйвер:

```
$ sudo crudini --set /etc/neutron/dhcp_agent.ini DEFAULT dhcp_driver neutron.agent.linux.dhcp.Dnsmasq
```

Настраиваем использование и удаление namespace:

```
$ sudo crudini --set /etc/neutron/dhcp_agent.ini DEFAULT use_namespaces True  
$ sudo crudini --set /etc/neutron/dhcp_agent.ini DEFAULT dhcp_delete_namespaces True
```

Настраиваем конфиг агента-метаданных **/etc/neutron/metadata_agent.ini**:

Указываем IP-адрес управляющего узла:

```
$ sudo crudini --set /etc/neutron/metadata_agent.ini DEFAULT nova_metadata_host 192.168.122.200
```

Указываем общий секрет, используемый для получения метаданных:

```
$ sudo crudini --set /etc/neutron/metadata_agent.ini DEFAULT metadata_proxy_shared_secret openstack
```

Дополнительные параметры конфигов сетевого (управляющего) узла:

```
$ sudo vi /etc/neutron/plugins/ml2/ml2_conf.ini  
...  
[ml2_type_flat]  
flat_networks = external  
$ sudo vi /etc/neutron/plugins/ml2/openvswitch_agent.ini  
...  
[ovs]  
local_ip = 192.168.122.200  
enable_tunneling = True  
[agent]  
tunnel_types = gre  
$ sudo vi /etc/neutron/l3_agent.ini  
[DEFAULT]  
external_network_bridge = br-ex  
use_namespaces = True  
interface_driver = neutron.agent.linux.interface.OVSInterfaceDriver  
router_delete_namespaces = True  
$ sudo vi /etc/neutron/dhcp_agent.ini  
[DEFAULT]  
interface_driver = neutron.agent.linux.interface.OVSInterfaceDriver  
use_namespaces = True  
dhcp_delete_namespaces = True  
dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq  
$ sudo vi /etc/neutron/metadata_agent.ini  
[DEFAULT]  
metadata_proxy_shared_secret = openstack  
nova_metadata_host = 192.168.122.200
```

Запускаем и включаем Open vSwitch:

```
$ sudo systemctl enable openvswitch.service  
Created symlink from /etc/systemd/system/multi-user.target.wants/openvswitch.service to  
/usr/lib/systemd/system/openvswitch.service.  
$ sudo systemctl start openvswitch.service  
$ sudo systemctl status openvswitch.service  
● openvswitch.service - Open vSwitch  
   Loaded: loaded (/usr/lib/systemd/system/openvswitch.service; enabled; vendor preset:  
           disabled)  
   Active: active (exited) since Вт 2022-12-06 16:37:38 MSK; 8s ago
```

Я тут понял, что в определенный момент перестал показывать скрины статусов, но не переживайте, все работает без ошибок.

Ухх, господа, теперь [надо создать сетевой интерфейс](#) для нашей так называемой внутренней сети (в книге он был с первой главы, в то время как у меня был только **eth0** на всех ВМ), так что выходим из нашей виртуалки на хост и начинаем создавать новую сеть типа bridge:

Первым делом создаем файл конфигурации сети **private.xml** (я просто скопировал содержимое файла **/etc/libvirt/qemu/networks/default.xml** и поменял ip и удалил части, которые генерируются автоматически), потому что **virsh** почему-то не может создать сеть через параметры командной строки (Привет это Артем и будущего, я ошибся в ip адресах и фиксили это один день своей жизни, не беспокойтесь скрины уже исправлены):

```
$ sudo vi private.xml
<network>
  <name>private</name>
  <bridge name='virbr3' stp='on' delay='0' />
  <ip address='10.100.1.1' netmask='255.255.255.0'>
    <dhcp>
      <range start='10.100.1.2' end='10.100.1.254' />
    </dhcp>
  </ip>
</network>
```

Создаем сеть с помощью **virsh**:

```
$ sudo virsh net-create private.xml
Network private created from private.xml
$ sudo virsh net-list
Name      State   Autostart
Persistent
-----
default   active   yes      yes
private   active   no       no
```

Включаем автостарт сети **private** (перед настройкой автостарта, нужно открыть конфиг сети через **net-edit**, чтобы первоначальные настройки сети применились):

```
$ sudo virsh net-edit private
Network private XML configuration edited.
$ sudo virsh net-autostart private
Network private marked as autostarted
```

Должен появится сетевой интерфейс **virbr3** на хосте:

```
$ sudo ip a
...
virbr3: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default qlen 1000
  link/ether 52:54:00:1e:d0:9c brd ff:ff:ff:ff:ff:ff
  inet 10.10.1.1/24 brd 10.10.1.255 scope global virbr3
    valid_lft forever preferred_lft forever
```

Теперь можно запускать **VM2 (controller.test.local)** и присоединить к ней сетевой интерфейс (команда выполняется на хосте):

```
$ sudo virsh attach-interface --domain VM3 --type network --source private --model virtio --config --live --persistent
Interface attached successfully
```

Внутри ВМ должен появится новый интерфейс **eth1**:

```
$ sudo ip a
...
eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether 52:54:00:85:fb:21 brd ff:ff:ff:ff:ff:ff
```

На этом все, сетевой интерфейс к машине присоединен, теперь можно перейти к его настройке для сетевого (у меня управляющего) узла для сервиса **Neutron**:

Создаем файл **/etc/sysconfig/network-scripts/ifcfg-eth1** и пишем в него следующие настройки и перезапускаем сеть (Опять Артем из будущего, короче с конфигами были тоже проблемы и пришлось добавить файл **/etc/sysconfig/network-scripts/ifcfg-br-ex** специально для моста):

```
$ sudo cat /etc/sysconfig/network-scripts/ifcfg-eth1
DEVICE=eth1
TYPE=OVSPort
DEVICETYPE=ovs
OVS_BRIDGE=br-ex
ONBOOT=yes
HWADDR=52:54:00:5e:ef:12
$ sudo cat /etc/sysconfig/network-scripts/ifcfg-br-ex
DEVICE=br-ex
DEVICETYPE=ovs
TYPE=OVSBridge
BOOTPROTO=static
IPADDR=10.100.1.250
NETMASK=255.255.255.0
GATEWAY=10.100.1.1
ONBOOT=yes
```

Теперь надо добавить мост **br-ex** для Open vSwitch, который будет находится на созданном интерфейсе **eth1**:

```
$ sudo ovs-vsctl add-br br-ex
[ 1041.773671] device ovs-system entered promiscuous mode
[ 1041.783030] device br-ex entered promiscuous mode
$ sudo ovs-vsctl add-port br-ex eth1
[ 1055.410159] device eth1 entered promiscuous mode
```

Проверяем все ли подключилось нормально при помощи **ovs-vsctl**:

```
$ sudo ovs-vsctl show
489d40ff-df55-4b53-ba2b-9e3ec849f55d
    Bridge br-ex
        Port br-ex
            Interface br-ex
                type: internal
        Port "eth1"
            Interface "eth1"
    ovs_version: "2.12.0"
```

Вроде все круто, так что можно запускать все службы **Neutron** для сетевого (сами знаете какого) узла (насколько я понял ovscleanup стартовать не надо):

```
$ sudo systemctl enable neutron-openvswitch-agent.service
Created symlink from /etc/systemd/system/multi-user.target.wants/neutron-openvswitch-
agent.service to /usr/lib/systemd/system/neutron-openvswitch-agent.service.
$ sudo systemctl enable neutron-l3-agent.service
Created symlink from /etc/systemd/system/multi-user.target.wants/neutron-l3-agent.service
to /usr/lib/systemd/system/neutron-l3-agent.service.
$ sudo systemctl enable neutron-dhcp-agent.service
Created symlink from /etc/systemd/system/multi-user.target.wants/neutron-dhcp-
agent.service to /usr/lib/systemd/system/neutron-dhcp-agent.service.
$ sudo systemctl enable neutron-metadata-agent.service
Created symlink from /etc/systemd/system/multi-user.target.wants/neutron-metadata-
agent.service to /usr/lib/systemd/system/neutron-metadata-agent.service.
$ sudo systemctl enable neutron-ovs-cleanup.service
Created symlink from /etc/systemd/system/multi-user.target.wants/neutron-ovs-
cleanup.service to /usr/lib/systemd/system/neutron-ovs-cleanup.service.
$ sudo systemctl start neutron-openvswitch-agent.service
$ sudo systemctl start neutron-l3-agent.service
$ sudo systemctl start neutron-dhcp-agent.service
$ sudo systemctl start neutron-metadata-agent.service
```

Вместо того, чтобы проверять статус каждой службы можно посмотреть все одной командой:

```
$ sudo openstack network agent list
+-----+-----+-----+-----+-----+-----+-----+
| ID      | Agent Type   | Host        | Availability Zone | Alive | State | Binary          |
+-----+-----+-----+-----+-----+-----+-----+
| 0764784b-80b4-4cf5-b0f9-bfffc154396f8 | Open vSwitch agent | controller.test.local | None    | (-)  | UP   | neutron-openvswitch-agent |
| 1fba8d1b-1ca0-4856-82f4-753b25a55af2 | L3 agent       | controller.test.local | nova   | (-)  | UP   | neutron-l3-agent      |
| 7de65c75-dba5-4150-9263-f2664d589e | Metadata agent  | controller.test.local | None    | (-)  | UP   | neutron-metadata-agent |
| dce6b888-a969-4c88-846d-546b9aec9a88 | DHCP agent     | controller.test.local | nova   | (-)  | UP   | neutron-dhcp-agent    |
+-----+-----+-----+-----+-----+-----+-----+
```

Теперь переходим к настройке служб **Neutron** на рабочих узлах **compute** и **compute-opt**:

Качаем необходимые пакеты (все уже обсуждались раньше):

```
$ sudo yum -y install openstack-neutron openstack-neutron-ml2 openstack-neutron-openvswitch
```

Теперь заполняем общие для всех узлов параметры конфигов для сервиса **Neutron**, и создаем ссылку на конфиг **ML2**:

```
$ sudo vi /etc/neutron/neutron.conf
[DEFAULT]
transport_url = rabbit://openstack:openstack@controller.test.local
auth_strategy = keystone
service_plugins = router
core_plugin = ml2
allow_overlapping_ips = True
...
[keystone_authtoken]
project_name = service
user_domain_name = default
project_domain_name = default
auth_type = password
username = neutron
password = openstack
www_authentication_uri = http://controller.test.local:5000
auth_url = http://controller.test.local:35357
...
[oslo_concurrency]
lock_path = /var/lib/neutron/tmp
$ sudo vi /etc/neutron/plugins/ml2/ml2_conf.ini
...
[securitygroup]
enable_security_group = True
enable_ipset = True
...
[ml2_type_gre]
tunnel_id_ranges = 1:1000
...
[ml2]
mechanism_drivers = openvswitch
tenant_network_types = gre
type_drivers = flat,gre,vlan,vxlan
```

```
$ sudo ln -s /etc/neutron/plugins/ml2/ml2_conf.ini /etc/neutron/plugin.ini
```

Базу выполнили, теперь изменяем параметры ядра **net.ipv4.conf.all.rp_filter** и **net.ipv4.conf.default.rp_filter** (отключение фильтрации пакетов), **net.bridge.bridge-nf-call-iptables** (пакеты с сетевого моста передаются на обработку **iptables**):

```
$ sudo vi /etc/sysctl.conf
...
net.ipv4.conf.all.rp_filter = 0
net.ipv4.conf.default.rp_filter = 0
net.bridge.bridge-nf-call-iptables = 1
$ sudo sysctl -p
net.ipv4.conf.all.rp_filter = 0
net.ipv4.conf.default.rp_filter = 0
net.bridge.bridge-nf-call-iptables = 1
```

Если последняя строка не выполняется то выполните следующую команду (включает модуль ядра **br_netfilter**), а потом опять **sysctl**:

```
$ sudo modprobe br_netfilter
$ sudo sysctl -p
net.ipv4.conf.all.rp_filter = 0
net.ipv4.conf.default.rp_filter = 0
net.bridge.bridge-nf-call-iptables = 1
```

Так, ну теперь пошли специфичные для рабочих узлов параметры конфигов:

Настраиваем конфиг **Open vSwitch /etc/neutron/plugins/ml2/openvswitch_agent.ini**:

Указываем локальный ip вычислительного узла (**compute-opt** — **192.168.122.215**):

```
$ sudo crudini --set /etc/neutron/plugins/ml2/openvswitch_agent.ini ovs local_ip  
192.168.122.210
```

Указываем тип используемой технологии туннеля:

```
$ sudo crudini --set /etc/neutron/plugins/ml2/openvswitch_agent.ini agent tunnel_types  
gre
```

Указываем драйвер брандмауэра (**iptables** на **Open vSwitch**):

```
$ sudo crudini --set /etc/neutron/plugins/ml2/openvswitch_agent.ini securitygroup  
firewall_driver neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver
```

Запускаем **Open vSwitch**, так как его настройка закончена:

```
$ sudo systemctl enable openvswitch.service  
Created symlink from /etc/systemd/system/multi-user.target.wants/openvswitch.service  
to /usr/lib/systemd/system/openvswitch.service.  
$ sudo systemctl start openvswitch.service
```

Настраиваем конфиг сервиса **Nova** **/etc/nova/nova.conf**:

Передаем полномочия (первые 3 команды) с службы **nova-network** на **Neutron**, а также отключаем службу брандмауэра в **Nova** (последняя команда):

```
$ sudo crudini --set /etc/nova/nova.conf DEFAULT network_api_class  
nova.network.neutronv2.api.API  
$ sudo crudini --set /etc/nova/nova.conf DEFAULT security_group_api neutron  
$ sudo crudini --set /etc/nova/nova.conf DEFAULT linuxnet_interface_driver  
nova.network.linux_net.LinuxOVSInterfaceDriver  
$ sudo crudini --set /etc/nova/nova.conf DEFAULT firewall_driver  
nova.virt.firewall.NoopFirewallDriver
```

Указываем параметры аутентификации **Neutron** (как и на управляемом узле):

```
$ sudo crudini --set /etc/nova/nova.conf neutron url  
http://controller.test.local:9696  
$ sudo crudini --set /etc/nova/nova.conf neutron auth_url  
http://controller.test.local:35357  
$ sudo crudini --set /etc/nova/nova.conf neutron auth_type password  
$ sudo crudini --set /etc/nova/nova.conf neutron project_domain_name default  
$ sudo crudini --set /etc/nova/nova.conf neutron user_domain_name default  
$ sudo crudini --set /etc/nova/nova.conf neutron region_name RegionOne  
$ sudo crudini --set /etc/nova/nova.conf neutron project_name service  
$ sudo crudini --set /etc/nova/nova.conf neutron username neutron  
$ sudo crudini --set /etc/nova/nova.conf neutron password openstack
```

Персональные параметры рабочих узлов:

```
$ sudo vi /etc/neutron/plugins/ml2/openvswitch_agent.ini
...
[ovs]
local_ip = 192.168.122.210
[agent]
tunnel_types = gre
[securitygroup]
firewall_driver = neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver
$ sudo vi /etc/nova/nova.conf
[DEFAULT]
...
firewall_driver = nova.virt.firewall.NoopFirewallDriver
network_api_class = nova.network.neutronv2.api.API
security_group_api = neutron
linuxnet_interface_driver = nova.network.linux_net.LinuxOVSIInterfaceDriver
[neutron]
url = http://controller.test.local:9696
auth_url = http://controller.test.local:35357
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = neutron
password = openstack
```

Перезапускаем службу **nova-compute** и запускаем агента **Open vSwitch**:

```
$ sudo systemctl restart openstack-nova-compute.service
$ sudo systemctl enable neutron-openvswitch-agent.service
Created symlink from /etc/systemd/system/multi-user.target.wants/neutron-openvswitch-
agent.service to /usr/lib/systemd/system/neutron-openvswitch-agent.service.
$ sudo systemctl start neutron-openvswitch-agent.service
```

Должны появится новые сетевые агенты в выводе команды, по одному на рабочий узел (я пока настроил только один):

ID	Agent Type	Host	Availability Zone	Alive	State	Binary
0764784b-80b4-4cf5-b0f9-bfffc154396f8	Open vSwitch agent	controller.test.local	None	(-)	UP	neutron-openvswitch- agent
1fba8d1b-1ca0-4856-82f4-753b25a55af2	L3 agent	controller.test.local	nova	(-)	UP	neutron-l3-agent
7de65c75-dba5-4150-9263-f2664d589efe	Metadata agent	controller.test.local	None	(-)	UP	neutron-metadata-agent
9c6a39b2-5209-4250-8710-4928ec1a2024	Open vSwitch agent	compute.test.local	None	(-)	UP	neutron-openvswitch- agent
dce6b888-a969-4c88-846d-546b9aec9a88	DHCP agent	controller.test.local	nova	(-)	UP	neutron-dhcp-agent

Все, с сервисом **Neutron** пока покончено, как и с основными (точнее 6 из 7, без Swift) сервисами **Openstack**. Осталось только поработать с созданием ВМ и сетей для них, а также установить доп сервисы для удобства (**Horizon**, **Gnocchi**, **Ceilometer**, **Aodh** и **Heat**). Есть также некоторые сомнения касательно подключения дополнительных сетевых интерфейсов для рабочих узлов (пока не подключаю), если их надо будет добавить, естественно, сделаю сноску (**не надо**).

10. Тестим все то, что на разворачивали.

10.1. Для начала создадим все необходимые компоненты сетевой инфраструктуры для запуска первой ВМ:

Создаем внешнюю сеть **ext-net**:

Field	Value
admin_state_up	UP
availability_zone_hints	
availability_zones	
created_at	2022-12-12T13:27:59Z
description	
dns_domain	None
id	574ac2c6-1d90-4c94-87ea-5335464ac5b0
ipv4_address_scope	None
ipv6_address_scope	None
is_default	False
is_vlan_transparent	None
location	cloud='', project.domain_id='', project.domain_name='Default', project.id='33ad52bb25284ad8be4b8681b8fe0063', project.name='admin', region_name='', zone=
mtu	1500
name	ext-net
port_security_enabled	False
project_id	33ad52bb25284ad8be4b8681b8fe0063
provider:network_type	flat
provider:physical_network	external
provider:segmentation_id	None
qos_policy_id	None
revision_number	1
router:external	External
segments	None
shared	True
status	ACTIVE
subnets	
tags	
updated_at	2022-12-12T13:27:59Z

Из-за опечатки в книге (вместо **--provider-physical-network external** было написано **--provider-physical-network datacentre**) я потратил очередные 2 часа своей жизни на исправление бага... А ведь это уже 4 издание книги.

- **--external** — сеть является внешней (есть средство внешней маршрутизации)
- **--share** — сеть доступна всем проектам
- **--provider-network-type** — физический механизм, с помощью которого реализуется виртуальная сеть (**flat, geneve, gre, local, vlan, vxlan**)
- **--provider-physical-network** — имя физической сети, в которой реализована виртуальная сеть (мы мапили это название с созданным мостом, а потом объявляли его возможным для использования в типе сети **flat**)

Теперь создадим подсеть, из которой будут выделяться плавающие IP-адреса (в реальности они должны быть видны из интернета).

\$ sudo openstack subnet create --network ext-net --no-dhcp --allocation-pool start=10.100.1.100,end=10.100.1.200 --gateway 10.100.1.1 --subnet-range 10.100.1.0/24 ext-subnet
+-----+-----+
Field Value
+-----+-----+
allocation_pools 10.100.1.100-10.100.1.200
cidr 10.100.1.0/24
created_at 2022-12-12T13:46:50Z
description
dns_nameservers
enable_dhcp False
gateway_ip 10.100.1.1
host_routes
id 30e9c86f-3770-4917-85bd-a4bc91521f31
ip_version 4
ipv6_address_mode None
ipv6_ra_mode None
location cloud='', project.domain_id=, project.domain_name='Default', project.id='33ad52bb25284ad8be4b8681b8fe0063', project.name='admin', region_name='', zone=
name ext-subnet
network_id 574ac2c6-1d90-4c94-87ea-5335464ac5b0
prefix_length None
project_id 33ad52bb25284ad8be4b8681b8fe0063
revision_number 0
segment_id None
service_types None
subnetpool_id None
tags
updated_at 2022-12-12T13:46:50Z
+-----+-----+

- **--network** — сеть родитель
- **--no-dhcp** — без DHCP
- **--allocation-pool** — доступные ip адреса
- **--gateway** — дефолтный шлюз во внешнюю сеть (ip маршрутизатора)
- **--subnet-range** — маска подсети

Теперь переквалифицируемся в пользователя demo и создаем сеть в его личном проекте:

\$ sudo keystonerc_usr
\$ sudo openstack network create demo-net
+-----+-----+
Field Value
+-----+-----+
admin_state_up UP
availability_zone_hints
availability_zones
created_at 2022-12-12T13:53:06Z
description
dns_domain None
id a3ba8800-8338-4558-aeca-3247df932f39
ipv4_address_scope None
ipv6_address_scope None
is_default False
is_vlan_transparent None
location cloud='', project.domain_id=, project.domain_name='Default', project.id='e420dacf94514ed587d878bf32cabf46', project.name='demo', region_name='', zone=
mtu 1458
name demo-net
port_security_enabled False
project_id e420dacf94514ed587d878bf32cabf46
provider:network_type None
provider:physical_network None
provider:segmentation_id None
qos_policy_id None
revision_number 1
router:external Internal
segments None
shared False
status ACTIVE
subnets
tags
updated_at 2022-12-12T13:53:06Z
+-----+-----+

Создаем подсеть (**demo-subnet**) нашей **demo-net**:

Field	Value
allocation_pools	172.16.0.2-172.16.0.254
cidr	172.16.0.0/24
created_at	2022-12-12T14:08:59Z
description	
dns_nameservers	
enable_dhcp	True
gateway_ip	172.16.0.1
host_routes	
id	59d8cece-795b-4618-8a10-e4d37e05d5ac
ip_version	4
ipv6_address_mode	None
ipv6_ra_mode	None
location	cloud='', project.domain_id='', project.domain_name='Default', project.id='e420dacf94514ed587d878bf32cabf46', project.name='demo', region_name='', zone=''
name	demo-subnet
network_id	a3ba8800-8338-4558-aeca-3247df932f39
prefix_length	None
project_id	e420dacf94514ed587d878bf32cabf46
revision_number	0
segment_id	None
service_types	
subnetpool_id	None
tags	
updated_at	2022-12-12T14:08:59Z

Создаем роутер (**demo-router**) для **demo-subnet**:

Field	Value
admin_state_up	UP
availability_zone_hints	
availability_zones	
created_at	2022-12-12T14:24:28Z
description	
external_gateway_info	null
flavor_id	None
id	e1a0f62f-46c3-472a-9b6d-5aa0c5611d5e
location	cloud='', project.domain_id='', project.domain_name='Default', project.id='e420dacf94514ed587d878bf32cabf46', project.name='demo', region_name='', zone=''
name	demo-router
project_id	e420dacf94514ed587d878bf32cabf46
revision_number	1
routes	
status	ACTIVE
tags	
updated_at	2022-12-12T14:24:28Z

Добавляем подсеть **demo-subnet** к роутеру **demo-router**:

```
$ sudo openstack router add subnet demo-router demo-subnet
```

Отступление от темы: у меня уже начинаются убиваться процессы на виртуалке из-за нехватки оперативки.

Теперь выставляем внешний шлюз для роутера **demo-router**:

```
$ sudo openstack router set --external-gateway ext-net demo-router
```

Вот примерные параметры роутера **demo-router**:

Field	Value
admin_state_up	UP
availability_zone_hints	
availability_zones	nova
created_at	2022-12-12T14:24:28Z
description	
external_gateway_info	{"network_id": "574ac2c6-1d90-4c94-87ea-5335464ac5b0", "enable_snat": true, "external_fixed_ips": [{"subnet_id": "30e9c86f-3770-4917-85bd-a4bc91521f31", "ip_address": "10.100.1.121"}]}
flavor_id	None
id	e1a0f62f-46c3-472a-9b6d-5aa0c5611d5e
interfaces_info	[{"subnet_id": "59d8cece-795b-4618-8a10-e4d37e05d5ac", "ip_address": "172.16.0.1", "port_id": "a3b98aca-1332-4156-83d9-62ee9c962c33"}]
location	cloud='', project.domain_id=, project.domain_name='Default', project.id='e420dacf94514ed587d878bf32cabf46', project.name='demo', region_name='', zone=
name	demo-router
project_id	e420dacf94514ed587d878bf32cabf46
revision_number	4
routes	
status	ACTIVE
tags	
updated_at	2022-12-12T14:39:07Z

10.2. Теперь переходим к созданию экземпляра ВМ:

А, попались, думали уже можно создать ВМ, неа еще надо создать **flavour** (шаблон виртуальной машины по выделяемым ресурсам) перед этим. Шаблоны по умолчанию может создавать только админ:

Field	Value
OS-FLV-DISABLED:disabled	False
OS-FLV-EXT-DATA:ephemeral	0
disk	1
id	df7814b9-4900-42de-b5f3-299b884cb8b5
name	m2.tiny
os-flavor-access:is_public	True
properties	
ram	300
rxtx_factor	1.0
swap	
vcpus	1

- **--ram** — оперативная память
- **--disk** — размер диска
- **--vcpus** — количество виртуальных ядер CPU
- **--public** — доступен всем пользователям
- **--rxtx-factor** — пропускная способность сети (только для гипервизора **XEN**)
- **--ephemeral** — размер второго диска (в отличии от **--disk** всегда удаляется при удалении ВМ)
- **--swap** — размер опционального swap-раздела

Выделил контроллер узлу еще 2GB оперативки, надеюсь хватит, но я уже вхожу в зону «выделил ресурсов больше чем на самом деле» ($2+2+6=8$)

Теперь создадим пару **ssh-ключей**, которую служба метаданных сервиса **Nova** будет передавать созданным ВМ:

```
$ sudo openstack keypair create demokey1 > ~/demokey1
$ sudo openstack keypair list
+-----+
| Name      | Fingerprint |
+-----+
| demokey1 | 49:3d:e1:43:82:58:ea:97:92:ac:ac:cf:0e:ca:36:59 |
+-----+
$ sudo cat demokey1
-----BEGIN RSA PRIVATE KEY-----
...
-----END RSA PRIVATE KEY-----
```

Вхожу в теневую зону (запускаю сразу 3 ВМ) и пытаюсь создать ВМ по созданному шаблону **m2.tiny**, передав в нее созданную пару ключей **demokey1** (если на этом отчет закончился, то это потому что у меня взорвался ноутбук):

При запуске обоих рабочих узлов, узел управления сразу падает, видимо kvm не любит избыточность ресурсов, поэтому прощай **compute-opt**. Лагануло при создании ВМ у меня знатно.

```
$ sudo openstack server create --image cirros-0.4.0-x86_64 --flavor m2.tiny --key-name demokey1 --nic net-id=demo-net myinstance1
+-----+
| Field          | Value        |
+-----+
| OS-DCF:diskConfig | MANUAL
| OS-EXT-AZ:availability_zone | None
| OS-EXT-SRV-ATTR:host | None
| OS-EXT-SRV-ATTR:hypervisor_hostname | None
| OS-EXT-SRV-ATTR:instance_name | None
| OS-EXT-STS:power_state | NOSTATE
| OS-EXT-STS:task_state | scheduling
| OS-EXT-STS:vm_state | building
| OS-SRV-USG:launched_at | None
| OS-SRV-USG:terminated_at | None
| accessIPv4 | None
| accessIPv6 | None
| addresses | None
| adminPass | RUDQzhYg5fpS
| config_drive | None
| created | 2022-12-12T16:26:31Z
| flavor | m2.tiny (df7814b9-4900-42de-b5f3-299b884cb8b5)
| hostId | None
| id | 3cb09e16-ea18-43b6-901b-a19cd147c13a
| image | cirros-0.4.0-x86_64 (57c57075-fd8c-4475-bd3d-9d1c745849eb)
| key_name | demokey1
| name | myinstance1
| progress | 0
| project_id | 33ad52bb25284ad8be4b8681b8fe0063
| properties | None
| security_groups | name='default'
| status | BUILD
| updated | 2022-12-12T16:26:31Z
| user_id | 549949769da94550a63fa6667741ab1b4
| volumes_attached | None
+-----+
```

Изливаю очередной бомбаж: я весь день (5 часов подряд) пытался пофиксить ошибку **UnicodeDecodeError** (ошибки можно найти в **логах**, выводе команд **journalctl** и **openstack server show имя_ВМ**), которая вылезала при создании ВМ, к 5 часу я уже редактировал питоновские файлы по гайдам на китайском языке, но все равно ошибка

коварно вылезала, а потом я перезагрузил (в 3 раз за 5 часов) виртуалки и все пофиксилось, вот как-то так...

Посмотреть статус созданной ВМ можно как с помощью **openstack cli**, так и при помощи **virsh** на рабочем узле:

```
$ sudo openstack server list
+-----+-----+-----+-----+-----+
| ID      | Name     | Status  | Networks   | Image      | Flavor   |
+-----+-----+-----+-----+-----+
| 5dc39d0e-5f1d-4fa8-b188-a39743cd5073 | myinstance1 | ACTIVE  | demo-net=172.16.0.10 | cirros-0.4.0-x86_64 | m2.tiny |
+-----+-----+-----+-----+-----+
```

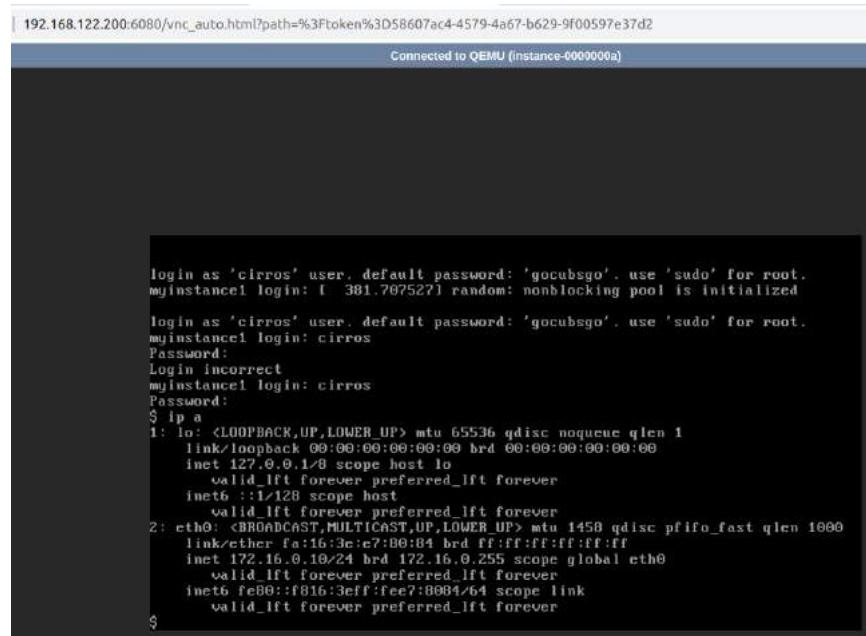
На рабочем узле:

```
$ sudo virsh list
+-----+-----+
| ID    | Имя           | Статус |
+-----+-----+
| 1     | instance-0000000a | работает |
+-----+-----+
```

Теперь можно подключится к ВМ через **noVNC** (**VNC** через браузер), получив URL с помощью **openstack cli**:

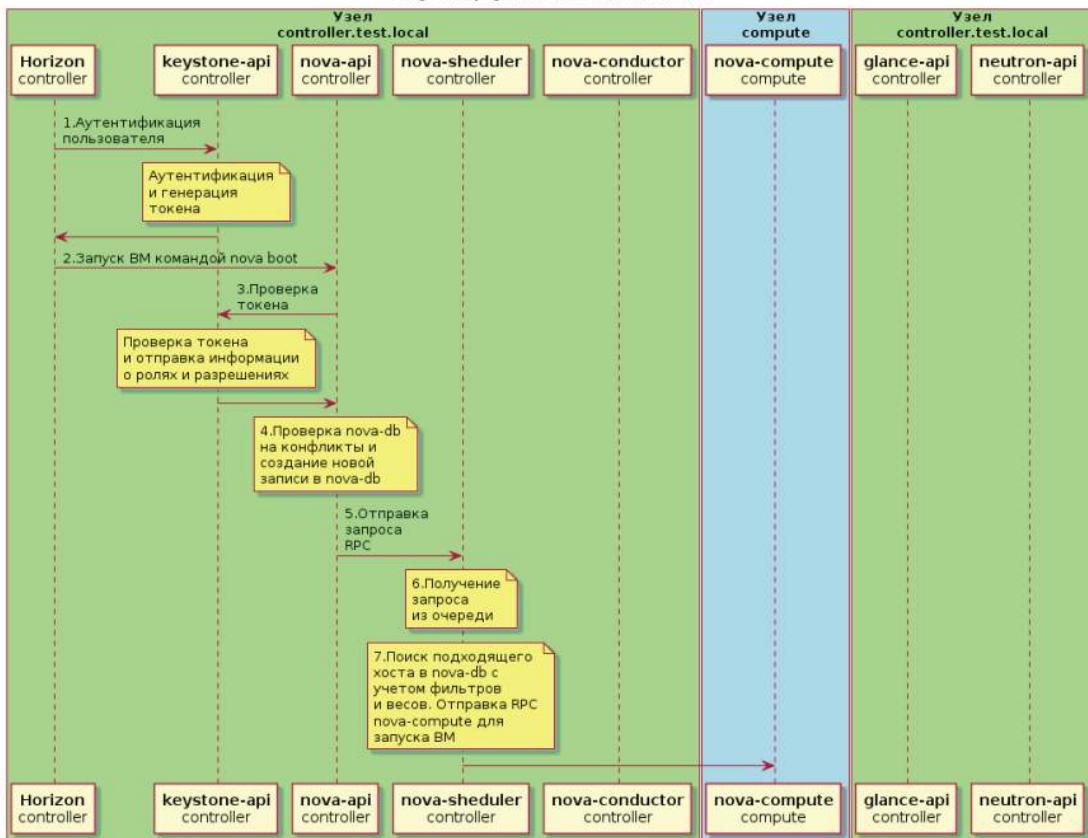
```
$ sudo openstack console url show myinstance1
+-----+
| Field | Value
+-----+
| type  | novnc
| url   | http://controller.test.local:6080/vnc_auto.html?path=%3Ftoken%3D58607ac4-4579-4a67-b629-9f00597e37d2 |
+-----+
```

Пишем полученный URL в строку браузера и получаем доступ к консоли ВМ **myinstance1** (у меня на хосте не настроено сопоставление DNS с ip, так что я заменил **controller.test.local** на **192.168.122.200**):

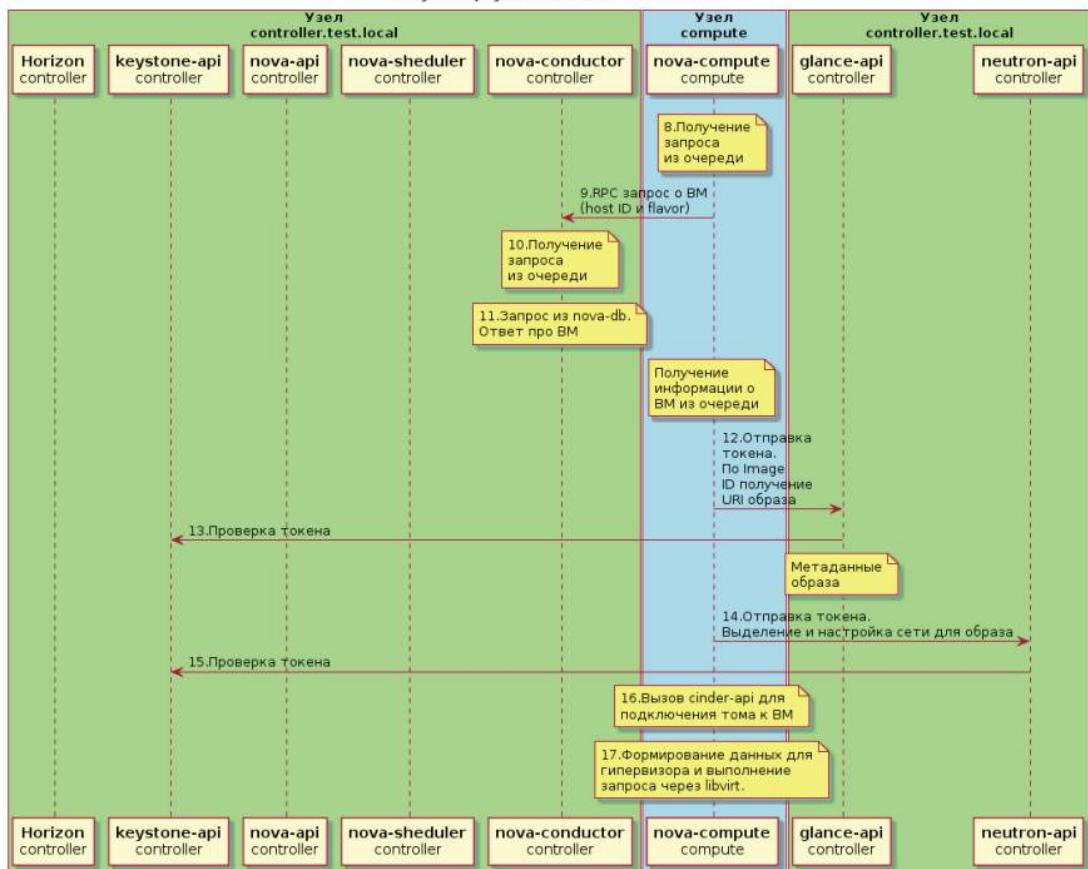


Вот скрины из книги о том как создается ВМ (запрос путешествует по всем сервисам):

Запуск виртуальной машины. Часть I



Запуск виртуальной машины. Часть II



Теперь раз ВМ включена то должна начать записываться статистика по потреблению ей ресурсов:

\$ sudo openstack usage list				
Usage from 2022-11-15 to 2022-12-14:				
Project	Servers	RAM MB-Hours	CPU Hours	Disk GB-Hours
demo	1	4804.63	16.02	16.02

Теперь надо добавить к ВМ сеть, чтобы к ней например можно было подключаться через SSH:

Первым делом надо создать группу безопасности **demo-sgroup** (набор правил брандмауэра, разрешающих доступ к тем или иным портам виртуальной машины), так как к ВМ автоматически добавилась группа **default**, которая ничего не разрешает:

\$ sudo openstack security group create demo-sgroup	
Field	Value
created_at	2022-12-13T11:26:49Z
description	demo-sgroup
id	77161e8f-f1bd-4cbd-b8e2-e73eb1420b02
location	cloud='', project.domain_id='', project.domain_name='Default', project.id='e420dacf94514ed587d878bf32cabf46', project.name='demo', region_name='', zone=
name	demo-sgroup
project_id	e420dacf94514ed587d878bf32cabf46
revision_number	1
rules	created_at='2022-12-13T11:26:49Z', direction='egress', ether_type='IPv4', id='b7fcfd269-e142-4a56-b560-6cdd4f95eafc', updated_at='2022-12-13T11:26:49Z' created_at='2022-12-13T11:26:49Z', direction='egress', ether_type='IPv6', id='f7381ee2-aaf1-4bd2-805f-de2fb6a9dbe6', updated_at='2022-12-13T11:26:49Z'
tags	[]
updated_at	2022-12-13T11:26:49Z

Теперь добавим 2 правила: первое разрешает доступ по SSH, второе разрешает протокол ICMP:

\$ sudo openstack security group rule create --protocol tcp --dst-port 22 demo-sgroup	
Field	Value
created_at	2022-12-13T11:41:16Z
description	
direction	ingress
ether_type	IPv4
id	4a7e3e7f-05e4-4676-bc53-5106b5eab6c7
location	cloud='', project.domain_id='', project.domain_name='Default', project.id='e420dacf94514ed587d878bf32cabf46', project.name='demo', region_name='', zone=
name	None
port_range_max	22
port_range_min	22
project_id	e420dacf94514ed587d878bf32cabf46
protocol	tcp
remote_group_id	None
remote_ip_prefix	0.0.0.0/0
revision_number	0
security_group_id	abae8161-7076-4d49-990d-cbb08164ce08
tags	[]
updated_at	2022-12-13T11:41:16Z

\$ sudo openstack security group rule create --protocol icmp demo-sgroup	
Field	Value
created_at	2022-12-13T11:45:27Z
description	
direction	ingress
ether_type	IPv4
id	de739628-3392-4197-aff8-4d6a9dec9d9d
location	cloud='', project.domain_id='Default', project.domain_name='Default', project.id='e420dacf94514ed587d878bf32cabf46', project.name='demo', region_name='', zone=''
name	None
port_range_max	None
port_range_min	None
project_id	e420dacf94514ed587d878bf32cabf46
protocol	icmp
remote_group_id	None
remote_ip_prefix	0.0.0.0/0
revision_number	0
security_group_id	abae8161-7076-4d49-990d-cbb08164ce08
tags	[]
updated_at	2022-12-13T11:45:27Z

Посмотреть установленные правила можно так:

\$ sudo openstack security group rule list demo-sgroup					
ID	IP Protocol	Ethertype	IP Range	Port Range	Remote Security Group
4a7e3e7f-05e4-4676-bc53-510b5eab6c7	tcp	IPv4	0.0.0.0/0	22:22	None
b8f57cf7-e527-4e0d-ba59-612ac7f89e52	None	IPv6	::/0		None
d7bf9aae4-b410-4b30-a40a-f51012d4a36d	None	IPv4	0.0.0.0/0		None
de739628-3392-4197-aff8-4d6a9dec9d9d	icmp	IPv4	0.0.0.0/0		None

Теперь можно добавить группу безопасности **demo-sgroup** ВМ **myinsatnce1** и удалить группу **default** (также можно указывать группу безопасности при запуске ВМ при помощи флага **--security-groups**), заодно держите удобный способ просмотра отдельных параметров вывода команды:

```
$ sudo openstack server add security group myinstance1 demo-sgroup
$ sudo openstack server remove security group myinstance1 default
$ sudo openstack server show myinstance1 | grep security_groups
| security_groups | name='demo-sgroup' |
```

Группу безопасности добавили теперь надо выделить ip из внешней сети **ext-net** и добавить его нашей ВМ для подключения:

\$ sudo openstack floating ip create ext-net	
Field	Value
created_at	2022-12-13T12:01:38Z
description	
dns_domain	None
dns_name	None
fixed_ip_address	None
floating_ip_address	10.100.1.172
floating_network_id	574ac2c6-1d90-4c94-87ea-5335464ac5b0
id	51ec175e-ec4b-4839-95a2-dc64bc2b386e
location	Munch({'project': Munch({'domain_name': 'Default', 'domain_id': None, 'name': 'demo', 'id': 'e420dacf94514ed587d878bf32cabf46'}), 'cloud': '', 'region_name': '', 'zone': None})
name	10.100.1.172
port_details	None
port_id	None
project_id	e420dacf94514ed587d878bf32cabf46
qos_policy_id	None
revision_number	0
router_id	None
status	DOWN
subnet_id	None
tags	[]
updated_at	2022-12-13T12:01:38Z

Не переживайте, что у меня ВМ вырублена, это просто чтобы ноут не зависал:

```
$ sudo openstack server add floating ip myinstance1 10.100.1.172
$ sudo openstack server list
+-----+-----+-----+-----+
| ID      | Name    | Status  | Networks          | Image        | Flavor   |
+-----+-----+-----+-----+
| 5dc39d0e-5f1d-4fa8-b188-a39743cd5073 | myinstance1 | SHUTOFF | demo-net=172.16.0.10, 10.100.1.172 | cirros-0.4.0-x86_64 | m2.tiny |
+-----+-----+-----+-----+
```

Теперь с помощью SSH можно подключится к созданной виртуалке по предоставленному ей плавающему ip (только перед этим надо выдать другие разрешения на файл ключей **demokey1**, а то SSH его проигнорирует из-за недостаточной безопасности):

```
$ sudo chmod 400 demokey1
$ sudo ssh -i demokey1 cirros@10.100.1.172
$ whoami
cirros
```

10.3. Переходим к тестированию создания снимков и резервных копий ВМ:

Когда-то там мы создавали мы создавали том **testvol1**, его и используем:

```
$ sudo openstack volume list
+-----+-----+-----+-----+
| ID            | Name    | Status  | Size | Attached to |
+-----+-----+-----+-----+
| 1c603790-ba71-476e-b46c-43ec0ebabb2d | testvol1 | available | 1 |           |
+-----+-----+-----+-----+
```

Добавим том **testvol1** к ВМ **myintstance1** (можно подключать во время создания ВМ при помощи флага **--block-device-mapping**), добавляем к ВМ по id (не по имени ВМ), потому что **admin** не видит ВМ **myintstance1**, так как она находится в другом проекте (опять китайцы помогли исправить баг, на этот раз **OSError: no such file or directory**, не знаю из-за устаревшей книги или из-за чего еще но в файл конфига сервиса **Cinder /etc/cinder/cinder.conf** надо добавить параметр **target_helper** то ли вместо **iscsi_helper** то ли вместе (че поделать все на китайском написано), я добавил вместе и все заработало):

```
$ sudo crudini --set /etc/cinder/cinder.conf lvm target_helper lioadm
```

```
$ sudo openstack server add volume 27dc81a-0833-4f94-9d85-676cc8e122b4 testvol1
```

Теперь он должен стать помечен как использующийся (**in-use**):

```
$ sudo openstack volume list
+-----+-----+-----+-----+
| ID      | Name    | Status  | Size | Attached to |
+-----+-----+-----+-----+
| fee37655-7f1d-4dbb-bc17-f246bb90afea | testvol1 | in-use | 1 | Attached to 27dc81a-0833-4f94-9d85-676cc8e122b4 on /dev/vdb |
+-----+-----+-----+-----+
```

Заходим в ВМ и смотрим наличие тома (потом можно смонтировать на него файловую систему и тд, но я делать этого не буду):

```
$ sudo ssh -i demokey1 cirros@10.100.1.172
$ lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
vda    253:0    0   1G  0 disk
└─vda1  253:1    0 1015M 0 part /
  └─vda15 253:15  0   8M  0 part
vdb    253:16   0   1G  0 disk
```

С томами все теперь переходим к снимкам. Важное уточнение оказывается снимок в **openstack** не является точкой восстановления для существующей ВМ (как в KVM), а является самостоятельным образом (как и скачанный нами **cirros**). Поэтому сники в **Openstack** используются для резервного копирования или создания нового базового образа. Так теперь надо сделать какое-нибудь видное изменение в ВМ **myinstance1**, которое можно было бы увидеть в новой ВМ **myinstance2** созданной из снимка ВМ **myinstance1** (можно сделать снимок тома (**volume snapshot**) и ВМ (**image create**)).

Создадим файл тест в **root** каталоге:

```
$ sudo ssh -i demokey1 cirros@10.100.1.172
$ vi test
test
```

Теперь создаем снимок **myinstance1_sn1** ВМ **myinstance1**, и проверяем его появление в сервисе **Glance**:

```
$ sudo openstack server image create myinstance1 --name myinstance_sn1
+-----+
| Field      | Value
+-----+
| checksum    | None
| container_format | None
| created_at  | 2022-12-14T13:44:49Z
| disk_format | None
| file        | /v2/images/7eleac4d-6c5d-445a-a032-85fe037783d2/file
| id          | 7eleac4d-6c5d-445a-a032-85fe037783d2
| min_disk    | 1
| min_ram    | 0
| name        | myinstance_sn1
| owner       | e420dacf94514ed587d878bf32cabf46
| properties  | base_image_ref='57c57075-fd8c-4475-bd3d-9d1c745849eb',
boot_roles='user', image_type='snapshot', instance_uui
d='27dc81a-0833-4f94-9d85-676cc8e122b4', os_hash_algo
=, os_hash_value=, os_hidden='False', owner_project_na
me='demo', owner_user_name='demo', user_id='b1457723dc
214b9aa669676043c7c7f2'
| protected   | False
| schema     | /v2/schemas/image
| size        | None
| status      | queued
| tags        |
| updated_at  | 2022-12-14T13:44:49Z
| virtual_size | None
| visibility  | private
+-----+
```

```
$ sudo openstack image list
+-----+-----+-----+
| ID           | Name            | Status |
+-----+-----+-----+
| 57c57075-fd8c-4475-bd3d-9d1c745849eb | cirros-0.4.0-x86_64 | active |
| 7eleac4d-6c5d-445a-a032-85fe037783d2 | myinstance_sn1     | active |
+-----+-----+-----+
```

Теперь запускаем новую ВМ **myinstance2**, используя созданный снимок **myinstance1_sn1**:

\$ sudo openstack server create --image myinstance_sn1 --flavor m2.tiny --network demo-net myinstance2	
+-----+-----+	
Field	Value
+-----+-----+	
OS-DCF:diskConfig	MANUAL
OS-EXT-AZ:availability_zone	
OS-EXT-STS:power_state	NOSTATE
OS-EXT-STS:task_state	scheduling
OS-EXT-STS:vm_state	building
OS-SRV-USG:launched_at	None
OS-SRV-USG:terminated_at	None
accessIPv4	
accessIPv6	
addresses	
adminPass	SRuzss7u4YUY
config_drive	
created	2022-12-14T13:50:06Z
flavor	m2.tiny (df7814b9-4900-42de-b5f3-299b684cb8b5)
hostId	
id	7ca55ea1-4513-4457-a27d-3999f63dc4bd
image	myinstance_sn1 (7e1eac4d-6c5d-445a-a032-85fe037783d2)
key_name	None
name	myinstance2
progress	0
project_id	e420dacf94514ed587d878bf32cabf46
properties	
security_groups	name='default'
status	BUILD
updated	2022-12-14T13:50:06Z
user_id	b1457723dc214b9aa669676043c7c7f2
volumes_attached	
+-----+-----+	

Заходим в нее через браузер и смотрим присутствуют ли изменения в ВМ **myinstance2**:

```
myinstance2 login: cirros
Password:
$ ls
test
$ cat test
test
```

Два совета из книги: если вы работаете с виртуальными машинами **Fedor**a, **CentOS**, **RHEL** или им подобными, необходимо перед созданием снимка удалить правило udev для сети, иначе новый экземпляр не сможет создать сетевого интерфейса:

```
$ sudo rm -f /etc/udev/rules.d/70-persistent.rule
```

Если виртуальная машина запущена во время создания снимка, необходимо сбросить на диск содержимое всех файловых буферов и исключить дисковый ввод/вывод каких-либо процессов:

```
$ sudo sync && fsfreeze -f / && sleep && fsfreeze -u /
```

Отличия резервных копий от снимков:

- резервная копия сохраняется в объектное хранилище, а не в сервис работы с образами

- резервная копия занимает столько места, сколько занимают данные, а моментальный снимок копирует весь образ полностью
- резервную копию можно восстановить на новый том, а из моментального снимка можно запустить новый экземпляр виртуальной машины
- в отличие от создания моментального снимка, нельзя делать резервную копию используемого тома

Узнав отличия снимков от резервных копий, можно перейти созданию резервной копии тома **testvol1** (**--force** нужен чтобы бэкапить том в состоянии **in-use**):

```
$ sudo openstack volume backup create --force testvol1
+-----+
| Field | Value
+-----+
| id    | c04aeb11-4715-4a61-9042-07c028eb3761 |
| name  | None
+-----+
```

Теперь тестируем восстановление тома (статус восстановления можно смотреть как и статус запуска ВМ), у меня бэкап не вышел (вылезает ошибка **MemoryError**, так что скорее всего просто мощности не хватает узлу), но команда вот:

```
$ sudo openstack volume restore c04aeb11-4715-4a61-9042-07c028eb3761 testvol1
```

10.4. Особая тема шифрования томов **Cinder**. Шифрование необходимо при общении сервисов **Cinder** и **Nova** и реализуется либо при помощи общего секрета (делаем так), либо при помощи Barbican (сервис управления ключами, у меня нет и не будет в ближайшем будущем). Понятное дело, что если секрет скомпрометирован, то злодей может получить доступ ко всем томам, так что в реальной жизни его хорошо прячем.

Задаем значение секрета в конфигах сервисов **Nova** (**/etc/nova/nova.conf**) и **Cinder** (**/etc/cinder/cinder.conf**) и перезапускаем сервисы:

На вычислительных узлах:

```
$ sudo crudini --set /etc/nova/nova.conf keymgr fixed_key 123456789
$ sudo systemctl restart openstack-nova-compute
```

На управляемом узле:

```
$ sudo crudini --set /etc/cinder/cinder.conf keymgr fixed_key 123456789
$ sudo systemctl restart openstack-cinder-volume
```

Создаем новый тип тома **LUKS**:

```
$ sudo openstack volume type create --encryption-provider
nova.volume.encryptors.luks.LuksEncryptor --encryption-cipher aes-xts-
plain64 --encryption-key-size 512 --encryption-control-location front-end
LUKS
+-----+
| Field      | Value
+-----+
| description | None
| encryption  | cipher='aes-xts-plain64', control_location='front-end',
|              | encryption_id='2e88818d-8cc9-4ae4-8e70-070a7ae3c313',
|              | key_size='512', provider='nova.volume.encryptors.luks
|              | .LuksEncryptor'
| id          | 24e1ed3c-d20a-4ab7-81ea-bc9e84428d1b
| is_public   | True
| name        | LUKS
+-----+
```

- **--encryption-provider** — драйвер обеспечивающий поддержку шифрования
- **--encryption-cipher** — алгоритм или режим шифрования
- **--encryption-key-size** — размер ключа шифрования
- **--encryption-control-location** — служба в которой выполняется шифрование

Теперь создаем сам зашифрованный том (создать его не получится, так как фича с ключами без использования сервиса Barbican устарела, и теперь надо обязательно его разворачивать, чего я пока делать не буду, но команда с примерным выводом вот):

```
$ sudo openstack volume create --size 1 --type LUKS myvolumeEncr
+-----+
| Field           | Value
+-----+
| attachments     | []
| availability_zone | nova
| bootable         | false
| consistencygroup_id | None
| created_at       | 2022-12-14T17:59:28.000000
| description       | None
| encrypted         | True
| id               | 28d8c180-a46a-4749-bd29-490cf1193c0
| migration_status | None
| multiattach       | False
| name              | myvolumeEncr
| properties        |
| replication_status | None
| size              | 1
| snapshot_id       | None
| source_volid       | None
| status             | creating
| type              | LUKS
| updated_at        | None
| user_id            | 549949769da94550a63fa6667741afb4
+-----+
```

- **--size** — размер тома в ГБ
- **--type** — тип тома

С томами все.

10.4. Переходим к тестированию квот на ресурсы. Они используются для управления вычислительными ресурсами и дисковым пространством в разрезе проекта, но можно и для всех пользователей.

Дефолтные квоты всех проектов:

\$ sudo openstack quota show --default	
Field	Value
backup-gigabytes	1000
backups	10
cores	20
fixed-ips	-1
floating-ips	50
gigabytes	1000
gigabytes_LUKS	-1
gigabytes__DEFAULT__	-1
groups	10
health_monitors	None
injected-file-size	10240
injected-files	5
injected-path-size	255
instances	10
key-pairs	100
l7_policies	None
listeners	None
load_balancers	None
location	Munch({'project': Munch({'domain_name': 'Default', 'domain_id': None, 'name': 'admin', 'id': u'33ad52bb25284ad8be4b8681b8fe0063'}), 'cloud': '', 'region_name': '', 'zone': None})
name	None
networks	100
per-volume-gigabytes	-1
pools	None
ports	500
project	None
project_name	admin
properties	128
ram	51200
rbac_policies	10
routers	10
secgroup-rules	100
secgroups	10
server-group-members	10
server-groups	10
snapshots	10
snapshots_LUKS	-1
snapshots__DEFAULT__	-1
subnet_pools	-1
subnets	100
volumes	10
volumes_LUKS	-1
volumes__DEFAULT__	-1

Также можно смотреть квоты для определенного проекта, например **demo**:

Field	Value
backup-gigabytes	1000
backups	10
cores	20
fixed-ips	-1
floating-ips	50
gigabytes	1000
gigabytes_LUKS	-1
gigabytes__DEFAULT__	-1
groups	10
health_monitors	None
injected-file-size	10240
injected-files	5
injected-path-size	255
instances	10
key-pairs	100
l7_policies	None
listeners	None
load_balancers	None
location	Munch({'project': Munch({'domain_name': 'Default', 'domain_id': None, 'name': 'admin', 'id': u'33ad52bb25284ad8be4b8681b8fe0063'}), 'cloud': '', 'region_name': '', 'zone': None})
name	None
networks	100
per-volume-gigabytes	-1
pools	None
ports	500
project	e420dacf94514ed587d878bf32cabf46
project_name	demo
properties	128
ram	51200
rbac_policies	10
routers	10
secgroup-rules	100
secgroups	10
server-group-members	10
server-groups	10
snapshots	10
snapshots_LUKS	-1
snapshots__DEFAULT__	-1
subnet_pools	-1
subnets	100
volumes	10
volumes_LUKS	-1
volumes__DEFAULT__	-1

Можно конечно самому повыставлять квот, но опыт **LimitRange** и **ResourceQuota** при работе с **Kubernetes** говорит мне переходить к следующему пункту.

10.5. Итак зоны доступности и агрегаторы вычислительных узлов. Зоны доступности позволяют группировать узлы **OpenStack** в логические группы с общим элементом доступности (например одна стойка или одно здание) и по умолчанию все узлы находятся в зоне **nova**. Агрегаторы узлов же – это логическое объединение узлов с привязкой дополнительных метаданных. Агрегаторы позволяют группировать узлы по

различным специфическим признакам (потом **Nova** автоматически будет разворачивать ВМ на более подходящих узлах). У меня один узел уже умер, но я решил все равно рассказать про эти объекты.

Для примера и тестирования создадим агрегатор **MyAggregate**:

Field	Value
availability_zone	None
created_at	2022-12-14T18:08:48.722328
deleted	False
deleted_at	None
hosts	None
id	1
name	MyAggregate
properties	None
updated_at	None

Добавим в агрегатор **MyAggregate** рабочий узел **compute.test.local** (один узел может состоять в нескольких агрегаторах):

Field	Value
availability_zone	None
created_at	2022-12-14T18:08:48.000000
deleted	False
deleted_at	None
hosts	compute.test.local
id	1
name	MyAggregate
properties	None
updated_at	None

Добавим метаданные агрегатору **MyAggregate**:

Field	Value
availability_zone	None
created_at	2022-12-14T18:08:48.000000
deleted	False
deleted_at	None
hosts	compute.test.local
id	1
name	MyAggregate
properties	mynewmetadata='true'
updated_at	None

Создадим новый тип (**flavour**) ВМ **m2.mynewmdvm** (с помощью него затестим агрегатор):

Field	Value
OS-FLV-DISABLED:disabled	False
OS-FLV-EXT-DATA:ephemeral	0
disk	1
id	0ee57176-e812-4fe1-ad25-f241ebbe4fa9
name	m2.mynewmdvm
os-flavor-access:is_public	True
properties	
ram	300
rxtx_factor	1.0
swap	
vcpus	1

Добавим к созданному типу ВМ **m2.mynewmdvm** метаданные, чтобы ВМ этого типа могли располагаться только на узлах определенного агрегатора **MyAggregate** и проверим получившийся тип:

\$ sudo openstack flavor set --property mynewmetadata=true m2.mynewmdvm	
\$ sudo openstack flavor show m2.mynewmdvm	
Field	Value
OS-FLV-DISABLED:disabled	False
OS-FLV-EXT-DATA:ephemeral	0
access_project_ids	None
disk	1
id	0ee57176-e812-4fe1-ad25-f241ebbe4fa9
name	m2.mynewmdvm
os-flavor-access:is_public	True
properties	mynewmetadata='true'
ram	300
rxtx_factor	1.0
swap	
vcpus	1

Все теперь ВМ этого типа будет стартовать только на узле **compute.test.local**, так как метаданные типа ВМ **m2.mynewmdvm** и агрегатора **MyAggregate** в котором состоит узел совпадают (поверьте на слово ибо одновременно два рабочих узла поднять я не могу).

Зона доступности, как можно понять, является недоагрегатором, и ее можно создать если приписать агрегатору определенное имя зоны доступности, вот пример (агрегатор — **MyAggregateZ**, имя зоны — **MyZone**):

\$ sudo openstack aggregate create --zone MyZone	
MyAggregateZ	
Field	Value
availability_zone	MyZone
created_at	2022-12-14T18:11:56.285021
deleted	False
deleted_at	None
hosts	None
id	2
name	MyAggregateZ
properties	None
updated_at	None

Потом можно добавить узел **compute-opt.test.local** в данный агрегатор **MyAggregateZ**, и зона доступности узла изменится:

```
$ sudo openstack aggregate add host MyAggregateZ
compute-opt.test.local
+-----+-----+
| Field | Value |
+-----+-----+
| availability_zone | MyZone
| created_at | 2022-12-14T18:11:56.000000
| deleted | False
| deleted_at | None
| hosts | compute-opt.test.local
| id | 2
| name | MyAggregateZ
| properties | availability_zone='MyZone'
| updated_at | None
+-----+-----+
```

```
$ sudo openstack host list
+-----+-----+-----+
| Host Name | Service | Zone |
+-----+-----+-----+
| controller.test.local | conductor | internal |
| controller.test.local | scheduler | internal |
| compute.test.local | compute | nova |
| compute-opt.test.local | compute | MyZone |
+-----+-----+-----+
```

Теперь можно запустить ВМ с указанием зоны доступности (**--availability-zone**) и она развернется исключительно на узле **compute-opt.test.local** (если он у вас конечно работает, у меня просто уже нет **compute-opt** узла так что держите только пример):

```
$ sudo openstack server create --image myinstance_s1 --flavor m2.tiny
--network demo-net --availability-zone MyZone myinstance3
```

Также можно изменить дефолтную зону доступности для кластера при помощи параметра **default_availability_zone** в **/etc/nova/nova.conf** на управляемом узле.

Тест закончен, удаляем узлы из агрегаторов и сами агрегаторы **MyAggregate** и **MyAggregateZ**:

```
$ sudo openstack aggregate remove host MyAggregateZ
compute-opt.test.local
+-----+-----+
| Field | Value |
+-----+-----+
| availability_zone | MyZone
| created_at | 2022-12-14T18:11:56.000000
| deleted | False
| deleted_at | None
| hosts |
| id | 2
| name | MyAggregateZ
| properties | availability_zone='MyZone'
| updated_at | None
+-----+-----+
```

```
$ sudo openstack aggregate remove host MyAggregate  
compute.test.local
```

Field	Value
availability_zone	None
created_at	2022-12-14T18:08:48.000000
deleted	False
deleted_at	None
hosts	
id	1
name	MyAggregate
properties	mynewmetadata='true'
updated_at	None

```
$ sudo openstack aggregate delete MyAggregateZ  
$ sudo openstack aggregate delete MyAggregate  
$ sudo openstack aggregate list  
...ПУСТОТА...
```

Также стоит отметить зоны доступности **Cinder**, чтобы монтировать подходящие для ВМ тома (принцип действия абсолютно такой же).

10.6. Тестируем живую миграцию ВМ. Есть два типа миграции:

- с общей системой хранения данных. Виртуальная машина перемещается между двумя вычислительными узлами с общим хранилищем, к которым оба узла имеют доступ (например **NFS**, **Ceph** или вариант, когда не используются временные диски (которые удаляются при удалении ВМ), а в качестве единственной системы хранения при создании виртуальных машин используется **Cinder**)
- без общей системы хранения данных. В этом случае на миграцию требуется больше времени, поскольку виртуальная машина копируется целиком с узла на узел по сети.

У меня миграцию произвести не выйдет, так как не могу одновременно запустить два рабочих узла, но шаги я опишу (правда без выводов команд).

Первым делом проверяем видят ли рабочие узлы **compute** и **compute-opt** друг друга:

На **compute-opt** узле:

```
$ sudo ping compute
```

На **compute** узле:

```
$ sudo ping compute-opt
```

Теперь запускаем ВМ **test** и смотрим на каком узле она находится, а также достаточно ли ресурсов на другом узле:

```
$ sudo openstack server create --image myinstance_sn1 --flavor m2.tiny --network demo-net test
```

```
$ sudo openstack server show test | grep OS-EXT-SRV-ATTR:host
```

```
$ sudo openstack host show compute(или)compute-opt.test.local
```

Разрешаем демону **libvirtd** слушать входящие подключения по сети, добавив следующую опцию в файл **/etc/sysconfig/libvirtd.conf** на обоих рабочих узлах:

```
$ sudo vi /etc/sysconfig/libvirtd.conf  
...  
LIBVIRTD_ARGS="--listen"
```

Разрешаем подключение без аутентификации и шифрования к демону **libvirtd**, редактируя файл конфига **/etc/libvirt/libvirtd.conf** (можно также использовать сертификаты или протокол **Kerberos**), после чего перезапускаем демон:

```
$ sudo vi /etc/libvirt/libvirtd.conf  
listen_tis = 0  
listen_tcp = 1  
auth_tcp= «none»  
$ sudo systemctl restart libvirtd
```

Далее надо добавить флаги миграции в файл конфига **/etc/nova/nova.conf** на обоих рабочих узлах (параметр уже устарел, так что я не смог найти подробную информацию) и перезапустить службу **nova-compute**:

```
$ sudo crudini --set /etc/nova/nova.conf DEFAULT block_migration_flag  
VIR_MIGRATE_UNDEFINE_SOURCE,VIR_MIGRATE_PEER2PEER,VIR_MIGRATE_LIV
```

```
$ sudo systemctl restart openstack-nova-compute
```

Теперь можно осуществить живую миграцию на другой узел (**--block-migrate** — миграция без общего дискового хранилища):

```
$ sudo openstack server migrate --block-migrate compute/compute-opt.test.local
```

Смотрим, где теперь работает ВМ (если все получилось, то на другом узле):

```
$ sudo openstack server show test | OS-EXT-SRV-ATTR:host
```

Информацию о миграции можно найти в параметрах ВМ, которая мигрирует:

```
$ sudo openstack server show test | grep status
```

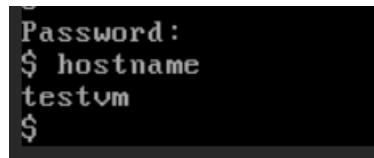
10.7. Удобный (или нет) способ конфигурировать ВМ при запуске по названию [cloud-init](#). Заключается в создании **YAML-файлов** с синтаксисом **cloud-config**, в которые записываются описываются модули, исполняемые во время загрузки ВМ (например модуль установки пакетов), после содержимое **YAML-файла** передаются при создании ВМ (при помощи флага **--user-data**) и конфигурирует виртуалку (например задает имя хоста или предустанавливает пакеты).

Вот пример содержимого YAML-файла **cloud-config**, задающего имя ВМ:

```
$ sudo vi test.yaml
#cloud-config
hostname: testvm
fqdn: testvm.test.local
manage_etc_hosts: true
```

После чего этот файл можно передать при создании ВМ **testvm**, и на ней как нетрудно догадаться имя хоста будет **testvm.test.local** (зайти в ВМ можно через **noVNC**):

```
$ sudo openstack server create --image myinstance_s1 --flavor m2.tiny --network demo-net --user-data test.yaml testvm
+-----+
| Field          | Value
+-----+
| OS-DCF:diskConfig | MANUAL
| OS-EXT-AZ:availability_zone |
| OS-EXT-STS:power_state | NOSTATE
| OS-EXT-STS:task_state | scheduling
| OS-EXT-STS:vm_state | building
| OS-SRV-USG:launched_at | None
| OS-SRV-USG:terminated_at | None
| accessIPv4 |
| accessIPv6 |
| addresses |
| adminPass | Ncwdm7bLSDED
| config_drive |
| created | 2022-12-14T19:02:30Z
| flavor | m2.tiny (df7814b9-4900-42de-b5f3-299b884cb8b5)
| hostId |
| id | 2f842dc3-10c8-403f-93de-c0880b9e7a8b
| image | myinstance_s1 (7e1eac4d-6c5d-445a-a032-85fe037783d2)
| key_name | None
| name | testvm
| progress | 0
| project_id | e420dac94514ed587d878bf32cabf46
| properties |
| security_groups | name='default'
| status | BUILD
| updated | 2022-12-14T19:02:30Z
| user_id | b1457723dc214b9aa669676043c7c7f2
| volumes_attached |
```



Все на этом можно закончить тесты базовой работы кластера и перейти к установке оставшихся сервисов.

11. Установка и настройка сервиса веб-панели управления Horizon.

11.1. Сервис представляет собой удобный графический интерфейс в виде веб-приложения (на **Python/Django**), через который можно работать с Openstack. Больше собственно сказать о нем нечего, разве что на нем пока представлено только 70% функционала **openstack cli**.

11.2. Теории не много, так что приступаем к установке и конфигурации.

Первым делом устанавливаем необходимые пакеты **httpd** (Apache HTTP сервер) **mod_ssl** (модуль **Apache**, обеспечивающий криптографию через протоколы SSL и TLS) **mod_wsgi** (модуль **Apache**, предоставляющий интерфейс для работы с **web-приложениями**, написанными на **Python**) **memcached** (сервис кэширования данных в оперативной памяти на основе хеш-таблицы) **python-memcached** (Python интерфейс для работы с **memcached**) **openstack-dashboard** (сервис **Horizon**) на **controller.test.local** узел (некоторые пакеты уже установлены, но все равно их перечислю, так как **Horizon** можно установить и на другие узлы и даже ВМ):

```
$ sudo yum -y install httpd mod_ssl mod_wsgi memcached python-memcached openstack-dashboard
```

Теперь конфигурируем конфиг **/etc/openstack-dashboard/local_settings** сервиса **Horizon**, изменяя опции

- **OPENSTACK_HOST** — IP-адрес узла, где запущен Keystone (или имя)
- **ALLOWED_HOSTS** — разрешенные ip для пользования **Horizon** ([“*”] - все могут, но желательно перечислить избранных через запятую)
- **SESSION_TIMEOUT** — время жизни сессии в **Horizon** в секундах, можно увеличить тут и в конфиге **Keystone** (**expiration=3600** секции **[Token]**) если образы ВМ очень большие и долго гружаются
- **WEBROOT** — URL панели мониторинга относительно ip хоста (этот параметр я вымучил кровью и потом, почему-то с определенного момента ни один шаг не проходит без каких-либо ошибок)
- **SESSION_ENGINE** — указание модуль обработки сеансов (**Django**) пользователя
- **CACHES** — сервис кэширования и его местоположение (ip адрес)
- **OPENSTACK_KEYSTONE_URL** — URL сервиса **Keystone**, но мы его редактируем, чтобы включить третью версию API (у меня редактировать не пришлось)

- **OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT** — поддержка доменов (openstack доменов, не путать с обычными)
- **OPENSTACK_API_VERSIONS** - поддерживаемые версии API сервисов
- **OPENSTACK_KEYSTONE_DEFAULT_DOMAIN** — имя домена по умолчанию

```
$ sudo vi /etc/openstack-dashboard/local_settings
...
ALLOWED_HOSTS = ['*']
...
OPENSTACK_HOST = "controller"
...
SESSION_ENGINE = 'django.contrib.sessions.backends.cache'
CACHES = {
    'default': {
        'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',
        'LOCATION': 'controller:11211',
    }
}
...
OPENSTACK_KEYSTONE_URL = "http://%s:5000/identity/v3" % OPENSTACK_HOST
...
OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = True
OPENSTACK_API_VERSIONS = {
    "identity": 3,
    "image": 2,
    "volume": 3,
}
OPENSTACK_KEYSTONE_DEFAULT_DOMAIN = "Default"
SESSION_TIMEOUT = 3600
WEBROOT = '/dashboard'
```

Теперь указываем параметр **WSGIApplicationGroup** (группа приложений в которой будет запускаться **Horizon**, в книге написано, что это фиксит некоторые баги) в файле конфига **/etc/httpd/conf.d/openstack-dashboard.conf** и перезапускаем службы **httpd** и **memcached**:

```
$ sudo vi /etc/httpd/conf.d/openstack-dashboard.conf
...
WSGIApplicationGroup %{GLOBAL}
```

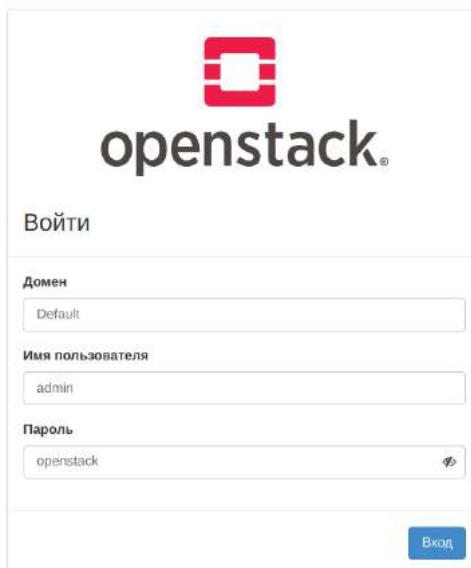
```
$ sudo systemctl enable memcached.service
Created symlink from /etc/systemd/system/multi-user.target.wants/memcached.service to
/usr/lib/systemd/system/memcached.service.
$ sudo systemctl restart httpd.service
$ sudo systemctl status httpd.service
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset:
disabled)
   Drop-In: /usr/lib/systemd/system/httpd.service.d
             └─openstack-dashboard.conf
   Active: active (running) since Чт 2022-12-15 15:25:42 MSK; 32s ago
```

```
$ sudo systemctl restart memcached.service
$ sudo systemctl status memcached.service
● memcached.service - memcached daemon
  Loaded: loaded (/usr/lib/systemd/system/memcached.service; enabled; vendor preset:
disabled)
  Active: active (running) since Чт 2022-12-15 15:25:28 MSK; 1min 3s ago
```

Веб-интерфейс Horizon будет настраиваться по адресу <http://controller.test.local/dashboard> (в моем случае <http://192.168.122.200/dashboard>):

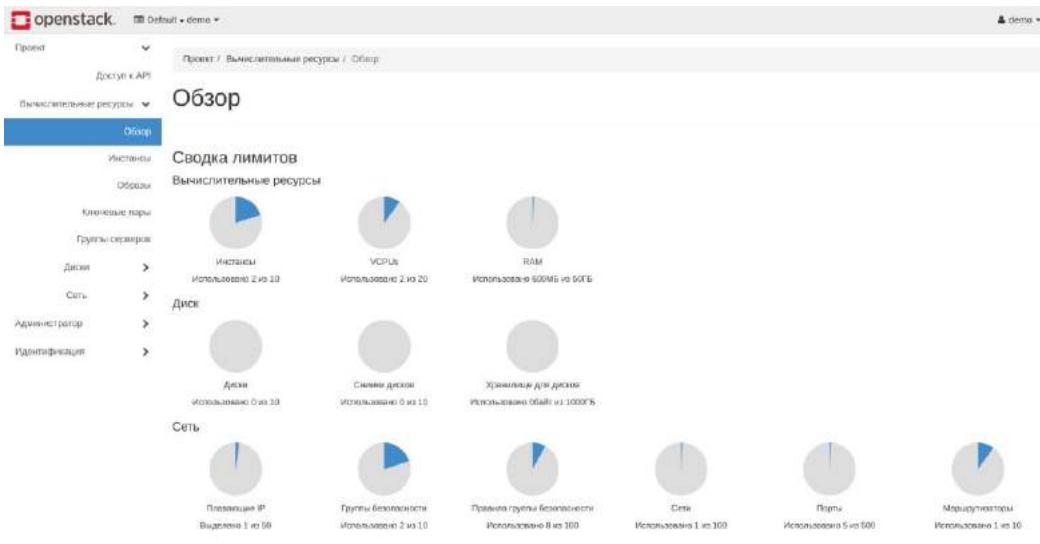
Как же тяжело даются последние сервисы, короче у меня при авторизации вылезала ошибка «**InternalServerError**», и мало, что она вылезала дак к этому еще файл логов ошибок **httpd /var/log/httpd/error_log** решил сломать вывод времени ошибки (выводил время ошибки на три часа раньше чем на самом деле). Вообщем оказалось, что проблема в **memcached**, который блокал весь трафик, кроме **localhost**, по итогу надо было поменять параметр **OPTIONS** и все заработало (еще я почистил кэш браузера мб это тоже помогло):

```
$ sudo vi /etc/sysconfig/memcached
PORT="11211"
USER="memcached"
MAXCONN="1024"
CACHESIZE="64"
OPTIONS="-l 0.0.0.0"
```



11.3. Horizon запустили теперь смотрим что внутри.

Заходим за пользователя **demo** и видим следующую картину:



Содержимое главной вкладки **Проект**:

- **Вычислительные ресурсы:**

- **Обзор** – общий обзор ресурсов пользователя. Можно посмотреть текущее потребление ресурсов, а также получить краткий отчет за это время
- **Инстансы** – список виртуальных машин для текущего проекта, а также возможность производить с виртуальными машинами различные действия, например создание новой виртуальной машины или подключение к консоли и просмотр вывода терминала виртуальной машины
- **Образы** – управление образами **Glance**. Тут же из образа можно стартовать виртуальную машину и создать том
- **Ключевые пары** — управление ключами доступа по SSH
- **Доступ к API** — URL, по которым доступны API соответствующих служб облака
- **Группы серверов** — управление группами серверов (придуманы, чтобы например запретить VM внутри группы подниматься на одном узле)

- **Диски**

- **Диски** — управление томами (все тома у нас в проекте **admin**, так что **demo** пользователь их не видит)
- **Снимок** — управление снимками томов
- **Группы** — управление **группами томов**
- **Снимки групп** — управление снимками групп томов

- **Сеть**

- **Сетевая топология** – топология сети проекта.
- **Сети** – список сетей проекта. Тут же создаются подсети, а также находится список портов с возможностью редактирования

- **Роутеры** – список маршрутизаторов проекта и управление ими
- **Плавающие IP** - управление «плавающими» IP-адресами
- **Группы безопасности** — управление группами безопасности
- **Объектное хранилище** (у меня нет, так как нет **Swift**)
 - **Контейнеры** – создание контейнеров и псевдопапок, а также загрузка в объектное хранилище файлов и их скачивание.
- **Оркестрация** (пока нет потом появится с установкой **Heat**)
 - **Стеки** – управление стеками, включая их запуск и мониторинг.

Содержимое вкладки **Администратор** (будет выводить инфу только для пользователя **admin**):

- **Обзор** – тут располагается статистика по всем проектам за заданный период времени;
- **Использование ресурсов** – вывод информации от **OpenStack Telemetry (Ceilometer)**, поэтому пока ее нет
- **Вычислительные ресурсы**:
 - **Гипервизоры** – информация по всем гипервизорам, управляемым **OpenStack** (кликнув по имени узла, можно посмотреть, какие виртуальные машины он обслуживает)
 - **Агрегаты узлов** – управление агрегатами, настройка группировок узлов как на логические группы, так и группировка их по физическому месторасположению
 - **Инстансы** – управление инстансами, прямо как во вкладке **Проект** только отображает все виртуальные машины
 - **Типы инстансов** – управление типами экземпляров виртуальных машин
 - **Образы** – управление образами **Glance**. Отображаются все образы
- **Диски** – аналогична одноименной во вкладке **Проект**, однако включает дополнительные подвкладки:
 - **Типы дисков** — определение типов томов и соответствующих им параметров качества обслуживания (**QoS**)
 - **Типы групп** — аналогично типам томов
- **Сеть**
 - **Сети** – аналогична одноименной во вкладке **Проект**, но включает в себя сети всех проектов, а также тут администратор также может задать внешнюю сеть
 - **Маршрутизаторы** – опять же как и во вкладке **Проект**, но включает в себя все маршрутизаторы

- Плавающие IP — как во вкладке Проект, только все IP
- Политики RBAC — управление политиками доступа [RBAC](#) к сетям
- Система
 - Параметры по умолчанию — можно задать квоты для проектов, действующие по умолчанию
 - Определения метаданных — управление собственными метаданными, которые потом можно использовать при создании ВМ (и не только) по умолчанию или по просьбе пользователей
 - Системная информация — информация о сервисах облака и их текущем состоянии

Последняя вкладка **Идентификация**:

- Проекты — управление всеми проектами
- Пользователи — управление всеми пользователями
- Группы — управление всеми группами пользователей
- Роли — управление всеми ролями
- Доступ для приложений — управление учетными данными приложений (**application credentials** придуманы, чтобы чтобы приложения могли проходить аутентификацию в **Keystone**)

На этом с **Horizon** все. Также можно еще менять тему интерфейса, но я пожалуй откажусь.

12. Установка и настройка сервиса сбора телеметрии Telemetry (Ceilometer, Aodh, Gnocchi)

12.1. Итак сервис **Openstack Telemetry** состоит из трех отдельных проектов связанных с мониторингом:

- **Ceilometer** — отвечает за мониторинг и сбор данных, поддерживает два способа сбора:
 - при помощи очереди сообщений через службу **ceilometer-collector** (запускается на одном или более управляющих узлах и отслеживает очередь сообщений, получает уведомления от других сервисов **Openstack** и, преобразовывая их в сообщения телеметрии, отправляет обратно в очередь сообщений)
 - через опрашивающие инфраструктуру агенты (через вызовы API агенты периодически запрашивают у сервисов **Openstack** необходимую информацию)

Состоит из следующих служб:

- **openstack-ceilometer-notification** — агент, отправляющий по протоколу AMQP метрики сборщику от различных сервисов

- **openstack-ceilometer-central** – агент, запускаемый на центральном сервере для запроса статистики по загрузке, не связанной с экземплярами виртуальных машин или вычислительными узлами (можно запустить несколько штук)
 - **openstack-ceilometer-compute** – агент, запускаемый на всех вычислительных узлах для сбора статистики по узлам и экземплярам виртуальных машин
- **Aodh** – отвечает за обработку триггеров (**alarms**). Триггеры срабатывают при достижении метрикой определенного значения, после чего в **Openstack** можно обратится по HTTP на определенный адрес или записать событие в журнал. Состоит из следующих служб:
 - **openstack-aodh-api** – API-сервер, который запускается на одном или более узлах. Служит для предоставления доступа к информации о сработавших триггерах (**alarms**)
 - **openstack-aodh-evaluator** – сервис, определяющий, сработал ли триггер при достижении метриками заданных значений в течение определенного измеряемого периода
 - **openstack-aodh-notifier** – сервис, запускающий те или иные действия при срабатывании триггера
 - **openstack-aodh-listener** – сервис, определяющий, когда триггер сработает (срабатывание определяется сравнением правил срабатывания триггера и событий, полученных агентами сбора телеметрии)
- **Gnocchi** – бэкэнд для хранения метрик, собранных сервисом телеметрии. Придуман чтобы хранить данные в **Ceph** (или файловой системе), так как это удобнее. Состоит из двух служб:
 - **openstack-gnocchi-api** — сервер приема API запросов на управляющем узле
 - **openstack-gnocchi-metricd** — агент на управляющем узле, который в реальном времени вычисляет статистику приходящих данных и записывает их в файловую систему, а также индексирует эти данные и записывает их индексы в БД (у нас **MariaDB**), для быстрого доступа к ним.

Сами же **метрики** бывают трех типов (если работали с **Prometeus**, то узнаете этих троих как родных):

- накопительные счетчики (**cumulative**) – постоянно увеличивающиеся со временем значения
- индикаторы (**gauge**) – дискретные и плавающие значения, например ввод/вывод диска или присвоенные «плавающие IP»
- дельта (**delta**) – изменение со временем, например пропускная способность сети

Грустно это признавать, но данный сервис (или сервисы) развернуть мне не удастся. Вы представляете 6 ГБ оперативки на управляющем узле уже не хватает (а ведь только в прошлом году я совершенно без проблем разворачивал **Kubernetes** на трех узлах), так что опять описание шагов, без собственного выполнения (может в новом году). Вот пруф плачевного состояния:

```
$ sudo free -h
total        used        free      shared  buff/cache   available
Mem:       5,7G       5,4G      142M       9,0M       173M       83M
Swap:      615M       1,0M      614M
```

12.2. Приступаем к установке служб **Gnocchi**, **Ceilometer** и **Aodh** на управляющем (**controller.test.local**) узле (можно на разных).

Установим службы **Gnocchi**:

```
$ sudo yum -y install openstack-gnocchi-api openstack-gnocchi-metricd python-gnocchiclient
```

Проведем классическую регистрацию сервисов **Gnocchi** и **Ceilometer** в **Keystone**:

```
$ sudo openstack user create --domain default --password openstack ceilometer
$ sudo openstack user create --domain default --password openstack gnocchi
$ sudo openstack role add --project service --user ceilometer admin
$ sudo openstack role add --project service --user gnocchi admin
$ sudo openstack service create --name gnocchi --description "Metric Service" metric
$ sudo openstack endpoint create --region RegionOne metric public http://controller.test.local:8041
$ sudo openstack endpoint create --region RegionOne metric internal
http://controller.test.local:8041
$ sudo openstack endpoint create --region RegionOne metric admin http://controller.test.local:8041
```

Создаем БД **gnocchi** для хранения индексов службы **Gnocchi**:

```
$ sudo mysql -u root -p
CREATE DATABASE gnocchi;
GRANT ALL PRIVILEGES ON gnocchi.* TO 'gnocchi'@'localhost' IDENTIFIED BY 'openstack';
GRANT ALL PRIVILEGES ON gnocchi.* TO 'gnocchi'@'%' IDENTIFIED BY 'openstack';
```

Переходим к настройке файла конфига **/etc/gnocchi/gnocchi.conf** сервиса **Gnocchi**:

Указываем параметры авторизации в **Keystone**:

```
$ sudo crudini --set /etc/gnocchi/gnocchi.conf api auth_mode keystone
$ sudo crudini --set /etc/gnocchi/gnocchi.conf keystone_authtoken auth_url
http://controller.test.local:5000/v3
$ sudo crudini --set /etc/gnocchi/gnocchi.conf keystone_authtoken auth_type password
$ sudo crudini --set /etc/gnocchi/gnocchi.conf keystone_authtoken project_domain_name
Default
$ sudo crudini --set /etc/gnocchi/gnocchi.conf keystone_authtoken user_domain_name
Default
$ sudo crudini --set /etc/gnocchi/gnocchi.conf keystone_authtoken project_name service
$ sudo crudini --set /etc/gnocchi/gnocchi.conf keystone_authtoken username gnocchi
$ sudo crudini --set /etc/gnocchi/gnocchi.conf keystone_authtoken password openstack
$ sudo crudini --set /etc/gnocchi/gnocchi.conf keystone_authtoken interface internalURL
$ sudo crudini --set /etc/gnocchi/gnocchi.conf keystone_authtoken region_name RegionOne
```

Указываем параметры подключения к БД:

```
$ sudo crudini --set /etc/gnocchi/gnocchi.conf indexer url  
mysql+pymysql://gnocchi:openstack@controller.test.local/gnocchi
```

Указываем место в файловой системе, где будут хранится данные метрик (можно хранить локально, можно например в Серф, если крутой, но я локально):

```
$ sudo crudini --set /etc/gnocchi/gnocchi.conf storage  
file_basepath /var/lib/gnocchi  
$ sudo crudini --set /etc/gnocchi/gnocchi.conf storage driver file
```

Инициализируем **Gnocchi**:

```
$ sudo gnocchi-upgrade
```

Запускаем и включаем все службы **Gnocchi**:

```
$ sudo systemctl enable openstack-gnocchi-api.service  
$ sudo systemctl enable openstack-gnocchi-metricd.service  
$ sudo systemctl start openstack-gnocchi-api.service  
$ sudo systemctl start openstack-gnocchi-metricd.service
```

Переходим к сервису **Ceilometer** и качаем его службы:

```
$ sudo yum -y install openstack-ceilometer-notification openstack-ceilometer-central
```

Настраиваем файл конфига **/etc/ceilometer/ceilometer.conf** сервиса **Ceilometer**:

Указываем информацию по подключению к сервису **RabbitMQ**:

```
$ sudo crudini --set /etc/ceilometer/ceilometer.conf DEFAULT transport_url  
rabbit://openstack:openstack@controller.test.local
```

Указываем реквизиты сервиса **Ceilometer**:

```
$ sudo crudini --set /etc/ceilometer/ceilometer.conf service_credentials auth_type  
password  
$ sudo crudini --set /etc/ceilometer/ceilometer.conf service_credentials auth_url  
http://controller.test.local:5000/v3  
$ sudo crudini --set /etc/ceilometer/ceilometer.conf service_credentials  
project_domain_id default  
$ sudo crudini --set /etc/ceilometer/ceilometer.conf service_credentials  
user_domain_id default  
$ sudo crudini --set /etc/ceilometer/ceilometer.conf service_credentials  
project_name service  
$ sudo crudini --set /etc/ceilometer/ceilometer.conf service_credentials username  
ceilometer  
$ sudo crudini --set /etc/ceilometer/ceilometer.conf service_credentials password  
openstack  
$ sudo crudini --set /etc/ceilometer/ceilometer.conf service_credentials interface  
internalURL  
$ sudo crudini --set /etc/ceilometer/ceilometer.conf service_credentials region_name  
RegionOne
```

Инициализируем **Ceilometer**:

```
$ sudo ceilometer-upgrade
```

Запускаем и включаем все службы **Ceilometer**:

```
$ sudo systemctl enable openstack-ceilometer-notification.service  
$ sudo systemctl enable openstack-ceilometer-central.service  
$ sudo systemctl start openstack-ceilometer-notification.service  
$ sudo systemctl start openstack-ceilometer-central.service
```

Теперь качаем службы для сервиса **Aodh** (**python-aodhclient** модуль для работы с **Python**):

```
$ sudo yum -y install openstack-aodh-api openstack-aodh-evaluator openstack-aodh-notifier  
openstack-aodh-listener openstack-aodh-expirer python-aodhclient
```

Создаем БД **aodh** для хранения информации о триггерах сервиса **Aodh**:

```
$ sudo mysql -u root -p  
CREATE DATABASE aodh;  
GRANT ALL PRIVILEGES ON aodh.* TO 'aodh'@'localhost' IDENTIFIED BY 'openstack';  
GRANT ALL PRIVILEGES ON aodh.* TO 'aodh'@'%' IDENTIFIED BY 'openstack';
```

Настраиваем файл конфига **/etc/aodh/aodh.conf** сервиса **Aodh**:

Регистрируем сервис **Aodh** в **Keystone**:

```
$ sudo openstack user create --domain default --password openstack aodh  
$ sudo openstack role add --project service --user aodh admin  
$ sudo openstack service create --name aodh --description "Telemetry" alarming  
$ sudo openstack endpoint create --region RegionOne alarming public  
http://controller.test.local:8042  
$ sudo openstack endpoint create --region RegionOne alarming internal  
http://controller.test.local:8042  
$ sudo openstack endpoint create --region RegionOne alarming admin  
http://controller.test.local:8042
```

Указываем параметры авторизации в **Keystone**:

```
$ sudo crudini --set /etc/aodh/aodh.conf DEFAULT auth_strategy keystone
$ sudo crudini --set /etc/aodh/aodh.conf keystone_auth_token www_authenticate_uri
http://controller.test.local:5000
$ sudo crudini --set /etc/aodh/aodh.conf keystone_auth_token auth_url
http://controller.test.local:35357
$ sudo crudini --set /etc/aodh/aodh.conf keystone_auth_token auth_type password
$ sudo crudini --set /etc/aodh/aodh.conf keystone_auth_token project_domain_id
default
$ sudo crudini --set /etc/aodh/aodh.conf keystone_auth_token user_domain_id
default
$ sudo crudini --set /etc/aodh/aodh.conf keystone_auth_token project_name service
$ sudo crudini --set /etc/aodh/aodh.conf keystone_auth_token username aodh
$ sudo crudini --set /etc/aodh/aodh.conf keystone_auth_token password openstack
```

Теперь указываем реквизиты самого **Aodh**:

```
$ sudo crudini --set /etc/aodh/aodh.conf service_credentials auth_type password
$ sudo crudini --set /etc/aodh/aodh.conf service_credentials auth_url
http://controller.test.local:5000/v3
$ sudo crudini --set /etc/aodh/aodh.conf service_credentials project_domain_id default
$ sudo crudini --set /etc/aodh/aodh.conf service_credentials user_domain_id default
$ sudo crudini --set /etc/aodh/aodh.conf service_credentials project_name service
$ sudo crudini --set /etc/aodh/aodh.conf service_credentials username aodh
$ sudo crudini --set /etc/aodh/aodh.conf service_credentials password openstack
$ sudo crudini --set /etc/aodh/aodh.conf service_credentials interface internalURL
$ sudo crudini --set /etc/aodh/aodh.conf service_credentials region_name RegionOne
```

И параметры подключения к БД **aodh** и **RabbitMQ**:

```
$ sudo crudini --set /etc/aodh/aodh.conf database connection
mysql+pymysql://aodh:openstack@controller.test.local/aodh
$ sudo crudini --set /etc/aodh/aodh.conf DEFAULT transport_url
rabbit://openstack:openstack@controller.test.local
```

Синхронизируем БД **aodh** с сервисом:

```
$ sudo aodh-dbsync
```

Теперь можно запустить все службы:

```
$ sudo systemctl enable openstack-aodh-api.service
$ sudo systemctl enable openstack-aodh-evaluator.service
$ sudo systemctl enable openstack-aodh-notifier.service
$ sudo systemctl enable openstack-aodh-listener.service
$ sudo systemctl start openstack-aodh-api.service
$ sudo systemctl start openstack-aodh-evaluator.service
$ sudo systemctl start openstack-aodh-notifier.service
$ sudo systemctl start openstack-aodh-listener.service
```

12.3. Переходим к установке службы **openstack-ceilometer-compute** на рабочих (**compute.test.local**) узлах (для всех узлов действия одинаковы):

Устанавливаем службу **openstack-ceilometer-compute**:

```
$ sudo yum -y install openstack-ceilometer-compute
```

Задаем параметры файлу конфига **/etc/ceilometer/ceilometer.conf** сервиса **Ceilometer**:

URL брокера сообщений **RabbitMQ**:

```
$ sudo crudini --set /etc/ceilometer/ceilometer.conf DEFAULT transport_url rabbit://openstack:openstack@controller.test.local
```

Указываем собственные реквизиты сервиса **Ceilometer**:

```
$ sudo crudini --set /etc/ceilometer/ceilometer.conf service_credentials auth_type password  
$ sudo crudini --set /etc/ceilometer/ceilometer.conf service_credentials auth_url http://controller.test.local:5000/v3  
$ sudo crudini --set /etc/ceilometer/ceilometer.conf service_credentials project_domain_id default  
$ sudo crudini --set /etc/ceilometer/ceilometer.conf service_credentials user_domain_id default  
$ sudo crudini --set /etc/ceilometer/ceilometer.conf service_credentials project_name service  
$ sudo crudini --set /etc/ceilometer/ceilometer.conf service_credentials username ceilometer  
$ sudo crudini --set /etc/ceilometer/ceilometer.conf service_credentials password openstack  
$ sudo crudini --set /etc/ceilometer/ceilometer.conf service_credentials interface internalURL  
$ sudo crudini --set /etc/ceilometer/ceilometer.conf service_credentials region_name RegionOne
```

Указываем в конфиге **/etc/nova/nova.conf** сервиса **Nova**, чтобы он начал отправлять сообщения через брокер сообщений:

```
$ sudo crudini --set /etc/nova/nova.conf DEFAULT instance_usage_audit_period hour  
$ sudo crudini --set /etc/nova/nova.conf DEFAULT notify_on_state_change vm_and_task_state  
$ sudo crudini --set /etc/nova/nova.conf oslo.messaging_notifications driver messagingv2
```

Настраиваем получение статистики по «**memory.usage**», «**disk.usage**» и «**disk.device.usage**»:

```
$ sudo crudini --set /etc/nova/nova.conf DEFAULT instance_usage_audit True
```

Запускаем службу **openstack-ceilometer-compute** и перезапускаем службу **openstack-nova-compute**:

```
$ sudo systemctl enable openstack-ceilometer-compute.service  
$ sudo systemctl start openstack-ceilometer-compute.service  
$ sudo systemctl restart openstack-nova-compute.service
```

12.4. Далее можно настраивать сервисы для отправки сообщений в сервис **Telemetry**. Я сделаю настройку сервисов **Glance** и **Cinder** (остальные сервисы [тут](#)):

Настраиваем файлы конфигов служб (**/etc/glance/glance-api.conf** и **/etc/glance/glance-registry.conf**) **Glance**:

Настраиваем службу отправки сообщений для **Ceilometer** по AMQP и URL принимающего брокера сообщений (**RabbitMQ**) в конфиге (**/etc/glance/glance-api.conf**) **glance-api**:

```
$ sudo crudini --set /etc/glance/glance-api.conf oslo_messaging_notifications  
driver messagingv2  
$ sudo crudini --set /etc/glance/glance-api.conf DEFAULT transport_url  
rabbit://openstack:openstack@controller.test.local
```

Настраиваем URL принимающего брокера сообщений (**RabbitMQ**) в конфиге (**/etc/glance/glance-registry.conf**) **glance-registry**:

```
$ sudo crudini --set /etc/glance/glance-registry.conf DEFAULT transport_url  
rabbit://openstack:openstack@controller.test.local
```

Перезапускаем настроенные службы **glance-api** и **glance-registry**:

```
$ sudo systemctl restart openstack-glance-api.service  
$ sudo systemctl restart openstack-glance-registry.service
```

Переходим к настройке конфига **/etc/cinder/cinder.conf** сервиса **Cinder** (если данный сервис находится на других узлах, то там его настройка аналогична):

Настраиваем службу отправки сообщений для **Ceilometer** по AMQP:

```
$ sudo crudini --set /etc/cinder/cinder.conf oslo_messaging_notifications  
driver messagingv2
```

Можно создать **cron-службу** (выполняет что-то периодически), которая будет раз в 5 минут собирать информацию о сервисе (в официальной документации такого пункта нет):

```
$ sudo */5 * * * * /path/to/cinder-volume-usage-audit --send_actions
```

Перезапускаем службы **Cinder**:

```
$ sudo systemctl restart openstack-cinder-api.service  
$ sudo systemctl restart openstack-cinder-scheduler.service  
$ sudo systemctl restart openstack-cinder-volume.service
```

Все, теперь должны пойти метрики.

12.5. Посмотреть метрики можно при помощи [следующих команд](#), а мы перейдем к созданию воображаемого триггера.

Перед созданием немного инфы о триггерах. Во-первых, триггер может быть в 3 состояниях: «Ok», «Тревога» и «Недостаточно данных». Во-вторых, правила триггеров могут объединяться с помощью **AND** и **OR**. В-третьих, к триггеру можно привязать два типа действий: отправление **POST-запроса** на какой-то URL или запись в файл журнала (только при отладке, в реальной жизни нет).

Теперь создаем триггер **myalarm1** для ВМ с id **35923211-ec9d-4f0a-8d5c-2cec96a427d8** (у вас может быть другой id), подробная информация по созданию [тут](#):

```
$ sudo openstack alarm create --type gnocchi_aggregation_by_resources_threshold  
--name myalarm1 --metric cpu_util --aggregation-method mean --comparison-  
operator 'ge' --threshold 60 --evaluation-periods 2 --granularity 300 --alarm-  
action 'log://' --resource-type instance --query '{"": {"id": "35923211-ec9d-  
4f0a-8d5c-2cec96a427d8"}}'
```

- **--type** — тип триггера (в нашем случае срабатывание по определенному порогу метрик)
- **--name** — имя триггера
- **--metric** — имя метрики
- **--aggregation-method** — метод объединения значений метрик (в нашем случае берется среднее значение за период)
- **--comparison-operator** — тип сравнения (в нашем случае **greater equal** — больше равно)
- **--threshold** — сравниваемое значение
- **--evaluation-periods** — количество периодов после возникновения события, которые нужно подождать перед совершением действия
- **--granularity** — длительность периода в секундах
- **--alarm-action** — действие, совершаемое триггером
- **--resource-type** — тип ресурса openstack
- **--query** — параметры передаваемые действию (в нашем случае в журнал запишется id ВМ)

С триггерами все переходим к последнему сервису (ну как так-то их еще минимум 5, но в гайде в ближайшее время их не будет).

13. Установка и настройка сервиса оркестрации Heat

13.1. Сервис **Heat** отвечает за автоматизацию управления жизненными циклами наборов облачных сервисов, объединяя их в так называемые стеки (**stack**). С помощью него можно как развернуть ВМ, так и стартовать комплексное приложение из многих машин и масштабировать его в зависимости от информации, передаваемой модулем телеметрии. Для описания стеков используются специальные описания ресурсов, их ограничений, зависимостей и параметров:

- **HTOT** (Heat Orchestration Template) – формат, предназначенный исключительно для OpenStack. Представляет собой документ формата YAML (с ним и работаем)
- **CFT** (AWS CloudFormation) – документ формата JSON в формате, совместимом с шаблонами сервиса [CloudFormation](#). Нужен, чтобы работать с уже существующими для AWS [шаблонами](#).

Службы **Heat**:

- **openstack-heat-engine** – основной сервис, обеспечивающий обработку шаблонов и отправляющий события пользователям API
- **openstack-heat-api** – сервис, отвечающий за предоставление основного REST API Heat. Взаимодействует с **openstack-heat-engine** через вызовы RPC
- **openstack-heat-api-cfn** – аналогичен **openstack-heat-api**, но обеспечивает работу с API, совместимым с **AWS CloudFormation**.
- **heat cli** – интерфейс взаимодействия с **Heat** API. Помимо командной строки, разработчики могут напрямую вызывать REST API или использовать веб-интерфейс **Horizon**.

13.2. Приступаем к воображаемой установке и настройке служб **Heat**:

Устанавливаем необходимые службы на узел управления (python2-heatclient модуль для взаимодействия **Python** с **Heat**):

```
$ sudo yum -y install openstack-heat-api openstack-heat-api-cfn openstack-heat-engine python2-heatclient openstack-heat-ui
```

Создаем БД **heat**:

```
$ sudo mysql -u root -p
CREATE DATABASE heat;
GRANT ALL PRIVILEGES ON heat.* TO 'heat'@'localhost' IDENTIFIED BY 'heat';
GRANT ALL PRIVILEGES ON heat.* TO 'heat'@'%' IDENTIFIED BY 'heat';
```

Регистрируем **Heat** в сервисе **Keystone**:

```
$ sudo openstack user create --domain default --password openstack heat
$ sudo openstack role add --project service --user heat admin
```

Уже что-то интересное, для **Heat** необходим отдельный домен **heat** для проектов и пользователей стека:

```
$ sudo openstack domain create --description "Stack projects and users" heat
```

В новом домене **heat** создаем админа **heat_domain_admin**:

```
$ sudo openstack user create --domain heat --password openstack heat_domain_admin
$ sudo openstack role add --domain heat --user-domain heat --user heat_domain_admin
admin
```

Создадим роли владельца стека **heat_stack_owner** и пользователя стека **heat_stack_user**. Также добавим роль владельца стека нашему пользователю **demo**:

```
$ sudo openstack role create heat_stack_owner
$ sudo openstack role add --project demo --user demo heat_stack_owner
$ sudo openstack role create heat_stack_user
```

Закончим регистрацию в **Keystone**, добавив два сервиса: **heat** (обычный **heat**) и **heat-cfn** (**heat** для **CloudFormation**). Ну и добавим точки входа в добавленные сервисы:

```
$ sudo openstack service create --name heat --description "Orchestration"
orchestration
$ sudo openstack service create --name heat-cfn --description "Orchestration"
cloudformation
$ sudo openstack endpoint create --region RegionOne orchestration public
http://controller.test.local:8004/v1/%\(\tenant_id\)\s
$ sudo openstack endpoint create --region RegionOne orchestration internal
http://controller.test.local:8004/v1/%\(\tenant_id\)\s
$ sudo openstack endpoint create --region RegionOne orchestration admin
http://controller.test.local:8004/v1/%\(\tenant_id\)\s
$ sudo openstack endpoint create --region RegionOne cloudformation public
http://controller.test.local:8000/v1
$ sudo openstack endpoint create --region RegionOne cloudformation internal
http://controller.test.local:8000/v1
$ sudo openstack endpoint create --region RegionOne cloudformation admin
http://controller.test.local:8000/v1
```

Приступаем к редактированию конфигурационного файла **/etc/heat/heat.conf**:

Прописываем путь к БД **heat**:

```
$ sudo crudini --set /etc/heat/heat.conf database connection
mysql+pymysql://heat:heat@controller.test.local/heat
```

Указываем URL брокера сообщений (**RabbitMQ**):

```
$ sudo crudini --set /etc/heat/heat.conf DEFAULT transport_url
rabbit://openstack:openstack@controller.test.local
```

Добавляем параметры авторизации в сервисе **Keystone**:

```
$ sudo crudini --set /etc/heat/heat.conf keystone_authtoken www_authenticate_uri http://controller.test.local:5000
$ sudo crudini --set /etc/heat/heat.conf keystone_authtoken auth_url http://controller.test.local:35357
$ sudo crudini --set /etc/heat/heat.conf keystone_authtoken auth_type password
$ sudo crudini --set /etc/heat/heat.conf keystone_authtoken project_domain_name default
$ sudo crudini --set /etc/heat/heat.conf keystone_authtoken user_domain_name default
$ sudo crudini --set /etc/heat/heat.conf keystone_authtoken project_name service
$ sudo crudini --set /etc/heat/heat.conf keystone_authtoken username heat
$ sudo crudini --set /etc/heat/heat.conf keystone_authtoken password openstack
$ sudo crudini --set /etc/heat/heat.conf trustee auth_type password
```

Добавляем реквизиты авторизации доверенного лица (**trustee**) в Keystone:

```
$ sudo crudini --set /etc/heat/heat.conf trustee auth_url http://controller.test.local:35357
$ sudo crudini --set /etc/heat/heat.conf trustee username heat
$ sudo crudini --set /etc/heat/heat.conf trustee password openstack
$ sudo crudini --set /etc/heat/heat.conf trustee user_domain_name default
```

Указываем, что в запросах клиента должен быть заголовок **authenticate_uri**:

```
$ sudo crudini --set /etc/heat/heat.conf clients_keystone www_authenticate_uri http://controller.test.local:5000
```

Задаем URL сервера метаданных и сервера доступности ресурсов (показывает, когда можно создать ресурс, создание которого зависит от существования других ресурсов):

```
$ sudo crudini --set /etc/heat/heat.conf DEFAULT heat_metadata_server_url http://controller.test.local:8000
$ sudo crudini --set /etc/heat/heat.conf DEFAULT heat_waitcondition_server_url http://controller.test.local:8000/v1/waitcondition
```

Указываем выделенный под запуск стеков домен и реквизиты администратора:

```
$ sudo crudini --set /etc/heat/heat.conf DEFAULT stack_domain_admin heat_domain_admin
$ sudo crudini --set /etc/heat/heat.conf DEFAULT stack_domain_admin_password openstack
$ sudo crudini --set /etc/heat/heat.conf DEFAULT stack_user_domain_name heat
```

Синхронизируемся с БД **heat**:

```
$ sudo -s /bin/sh -c "heat-manage db_sync" heat
```

Запускаем службы **Heat**:

```
$ sudo systemctl enable openstack-heat-api.service
$ sudo systemctl enable openstack-heat-api-cfn.service
$ sudo systemctl enable openstack-heat-engine.service
$ sudo systemctl start openstack-heat-api.service
$ sudo systemctl start openstack-heat-api-cfn.service
$ sudo systemctl start openstack-heat-engine.service
```

Команда проверки работы сервисов — [BOT](#).

13.3. Теперь можно создать первый стек, для этого на создать YAML-файл **test-server.yml** следующего содержания (создает стек, состоящий из одной виртуальной машины, которой во время старта передается скрипт, выводящий сообщение «Instance STARTED!» на стандартный вывод):

```
heat_template_version: 2014-10-16
description: >
    OpenStack. Практическое знакомство с облачной операционной системой.
    Пример запуска одной ВМ

parameters:
  network:
    type: string
    description: Сеть экземпляра ВМ
    default: demo-net
  image:
    type: string
    description: Образ для запуска ВМ
    default: cirros-0.3.3-x86_64

resources:
  my_server:
    type: OS::Nova::Server
    properties:
      flavor: m2.tiny
      key_name: demokey1
      networks:
        - network: { get_param: network }
      image: { get_param: image }
      user_data: |
        #!/bin/sh
        echo "Instance STARTED!"
      user_data_format: RAW

outputs:
  instance_name:
    description: Имя экземпляра ВМ
    value: { get_attr: [my_server, name] }
  private_ip:
    description: IP-адрес ВМ в частной сети
    value: { get_attr: [ my_server, first_address ] }
```

- **heat_template_version** — версия шаблона **Heat** (соответствует дате релиза актуальной версии Openstack, в файле версия устарела)
- **description** — описание шаблона
- **parameters** — определение переменных окружения (параметров), для дальнейшего использования
 - **network** — параметр сети
 - **type** — тип параметра (в данном случае строка)

- **description** — описание параметра
- **default** — дефолтное значение параметра, если через флаги команды создания стека не передано другое значение
- **image** — параметр образа
- **resources** — описание ресурсов шаблона
 - **my_server** — имя ресурса
 - **type** - тип ресурса (в данном случае ВМ (**server**))
 - **properties** — определение параметров ВМ my_server
 - **flavor** — тип тома ВМ
 - **key_name** — пара SSH ключей ВМ
 - **networks** — массив сетей подключаемых к ВМ
 - **network** — сеть подключенная к ВМ
 - **get_param** — получение переменных окружения (параметров)
 - **image** — базовый образ ВМ
 - **user_data** — действия выполняемые при запуске ВМ (в данном случае запуск скрипта)
 - **user_data_format** — формат, в котором были записаны действия (в данном случае в виде строки)
 - **outputs** — параметры выводимые пользователю в **Horizon** или через API
 - **instance_name** — название ВМ
 - **description** — описание параметра
 - **value** — значение параметра
 - **get_attr** — получение значения атрибута ресурса (задается имя ресурса (**my_server**) и конкретный атрибут (**name**))
 - **private_ip** — плавающий IP ВМ

Теперь можно создать стек используя созданный YAML-файл:

```
$ sudo openstack stack create --parameter network=demo-net --parameter image=c8ccc9b3-29bb-4220-be38-8f261ac8b99a -t test-server.yml teststack
```

- **--parameter** — передать значение параметра
- **-t** — использовать файл шаблона

[Данная команда](#) показывает статус создания стека.

Также можно посмотреть создалась ли ВМ, а также информацию о ней:

```
$ sudo openstack server list  
$ sudo openstack stack show teststack
```

Проверка срабатывания скрипта:

```
$ sudo openstack console log show teststack-my_server-g6m7qyi7jsn4
```

После чего можно посмотреть список ресурсов стека и потом его удалить:

```
$ sudo openstack stack resource list teststack  
$ sudo openstack stack delete teststack
```

14. Что происходит под капотом Neutron (бонусная глава)

14.1. Давайте выясним как у нас в кластере на самом деле создаются сети и почему на рабочих узлах достаточно одного сетевого интерфейса. Для начала разберемся, что из себя представляет **Open vSwitch** (виртуальный коммутатор), а также **сетевые пространства имен**:

14.2. **Сетевые пространства имен** являются одной из разновидностей **пространств имен Linux** (придуманы, чтобы изолировать друг от друга процессы), которые включают в себя:

- **PID**, Process ID – изоляция иерархии процессов
- **NET**, Networking – изоляция сетевых интерфейсов
- **PC**, InterProcess Communication – управление взаимодействием между процессами
- **MNT**, Mount – управление точками монтирования
- **UTS**, Unix Timesharing System – изоляция ядра и идентификаторов версии

Так вводим три следующие команды (список **сетей** и **маршрутизаторов openstack**, а также список **сетевых пространств Linux**) и анализируем вывод;

```
$ sudo openstack network list  
+-----+  
| ID      | Name    | Subnets |  
+-----+  
| 574ac2c6-1d90-4c94-87ea-5335464ac5b0 | ext-net | 30e9c86f-3770-4917-85bd-a4bc91521f31 |  
| a3ba8800-8338-4558-aeca-3247df932f39 | demo-net | 59d8cece-795b-4618-8a10-e4d37e05d5ac |  
+-----+  
  
$ sudo openstack router list  
+-----+  
| ID      | Name    | Status | State | Project          | Distributed | HA |  
+-----+  
| e1a0f62f-46c3-472a-9b6d-5aa0c5611d5e | demo-router | ACTIVE | UP   | e420dacf94514ed587d878bf32cabf46 | False     | False |  
+-----+  
  
$ sudo ip netns  
qrouter-e1a0f62f-46c3-472a-9b6d-5aa0c5611d5e (id: 1)  
qdhcp-a3ba8800-8338-4558-aeca-3247df932f39 (id: 0)
```

Что же мы видим, а именно, что служба **neutron-l3-agent** создает сетевые пространства для каждой сети (**qdhcp-id**) и маршрутизатора (**qrouter-id**) **Openstack** (id совпадают), но только для тех, к которым подключены ВМ напрямую (сетевого

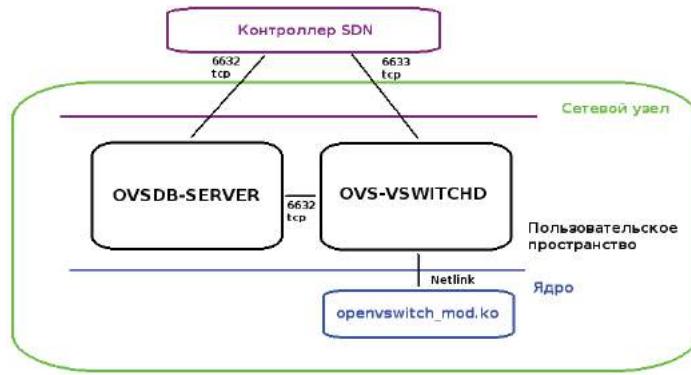
пространства **ext-net** нет). Также стоит отметить, что сетевые пространства автоматически не удаляются.

С сетевыми пространствами все.

14.3. Итак, **Open vSwitch**, данная технология состоит из следующих компонентов:

- **Openswitch_mod.ko** — модуль ядра, отвечающий за работу с пакетами
- **Ovs-vswitchd** — демон, отвечающий за управление, программирование логики пересылки пакетов, VLAN'ы и объединение сетевых карт
- **Ovsdb-server** — сервер базы данных, отвечающий за ведение базы данных с конфигурацией
- **Контроллер SDN (Software Defined Network)** — API сервер для работы с коммутатором
- OpenFlow — протокол, при помощи которого **контроллер SDN** может удаленно контролировать таблицы потоков на коммутаторах и маршрутизаторах

В совокупности все это выглядит так:



Основные команды для работы с Open vSwitch:

- **ovs-vsctl show** – вывод общей информации по коммутатору
- **ovs-vsctl add-br/del-br** – добавить или удалить мост
- **ovs-vsctl add-port/del-port** – добавить или удалить порт
- **ovs-ofctl dump-flows** – вывод запрограммированных потоков для конкретного коммутатора (за их определение отвечает служба **neutron-openvswitch-agent**)
- **ovsdb-tool show-log** – все команды настройки, отданные OVS, при помощи утилит пространства пользователя (с ее помощью **Neutron** работает с Open vSwitch)

Теперь сравним вывод списка портов **openstack** с выводом конфигурации **Open vSwitch** (на **controller** и **compute** узлах). Вывод **Open vSwitch** я немножко подрезал:

ID	Name	MAC Address	Fixed IP Addresses	Status
106645c0-cdb8-42ee-9dcf-e995d5e87bbd		fa:16:3e:2a:de:31	ip_address='10.100.1.121', subnet_id='30e9c86f-3770-4917-85bd-a4bc91521f31'	ACTIVE
3ac48b23-6b91-46d9-9097-e41351ab5a5f		fa:16:3e:f0:e3:1e	ip_address='10.100.1.172', subnet_id='30e9c86f-3770-4917-85bd-a4bc91521f31'	N/A
87118d9b-845b-449f-b166-9120b2d86b12		fa:16:3e:95:63:92	ip_address='172.16.0.72', subnet_id='59d8cece-795b-4618-8a10-e4d37e05d5ac'	ACTIVE
9c0640c5-92a8-4e89-830f-c7e3c5fc0e10		fa:16:3e:15:99:42	ip_address='172.16.0.25', subnet_id='59d8cece-795b-4618-8a10-e4d37e05d5ac'	ACTIVE
a3b98aca-1332-4156-83d9-62ee9c962c33		fa:16:3e:9c:0c:c7	ip_address='172.16.0.1', subnet_id='59d8cece-795b-4618-8a10-e4d37e05d5ac'	ACTIVE
ef1fc61d-5640-4922-b23b-de459f40fef		fa:16:3e:52:4b:53	ip_address='172.16.0.2', subnet_id='59d8cece-795b-4618-8a10-e4d37e05d5ac'	ACTIVE

```
$ sudo ovs-vsctl show
489d40ff-df55-4b53-ba2b-9e3ec849f55d
    Manager "ptcp:6640:127.0.0.1"
        is_connected: true
    Bridge br-tun
        Controller "tcp:127.0.0.1:6633"
        ...
        Port patch-int
            Interface patch-int
                type: patch
                options: {peer=patch-tun}
        Port "gre-c0a87ad2"
            Interface "gre-c0a87ad2"
                type: gre
                options: {..., remote_ip="192.168.122.210"}
        Port br-tun
            Interface br-tun
                type: internal
        Port "gre-c0a87ad7"
            Interface "gre-c0a87ad7"
                type: gre
                options: {..., remote_ip="192.168.122.215"}
    Bridge br-ex
        Controller "tcp:127.0.0.1:6633"
        ...
        Port phy-br-ex
            Interface phy-br-ex
                type: patch
                options: {peer=int-br-ex}
        Port "eth1"
            Interface "eth1"
        Port br-ex
            Interface br-ex
                type: internal
    Bridge br-int
        Controller "tcp:127.0.0.1:6633"
        ...
        Port int-br-ex
            Interface int-br-ex
                type: patch
                options: {peer=phy-br-ex}
        Port "qr-a3b98aca-13"
            tag: 1
            Interface "qr-a3b98aca-13"
                type: internal
        Port "qg-106645c0-cd"
            tag: 2
            Interface "qg-106645c0-cd"
                type: internal
        Port br-int
            Interface br-int
                type: internal
    Port patch-tun
        Interface patch-tun
            type: patch
            options: {peer=patch-int}
    Port "tape1fc61d-56"
        tag: 1
        Interface "tape1fc61d-56"
            type: internal
```

```

$ sudo ovs-vsctl show
ccb3e4c4-c446-453d-b793-77f35dd1a27
    Manager "ptcp:6640:127.0.0.1"
        is_connected: true
    Bridge br-int
        Controller "tcp:127.0.0.1:6633"
        ...
        Port "qvo87118d9b-84"
            tag: 1
            Interface "qvo87118d9b-84"
            Port "qvo9c0640c5-92"
                tag: 1
                Interface "qvo9c0640c5-92"
            Port patch-tun
                Interface patch-tun
                    type: patch
                    options: {peer=patch-int}
            Port br-int
                Interface br-int
                    type: internal
    Bridge br-tun
        Controller "tcp:127.0.0.1:6633"
        ...
        Port "gre-c0a87ad7"
            Interface "gre-c0a87ad7"
                type: gre
                options: {..., remote_ip="192.168.122.215"}
        Port "gre-c0a87ac8"
            Interface "gre-c0a87ac8"
                type: gre
                options: {..., remote_ip="192.168.122.200"}
        Port br-tun
            Interface br-tun
                type: internal
        Port patch-int
            Interface patch-int
                type: patch
                options: {peer=patch-tun}

```

Такаак, в первую очередь можно заметить, что на узле управления у нас находятся три моста (**br-int**, **br-tun** и **br-ex**), а на рабочем узле, где запущена ВМ два моста (**br-int** и **br-tun**). Что из себя представляют данные мосты:

- **Br-int** – интеграционный мост, предназначенный для подключения ВМ. Он осуществляет VLAN-тегирование трафика приходящего с/на вычислительные узлы. Находится как на вычислительных так и сетевых (у меня управляющем) узлах и создается автоматически при первом старте **neutron-openvswitch-agent**. На управляющем узле к нему подключены шесть портов:
 - **tapef1fc61d-56** — порт моста к ВМ **myinstance1** (ip моста — **172.16.0.2**), для **testvm** нет моста, так как она в данный момент выключена
 - **patch-tun** – соединение с мостом **tun**
 - **qr-a3b98aca-13** – подключение к единственному маршрутизатору (ip маршрутизатора в подсети — **172.16.0.1**)
 - **qg-106645c0-cd** — подключение к внешнему шлюзу маршрутизатора (ip внешнего шлюза в настройках маршрутизатора — **10.100.0.121**)
 - **int-br-ex** – подключение к мосту **br-ex**
 - **br-init** — собственный порт моста **init**

- На рабочем узле есть четыре порта:
 - **qvo87118d9b-84** — интерфейс ВМ **myinstance1** (локальный ip ВМ — **172.16.0.25**)
 - **qvo9c0640c5-92** — интерфейс ВМ **testvm** (локальный ip ВМ — **172.16.0.72**)
 - **patch-tun** — аналогично управляющему узлу соединение с мостом **tun**
 - **br-int** — аналогично управляющему узлу порт моста **init**
- **br-tun** — в нашем случае это GRE-туннель. Связывает сетевые и вычислительные узлы, передавая тегированный трафик с интеграционного моста, используя правила OpenFlow. Имеет следующие порты (на обоих узлах):
 - **gre-c0a87ad7** — gre-туннель до узла **compute-opt**
 - **gre-c0a87ac8** — gre-туннель до узла **controller**
 - **gre-c0a87ad2** — gre-туннель до узла **compute**
 - **br-tun** — собственный порт моста **tun**
 - **patch-int** — подключение к мосту **br-int**
- **Br-ex** — мост, осуществляющий взаимодействие с внешним миром. Существует только на сетевых (управляющих) узлах. Содержит порты:
 - **eth1** — физический сетевой интерфейс
 - **phy-br-ex** — порт связи моста и физического сетевого интерфейса
 - **br-ex** — собственный порт моста **ex**

Вот так можно посмотреть сетевые пространства имен нашего виртуального маршрутизатора:

```
$ sudo ip netns exec qrouter-e1a0f62f-46c3-472a-9b6d-5aa0c5611d5e ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
  link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
      valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
      valid_lft forever preferred_lft forever
2: gre0@NONE: <NOARP> mtu 1476 qdisc noop state DOWN group default qlen 1000
  link/gre 0.0.0.0 brd 0.0.0.0
3: gretap0@NONE: <BROADCAST,MULTICAST> mtu 1462 qdisc noop state DOWN group default qlen 1000
  link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
14: qr-a3b98aca-13: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1458 qdisc noqueue state UNKNOWN group default qlen 1000
  link/ether fa:16:3e:9c:0c:c7 brd ff:ff:ff:ff:ff:ff
    inet 172.16.0.1/24 brd 172.16.0.255 scope global qr-a3b98aca-13
      valid_lft forever preferred_lft forever
    inet6 fe80::f816:3eff:fe9c:cc7/64 scope link
      valid_lft forever preferred_lft forever
15: qg-106645c0-cd: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN group default qlen 1000
  link/ether fa:16:3e:2a:de:31 brd ff:ff:ff:ff:ff:ff
    inet 10.100.1.121/24 brd 10.100.1.255 scope global qg-106645c0-cd
      valid_lft forever preferred_lft forever
    inet 10.100.1.172/32 brd 10.100.1.172 scope global qg-106645c0-cd
      valid_lft forever preferred_lft forever
    inet6 fe80::f816:3eff:fe2a:de31/64 scope link
      valid_lft forever preferred_lft forever
```

14.4. Из-за того, что **Open vSwitch** не может работать с правилами **iptables**, которые применяются на виртуальный интерфейс, непосредственно подключенный к порту

коммутатора, группы безопасности (набор настраиваемых разрешающих правил прохождения трафика, которые возможно назначать на порты) применяются к мосту **qbr-id** с помощью [LinuxBridge](#) (что-то типа костыля), выведем все мосты рабочего узла:

```
$ sudo brctl show
bridge name      bridge id      STP enabled      interfaces
qbr87118d9b-84  8000.6a47ffa4ab15  no            qvb87118d9b-84
qbr9c0640c5-92  8000.be7d9ed994a1  no            qvb9c0640c5-92
virbr0          8000.525400cb82c3  yes           tap9c0640c5-92
                                         virbr0-nic
```

Тут видно три моста:

- **qbr87118d9b-84** — мост к ВМ **testvm**
- **qbr9c0640c5-92** — мост к ВМ **myinstance1**
- **virbr0** — мост к хост машине

Посмотрим правила брандмауэра (на самом деле команды, которые автоматически ввела служба **neutron-openvswitch** при создании ВМ) для интерфейса **tap9c0640c5-92** (сетевой интерфейс ВМ **myinstance1**) с помощью [iptables](#) (-S — все правила):

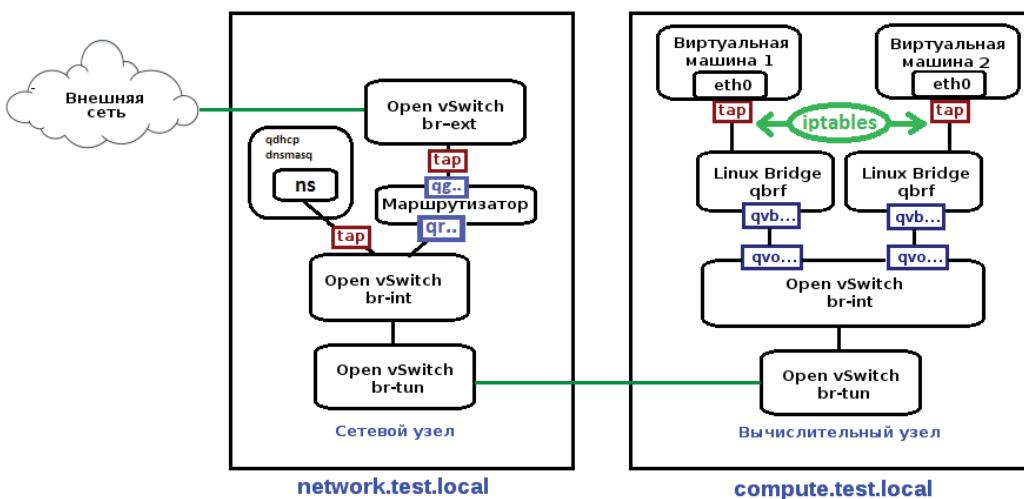
```
$ sudo iptables -S | grep tap9c0640c5-92
-A neutron-openvswi-FORWARD -m physdev --physdev-out tap9c0640c5-92 --physdev-is-bridged -m comment --comment "Direct traffic from the VM interface to the security group chain." -j neutron-openvswi-sg-chain
-A neutron-openvswi-FORWARD -m physdev --physdev-in tap9c0640c5-92 --physdev-is-bridged -m comment --comment "Direct traffic from the VM interface to the security group chain." -j neutron-openvswi-sg-chain
-A neutron-openvswi-INPUT -m physdev --physdev-in tap9c0640c5-92 --physdev-is-bridged -m comment --comment "Direct incoming traffic from VM to the security group chain." -j neutron-openvswi-o9c0640c5-9
-A neutron-openvswi-sg-chain -m physdev --physdev-out tap9c0640c5-92 --physdev-is-bridged -m comment --comment "Jump to the VM specific chain." -j neutron-openvswi-i9c0640c5-9
-A neutron-openvswi-sg-chain -m physdev --physdev-in tap9c0640c5-92 --physdev-is-bridged -m comment --comment "Jump to the VM specific chain." -j neutron-openvswi-o9c0640c5-9
```

- **-A** — добавить правило в цепочку с именем (например **neutron-openvswi-FORWARD**)
- **-m** — имя дополнительного используемого модуля (**physdev** — модуль устройств ввода и вывода порта моста, **comment** — модуль добавления комментариев)
- **--physdev-out** — имя порта моста, через который будет отправлен пакет
- **--physdev-in** — имя порта моста, через который принимается пакет
- **--physdev-is-bridged** — пакет передается по мосту и не маршрутизируется
- **--comment** — содержимое комментария к правилу
- **-j** - имя цели (цепочки) правила (что делать с пакетом, в нашем случае прейти к цепочке с таким-то именем). Вот вывод инфы о цепочке входящего в ВМ трафика:

```
$ sudo iptables -L neutron-openvswi-i9c0640c5-9
Chain neutron-openvswi-i9c0640c5-9 (1 references)
target    prot opt source          destination
RETURN   all -- anywhere        anywhere
associated with a known session to the RETURN chain. */
RETURN   udp  -- anywhere       172.16.0.25
RETURN   udp  -- anywhere       255.255.255.255
RETURN   tcp  -- anywhere        anywhere
RETURN   icmp -- anywhere       anywhere
DROP     all  -- anywhere        anywhere
state RELATED,ESTABLISHED /* Direct packets
state INVALID /* Drop packets that appear
related to an existing connection (e.g. TCP ACK/FIN) but do not have an entry in conntrack. */
neutron-openvswi-sg-fallback all  -- anywhere      anywhere
                                         /* Send unmatched traffic
to the fallback chain. */
```

Тут видно нашу настройку приема **icmp** пакетов, **ssh** подключений, а также дефолтный прием всех **udp** пакетов на ip BM, ну и правила работы с разными тегами трафика (**RELATED** (пакет начинает новое соединение (например **FTP**), но связан с существующим соединением. **принимаем**), **ESTABLISHED** (пакет связан с соединением, которое видело пакеты в обоих направлениях. **принимаем**), **INVALID** (пакет связан с неизвестным соединением. **отбрасываем**) и **все остальное** (отправляет в цепочку **neutron-openvswi-sg-fallback**)).

Вот так должна выглядеть наша сеть (**qvb** - подключение в сторону моста **qbr**, **qvo** - подключение в сторону **OVS**):



Ну вроде все единственное, что можно сказать это что у **openstack** есть служба **openstack-neutron-fwaas** (файерволл как сервис), которая помогает снизить нагрузку на сетевые интерфейсы BM, оправляя им уже проверенный трафик.

14.4. Для того чтобы красиво визуализировать сеть узла можно скачать специальную утилиту сканирования сети **plotnetcfg**, которая создаст конфиг сети, после чего визуализировать этот конфиг в виде схемы при помощи утилиты **dot**, входящей в пакет **graphviz** (можно скачать на каждый узел, чтобы составить их схемы, но я скачиваю только на узел **compute**).

Перед установкой пакетов надо опять **включить** репозиторий **epel**, который мы вырубали. После чего надо скачать пакеты **plotnetcfg** и **graphviz**:

```
$ sudo yum -y install plotnetcfg graphviz
```

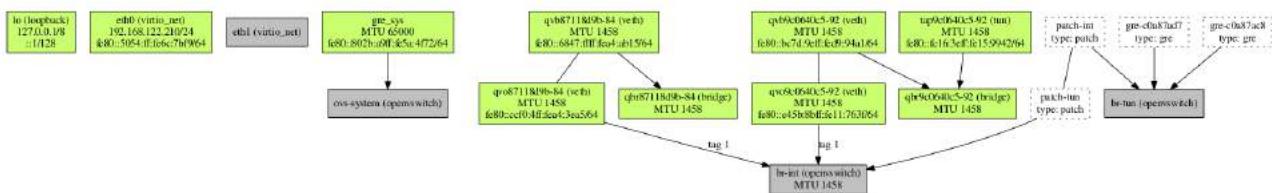
Теперь можно визуализировать сетевую конфигурацию узла **compute** (-T — тип файла, в который сохранится схема):

```
$ sudo plotnetcfg | dot -Tpdf > config.pdf
```

Скачиваем на хост машину файл и смотрим че вышло:

```
$ sudo scp root@192.168.122.210:~/config.pdf config.pdf
```

Скрин части схемы (**eth1** мне было просто лень убрать, так-то его не должно быть на вычислительном узле):



14.5. Еще интересная тема так называемый мониторинг трафика ВМ. Попробуем проанализировать трафик нашей ВМ **myinstance1**:

Запускаем из ВМ поток пакетов ICMP на ip виртуального маршрутизатора (и не останавливаем):

```
$ sudo ping 172.16.0.1
PING 172.16.0.1 (172.16.0.1): 56 data bytes
 64 bytes from 172.16.0.1: seq=0 ttl=64 time=3.395 ms
 64 bytes from 172.16.0.1: seq=1 ttl=64 time=1.322 ms
 64 bytes from 172.16.0.1: seq=2 ttl=64 time=2.715 ms
...
...
```

Теперь заходим на узел, на котором она поднялась, качаем [tcpdump](#) (-n — начать сразу же, -p — выводить все в консоль, -i — имя интерфейса -vvv — очень детальный вывод, -s — размер снимка пакета, -w — записать в файл) и пытаемся перехватить трафик (интерфейсы **patch-tun** и **br-int**) ВМ (должны вылезти ошибки):

```
$ sudo yum -y install tcpdump
tcpdump: -npi patch-tun -vvvs0 -w /tmp/dump.cap
tcpdump: patch-tun: No such device exists
(SIOCGIFHWADDR: No such device)
$ sudo tcpdump -npi br-int -vvvs0 -w /tmp/dump.cap
tcpdump: br-int: That device is not up
```

Почему же вылезают ошибки, а потому что внутренние устройства **Open vSwitch** невидимы для большинства утилит за пределами OVS, так как **Open vSwitch** не может работать с правилами **iptables**, которые применяются на виртуальный интерфейс,

непосредственно подключенный к порту коммутатора (из-за этого все больше любят **Linux Bridge**, с которым таких проблем нет). Поэтому люди придумали костыль, а именно **dummy-интерфейс** (локальный интерфейс, тоже самое что и **loopback**), который будет зеркаливать весь трафик моста **br-int**.

Создаем интерфейс **br-int-tcpdump** и поднимаем его:

```
$ sudo ip link add name br-int-tcpdump type dummy  
$ sudo ip link set dev br-int-tcpdump up
```

Добавляем созданный интерфейс **br-int-tcpdump** к мосту **br-int**:

```
$ sudo ovs-vsctl add-port br-int br-int-tcpdump
```

Он должен появится в конфиге OVS в секции моста **br-int** (управляющего узла):

```
$ sudo ovs-vsctl show  
ccb3e4c4-c446-453d-b793-77f35ddc1a27  
    Manager "ptcp:6640:127.0.0.1"  
        is_connected: true  
    Bridge br-int  
        Controller "tcp:127.0.0.1:6633"  
            is_connected: true  
            fail_mode: secure  
            datapath_type: system  
        Port "qvo87118d9b-84"  
            tag: 1  
            Interface "qvo87118d9b-84"  
        Port "qvo9c0640c5-92"  
            tag: 1  
            Interface "qvo9c0640c5-92"  
        Port br-int-tcpdump  
            Interface br-int-tcpdump  
        Port patch-tun  
            Interface patch-tun  
                type: patch  
                options: {peer=patch-int}  
        Port br-int  
            Interface br-int  
                type: internal  
    ...
```

Теперь настраиваем зеркалирование всего трафика с моста **br-int** на интерфейс **br-int-tcpdump** (эта сложная команда включает в себя 4 подкоманды, которые передают друг другу значения):

```
$ sudo ovs-vsctl -- set Bridge br-int mirrors=@m \  
--id=@br-int-tcpdump get Port br-int-tcpdump \  
--id=@br-int get Port br-int \  
--id=@m create Mirror name=mirrortest select-dst-port=@br-int select-src-port=@br-int  
output-port=@br-int-tcpdump select_all=1
```

- **mirrors** — список зеркаливаний интерфейса

- **--id=@имя_переменной** — локальная переменная, в которую запишется вывод подкоманды
- **name** — имя зеркалирования
- **select-dst-port** — интерфейс, на который приходят пакеты
- **select-src-port** — интерфейс, с которого приходят пакеты
- **output-port** — интерфейс, на который пакеты будут зеркалироваться
- **select_all** — все пакеты, если 1

Теперь трафик должен начать перехватываться (его можно записать в файл, скинуть на хост машину и проанализировать, например, в [Wireshark](#)):

```
$ sudo tcpdump -npi br-int-tcpdump -vvvs0 -w /tmp/dump.cap
tcpdump: listening on br-int-tcpdump, link-type EN10MB (Ethernet),
capture size 262144 bytes
Got 32
40 packets captured
40 packets received by filter
0 packets dropped by kernel
```

14.6. Ну что ж последний пункт затрагивает такое понятие как балансировщик нагрузки как сервис (**LbaaS**). Вроде у **Openstack** уже есть свой сервис балансировки нагрузки по имени **Octavia**, но я последнюю книге и разверну **HAProxy** (подключаемый модуль **Neutron**). Балансировщик нагрузки отвечает за балансировку входящих сетевых подключений между экземплярами виртуальных машин, входящих в один кластер, чтобы не было такого, что одна ВМ задыхается от запросов, пока вторая не знает чем ей заняться. Далее в книге была показана конфигурация HAProxy, в ходе которой я выяснил, что его уже никто почти не использует, так как есть **Octavia**, так что пункт отменяется до момента настройки мной данного сервиса.

Пока можно сказать три вещи:

- Есть три типа политики балансировки нагрузки:
 - подключения принимаются по очереди всеми виртуальными машинами
 - с одного IP-адреса подключения принимает одна и та же виртуальная машина
 - подключение принимает машина с наименьшим числом подключений
- Lbaas создает сетевые пространства имен для ВМ в так называемом пуле балансировки
- Порядок организации балансировки нагрузки для нескольких ВМ:
 - Создается специальный пул, в котором все ВМ будут делить запросы
 - В пул добавляются ВМ по их ip
 - Создается сервис мониторинга доступности ВМ

- Создается виртуальный ip пула
- Виртуальному ip пула добавляется плавающий ip, чтобы к кластеру можно было подключаться извне
- Таким образом запросы приходят на плавающий ip и распределяются согласно политике между участниками пула

Поздравляю господа, на этом минимальную настройку можно объявить законченной.

