TIONIX

Установка и первичная настройка облачной платформы

Tionix Cloud Platform

05/13/2022

Содержание

Требования к облачной платформе Tionix

Требования к построению кластера

Введение

Платформа TCP в продуктивной среде должна работать с адекватным уровнем отказоустойчивости – параметре, который определяет корректность работы ПО при отказе части компонентов и при их восстановлении. Для повышения уровня отказоустойчивости используются различные средства кластеризации – объединения экземпляров сервисов, запущенных на разных узлах в один метасервис с единой точкой входа. Этот раздел кратко объяснит требования к кластеризируемым сервисам и к инфраструктуре.

Управляющие узлы

Различные сервисы платформы используют свои механизмы кластеризации.

Pacemaker

Pacemaker является основным средством запуска сервисов в режиме отказоустойчивости. В качестве средства синхронизации состояния сервисов используется Corosync[1].

Для корректной работы Pacemaker требуется:

- Минимальное количество экземпляров в кластере: 3, рекомендуется нечетное количество.
- Необходима настройка протокола STONITH[2] с использованием протокола IPMI (при технической возможности) и статуса сервисов.
- Управление статусом всех сервисов платформы должно производиться через механизмы Pacemaker.
- Рекомендуется дублировать данные Corosync через сеть репликации.

MariaDB

MariaDB содержит встроенную систему кластеризации в режиме Active/Active, которая называется Galera. Galera использует протокол wprep и rsync для репликации данных между узлами. В Pacemaker сервисы MariaDB добавляются в режиме Active/Active.

Для корректной работы Galera требуется:

- MariaDB Galera является сервисом с хранением данных и алгоритмом репликации на основе Raft, поэтому при построении кластера нужно учесть правила кворума.
- Минимальное количество экземпляров в кластере: 3, рекомендуется нечетное количество.
- Для хранения данных БД требуется быстрый локальный носитель на базе твердотельного диска.
- Рекомендуется дублировать данные через сеть репликации.

RabbitMQ

RabbitMQ имеет встроенные средства кластеризации с репликацией данных очередей сообщений. В Pacemaker сервисы MariaDB добавляются в режиме Active/Backup.

Для корректной работы кластера RabbitMQ требуется:

- Минимальное количество экземпляров в кластере: 3, рекомендуется нечётное количество.
- Для полноценной репликации необходимо включить функцию персистентных очередей сообщений (durable queue), которые должны сохраняться на дисках узлов управления и реплицироваться между собой.
- Для хранения данных БД желательно использовать быстрый локальный носитель на базе твердотельного диска.

1 http://corosync.github.io/corosync/
2 https://en.wikipedia.org/wiki/STONITH

- Рекомендуется дублировать данные через сеть репликации.

memcached

memcached не имеет встроенных средств кластеризации, добавляется в Pacemaker в режиме Active/Backup без синхронизации данных кэша между инстансами memcached. Принято, что данные кэша не являются важными и их можно терять.

Сервисы OpenStack

- Большинство сервисов OpenStack не хранят свое состояние (речь про состояние самого сервиса, а не про данные облачной платформы, хранимые в БД). Поэтому они должны запускаться в Pacemaker в режиме Active/Active.
- Минимальное количество экземпляров сервисов: 2.
- Конфигурация сервисов OpenStack между узлами кластера должна быть эквивалентной.
- В точках доступа сервисов OpenStack обязательно нужно использовать DNS-имя контроллера облака с резолвингом на виртуальный IP-адрес или динамическое изменение адреса DNS управляющего узла.
- Сервисы cinder-volume и nova-novncproxy должны запускаться в единственном экземпляре, поэтому они должны быть добавлены в Pacemaker в режиме Active/Backup из-за особенностей работы в кластерном окружении.

Модули TIONIX

- Большинство сервисов TIONIX не хранят свое состояние (речь про состояние самого сервиса, а не про данные облачной платформы, хранимые в БД). Поэтому они должны запускаться в Pacemaker в режиме Active/Active.
- Минимальное количество экземпляров сервисов: 2
- Конфигурация сервисов Tionix между узлами кластера должна быть эквивалентной.
- Сервис tionix-node-control-node-sync должен запускаться в единственном экземпляре, поэтому он должен быть добавлен в Pacemaker в режиме Active/Backup из-за особенностей работы в кластерном окружении.

Вычислительные узлы

Сервисы вычислительных узлов должны работать вне кластера управления. Сервис nova-compute должен быть настроен на единый виртуальный адрес кластера.

Требования к вычислительным ресурсам

Серверный комплекс платформы состоит из двух видов узлов в соответствии с их функциональным назначения:

- Управляющие узлы или контроллеры (далее – УУ). Используются для обеспечения вычислительными ресурсами СУ ОП ПВ КРТ;
- Вычислительные узлы (далее – ВУ). Используются для предоставления виртуализованных вычислительных ресурсов, в виде экземпляров виртуальных машин, прикладным информационным системам.

Минимальная, рекомендованная конфигурации узлов, определяющая выделение ресурсов для кластера управления, представлены в данной таблице:

Минимальная конфигурация	**Рекомендуемая базовая**
3 физических узла	
2 x CPU sockets (6 CPU cores (x86_64))	2 x CPU sockets (10 CPU cores (x86_64))
64 GB RAM	128 GB RAM
2 x 300GB SSD (DWPD >= 3) RAID1	2 x 300GB SSD (DWPD >= 3) RAID1
2 x 1GbE, 2 x 10GbE ports	2 x 10 GbE, 2 x 10 GbE

Конфигурация ВУ определяется из требований к общему количеству необходимых вычислительных ресурсов.

Требования к сети

Сегментация сети

В архитектуре облачной платформы необходимо использовать несколько физических сегментов сети, которые должны обрабатывать трафик с различным функциональным назначением. Принят следующий список сетей:

- **Сеть управления** (management network). Предназначен для трафика с содержанием команд управления облака и контроля над вычислительными ресурсами.
  - Минимальная скорость интерфейсов сети управления: 1Gbit/s.
  - Интерфейсы сети управления должны иметься на всех узлах облачной платформы.
  - Jumboframe не обязателен.
  - Можно использовать для цели репликации данных кластера управляющих компонентов.
- **Сеть вычислений** (compute network). Используется для обмена трафиком между виртуальными сетями облачной платформы (иными словами, между виртуальными машинами).
  - Минимальная скорость интерфейсов сети вычислений: 10Gbis/s.
  - Интерфейсы сети вычислений должны иметься на всех вычислительных узлах.
  - Необходимо включение Jumbo-кадров, равным 9000 байтов.
- **Сеть хранения** (storage network). Используется для получения доступа к данным ВМ, которые хранятся во внешних системах хранения.
  - Может использовать как Ethernet, так и FibreChannel в качестве протоколов канального уровня.
  - Минимальная скорость интерфейсов сети хранения: 8Gbit/s (FC), 10Gbit/s (Ethernet).
  - Интерфейсы сети хранения должны иметься на всех вычислительных узлах и в узлах хранения, если они являются обычными узлами на ОС на базе ядра Linux.
  - Управление внешней системой хранения должно осуществляться через сеть управления.
  - Необходимо включение Jumbo-кадров, равным 9000 байтов.
- **Сеть VDI** (VDI network). При наличии VDI-функций используется для доступа до VDI-сессий (пока только SPICE).
  - Минимальная скорость интерфейсов сети хранения: 10Gbit/s.
  - Интерфейсы сети VDI должны иметься на вычислительных узлах, которые добавлены в проекты VDI.
  - В этой сети можно включить Jumbo-кадры, равные 9000 байтов, однако необходимо иметь в виду, что включение Jumbo негативно влияет на задержки в работе VDI-протоколов.

Специализированная сеть передачи данных FibreChannel является рекомендуемой, в связи с отсутствием ethernet задержек и специализированности данных сетей для передачи данных.

Тем не менее вполне допустимо использование Ethernet сетей для взаимодействия с хранилищами, при обеспечении достаточной отказоустойчивости и производительности.

Требования к дисковому пространству

В облачной платформе могут быть использованы два вида хранения:

- Блочное устройство Cinder.
- Эфемерные диски - виртуальные диски, расположенные в файловой системе ВУ.

Требования к блочным устройствам Cinder

Использование блочных устройств Cinder является рекомендуемым вариантом для хранения пользовательских данных. Требования к СХД, подключаемых к сервисам Cinder, могут отличаться в зависимости от используемого драйвера.

Общие требования к СХД с Ethernet и FC

- Количество экземпляров СХД должно быть не менее двух. Потеря одного экземпляра СХД не должна приводить к потере доступа к данным.
- СХД должны использовать сеть хранения для доступа к пользовательским данным.
- СХД должны быть доступны по множественным путям сети с использованием протокола Multipath.
- При возможности следует использовать TIONIX Driver (см. стр. 170), в противном случае необходимо использовать драйвер, предоставленный вендором СХД.
- Необходимо использовать только образы формата RAW для запуска виртуальных машин. При использовании остальных форматов следует предоставить УУ дисковое пространство для конвертации образа в формат RAW.

Общие требования к эфемерным дискам

- Эфемерные диски доступны только при наличии дисков в ВУ. При сетевой загрузке ВУ эфемерные диски недоступны.
- Эфемерные диски не предназначены для долговременного хранения пользовательских данных.
- Необходимо использовать формат QCOW2 или другой формат, поддерживающий дельта-файлы.

Требования к хостовым ОС

Системные пакеты

Дистрибутив	Версия QEMU	Поддержка KVM	Версия libvirt	Версия OVS	Версия RabbitMQ	Версия memcached	Версия MariaDB
Almalinux 8.4	5.0	да	6.0.0	2.12	3.8.0	1.6.0	10.3

Модули TIONIX

Модуль	Версия
NodeControl	⩾3.0
Dashboard	⩾3.0
Monitor	⩾3.0
Scheduler	⩾3.0
Client	⩾3.0
Drivers	⩾3.0
Agent	⩾3.0
PointMeter	⩾3.0

Настройка окружения

Миграция с CentOS 8 на Almalinux

Введение

Начиная со 2 февраля 2022 года продукты Tionix перешел на использование ОС Almalinux 8.4 вместо CentOS 8.4, поддержка которой закончилась 31 декабря 2021 года. Поэтому для текущих инсталляций может потребоваться миграция ОС. Данная статья описывает основные шаги такой миграции.

Примечания к процессу обновления:

- Минимальная версия CentOS для миграции: 8.4. Версии ниже не поддерживаются.
- Осуществляется обновление на версию Almalinux 8.5.

Предварительная настройка

Выключение сервисов платформы

Перед началом миграции необходимо полностью выключить все сервисы OpenStack и Tionix:

```
systemctl stop tionix-* openstack-* neutron-* mysqld rabbitmq httpd
```

Изменение конфигурации dnf

В `/etc/dnf/dnf.conf` необходимо поменять параметр best на False:

```
best=False
```

На этом предварительная настройка закончена.

Миграция дистрибутива

Для миграции необходимо использовать скрипт almalinux-deploy. Получите скрипт с git-репозитория проекта:

```
git clone https://github.com/AlmaLinux/almalinux-deploy
```

Перейдите на каталог almalinux-deploy и запустите сам скрипт:

```
cd almalinux-deploy
chmod +x almalinux-deploy
./almalinux-deploy
```

Скрипт работает полностью автоматически. Процесс выглядит примерно так:

- Вначале скрипт проверяет окружение, версию CentOS и иные параметры на соответствие требованиям скрипта.
- Меняет стандартные репозитории CentOS на свои.
- Обновляет список репозиториев и включает на обновление все пакеты, которые относятся к стандартным пакетам CentOS. Прочие репозитории тронуты не будут.
- Запускает процесс обновления, он займет некоторое время.

После окончания процесса переустановки пакетов с репозитория Almalinux потребуется перезапуск ОС:

```
reboot
```

После перезапуска проверьте, что система действительно сменилась на Almalinux:

```
cat /etc/redhat-release
```

Настройка сетевых интерфейсов

Перед тем, как начать установку программного обеспечения, необходимо настроить имеющиеся сетевые интерфейсы. Эта статья описывает общий алгоритм настройки сетевых интерфейсов.

Конфигурация сетевых интерфейсов находится по пути:

- `/etc/sysconfig/network-scripts/ifcfg-{interface-name}`

Далее в качестве примера будет использовано имя интерфейса *eth0.* В качестве бэкенда управления сетью используется NetworkManager[3].

Минимальная конфигурация

1. При наличии DHCP для сети, куда подключен сетевой интерфейс, достаточно использовать следующую конфигурацию:
   Файл конфигурации

   ```
   TYPE=Ethernet
   BOOTPROTO=dhcp
   IPV4_FAILURE_FATAL=no
   NAME=eth0
   DEVICE=eth0
   ONBOOT=yes
   ```

2. В этом случае сетевой интерфейс получит адрес и сетевую маску. Если необходимо, чтобы интерфейс использовался для маршрута по умолчанию, то добавьте следующую строку:

   ```
   DEFROUTE=yes
   ```

   Убедитесь, что только один интерфейс настроен со включенным параметром маршрута по умолчанию.

3. Перезапустите сервис NetworkManager:

   ```
   systemctl restart NetworkManager
   ```

Смена имени машины

В этом Руководстве в качестве основного имени машины используется имя controller. Для изменения имени машины следует выполнить следующую команду:

```
hostnamectl set-hostname controller
```

Для визуального изменения имени машины (например, при работе с узлом через SSH-протоколо) в оболочке нужно перезайти на узел.

Регистрация доменных имён узлов

По умолчанию адреса всех узлов инфраструктуры должны быть зарегистрированы в DNS и предоставлены доменные имена. Для тестовых целей доменные имена можно указать в /etc/hosts. В частности, для controller нужно указать адрес, который прописан в mgmt-интерфейсе:

```
10.0.0.11 controller
```

Настройка репозиториев Almalinux

Введение

Перед началом установки облачной платформы необходимо добавить дополнительные репозитории для Almalinux 8. Они должны быть добавлены на всех узлах облачной платформы.

3 https://en.wikipedia.org/wiki/NetworkManager

Если вы нам необходимо произвести миграцию ОС с CentOS 8 на Almalinux 8, то используйте эту инструкцию. (см. стр. 9)

Включение репозиториев

Основные репозитории Almalinux не требуют какой-либо настройки.

OpenStack

Для установки OpenStack требует установить дополнительные репозитории:

- Репозиторий Powertools в составе Almalinux, который выключен по умолчанию.
- EPEL, содержащий дополнительные пакеты для Almalinux.
- Репозитории CentOS SIG, расположенные в CentOS Vault, которые содержат компоненты OpenStack.

Репозиторий Powertools

Для некоторых пакетов OpenStack требуются пакеты из репозитория Powertools. Включите этот репозиторий:

```
dnf config-manager --enable powertools
```

EPEL

Репозиторий EPEL доступен в виде пакета, установите его:

```
dnf -y install epel-release
```

Репозитории CentOS SIG

На данный момент Almalinux не содержит пакеты OpenStack, их необходимо брать с репозиториев CentOS SIG. Для этого создайте файл `/etc/yum.repos.d/openstack.repo` со следующим содержимым:

```
[centos84-openstack-victoria]
name=CentOS SIG OpenStack Victoria Repository
baseurl=http://mirror.nsc.liu.se/centos-store/8.4.2105/cloud/x86_64/openstack-victoria/
gpgcheck=0
enabled=1

[centos84-adv-virt]
name=CentOS SIG Advanced Virtualization Repository
baseurl=http://mirror.nsc.liu.se/centos-store/8.4.2105/virt/x86_64/advanced-
virtualization/
gpgcheck=0
enabled=1

[centos84-rabbitmq38]
name=CentOS SIG RabbitMQ 3.8 Repository
baseurl=http://mirror.nsc.liu.se/centos-store/8.4.2105/messaging/x86_64/rabbitmq-38/
gpgcheck=0
enabled=1

[centos84-ceph-pacific]
name=CentOS SIG Ceph Pacific Repository
baseurl=http://mirror.nsc.liu.se/centos-store/8.4.2105/storage/x86_64/ceph-pacific/
gpgcheck=0
enabled=1

[centos84-openvswitch]
name=CentOS SIG Open vSwitch Repository
baseurl=http://mirror.nsc.liu.se/centos-store/8.4.2105/nfv/x86_64/openvswitch-2/
gpgcheck=0
enabled=1

[centos84-haproxy]
name=CentOS SIG HAProxy 2.2 Repository
baseurl=http://mirror.nsc.liu.se/centos-store/8.4.2105/nfv/x86_64/network-extras/
gpgcheck=0
enabled=1
```

Обновите систему:

```
dnf -y update
```

TIONIX

Модули TIONIX распространяются отдельно. Создайте файл `/etc/yum.repos.d/tionix-3-0.repo` со следующим содержанием:

```
[tionix-modules]
baseurl=http://rpm-centos.tionix.ru/3.0/x86_64/
enabled=1
gpgcheck=0
name=Tionix Modules 3.0 for EL8

[tionix-extras]
baseurl=http://rpm-centos.tionix.ru/extras/el8/x86_64/
enabled=1
gpgcheck=0
name=Tionix Modules 3.0 for EL8 (Extra Packages)
```

На этом настройка репозиториев закончена.

> После добавления репозиториев TIONIX возможны проблемы обновления системы или установки некоторых пакетов. В этом случае запустите утилиту dnf с параметром --nobest.

Установка системных пакетов

После добавления репозиториев необходимо установить системные пакеты, которые потребуются для дальнейших шагов настройки.

OpenStack

1. Всё взаимодействие с облачной платформой производится через клиент openstackclient:

```
dnf -y install python3-openstackclient
```

2. Установите политики SELinux[4] для OpenStack:

```
dnf -y install openstack-selinux
```

> **Примечание**
>
> Для продуктивных систем крайне нежелательно выключать систему мандатного доступа SELinux[5].

TIONIX

Для модулей TIONIX необходимы следующие пакеты:

1. Установите пакет лицензирования модулей:

> **Важно**
>
> Открытые лицензии работают в течение 3 месяцев после его генерации. Лицензии для коммерческих инсталляций генерируются отдельно и по запросу.

   a. **открытая лицензия:**

```
dnf -y install python3-tionix_licensing-3.0.0
dnf -y install tionix-license
```

   b. **коммерческая лицензия:**

> **Примечание**
>
> При наличии уже установленной открытой лицензии необходимо ее предварительно удалить:
>
> ```
> dnf remove tionix-license
> ```

```
dnf -y install tionix-license-3.0.0-20211208.el8.noarch.rpm
```

   Где: `tionix-license-3.0.0-20211208.el8.noarch.rpm` - файл пакета лицензии.

2. Установите пакет Setuptools:

```
dnf -y install python3-setuptools
```

3. Установите пакет distro:

```
dnf -y install python3-distro
```

Настройка сервиса NTP

4 https://conf.tionix.ru/pages/viewpage.action?pageId=174030874#id-Глоссарий-selinux-term
5 https://conf.tionix.ru/pages/viewpage.action?pageId=174030874#id-Глоссарий-selinux-term

Введение

Любая облачная платформа очень чувствительна к тому выставленному в часах узлов времени. Очень важно при взаимодействии между сетевыми сервисами получать одни и те же значения времени. Это касается даже очень простых инсталляций, где отдельно есть один управляющий и один вычислительный узел, не говоря уже о кластерных вариантах и референсной архитектуре. В качестве протокола точного времени в TCP используется стандартный протокол NTP[6] (Network Time Protocol), а в качестве реализации этого протокола - chrony[7]. Выбор chrony прост: на данный момент это стандарт де-факто с гибкой конфигурацией и поддержкой сервера NTP.

> При использовании референсной архитектуры chrony необходимо устанавливать только на железные узлы. Внутри контейнеров chrony устанавливать не надо

Для лучшего функционирования NTP предлагается использовать следующий вариант настройки:

- В управляющем узле или в корпоративной сети настраивается NTP-клиент, который настроен на использование географически ближайшего пула NTP-серверов. Например, ru.pool.ntp.org или любой NTP-сервер со stratum <=2.
- Одновременно этот сервис NTP должен быть настроен как сервер.
- Все остальные NTP-клиенты, установленные в узлах облачной платформы, должны быть настроены на использование этого локального NTP-сервера.

Данная конфигурация позволяет:

- Получить меньшие уровни jitter при работе с удаленными NTP-серверами.
- Позволяет получить точное время при отсутствии доступа к Интернету для узлов облачной платформы через сеть управления (mgmt).

Однако нужно учесть, что уровень stratum, который влияет на уровень точности синхронизации, для конечных NTP-клиентов не должен опускаться ниже 3-4, иначе между узлами рассинхронизация времени может превысить допустимый предел. Убедитесь, что в используемом пуле NTP-серверов используются stratum не ниже 2, в этом случае stratum у локального NTP-сервера не будет больше 3-4, что вполне допустимо. Подробнее об уровнях stratum можно узнать здесь[8].

Для примера решим, что NTP-сервер ставится на управляющий узел.

NTP-сервер использует порт 123/UDP.

> При наличии корпоративного NTP-сервера пропустите шаг установки сервера NTP, а в клиентах укажите адрес корпоративного сервера в качестве основного по аналогии с настройкой сервера на базе chrony.

Установка сервера NTP

Установка NTP-сервера достаточно проста.

1. Установите пакет chrony:

   ```
   dnf -y install chrony
   ```

   > Стандартные пути конфигурации:
   >
   > - `/etc/chrony.conf` - основной файл конфигурации.

2. В основном файле конфигурации укажите сервер **ru.pool.ntp.org** в качестве основного пула NTP-серверов (все остальные настроенные пулы и серверы необходимо удалить или закомментировать):

   ```
   pool ru.pool.ntp.org iburst
   ```

   > Параметр iburst уменьшает интервал между первыми четырьмя запросами к пулу NTP-серверов после запуска сервиса до 2 и менее секунд. Это необходимо для более быстрой первой синхронизации времени (по умолчанию минимальный интервал между запросами равен 64 секундам).

6 https://ru.wikipedia.org/wiki/NTP
7 https://chrony.tuxfamily.org/
8 https://habr.com/ru/post/79461/

3. Так как chrony в управляющем узле должен работать в режиме сервера, необходимо разрешить клиентам с подсетей платформы подключаться к нему:

```
allow 10.0.0.0/8
```

> Вместо 10.0.0.0/8 укажите подсеть узлов облачной платформы, к которым необходимо настроить доступ до этого сервера NTP. Разрешено использовать несколько директив allow в отдельных строках.

4. Запустите сервис chrony и добавьте его в автозапуск:

```
systemctl start chronyd.service
systemctl enable chronyd.service
```

Установка клиента NTP

Установка клиента NTP на всех остальных узлах облачной платформы проводится так же.

1. Установите пакет chrony:

```
dnf -y install chrony
```

> Стандартные пути конфигурации:
>
> - `/etc/chrony/chrony.conf` - основной файл конфигурации.

2. Укажите локальный сервер NTP в конфигурацию с параметром iburst (все остальные серверы и пулы должны быть закомментированы или удалены):

```
server controller iburst
```

> Если локальных серверов несколько, то можно добавить несколько директив server в отдельных строках.

3. Запустите сервис chrony и добавьте его в автозапуск:

```
systemctl start chronyd.service
systemctl enable chronyd.service
```

Проверка установки

В обоих случаях проверка работы сервиса NTP сводится к получению данных о доступных серверах NTP.

1. Запустите команду получения списка доступных NTP-серверов:

```
chronyc sources
```

2. Для сервера NTP команда должен вернуть примерно такой вывод:

```
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^+ 78-36-18-184.dynamic.mur>     1  10   377   978  +5123us[+6066us] +/-   28ms
^+ yggnode.cf                    2  10   377  1123  -1271us[ -330us] +/-   17ms
^+ 79.120.30.43                  1  10   377   222   -683us[ -668us] +/-   24ms
^* 128.0.142.251                 2   8   377   167  -1762us[-1747us] +/-   22ms
```

3. Аналогично для клиента NTP:

```
210 Number of sources = 1
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^* controller                    3    9   377   421    +15us[  -87us] +/-   15ms
```

Сервер, отмеченный знаком *, является выбранным для получения времени.

Иногда даже с iburst обновление времени после запуска сервиса происходит не мгновенно. В этом случае можно явно попросить сервер сделать шаг синхронизации:

```
chronyc makestep 1 0.3
```

Установка балансировщика нагрузки HAProxy

Для части сервисов OpenStack имеются проблемы в реализации протокола шифрования протокола TCP. Для обхода этих проблем в референсной архитектуре было решено использовать балансировщик нагрузки HAProxy с функцией SSL Termination.

HAProxy[9] - это проект с открытым исходным кодом, предоставляющий возможности балансировщика нагрузки методом перенаправления запросов на экземпляры сетевой службы по определенному алгоритму.

В HAProxy используется два ключевых понятия:

- фронтенд (frontend) - это открываемый HAProxy сетевой порт, предназначенный для принятия запросов с клиентских приложений;
- бэкенд (backend) - это сетевые сервисы, в которые будут перенаправлены запросы с клиентских приложений по алгоритму распределения, указанного в конфигурации балансировщика нагрузки.

Установка HAProxy

1. Установите пакет HAProxy:

```
dnf -y install haproxy
```

Стандартные пути конфигурации:

- `/etc/haproxy` - каталог с конфигурацией;
- `/etc/haproxy/haproxy.cfg` - основной конфигурационный файл.

2. Приведите конфигурацию HAProxy к следующему:

9 http://www.haproxy.org/

```
global
  log stderr daemon
  maxconn 10000
  ssl-default-bind-ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-
SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-
CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-
AES256-GCM-SHA384
  ssl-default-bind-ciphersuites
TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256
  ssl-default-bind-options prefer-client-ciphers no-sslv3 no-tlsv10 no-tlsv11 no-
tls-tickets
  ssl-default-server-ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-
SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-
CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-
AES256-GCM-SHA384
  ssl-default-server-ciphersuites
TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256
  ssl-default-server-options no-sslv3 no-tlsv10 no-tlsv11 no-tls-tickets
  ssl-dh-param-file /usr/local/etc/haproxy/dhparam

defaults
  log global
  mode http
  option httplog
  timeout connect 3s
  timeout client 3m
  timeout server 3m
  timeout tunnel 1h
```

Таблица конфигурации

ⓘ Легенда таблицы доступна на этой странице[10].

Имя параметра	Описание	Примечания
global	Глобальные настройки сервиса HAProxy.	
log	Параметры настройки журналирования.	Указанное значение отправляет журнал сервиса в stderr, что позволит перенаправить его в систему управления контейнера. Для обычной установки требуется[11] установка syslog-сервиса и указание его адреса: • log 127.0.0.1 local0 Параметр global - это имя канала журналирования. Можно указать несколько параметров log с разными каналами, если требуется разделение журналов для фронтэндов.
maxconn	Максимальное количество соединений к сервису.	По умолчанию значение равно 2000. ⚠ Не забудьте повысить лимиты на количество открытых файлов (nofile) для сервиса до значения maxconn.

10 https://conf.tionix.ru/pages/viewpage.action?pageId=205881354#id-Условныеобозначения-table-format-desc
11 https://www.haproxy.com/blog/introduction-to-haproxy-logging/

Имя параметра	Описание	Примечания
ssl-default-bind-ciphers	Список доступных алгоритмов шифрования TLS v1.2 для фронтэндов (используется для директив bind).	По поводу списка алгоритмов шифрования ознакомьтесь с общим примечанием после этой таблицы.
ssl-default-bind-ciphersuites	Список доступных алгоритмов шифрования TLS v1.3 для фронтэндов (используется для директив bind).	По поводу списка алгоритмов шифрования ознакомьтесь с общим примечанием после этой таблицы.
ssl-default-bind-options	Список доступных версий протокола TLS для фронтэндов (используется для директив bind).	В этом параметре явно выключаются все небезопасные версии протокола TLS для фронтэндов. Тикеты TLS необходимо выключить из-за их проблем с безопасностью[12] в версии TLS v1.2 (для TLS v1.3+ этот параметр не применим).
ssl-default-server-ciphers	Список доступных алгоритмов шифрования TLS v1.2 для бэкендов (используется для директив server).	По поводу списка алгоритмов шифрования ознакомьтесь с общим примечанием после этой таблицы.
ssl-default-server-ciphersuites	Список доступных алгоритмов шифрования TLS v1.3 для бэкендов (используется для директив server).	По поводу списка алгоритмов шифрования ознакомьтесь с общим примечанием после этой таблицы.
ssl-default-server-options	Список доступных версий протокола TLS для бэкендов (используется для директив server).	В этом параметре явно выключаются все небезопасные версии протокола TLS для фронтэндов. Тикеты TLS необходимо выключить из-за их проблем с безопасностью[13] в версии TLS v1.2 (для TLS v1.3+ этот параметр не применим).
ssl-dh-param-file	Файл параметров протокола Диффи-Хеллмана.	
defaults	Общие параметры для фронтэндов.	
log	Указание канала журналирования по умолчанию.	
mode	Протокол для проксирования по умолчанию	Почти все сервисы в облачной платформе использует протокол HTTP, поэтому его указываем по умолчанию. При необходимости указать протокол TCP, то явно укажите директиву `mode tcp` в теле фронтенда и бэкенда.
option	Включение различных функций балансировщика.	Директива `option` предназначена для включения определенных функций. Например, `httplog` включает функцию регистрации подключений к фронтэнду по протоколу HTTP в журналах сервиса HAProxy.

12 https://blog.filippo.io/we-need-to-talk-about-session-tickets/
13 https://blog.filippo.io/we-need-to-talk-about-session-tickets/

Имя параметра	Описание	Примечания
timeout connect	Таймаут подключения к бэкенду HAProxy по умолчанию.	Параметр не относится к фронтэнду.
timeout client	Таймаут ожидания ответа на запрос клиента в фронтэнде.	
timeout server	Таймаут ожидания ответа сервера (узла, указанного в директиве server в привязанном бэкенде).	
timeout tunnel	Таймаут взаимодействия между клиентом и сервера после установления соединения ("туннеля").	Этот параметр предназначен для долгоживущих соединений для предотвращения раннего разрыва соединения со стороны балансировщика нагрузки.

ⓘ В списке chiphers содержатся современные варианты AES на базе алгоритмов ECDSA (с поддержкой аппаратного ускорения AES-NI), алгоритмы CHACHA20-POLY1305 и классический AES с использованием протокола Диффи-Хеллмана (DH). Алгоритм CHACHA20-POLY1305 необходим для более быстрой программной обработки шифрованного соединения при невозможности использования AES с ECDSA, а DHE-AES используется в качестве legacy-варианта.

4. Проверьте корректность конфигурации HAProxy:

```
haproxy -c -f /etc/haproxy/haproxy
```

5. Запустите сервис HAProxy:

```
systemctl start haproxy && \
systemctl enable haproxy
```

Настройка фронтенда и бэкенда

Для большинства сервисов облачной платформы необходимо настроить отдельные фронтенды и бэкенды в HAProxy. В этом разделе будет показана общая настройка, а конфигурация будет предложена в разделах самих сервисов.

ⓘ Для сервисов со своей настройкой балансировщика нагрузки конфигурация будет указана отдельно с полным описанием директив.

1. Общая конфигурация фронтенда выглядит следующим образом:

```
frontend service_name
  bind "$IP:PORT" ssl crt /usr/local/etc/haproxy/cert.pem alpn h2,http/1.1
  http-request set-header X-Forwarded-Proto https
  default_backend service_name_backend
```

Таблица конфигурации

ⓘ Легенда таблицы доступна на этой странице[14].

14 https://conf.tionix.ru/pages/viewpage.action?pageId=205881354#id-Условныеобозначения-table-format-desc

Имя параметра	Описание	Примечания
frontend	Начало секции описания параметров фронтенда.	service_name является названием фронтенда.
bind	Адрес и порт для фронтенда.	в bind должен быть указан адрес mgmt-интерфейса. Можно указать несколько директив bind.
ssl	Параметр bind, включающий режим шифрования для фронтенда.	
crt	Параметр для bind, путь до файла сертификата с ключом в формате pem. Обязателен, если указан ssl	
alpn	Параметр для bind, протокол выбора версии HTTP.	Позволяет клиенту сообщить о поддерживаемых протоколах HTTP в сервере. После этого параметра нужно указать список поддерживаемых версий через запятую.
http-request	Директива, позволяющая манипулировать запросами HTTP.	
default_backend	Бэкенд по умолчанию для перенаправления запросов клиента, переданных фронтенду.	

2. Общая конфигурация бэкенда выглядит следующим образом:

```
backend service_name_backend
  server glance 127.0.0.1:9292
```

Таблица конфигурации

ⓘ Легенда таблицы доступна на этой странице[15].

Имя параметра	Описание	Примечания
backend	Начало секции описания параметров бэкенда.	service_name_backend является именем бэкенда.
server	Адрес сетевого сервиса, расположенного за балансировщиком нагрузки.	В первой позиции после этого параметра указывается имя сервера, во второй - IP-адрес и порт. Можно указать несколько серверов.

3. После добавления или изменения фронтенда/бэкенда вначале проверьте правильность конфигурации:

```
haproxy -c -f /etc/haproxy/haproxy
```

4. Перезагрузите конфигурацию сервиса HAProxy

```
systemctl reload haproxy
```

15 https://conf.tionix.ru/pages/viewpage.action?pageId=205881354#id-Условныеобозначения-table-format-desc

⚠ Во время перезагрузки конфигурации возможны единичные обрывы установленных соединений.

Настройка SSL Termination в HAProxy

Введение

Информация

См. также: Установка балансировщика нагрузки HAProxy (см. стр. 16).

По принятой референсной архитектуре шифрование сервисов OpenStack, кроме Keystone, не производится на уровне самого сервиса или на уровне веб-сервера Apache, а вместо этого используется функция SSL Termination[16] на уровне балансировщика нагрузки HAProxy[17]. Эта статья кратко опишет эту часть настройки балансировщика.

Если кратко, SSL Termination позволяет устанавливать шифрованное соединение между клиентом и самим балансировщиком нагрузки, а не с самим сервисом. Это упрощает настройку конечных сервисов OpenStack, позволяет оффлоадить нагрузку на шифрование, однако для некоторых типов нагрузки, как показывает опыт, такой вариант терминирования SSL не подходит (в основном, относится к протоколам, базирующихся на TCP и крайне чувствительных к единичным потерям пакетов).

В основной статье уже предоставлена конфигурация с SSL Termination, здесь чуть более подробно описан сам механизм.

Настройка SSL Termination

Схематически схема терминирования SSL в балансировщике выглядит так:

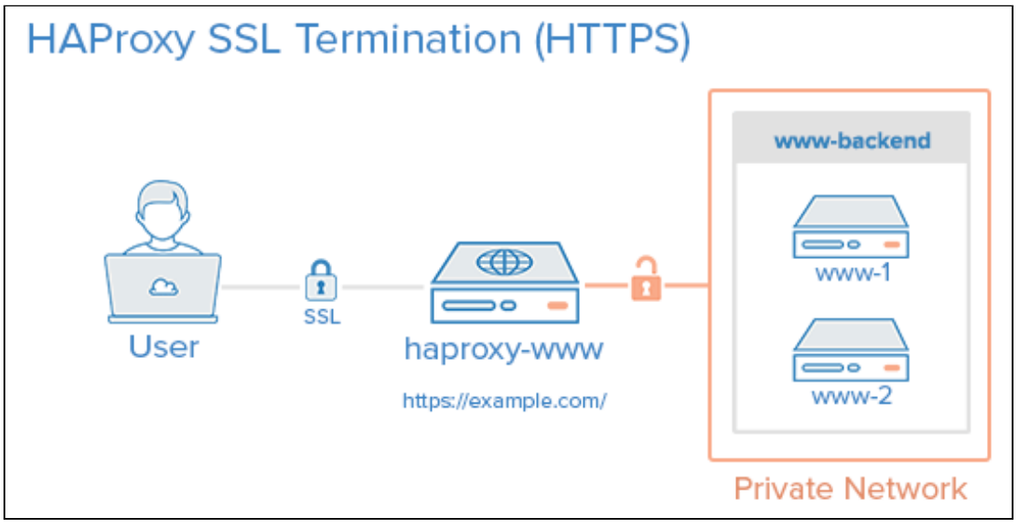


Схема терминирования SSL

Пользователь при соединении к (данном случае) к веб-серверу по шифрованному соединению соединяется с балансировщиком нагрузки.

В фронтенде балансировщика указаны следующие настройки:

```
frontend service_name
bind "IP:PORT" ssl crt /usr/local/etc/haproxy/cert.pem ...
...
```

Параметр ssl указывает на то, что HAProxy ожидает подключения по протоколу TLS, а crt указывает, какой ключ для шифрования следует использовать.

Пользовательский запрос в фронтенде расшифровывается и далее передается бэкенду, в котором указаны адреса сетевых сервисов. Этот бэкенд по умолчанию не использует шифрование при подключении. Он содержит следующую конфигурацию:

16 https://www.haproxy.com/blog/haproxy-ssl-termination/
17 https://www.haproxy.com/

```
backend service_name_backend
  server service IP:PORT
```

В параметрах server директива ssl не указана, в этом случае HAPRoxy при соединении с сетевым сервисом не будет пытаться использовать протокол TLS, а воспользуется чистым протоколом HTTP (при mode http).

Эта настройка и является включением терминирования SSL.

Установка и настройка СУБД MariaDB

Введение

Всё состояние облачной платформы хранится в едином источнике и хранилище данных — базе данных SQL. В качестве референсной системы управления базами данных (СУБД) в референсной архитектуре используется MariaDB.

MariaDB[18] - это СУБД с открытым исходным кодом, которая является форком проекта MySQL[19] после перехода прав на этот продукт компании Oracle[20]. На данный момент продолжает активно развиваться[21] и де-факто является стандартом для большинства распространённых дистрибутивов Linux. Выбор MariaDB связан с тем, что он лучше всего протестирован как СУБД для сервисов OpenStack.

Второй компонент, используемый в референсной архитектуре — это Galera[22]. Это расширение для MariaDB, позволяющее разворачивать кластерные варианты СУБД MariaDB в режиме Master/Master: все экземпляры такого кластера могут принимать запросы как на чтение, так и на запись. До версии 10.6 Galera умела реплицировать только пользовательские данные в формате InnoDB[23], с 10.6 появилась поддержка движка хранения Aria[24] (в экспериментальном режиме, как и MyISAM[25]), в котором хранятся все системные таблицы MariaDB. С версии 10.1 Galera включена в состав сервера MariaDB и отдельно его устанавливать не требуется.

Сервер MariaDB использует порт 3306/TCP[26], Galera - порты[27] 4567/TCP, 4568/TCP и 4444/TCP.

Установка MariaDB

Установка MariaDB производится в несколько шагов.

1. Установите пакет MariaDB и библиотеку Python для работы с этой СУБД:

   ```
   dnf -y install mariadb mariadb-server python3-PyMySQL
   ```

   Стандартные пути до конфигурации:

   - `/etc/my.cnf.d` - каталог с конфигурацией.
   - `/etc/my.cnf.d/my.cnf` - основной конфигурационный файл. Этот файл менять не следует.
   - `/etc/my.cnf.d/openstack.cnf` - файл конфигурации СУБД для сервисов OpenStack.

2. В файл `/etc/my.cnf.d/openstack.cnf` добавьте следующие параметры (описание (см. стр. 26)):

18 https://mariadb.org/
19 https://www.mysql.com/
20 https://www.oracle.com/index.html
21 https://github.com/MariaDB/server
22 https://mariadb.com/kb/en/what-is-mariadb-galera-cluster/
23 https://mariadb.com/kb/en/innodb/
24 https://mariadb.com/kb/en/aria-storage-engine/
25 https://mariadb.com/kb/en/myisam-storage-engine/
26 https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml?search=3306#Monty
27 https://galeracluster.com/library/documentation/firewall-settings.html

```
[mysqld]
bind-address = 10.0.0.11

default-storage-engine = innodb
innodb_file_per_table = on
max_connections = 4096
character-set-server = utf8
collation-server = utf8_general_ci
```

3. Для сервера MariaDB необходимо поднять лимиты на открытые файловые дескрипторы в ОС. Для этого создайте файл по пути `/etc/systemd/system/mariadb.service.d/limits.conf` и укажите следующее:

```
[Service]
LimitNOFILE=10000
```

4. Перезагрузите конфигурацию systemd для обновления юнита MariaDB:

```
systemctl daemon-reload
```

5. После сохранения конфигурации запустите сервис MariaDB и добавьте его в автозапуск:

```
systemctl start mariadb
systemctl enable mariadb
```

Обычная установка на этом заканчивается.

Настройка кластера Galera

Для кластерного варианта MariaDB с использованием Galera необходимо добавить параметра для протокола синхронизации wsrep[28].

> ⓘ Galera встроена в дистрибуцию MariaDB, поэтому отдельно устанавливать пакеты не требуется. Пакеты MariaDB должны быть установлены на всех узлах, где будут запущены экземпляры кластера Galera.

> ⚠ Для продуктивных систем обязательно нужно использовать как минимум три экземпляра кластера Galera. По умолчанию Galera следит за тем, что большинство экземпляров (>50% от общего количества узлов в кластере) может принимать запросы, иначе доступ к БД (даже чтение) будет заблокирован (подробнее[29]). Возможен вариант с использованием арбитра[30], однако такой способ инсталляции в Tionix Cloud Platform на данный момент официально не поддерживается.

1. Установите MariaDB на всех узлах как при обычной установке, где предполагается запустить экземпляры кластера и примите минимальную конфигурацию.
2. Создайте файл `/etc/my.cnf.d/galera.cnf` со следующим содержимым (описание (см. стр. 26)):

```
[galera]
wsrep_on = ON
wsrep_cluster_address = gcomm://controller1,controller2,controller3
wsrep_provider = /usr/lib/galera/libgalera_smm.so
binlog_format = ROW
default_storage_engine = InnoDB
innodb_autoinc_lock_mode = 2
innodb_doublewrite = 1
innodb_flush_log_at_trx_commit = 0
innodb_buffer_pool_size=2G
```

3. После этого необходима инициализация кластера: один из узлов с экземпляром MariaDB должен запуститься и стать условным мастером, с которого остальные экземпляры получат реплику. Выберите такой узел и в нём запустите команду:

---

28 https://galeracluster.com/library/documentation/architecture.html
29 https://galeracluster.com/library/documentation/crash-recovery.html
30 https://galeracluster.com/library/documentation/arbitrator.html

```
galera_new_cluster
```

4. В случае успеха прошлая команда должна вернуть пустой вывод. Через systemctl проверьте статус сервиса mariadb (статус сервиса должен быть "Active"):

```
systemctl status mariadb
```

5. Во всех остальных узлах нужно просто запустить сервис mariadb:

```
systemctl start mariadb
```

6. В любом узле зайдите в интерактивную сессию с СУБД:

```
mysql
```

7. Получите количество добавленных в кластер узлов:

```
SHOW GLOBAL STATUS LIKE 'wsrep_cluster_size';
```

Вы должны получить примерно следующий вывод:

```
+--------------------+-------+
| Variable_name      | Value |
+--------------------+-------+
| wsrep_cluster_size | 3     |
+--------------------+-------+
```

Теперь при обращении к любому адресу из списка кластеров Galera вы должны получить одинаковый ответ от СУБД.

8. Выполните команду для указания базовых параметров безопасности:

```
mysql_secure_installation
```

На этом первичная настройка кластера Galera закончена.

Настройка Galera в HAProxy

> *См. также:* Установка балансировщика нагрузки HAProxy (см. стр. 16).

В *референсной архитектуре* доступ до сервисов Galera предоставляется через балансировщика нагрузки. Конфигурация для HAProxy выглядит следующим образом (описание (см. стр. 27)):

```
global
  log stderr daemon
  maxconn 10000

defaults
  log global
  mode tcp
  option tcplog
  timeout connect 3s
  timeout client 3m
  timeout server 3m
  timeout tunnel 1h

resolvers k8s
  parse-resolv-conf
  accepted_payload_size 8192

frontend galera
  bind :3306
  default_backend galera_backend

# Send all traffic to a "master" server. HAProxy should acquire new address if current
one is down.
# https://dba.stackexchange.com/questions/203956/mysql-galera-cluster-mass-update-delay
# https://galeracluster.com/library/kb/deadlock-found.html
# https://ghostaldev.com/2016/05/22/galera-gotcha-mysql-users
backend galera_backend
  option mysql-check user password
  server master galera.domain.loc:3306 check resolvers k8s init-addr none
```

Отметим неописанные части конфигурации.

Описание настройки бэкенда

В настройках бэкенда можно заметить следующие нюансы настройки.

1. Данная строка включает параметр проверки доступа к базе данных методом аутентификации в СУБД указанным пользователем и паролем:

   ```
   option mysql-check user password
   ```

   > ⓘ Сервер в конфигурации указывается один, потому что в референсной архитектуре принято, что адреса узлов кластера Galera регистрируются в сервере DNS по указанному в конфигурации бэкенда адресу.

2. При *обычной установке* можно просто указать три узла Galera явно, в этом случае конфигурация примет следующий вид:

   ```
   backend galera_backend
        balance source
        option mysql-check user haproxy
        server node1 192.168.1.1:3306 check weight 1
        server node2 192.168.1.2:3306 check weight 1
        server node3 192.168.1.3:3306 check weight 1
   ```

   Алгоритм балансировки source позволяет привязывать сессии TCP клиентских приложений к конкретному экземпляру СУБД в кластере Galera.

Таблицы конфигурации

> ⓘ Легенда таблиц доступна на этой странице[31].

31 https://conf.tionix.ru/pages/viewpage.action?pageId=205881354#id-Условныеобозначения-table-format-desc

Общие параметры MariaDB

Имя параметра	Описание	Примечания
[mysqld]	Глобальные параметры сервера MariaDB.	
default-storage-engine	Указание движка хранения базы данных.	OpenStack поддерживает только InnoDB.
innodb_file_per_table	Хранение таблиц базы данных в отдельных файлах.	Этот параметр указывается как оптимизация производительности.
max_connections	Максимальное количество соединений к СУБД.	Для референсной архитектуры в качестве дефолта принято количество в 4096 соединений. В большинстве случаев этого достаточно. Однако для очень больших инсталляций этот параметр можно увеличивать и далее. Не забудьте так же увеличить лимиты ОС nofile (об этом ниже).
character-set-server	Кодировка символов, используемых в базе данных.	Этот параметр указывает общую кодировку символов, которые будут использованы при хранении текстовых данных внутри БД. OpenStack поддерживает только эту кодировку (MariaDB использует latin1 по умолчанию), поэтому её необходимо явно указать. Подробнее об этом можно узнать здесь[32].
collation-server	Представление символов, используемых в базе данных.	Этот параметр то, каким подсемейством указанной кодировки символы будут представлены для каких-либо операций, например, сортировки полей. Подробнее об этом можно узнать здесь[33].

Параметры кластеризации Galera

Имя параметра	Описание	Примечания
[galera]	Параметры репликации Galera	
wsrep_on	Включение протокола wsrep API	wsrep API является внутренним механизм репликации данных БД между экземплярами кластера. По умолчанию он выключен.
wsrep_cluster_address	Список адресов экземпляров кластера Galera.	Список серверов нужно указывать через запятую, порт можно указать, добавив к домену знак двоеточия.
wsrep_provider	Путь до библиотеки с реализацией протокола wsrep.	Убедитесь, что путь до библиотеки корректен.
binlog_format	Формат бинарных журналов[34]состояния БД.	На момент написания Galera официально поддерживает только формат ROW[35] в силу того, что он является самым безопасным форматом хранения бинарных журналов.

32 https://mariadb.com/kb/en/character-sets/
33 https://mariadb.com/kb/en/character-sets/
34 https://mariadb.com/kb/en/binary-log/
35 https://mariadb.com/kb/en/mariadb-galera-cluster-known-limitations/

Имя параметра	Описание	Примечания
default_storage_engine	Стандартный движок хранения базы данных	Galera поддерживает репликацию данных только с движком InnoDB.
innodb_autoinc_lock_mode	Режим блокировки автоинкремента[36]	Если кратко, автоинкремент необходим для генерации идентификаторов для объектов, добавляемых в базу данных. Параметр 2 ("interleaved") выключает блокировку при командах INSERT, что позволяет запускать сразу несколько инструкций INSERT одновременно. Это безопасно при использовании ROW-формата для бинарных журналов и позволяет повысить быстродействие.[37] Также нужно отметить, что при работе автоинкремента ID объектов будут увеличиваться на шаг, равный числу узлов кластера Galera.
innodb_doublewrite	Включение буфера двойной записи.	Этот параметр позволяет увеличить надежность кластера Galera: при сбросе страницы данных на диск MariaDB вначале запишет данные в буфер двойной записи и лишь после того, как она убедится, что запись была выполнена успешно страница будет записана в конечной таблице базы данных. При восстановлении данных MariaDB сравнит успешно записанные элементы в буфере с данными в конечной таблице данных. Если записи будут отличаться, значит, запись в конечной таблице будет считаться невалидной. Подробнее здесь[38]. Отрицательно влияет на производительность кластера из-за двойной записи на диск.
innodb_flush_log_at_trx_commit	Политика записи данных транзакции в буферном журнале.	При режиме 2 в буферный журнал будет попадать информация о транзакциях, в диск эта информация будет сбрасываться каждую секунду. Для обычных установок — это опасный параметр, однако в случае Galera данные о транзакциях реплицируются между узлами в синхронном режиме, поэтому нет надобности в сбросе буферного журнала при каждой транзакции (режим 1, по умолчанию). Очень сильно положительно влияет на производительность кластера.
innodb_buffer_pool_size	Размера буферного пула в ОЗУ для промежуточной записи транзакций к БД.	Это параметр для оптимизации быстродействия, определяющий, сколько ОЗУ может занять буферный пул MariaDB. Разработчики рекомендуют указывать[39] размер пула до 80% от доступного объема ОЗУ. Однако размеры БД OpenStack не столь большие, поэтому в большинстве случаев достаточно указания 2 гигабайт (даже это сверхмного, в идеале размер буфера должен составлять размер всех баз +10-15% сверху).

Настройки HAProxy

ⓘ Легенда таблицы доступна на этой странице[40].

36 https://mariadb.com/kb/en/auto_increment/
37 https://www.percona.com/blog/2017/07/26/what-is-innodb_autoinc_lock_mode-and-why-should-i-care/
38 https://mariadb.com/kb/en/innodb-doublewrite-buffer/
39 https://mariadb.com/kb/en/innodb-buffer-pool/
40 https://conf.tionix.ru/pages/viewpage.action?pageId=205881354#id-Условныеобозначения-table-format-desc

Здесь включены только строки, которые нужно добавить в стандартную конфигурацию (см. стр. 16) HAProxy.

Имя параметра	Описание	Примечания
resolvers	Параметры резолвинга доменных имён	Рекомендуется использовать внутреннюю реализацию резолвинга DNS[41] вместо системной на базе libc.
parse-resolv-conf	Добавление всех серверов DNS, указанных в /etc/resolv.conf, в список nameservers для HAProxy.	
accepted_payload_size	Указание разрешённого размера тела запроса DNS.	Необходимо для включения EDNS[42].

Установка сервиса memcached

memcached[43] - это простой сетевой сервис кэширования данных. Многие сервисы OpenStack могут кэшировать часть данных в этом сервисе, в частности, проекты Keystone и Nova. memcached хранить информацию в оперативной памяти в виде хеш-таблицы[44]. Кэширование на диск не поддерживается.

Сервис использует порт 11211/TCP.

Установка сервиса

Установка memcached тривиальна.

1. Установите пакет memcached и библиотеку для Python:

```
dnf -y install memcached python3-memcached
```

Стандартные пути до конфигурации:

- `/etc/sysconfig/memcached` - основной файл конфигурации.

2. В основном файле конфигурации кажите адрес mgmt-интерфейса в параметры прослушивания:

```
OPTIONS="-l 127.0.0.1,::1,controller"
```

Дополнительные адреса можно указывать через запятую.

3. Запустите сервис memcached и добавьте его в автозапуск:

```
systemctl start memcached.service
systemctl enable memcached.service
```

У memcached нет встроенных средств обеспечения HA. В продуктивных средах можно использовать несколько отдельных memcached-серверов и указать их адреса в конфигурацию сервисов OpenStack. В референсной архитектуре memcached запускается в единичном экземпляре в своём контейнере и функции условной отказоустойчивости обеспечивается созданием нового экземпляра контейнера с сервисом при проблемах со старым.

Проверка работы сервиса

1. Проверьте статус юнита memcached:

41 https://www.haproxy.com/blog/dns-service-discovery-haproxy/
42 https://ru.wikipedia.org/wiki/EDNS
43 https://memcached.org/
44 https://ru.wikipedia.org/wiki/%D0%A5%D0%B5%D1%88-%D1%82%D0%B0%D0%B1%D0%BB%D0%B8%D1%86%D0%B0

```
systemctl status memcached
```

В ответ на команду вы должны получить примерно следующий вывод:

```
● memcached.service - memcached daemon
   Loaded: loaded (/usr/lib/systemd/system/memcached.service; enabled;
vendor preset: disabled)    <---- Юнит должен иметь статус enabled (не в
 vendor preset)
  Drop-In: /run/systemd/system/memcached.service.d
           └─zzz-lxc-service.conf
   Active: active (running) since Sun 2021-11-07 21:02:31 UTC; 7s ago
<---- Должен быть статус active (running)
 Main PID: 733 (memcached)
    Tasks: 10 (limit: 204240)
   Memory: 1.6M
   CGroup: /system.slice/memcached.service
           └─733 /usr/bin/memcached -p 11211 -u memcached -m 64 -c 1024 -l
127.0.0.1,::1,controller

Nov 07 21:02:31 tnx-mgmt-almalinux systemd[1]: Started memcached daemon.
```

2. Проверьте статус порта:

```
ss -tnlp | grep 11211
```

В ответ на команду вы должны получить примерно следующий вывод:

```
LISTEN 0      1024   10.236.64.231:11211      0.0.0.0:*    users:
(("memcached",pid=733,fd=28))
LISTEN 0      1024       127.0.0.1:11211      0.0.0.0:*    users:
(("memcached",pid=733,fd=26))
LISTEN 0      1024           [::1]:11211         [::]:*    users:
(("memcached",pid=733,fd=27))
```

Количество LISTEN-портов должно совпадать с количеством адресов, указанных в OPTIONS основного файла конфигурации сервиса.

3. Подключитесь к memcached-серверу через утилиту netcat и вызовите команду version после установления соединения с сервисом:

```
nc -v controller 11211
------ После установления соединения ------
version
```

В ответ на команду вы должны получить примерно следующий вывод:

```
nc -v controller 11211
Ncat: Version 7.70 ( https://nmap.org/ncat )
Ncat: Connected to 10.236.64.231:11211.
version
VERSION 1.5.22
quit
```

Установка сервиса RabbitMQ

Введение

OpenStack состоит из множества сервисов, которые должны обмениваться между собой информацией. Для этого они используют протокол AQMP в лице сервера RabbitMQ.

RabbitMQ[45] - это проект с открытым исходным кодом, реализацию сетевой сервис обмена сообщениями в режиме Publisher-Subscriber (Pub/Sub)[46]. Выделяется хорошей производительностью, достаточно простой конфигурацией и доступен во всех известных дистрибутивах.

RabbitMQ использует порт 5672/TCP[47].

Установка сервиса

RabbitMQ доступен из официальных репозиториев

1. Установите пакет с сервисом:

```
dnf -y install rabbitmq-server
```

> **Информация**
>
> Стандартные пути до конфигурации:
>
> - `/etc/rabbitmq` - каталог конфигурации;
> - `/etc/rabbitmq/rabbitmq.conf` - основной файл конфигурации.
>
> Эти пути могут отсутствовать в файловой системе. В этом случае их можно создать с правами для пользователя rabbitmq:
>
> ```
> mkdir /etc/rabbitmq
> > /etc/rabbitmq/rabbitmq.conf
> chown -R rabbitmq:rabbitmq /etc/rabbitmq
> ```

2. RabbitMQ по умолчанию слушает все доступные сетевые интерфейсы. В продуктивных средах необходимо слушать только mgmt-интерфейс, указав IP-адрес узла в этой сети:

```
listeners.tcp.1 = 10.0.0.11:5672
```

> Примечания:
>
> - В референсной архитектуре этот параметр не требуется, так как сетевой доступ до RabbitMQ контролируется контейнерной виртуализацией.
> - Не используйте доменные имена для listeners, иначе вы получите ошибку запуска.

3. Запустите сервис и добавьте его в автозапуск:

```
systemctl start rabbitmq-server.service
systemctl enable rabbitmq-server.service
```

Проверка работы сервиса

1. Проверьте статус юнита memcached:

```
systemctl status rabbitmq-server
```

45 https://www.rabbitmq.com/

46 https://ru.wikipedia.org/wiki/%D0%98%D0%B7%D0%B4%D0%B0%D1%82%D0%B5%D0%BB%D1%8C-%D0%BF%D0%BE%D0%B4%D0%BF%D0%B8%D1%81%D1%87%D0%B8%D0%BA_(%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)

47 https://ru.adminsub.net/tcp-udp-port-finder/5672

В ответ на команду вы должны получить примерно следующий вывод:

```
● rabbitmq-server.service - RabbitMQ broker
   Loaded: loaded (/usr/lib/systemd/system/rabbitmq-server.service;
enabled; vendor preset: disabled)    <---- Юнит должен иметь статус enabled
(не в vendor preset)
  Drop-In: /run/systemd/system/rabbitmq-server.service.d
           └─zzz-lxc-service.conf
   Active: active (running) since Sun 2021-11-07 21:13:32 UTC; 56s ago
<---- Должен быть статус active (running)
 Main PID: 1955 (beam.smp)
   Status: "Initialized"    <---- Должен быть статус "Initialized"
    Tasks: 91 (limit: 204240)
   Memory: 74.2M
   CGroup: /system.slice/rabbitmq-server.service
           ├─1955 /usr/lib64/erlang/erts-10.6.4/bin/beam.smp -W w -A 64
 -MBas ageffcbf -MHas ageffcbf -MBlmbcs 512 -MHlmbcs 512 -MMmcs 30 -P
1048576 -t 5000000 -stbt db -zdbbl 128000 >
           ├─2057 /usr/lib64/erlang/erts-10.6.4/bin/epmd -daemon
           ├─2211 erl_child_setup 1024
           ├─2268 inet_gethost 4
           └─2269 inet_gethost 4
```

2. Проверьте статус порта RabbitMQ:

```
ss -tnlp | grep 5672
```

В ответ на команду вы должны получить примерно следующий вывод:

```
LISTEN 0      128    10.236.64.231:5672       0.0.0.0:*    users:
(("beam.smp",pid=1955,fd=90))
LISTEN 0      128          0.0.0.0:25672      0.0.0.0:*    users:
(("beam.smp",pid=1955,fd=77))
```

Портов должно быть как минимум 2:

- 5672 - требуется для доступа к RabbitMQ сервисами облачной платформы;
- 25672 - требуется для общения между инстансами RabbitMQ (функции кластеризации) и для доступа к управлению сервиса через CLI. Номер порта генерируется увеличением основного порта на число 20000 (5672+20000=25672)

Подробнее о сетевых настройках RabbitMQ можно узнать здесь[48].

3. Проверьте статус порта сервиса epmd:

```
ss -tnlp | grep 4369
```

В ответ на команду вы должны получить примерно следующий вывод:

```
LISTEN 0      128          0.0.0.0:4369       0.0.0.0:*    users:(("epmd",p
id=2057,fd=3))
LISTEN 0      128             [::]:4369          [::]:*    users:(("epmd",p
id=2057,fd=4))
```

epmd по умолчанию прослушивает все интерфейсы.

Подробнее о сетевых настройках RabbitMQ можно узнать здесь[49].

4. Получите данные состояния RabbitMQ:

```
rabbitmqctl status
```

48 https://www.rabbitmq.com/networking.html
49 https://www.rabbitmq.com/networking.html

В ответ на команду вы должны получить примерно следующий вывод:

```
Status of node rabbit@tnx-mgmt-almalinux ...
Runtime

OS PID: 1955
OS: Linux
Uptime (seconds): 607
RabbitMQ version: 3.8.3
...
```

Создание начальных объектов в RabbitMQ

Для сервисов OpenStack и TIONIX должны использоваться свои учетные записи для доступа к сервису RabbitMQ

Более подробно об аутентификации в RabbitMQ можно узнать по этой ссылке[50].

1. Создайте пользователя openstack и предоставьте все права:

```
rabbitmqctl add_user openstack RABBIT_PASS
rabbitmqctl set_permissions openstack ".*" ".*" ".*"
```

2. Для сервисов TIONIX настройка чуть более сложная. Вначале создайте пользователя и виртуальный узел[51] tionix:

```
rabbitmqctl add_user tionix TIONIX_RABBIT_PASS
rabbitmqctl add_vhost tionix
```

Новый виртуальный узел RabbitMQ необходим для разделения очереди сообщений сервисов TIONIX от сервисов OpenStack, которые будут использовать виртуальный узел "/".

3. Предоставьте пользователю tionix необходимые права:

```
rabbitmqctl set_permissions -p tionix tionix ".*" ".*" ".*"
rabbitmqctl set_permissions tionix ".*" ".*" ".*"
```

Готово, настройка завершена.

Старайтесь не использовать классические HA-функции RabbitMQ. Практика показала, что они работают неустойчиво и из-за репликации всех сообщений между экземплярами кластера сильно снижается скорость обработки очередей. В референсной архитектуре RabbitMQ запускается в единственном экземпляре и условная отказоустойчивость гарантируется запуском нового экземпляра контейнера с этим сервисом при проблемах со старым. В продуктивных средах без контейнерной виртуализации можно использовать Pacemaker.

В будущем планируется использовать новый вариант кластеризации для RabbitMQ на базе протокола Raft.

50 https://www.rabbitmq.com/access-control.html
51 https://www.rabbitmq.com/vhosts.html

Установка и настройка служб OpenStack

OpenStack Keystone

Информация о сервисе Keystone

Обзор сервиса

OpenStack Keystone[52] (далее — просто Keystone) — это сервис предоставления функций аутентификации и авторизации, а также хранения данных всех сервисов OpenStack, зарегистрированных в облачной платформе.

Keystone - это первый сервис, с которым пользователь столкнётся при взаимодействии с облачной платформой. После аутентификации пользователь сможет использовать все доступные функции других сервисов. Сервисы OpenStack могут использовать функции Keystone для аутентификации и получения прав на выполнение тех или иных операций, необходимых пользователю. Keystone может использовать как внутреннюю базу данных пользователей, так и использовать внешние сервисы (например, серверы на базе протокола LDAP).

Keystone может предоставить пользователю информацию о том, какие сервисы OpenStack зарегистрированы в облачной платформе. У каждого сервиса могут иметься различные точки входа (endpoint) - адреса сервисов, по которым они будут доступны. Каждая точка доступа может быть доступна через отдельную сеть, что позволяет разделять трафик различных уровней доступа до облачной платформы.

По умолчанию доступны три типа точек входа:

- **admin API network** — эта точка входа предоставляет доступ к функциям управления облачной платформы и предназначен для администраторов.
- **public API network** — рассчитан на возможность предоставления функций облачной платформы пользователя через Интернет.
- **internal API network** — предназначен для межсервисного общения между компонентами облачной платформы.

Для повышения масштабируемости Keystone поддерживает функцию разделения на регионы, которые будут содержать точки входа в сервисы своих облачных платформ. В данном руководстве для упрощения принято, что Keystone содержит один регион, который называется **RegionOne**, а в нём для каждого сервиса регистрируется три точки входа с разным уровнем доступности функций.

Установка Keystone обязательна и требуется для всех остальных сервисов OpenStack, поэтому его нужно установить в первую очередь.

Сам сервис Keystone состоит из трёх больших частей:

- **Сервер Keystone** — это централизованная служба, которая предоставляет функции аутентификации и авторизации через интерфейс RESTful.
- **Драйверы** — это специальные компоненты сервиса Keystone, обеспечивающие поддержку различных внешних сервисов по отношению к OpenStack, например, LDAP-серверов.
- **Модули Keystone** (иначе middleware modules) — это части сервиса Keystone, которые выполняют какую-либо часть работы: принятие и обработка запроса, выделение данных пользователя из него, генерирование данных авторизации, например, токенов, и так далее. Интеграция между этими модулями и другими компонентами OpenStack осуществляется через пограничный интерфейс веб-сервера Python (WSGI).

Краткий список терминов

В Keystone принята своя терминология для тех или иных типов данных.

По обхвату сверху вниз объекты Keystone разделяются на следующие типы:

- **Регион (Region)** — это самая крупная абстракция, обозначающая отдельную облачную платформу со своими сервисами и инфраструктурой. Иными словами, Keystone через регионы может быть провайдером аутентификации и авторизации сразу для нескольких облачных платформ. По умолчанию первый регион в Keystone называется RegionOne.

52 https://docs.openstack.org/keystone/victoria/

- **Домен (Domain)** — это абстракция, разделяющая крупные списки пользователей, групп и проектов между собой. В общем случае домены привязываются к конкретному источнику данных о пользователях (отдельная база данных SQL или, например, сервер LDAP. Подробнее здесь).
- **Проект (Project)** — это абстракция, включающая разделение пользователей и групп между собой по каким-либо критериям. Так же на этом уровне начинает работать ролевая модель OpenStack, так как конкретный пользователь регистрируется в облаке с указанной ролью.
- **Пользователь (User)** — это уникальный объект физического пользователя, с помощью которого он может войти в облачную платформу и получить соответствующие права в рамках какого-либо проекта.
- **Роль (Role**) — абстракция для пользователя, позволяющая определить ограничения по разрешенным функциям в облачной платформе в рамках какого-либо проекта. При регистрации пользователя в проекте с какой-либо ролью создает так называемую "ролевую привязку" (role assignment). Подробнее об этом здесь[53].
- **Группа (Group)** — обособленная абстракция, объединяющая пользователей по какому-либо признаку. Могут быть включены в проект с указанием общей роли для всех пользователей в проекте.

В Keystone есть объекты вне этой иерархии:

- **Пароль (Password)** — это уникальный идентификатор для конкретного пользователя, предназначенный для его аутентификации.
- **Токен (Token)** — это генерируемый после успешного процесса аутентификации и авторизации уникальный идентификатор, привязанный к пользователю в проекте. Токен позволяет узнать любому сервису OpenStack, какими правами в проекте обладает авторизованный пользователь. Токен является временным объектом, через некоторое время токен становится недействительным.
- **Сервис (Service)** — это абстракция с информацией о зарегистрированных сервисах OpenStack в облачной платформе. Сервис содержит точку входа в любой сервис OpenStack. Список сервисов формирует каталог (catalog).

Работа основной функции сервиса

Конечной функцией Keystone с точки зрения пользователя и сервисов OpenStack является генерирование токена авторизации, предоставляющий доступ к части или ко всем функциями облачной платформы, и его дальнейшая проверка на валидность. Для получения такого токена должно пройти несколько этапов работы сервиса Keystone. Кратко опишем их.

- Пользователь для входа в платформу (не важно, через веб-интерфейс или напрямую через API) использует три типа данных:
  - **Имя домена**, где находится проект или проекты, в которых пользователь зарегистрирован.
  - **Имя пользователя**.
  - **Пароль пользователя**.
- После запроса на вход Keystone, используя полученные данные, делает запрос в свою внутреннюю базу данных или во внешнюю систему хранения данных пользователей для проверки наличия пользователя.
- Keystone получает ответ на этот запрос. При отрицательном ответе (нет такого пользователя, неверный пароль, неверно указан домен и т. д.), запрос отклоняется и пользователь получит сообщение о невозможности входа.
- Если запрос возвращает положительный ответ, то Keystone обращается к провайдеру токенов. На этом шаге заканчивается процесс аутентификации и начинается процесс авторизации.
- Keystone проверяет роль пользователя, после чего обращается к провайдеру токенов авторизации. Если пользователь не имеет необходимых прав в рамках своей роли, то на этом шаге может возникнуть ошибка.
- Провайдер токена генерирует уникальный идентификатор и передает его сервису Keystone.
- Сервис Keystone перенаправляет токен пользователю.
- Пользователь передаёт этот токен сервисам OpenStack.
- Сервисы OpenStack проверяют валидность токена, после чего предоставляют доступные функции пользователю. Процесс авторизации на этом заканчивается. Токены fernet содержат всю основную информацию в самом теле токена, поэтому сервису Keystone нет необходимости производить отзыв токена.
- При окончании времени жизни токена весь этот процесс повторяется с самого начала.

Дополнительные материалы

- Главная страница[54] официальной документации проекта.
  - Подробная архитектура сервиса[55].
- Официальный репозиторий[56] проекта.

53 https://conf.tionix.ru/pages/viewpage.action?pageId=164102171
54 https://docs.openstack.org/keystone/victoria/
55 https://docs.openstack.org/keystone/victoria/getting-started/architecture.html
56 https://opendev.org/openstack/keystone

- Описание API[57] сервиса.

Установка сервиса Keystone

Установка сервиса Keystone состоит из нескольких шагов.

Настройка окружения

Перед самой установкой сервиса нужно предварительно настроить некоторые компоненты инфраструктуры.

Подготовка базы данных keystone

См. также: Установка и настройка СУБД MariaDB (см. стр. 22).

Всю информацию о данных для аутентификации и авторизации по умолчанию Keystone хранит в базе данных MariaDB.

1. Войдите в окружение базы данных:

```
mysql -u root -p
```

2. Создайте базу данных keystone:

```
create database keystone;
```

3. Предоставьте доступ к этой базе данных пользователю keystone в СУБД (для localhost и всем остальным адресам отдельно, вместо KEYSTONE_DBPASS используйте свой пароль):

```
grant all privileges on keystone.* to 'keystone'@'localhost' identified by
'KEYSTONE_DBPASS';
grant all privileges on keystone.* to 'keystone'@'%' identified by
'KEYSTONE_DBPASS';
```

4. Выйдите из сессии СУБД:

```
exit;
```

Установка зависимых пакетов

Для своей работы Keystone использует несколько внешних компонентов:

- Apache[58] - веб-сервер в качестве реализации HTTP-протокола;
- **модуль WSGI mod_wsgi** для Apache - для поддержки пограничного интерфейса веб-сервера, позволяющий запустить Keystone как сетевой сервис;
- **модуль SSL mod_ssl** для Apache - для поддержки шифрования соединений до сервиса Keystone по протоколу TLS;
- memcached[59] - сервис кэширования объектов для ускорения часто используемых запросов в Keystone. Установка описана здесь (см. стр. 28).

Apache и модули

1. Установите пакеты веб-сервера Apache и модулей к нему:

```
dnf -y install httpd python3-mod_wsgi mod_ssl
```

2. Добавьте сервис Apache в автозапуск:

57 https://docs.openstack.org/api-ref/identity/v3/index.html
58 https://httpd.apache.org/
59 https://memcached.org/

```
systemctl enable httpd
```

3. Укажите имя сервера Apache в конфигурационном файле `/etc/httpd/conf/httpd.conf`. Желательно использовать фактический адрес DNS управляющего узла или имя машины, где установлен Apache:

```
ServerName controller
```

Установка сервиса Keystone

> См. также: Настройка репозиториев Almalinux (см. стр. 10).

После установки всех необходимых внешних сервисов можно приступить к установке самого Keystone.

1. Установите пакет сервиса:

```
dnf -y install openstack-keystone
```

> Стандартные пути файлов конфигурации:
>
> - `/etc/keystone` – Каталог конфигурации Keystone;
> - `/etc/keystone/keystone.conf` – основной файл конфигурации.

2. Очистите основной файл конфигурации, который был добавлен после установки пакета:

```
> /etc/keystone/keystone.conf
```

3. В основной файл добавьте следующую конфигурацию (описание (см. стр. 48)):

```
[DEFAULT]
debug = False
transport_url = rabbit://openstack:RABBIT_PASS@controller:5672
log_dir = /var/log/keystone/keystone.log
#use_stderr = True

[oslo_middleware]
enable_proxy_headers_parsing = True

[database]
connection = mysql+pymysql://keystone:KEYSTONE_DBPASS@controller/keystone
connection_recycle_time = 10
max_pool_size = 1
max_retries = -1

[oslo_messaging_notifications]
driver = messagingv2
transport_url = rabbit://openstack:RABBIT_PASS@controller:5672

[identity]
driver = sql
domain_specific_drivers_enabled = True
domain_config_dir = /etc/keystone/domains
caching = True

[token]
revoke_by_id = False
provider = fernet
expiration = 86400

[cache]
backend = oslo_cache.memcache_pool
enabled = True
servers = controller:11211
```

4. При настройке службы обратите внимание на следующие параметры:
   a. Настройки сервиса RabbitMQ в параметре **DEFAULT/transport_url** и **oslo_messaging_notifications/transport_url**;

b. Параметры журналирования сервиса (см. стр. 49), которые зависят от метода установки;
c. Параметры подключения к базе данных в параметре **database/connection**;
d. Параметры сервиса кэширования в параметре **cache/servers**.

5. Инициализируйте базу данных keystone:

```
su -s /bin/sh -c "keystone-manage db_sync" keystone
```

6. При использовании провайдера fernet (по умолчанию) необходимо произвести его настройку:

```
keystone-manage fernet_setup --keystone-user keystone --keystone-group keystone
keystone-manage credential_setup --keystone-user keystone --keystone-group
keystone
```

> Эти команды создадут Primary и Secondary ключи в `/etc/keystone/fernet-keys`, позволяющие генерировать токены этого формата.

7. Инициализируйте сервис Keystone созданием пользователя admin и стандартных точек входа API сервиса:

```
keystone-manage bootstrap \
  --bootstrap-admin-url http://controller:5000/v3/ \
  --bootstrap-internal-url http://controller:5000/v3/ \
  --bootstrap-public-url http://controller:5000/v3/ \
  --bootstrap-region-id RegionOne \
  --bootstrap-password ADMIN_PASS
```

> Вместо ADMIN_PASS наберите пароль для пользователя admin. Укажите свои DNS-имена, если они отличаются от имени "controller". При наличии кластера укажите общее DNS-имя.

> ⚠ Для разных типов точек входа API можно указать разные адреса. Это может быть необходимо в случае разделения, например, публичного трафика от внутреннего. Использование IP-адресов не рекомендуется.

8. Для включения сервиса Keystone в веб-сервере нужно создать символическую ссылку в каталоге конфигурации Apache:

```
ln -s /usr/share/keystone/wsgi-keystone.conf /etc/httpd/conf.d/
```

Финализация установки

1. Перезапустите сервис Apache:

```
systemctl restart httpd
```

> Если ранее был установлен и запущен HAProxy, но его настройка ещё не была осуществлена, то перезапуск httpd в этом шаге будет неудачным. Причина в том, что в конфигурации HAProxy по умолчанию настраивается фронтенд с открытием порта 5000. В этом случае удалите этого фронтенд из конфигурации /etc/haproxy/haproxy.cfg, перезапустите HAProxy, после чего перезапустите httpd.

Проверка работы сервиса

1. Проверьте наличие открытого порта 5000:

```
ss -tnlp | grep 5000
```

Ответ должен быть примерно таким:

```
LISTEN 0      128                *:5000             *:*    users:(("httpd",
pid=20299,fd=11),("httpd",pid=20298,fd=11),("httpd",pid=20297,fd=11),
("httpd",pid=20289,fd=11))
```

По умолчанию Keystone слушает все интерфейсы.

2. Проверьте, что по этому порту отвечает сервис Keystone:

```
curl http://controller:5000/v3
```

Ответ должен быть примерно таким:

```
{"version": {"id": "v3.14", "status": "stable", "updated":
"2020-04-07T00:00:00Z",...
```

Файл настройки системного окружения

1. Перед тем, как продолжить работу над сервисом с помощью клиента для командной строки *openstack* создайте файл с настройками окружения для более простого входа в платформу. В корневом каталоге вашего пользователя (обычного или root) в операционной системе создайте файл `$HOME/admin-openrc` со следующим содержимым (описание (см. стр. 51)):

```
export OS_USERNAME=admin
export OS_PASSWORD=ADMIN_PASS
export OS_PROJECT_NAME=admin
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_DOMAIN_NAME=Default
export OS_AUTH_URL=http://controller:5000/v3
export OS_IDENTITY_API_VERSION=3
```

Вместо ADMIN_PASS укажите пароль пользователя admin, который хранится в сервисе Keystone.

2. Для применения этих настроек запустите следующую команду:

```
source $HOME/admin-openrc
```

3. Эту команду необходимо выполнять каждый раз при входе в облачную платформу. Чтобы автоматизировать это, добавьте эту команду в файл $HOME/.bashrc:

```
echo 'source "${HOME}/admin-openrc"' >> "${HOME}/.bashrc"
```

4. Перезайдите в сессию командной строки для принятия изменений.
5. Такой файл можно создать для любого другого пользователя, если необходимо войти под ним. К примеру, для пользователя user (см. стр. 39):

```
export OS_USERNAME=user
export OS_PASSWORD=USER_PASS
export OS_PROJECT_NAME=user-project
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_DOMAIN_NAME=Default
export OS_AUTH_URL=http://controller:5000/v3
export OS_IDENTITY_API_VERSION=3
```

Вместо USER_PASS укажите пароль пользователя (в примере - user), который хранится в сервисе Keystone.

6. Для применения этих настроек запустите следующую команду:

```
source $HOME/user-openrc
```

7. Вы можете переключаться между пользователя, просто исполняя команду source со соответствующим файлом. Старые параметры при применении нового файла окружения перезаписываются.
8. После применения файла окружения получите список пользователей в Keystone:

```
openstack user list
```

> В ответ вы должны получить список пользователя, который должен содержать пользователя admin:
>
> ```
> +----------------------------------+-------+
> | ID                               | Name  |
> +----------------------------------+-------+
> | e8f0a2078a9548029f7f9025beeec983 | admin |
> +----------------------------------+-------+
> ```

Создание первых объектов

После инициализации и запуска сервиса Keystone необходимо создать несколько новых объектов.

Создание проекта service

1. Обязательным шагом после инициализации и запуска Keystone является создание проекта service в проекте default:

```
openstack project create --domain default --description "Service Project" service
```

> При успешном выполнении команды вы должны получить примерно следующий вывод:
>
> ```
> +-------------+----------------------------------+
> | Field       | Value                            |
> +-------------+----------------------------------+
> | description | Service Project                  |
> | domain_id   | default                          |
> | enabled     | True                             |
> | id          | fbba4d8ef4924ee9852f09d48503a43e |
> | is_domain   | False                            |
> | name        | service                          |
> | options     | {}                               |
> | parent_id   | default                          |
> | tags        | []                               |
> +-------------+----------------------------------+
> ```

> Данный проект предназначен для регистрации сервисных пользователей, с помощью которых сервисы OpenStack будут аутентифицироваться в Keystone. Все эти пользователи должны регистрироваться только в проекте service.

Создание нового домена

1. По умолчанию после инициализации будет создан домен default. Если вы хотите создать новый домен, то выполните следующую команду:

```
openstack domain create --description "A Production Domain" production
```

При успешном выполнении команды вы должны получить следующий вывод:

```
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | A Production Domain              |
| enabled     | True                             |
| id          | ee598d687d744c93869b53175d2ade22 |
| name        | production                       |
| options     | {}                               |
| tags        | []                               |
+-------------+----------------------------------+
```

При использовании домен-специфичных драйверов[60] для серверов LDAP отдельно создавать домены не требуется.

Создание нового проекта

По умолчанию после инициализации Keystone будет доступен проект admin, предназначенный для пользователей с административными правами.

Для обычных задач рекомендуется использовать обычного пользователя, зарегистрированного не в административном проекте. С

1. Для создания проекта в домене default необходимо выполнить команду:

```
openstack project create --domain default --description "User Project" user-
project
```

При успешном выполнении команды вы должны получить следующий вывод:

```
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | User Project                     |
| domain_id   | default                          |
| enabled     | True                             |
| id          | e8cb062fd4b442aa98cffee063247cb6 |
| is_domain   | False                            |
| name        | user-project                     |
| options     | {}                               |
| parent_id   | default                          |
| tags        | []                               |
+-------------+----------------------------------+
```

Создание обычного пользователя и указание роли в проекте

Для обычных задач рекомендуется использовать обычного пользователя, зарегистрированного не в административном проекте.

1. Создайте обычного пользователя:

```
openstack user create --domain default --password-prompt user
```

При выполнении команды интерактивно будет запрошен пароль для пользователя

60 https://conf.tionix.ru/pages/viewpage.action?pageId=164102237#id-Использованиесвязанныхсдоменомсистемхраненияданныхпользователей-ks-domain-specific-drivers

При успешном выполнении команды вы должны получить следующий вывод:

```
+---------------------+----------------------------------+
| Field               | Value                            |
+---------------------+----------------------------------+
| domain_id           | default                          |
| enabled             | True                             |
| id                  | 65d58683983a49d0979652be4ae6d4a2 |
| name                | user                             |
| options             | {}                               |
| password_expires_at | None                             |
+---------------------+----------------------------------+
```

2. Создайте новую роль:

```
openstack role create user-role
```

При успешном выполнении команды вы должны получить следующий вывод:

```
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | None                             |
| domain_id   | None                             |
| id          | 04261b93e23c47fb869f97d0e98dcaf7 |
| name        | user-role                        |
| options     | {}                               |
+-------------+----------------------------------+
```

3. Добавьте пользователя в проект под созданной ролью:

```
openstack role add --project user-project --user user user-role
```

Для предоставления административных парв пользователю можете указать роль admin.

При успешном выполнении команда должна вернуть пустой вывод.

Все создаваемые роли по умолчанию равны роли member. Для создания отдельных правил для новой роли необходимо применить изменения в ролевой модели OpenStack. Подробнее о ролевой модели можно узнать в этой статье[61].

Верификация работы сервиса

После первичной настройки необходимо убедиться, что базовые функции Keystone работают исправно. Для этого достаточно проверить корректность генерации токена.

Проверка генерации токена от роли администратора

1. Настройте параметры окружения (см. стр. 38) для входа в облачную платформу как пользователь admin:

```
source $HOME/admin-openrc
```

2. Выполните команду генерирования и выпуска токена (команда спросит пароль):

```
openstack token issue
```

61 https://conf.tionix.ru/pages/viewpage.action?pageId=164102171

В ответ вы должны получить данные сгенерированного токена:

```
+------------
+-------------------------------------------------------------------+
| Field      | Value
|
+------------
+-------------------------------------------------------------------+
| expires    | 2016-02-12T20:14:07.056119Z
|
| id         | gAAAAABWvi7_B8kKQD9wdXac8MoZiQldmjEO643d-e_j-
XXq9AmIegIbA7UHGPv |
|            |
atnN21qtOMjCFWX7BReJEQnVOAj3nclRQgAYRsfSU_MrsuWb4EDtnjU7HEpoBb4 |
|            | o6ozsA_NmFWEpLeKy0uNn_WeKbAhYygrsmQGA49dclHVnz-OMVLiyM9ws
|
| project_id | 343d245e850143a096806dfaefa9afdc
|
| user_id    | ac33776331494012 96f6c0d92d79dc16
|
+------------
+-------------------------------------------------------------------+
```

Проверка генерации токена от роли пользователя

Отдельно необходимо проверить генерацию токена пользователем, имеющий роль обычного пользователя.

1. Настройте параметры окружения (см. стр. 38) для входа в облачную платформу как пользователь без административных прав (в примере - user):

```
source $HOME/user-openrc
```

2. Выполните команду генерирования и выпуска токена (команда спросит пароль):

```
openstack token issue
```

3. В ответ вы должны получить данные сгенерированного токена:

```
+------------
+-------------------------------------------------------------------+
| Field      | Value
|
+------------
+-------------------------------------------------------------------+
| expires    | 2016-02-12T20:14:07.056119Z
|
| id         | gAAAAABWvi7_B8kKQD9wdXac8MoZiQldmjEO643d-e_j-
XXq9AmIegIbA7UHGPv |
|            |
atnN21qtOMjCFWX7BReJEQnVOAj3nclRQgAYRsfSU_MrsuWb4EDtnjU7HEpoBb4 |
|            | o6ozsA_NmFWEpLeKy0uNn_WeKbAhYygrsmQGA49dclHVnz-OMVLiyM9ws
|
| project_id | 343d245e850143a096806dfaefa9afdc
|
| user_id    | ac33776331494012 96f6c0d92d79dc16
|
+------------
+-------------------------------------------------------------------+
```

Шифрование сервиса

Сервис Keystone является публичным сервисом с критичными для платформы функциями, поэтому в продуктивных средах обязательно включения шифрования запросов API протоколом TLSv1.2 и выше.

В отличие от остальных (см. стр. 21) сервисов, шифрование Keystone настраивается на уровне веб-сервера Apache. Такое решение было принято потому, что такой способ является нативным для сервиса. Для остальных сервисов OpenStack использование SSL Termination[62]является обходным путём проблем шифрования на уровне библиотек Python.

Включение протокола TLS в веб-сервере

1. В референсной архитектуре для включения шифрования в веб-сервере Apache используется следующая конфигурация, которая находится по пути `/etc/httpd/conf.d/wsgi-keystone.conf` (описание (см. стр. 45)):

62 https://conf.tionix.ru/pages/viewpage.action?pageId=174030874#id-Глоссарий-lb-ssl-term

```
ServerName controller
ServerRoot "/etc/httpd"
Include conf.modules.d/*.conf
User apache
Group apache
LogLevel warn
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
ErrorLog /dev/stderr
CustomLog /dev/stdout combined
TypesConfig /etc/mime.types
AddDefaultCharset UTF-8
EnableSendfile on

<Directory />
  AllowOverride none
  Require all denied
</Directory>

SSLProtocol all -SSLv3 -TLSv1 -TLSv1.1
SSLCipherSuite ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-
ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-
POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-
SHA384
SSLHonorCipherOrder off
SSLSessionTickets off

Listen 5000

<VirtualHost *:5000>
  WSGIDaemonProcess keystone-public processes=5 threads=1 user=keystone
group=keystone display-name=%{GROUP}
  WSGIProcessGroup keystone-public
  WSGIScriptAlias / /usr/bin/keystone-wsgi-public
  WSGIApplicationGroup %{GLOBAL}
  WSGIPassAuthorization On
  LimitRequestBody 114688
  ErrorLogFormat "%{cu}t %M"
  ErrorLog /dev/stderr
  CustomLog /dev/stdout combined
  SSLEngine on
  SSLCertificateFile certs/cert.pem
  SSLCertificateKeyFile certs/privkey.pem
  Protocols h2 http/1.1
  <Directory /usr/bin>
    Require all granted
  </Directory>
</VirtualHost>

Alias /identity /usr/bin/keystone-wsgi-public
<Location /identity>
    SetHandler wsgi-script
    Options +ExecCGI
    WSGIProcessGroup keystone-public
    WSGIApplicationGroup %{GLOBAL}
    WSGIPassAuthorization On
</Location>
```

2. После принятия этой конфигурации необходимо перезапустить сервер Apache:

```
systemctl restart httpd
```

3. После этого убедитесь, что шифрованные соединения доступны по порту 5000, например, командой curl:

```
curl -v https://keystone.{domain}:5000
```

Если сертификат самоподписанный, то укажите:

- параметр -k, в этом случае верификация сертификата будет выключена;
- или параметр --capath до файла CA вашего центра сертификации.

Функционирование продуктивных сред с самоподписанными сертификатами официально не поддерживается.

Таблица конфигурации

Легенда таблицы доступна на этой странице[63].

Имя параметра	Описание	Примечания
ServerName	Имя сервера Apache	В референсной архитектуре ServerName генерируется во время деплоя контейнера и выглядит как keystone.$K8S_DOMAIN. При обычной установке ServerName нужно указать вручную, обычно он совпадает с именем узла.
ServerRoot	Путь до корневого каталога веб-сервера.	Не меняйте этот параметр без необходимости.
Include	Включение дополнительных файлов конфигурации, расположенных по указанному пути.	
User	Пользователь операционной системы, от имени которого будут запущены процессы веб-сервера Apache.	
Group	Группа операционной системы, от имени которой будут запущены процессы веб-сервера Apache.	
LogFormat	Формат журналирования событий.	
ErrorLog	Путь для хранения журналов ошибок.	Есть три варианта использования: • **/dev/stderr** - для референсной архитектуры, перенаправит журналы в систему управления контейнерами. • **/var/log/httpd/keystone-error.log** - для обычной установки, перенаправит журнал по указанному пути. • **journald** - для обычной установки, перенаправит журнал в сервис journald. Требуется включение модуля mod_journald[64].

63 https://conf.tionix.ru/pages/viewpage.action?pageId=205881354#id-Условныеобозначения-table-format-desc
64 https://httpd.apache.org/docs/trunk/mod/mod_journald.html

Имя параметра	Описание	Примечания
CustomLog	Путь для хранения журналов запросов к веб-серверу.	Есть три варианта использования: • **/dev/stdout** - для референсной архитектуры, перенаправит журналы в систему управления контейнерами. • **/var/log/httpd/keystone-custom.log** - для обычной установки, перенаправит журнал по указанному пути. • **journald** - для обычной установки, перенаправит журнал в сервис journald (журналы будут доступны юниту httpd). Требуется включение модуля mod_journald[65].
SSLProtocol	Указание поддерживаемых версий протоколов TLS/SSL.	Необходимо явно выключить все старые версии SSL и TLS (до версии 1.1 включительно). По умолчанию остаются версии протокола TLS v1.2+.
SSLCipherSuite	Список доступных алгоритмов шифрования.	Данный список был составлен по определенным критериям, в основном, по итоговой производительности и уровню безопасности в целом. Вначале используются EC[66]-варианты AES[67] в режиме GCM[68] и возможностью аппаратного ускорения (через AES-NI), затем ChaCha20-Poly1305 как эффективный программный алгоритм, и в конце списка — более классический AES с использованием протокола Диффи-Хелмана[69] в качестве legacy-варианта.
SSLHonorCipherOrder	Ограничение поддержки алгоритмов шифрования списков SSLCipherSuite.	При включении этого параметра клиент может использовать только те алгоритмы, которые указаны в **SSLCipherSuite**.
SSLSessionTickets	Включение поддержки тикетов сессий TLS (session tickets).	Session ticket является механизмом оптимизации процесса хендшейка, когда клиент может передать session ticket с данными прошлой сессии серверу и сразу, в один запрос установить соединение. Этот параметр действителен только для TLS v1.2, в TLS v1.3+ используется свой механизм session ticket. В случае TLS v1.2 из-за наличия проблем[70] эту директиву следует выключить.
Listen	Адрес и порт прослушивания для виртуального узла Apache.	Через указанный порт будет доступно WSGI-приложение с API сервиса Keystone. По умолчанию порт равен 5000. ⚠ При указании только порта Apache будет слушать его во всех доступных интерфейсах. Если необходимо открыть этот порт только в определённом интерфейсе, то явно укажите адрес: • Listen controller:5000

65 https://httpd.apache.org/docs/trunk/mod/mod_journald.html
66 https://ru.wikipedia.org/wiki/%DO%AD%DO%BB%DO%BB%DO%B8%DO%BF%D1%82%DO%B8%D1%87%DO%B5%D1%81%DO%BA%DO%BO%D1%8F_%DO%BA%D1%8O%DO%B8%DO%BF%D1%82%DO%BE%DO%B3%D1%8O%DO%BO%D1%84%DO%B8%D1%8F
67 https://ru.wikipedia.org/wiki/AES_(%D1%81%D1%82%DO%BO%DO%BD%DO%B4%DO%BO%D1%8O%D1%82_%D1%88%DO%B8%D1%84%D1%8O%DO%BE%DO%B2%DO%BO%DO%BD%DO%B8%D1%8F)
68 https://ru.wikipedia.org/wiki/Galois/Counter_Mode
69 https://ru.wikipedia.org/wiki/%DO%9F%D1%8O%DO%BE%D1%82%DO%BE%DO%BA%DO%BE%DO%BB_%DO%94%DO%B8%D1%84%D1%84%DO%B8_%E2%8O%94_%DO%A5%DO%B5%DO%BB%DO%BB%DO%BC%DO%BO%DO%BD%DO%BO
70 https://blog.filippo.io/we-need-to-talk-about-session-tickets/

Имя параметра	Описание	Примечания
VirtualHost	Директива описания виртуального узла Apache.	В этой секции описываются параметры процесса, реализующий WSGI-приложение.
WSGIDaemonProcess	Имя запускаемого процесса	В этом параметре указываются параметры запуска сервиса Keystone как WSGI-приложения.
WSGIProcessGroup	Группа процессов WSGI, к которому относится WSGI-приложение с сервисом Keystone.	
WSGIScriptAlias	Ссылка на путь до WSGI-приложения.	В данном случае указывается, по какому пути URL Apache должен будет обратиться к WSGI-приложению.
WSGIApplicationGroup	Группа приложений WSGI, к которому относится WSGI-приложение с сервисом Keystone.	
WSGIPassAuthorization	Включение обработки заголовков авторизации в запросах HTTP.	
LimitRequestBody	Максимальный размер запроса HTTP (в байтах).	
ErrorLogFormat	Формат журнала ошибок.	
SSLEngine	Включение режима шифрования соединений для виртуального узла Apache.	
SSLCertificateFile	Путь до файла сертификата для шифрования соединения. Обязателен, если указан **SSLEngine on**.	Можно указать: • абсолютный путь с указанием "/" в начале пути. • относительный путь, который начинается с пути **ServerRoot**. В этом случае "/" в начале пути не указывается. ⚠ Файл сертификата должен быть доступен для чтения процессам Apache.
SSLCertificateKeyFile	Путь до файла ключа к сертификату для шифрования соединения. Обязателен, если указан **SSLEngine on**.	Можно указать: • абсолютный путь с указанием "/" в начале пути. • относительный путь, который начинается с пути **ServerRoot**. В этом случае "/" в начале пути не указывается. ⚠ Файл сертификата должен быть доступен для чтения процессам Apache.
Protocols	Доступные версии протокола HTTP.	ⓘ При указании протокола h2 автоматически включается протокол APLN, позволяющий клиенту узнать, какие версии протокола HTTP поддерживается сервером.

Имя параметра	Описание	Примечания
Directory	Директива доступа к файловой системе для виртуального узла Apache.	Данная директива необходима для того, чтобы WSGI-модуль смог получить доступ к исполняемому файлу сервиса Keystone.
Require	Права доступа к указанному в директиве пути.	В данном случае предоставлятся все права к каталогу /usr/bin.
Alias	Глобальный алиас на ресурс при запросе ссылки HTTP.	В случае сервиса Keystone необходимо создать алиас /identity, откуда по умолчанию должен быть доступен его API.
Location	Параметры ссылки HTTP.	
SetHandler	Указание обработчика при вызове ссылки.	wsgi-script включает обработчик для исполнения WSGI-приложения.
Options	Включение дополнительных параметров при обработке запроса.	

Описание файла конфигурации Keystone

В описании процесса по установке сервиса Keystone предложена конфигурация для сервиса Keystone, а так же настройки входа в сервис. Это страница содержит подробное описание этих настроек.

При изменении конфигурации необходимо перезапустить веб-сервер Apache:

```
systemctl restart httpd
```

Таблица конфигурации

Сервис Keystone

ⓘ Легенда таблицы доступна на этой странице[71].

ⓘ Путь до конфигурации: `/etc/keystone/keystone.conf`

Имя параметра	Описание	Примечания
DEFAULT	Глобальные параметры сервиса Keystone.	
debug	Включение отладочного режима журналирования.	В продуктивных средах используйте только для тестирования и выявления проблем. ⚠ Отладочный режим может повлиять на производительность и конфиденциальность данных.

71 https://conf.tionix.ru/pages/viewpage.action?pageId=205881354#id-Условныеобозначения-table-format-desc

Имя параметра	Описание	Примечания
[oslo_messaging_notifications] **driver** **transport_url**	Включение поддержки уведомлений о событиях в сервисе с использованием RabbitMQ.	Эти параметры необходимы для сервиса Journal (см. стр. 121), входящий в состав модуля TIONIX Client (см. стр. 117). Несколько адресов указываются в transport_url через запятую. Подключения к ним будут происходить последовательно.
use_stderr **log_dir** **use_journal**	Параметры журналирования сервиса.	Необходимо выбрать **один из трёх** указанных параметров: • **use_stderr = True**, если установка *производится в рамках референсной архитектуры*, в этом случае журналы будут перенаправлены в виртуальное устройство /dev/stderr. Это позволяет получить логи системе управления контейнерами. • **log_dir = /var/log/keystone**, если производится *обычная установка*, в этом случае журналы будут сохраняться в файлах в указанном каталоге. • **use_journal = True**, если производится *обычная установка*, в этом случае журналы будут перенаправлены в службу управления журналами systemd-journald.[72] ⚠ Режим use_syslog для перенаправления журналов в службу syslog переведён в разряд устаревших, поэтому официально не поддерживается.
oslo_middleware	Параметры обработки запросов к API сервиса Keystone.	
enable_proxy_headers_parsing	Обработка заголовков прокси-сервера.	⚠ Этот параметр обязателен при использовании балансировщика нагрузки HAProxy.
database	Параметры подключения к базе данных keystone.	
connection	Адрес базы данных keystone.	Можно указать только один адрес.
connection_recycle_time	Время жизни соединений к базе данных, которые имеются в пуле соединений.	
max_pool_size	Максимальный размер пула соединений к базе данных.	
max_retries	Максимальное количество попыток соединений в пуле.	Значение "-1" выключает ограничение по количеству попыток.

72 https://www.freedesktop.org/software/systemd/man/systemd-journald.service.html

Имя параметра	Описание	Примечания
identity	Основные параметры обработки данных аутентификации в Keystone.	
driver	Метод хранения и доступа к данным аутентификации.	driver может иметь два значения для трёх ситуаций: • **"sql"** - в этом случае данные аутентификации будут сохранены в базе данных Keystone в СУБД MariaDB для всех доменов OpenStack. • **"ldap"** - в этом случае данные аутентификации пользователей и групп облачной платформы будут получены с LDAP-сервера в рамках домена "default" по умолчанию. • **"sql"** *и* включение домен-специфичных драйверов[73] - в этом случае при сохранении доменов при отсутствии отдельных конфигураций в Keystone будут сохранены в базе данных, при наличии данные пользователей будут браться с LDAP-сервера. • "**tnx_ldap**" - это расширенный компанией TIONIX вариант драйвера (см. стр. 123) "**ldap**". Может работать совместно с домен-специфичными драйверами[74].
domain_specific_drivers_enabled	Включение домен-специфичных драйверов[75].	Эта функция позволяет для каждого домена OpenStack указать свои данные подключения в LDAP-серверам.
domain_config_dir	Каталог конфигурации домен-специфичных драйверов. Обязателен, если включены домен-специфичные драйверы.	
caching	Включение кэширования для создания и валидации токена.	Требует включения глобальной функции кэширования в секции cache.
token	Параметры генерирования токенов авторизации.	
revoke_by_id	Включение отзыва токена по его ID.	
provider	Алгоритм генерирования токена.	Для продуктивных систем всегда используйте алгоритм fernet. Работа с другими типами токенов в продуктивных системах официально не поддерживается.

73 https://conf.tionix.ru/pages/viewpage.action?pageId=164102237#id-Использованиесвязанныхсдоменомсистемхранениянданныхпользователей-ks-domain-specific-drivers
74 https://conf.tionix.ru/pages/viewpage.action?pageId=164102237#id-Использованиесвязанныхсдоменомсистемхранениянданныхпользователей-ks-domain-specific-drivers
75 https://conf.tionix.ru/pages/viewpage.action?pageId=164102237#id-Использованиесвязанныхсдоменомсистемхранениянданныхпользователей-ks-domain-specific-drivers

Имя параметра	Описание	Примечания
expiration	Время жизни токена.	⚠ В продуктивных системах не используйте слишком большой интервал жизни токена (не больше 24 часов).
cache	Глобальные параметры кэширования сервиса Keystone	
backend	Параметр реализации кэширования. Обязателен, если указан enabled в секции cache.	Может иметь два параметра: • **dogpile.cache.memcached** - рекомендуется для небольших инсталляций. • **oslo_cache.memcache_pool** - рекомендуется для высоконагруженных систем из-за поддержки пулинга соединений. ⚠ По умолчанию для продуктивных систем используйте **oslo_cache.memcache_pool**.
enabled	Включения режима кэширования.	
servers	Список серверов Memcached.	Настраивается в виде списка addr:port, разделенные запятыми.

Файл экспорта параметров входа

ⓘ Легенда таблицы доступна на этой странице[76].

Путь до конфигурации: `$HOME/{OS_USER}-openrc.`

Имя параметра	Описание	Примечания
OS_USERNAME	Имя пользователя в облачной платформе.	
OS_PASSWORD	Пароль пользователя.	
OS_PROJECT_NAME	Имя проекта.	
OS_USER_DOMAIN_NAME	Имя проекта, в котором зарегистрирован пользователь.	
OS_PROJECT_DOMAIN_NAME	Имя домена, где зарегистрирован проект.	
OS_AUTH_URL	Адрес сервиса Keystone.	
OS_IDENTITY_API_VERSION	Версия Keystone API.	

OpenStack Glance

Информация о сервисе Glance

OpenStack Glance[77] - это служба предоставления и хранения наборов данных, необходимых конечным пользователям для создания вычислительных объектов облачной платформы. Иными словами, в Glance

76 https://conf.tionix.ru/pages/viewpage.action?pageId=205881354#id-Условныеобозначения-table-format-desc
77 https://docs.openstack.org/glance/victoria/

хранятся различные типы образов, необходимые для запуска виртуальных машин и контейнеров, а также метаданные, определяющие различные параметры запуска и функционирования ресурсов OpenStack.

Для версии Victoria поддерживается версия Glance API v2.

Образы

Сервис Glance главным образом призван находить, регистрировать и предоставлять образы, предназначенные для запуска виртуальных машин. У сервиса имеется RESTful API, позволяющий получать как образы в виде потока данных. Образы могут иметь различный формат (в том числе поддерживаются образы ядра, инициализации виртуальной фс и так далее), так и могут располагаться в различных системах хранения: начиная с обычной локальной файловой системы и заканчивая распределенными объектными системами хранения.

Метаданные

Вторая функция сервиса Glance - это предоставление каталога определённых метаданных (metadefs). Они предназначены для указания различных параметров, которые могут быть применены при работе с различными ресурсами OpenStack (обычно это касается запуска ВМ и работе с дисками). Представляет собой обычный каталог, содержащие строки типа "ключ=значение" и сами по себе они ничего не делают, сервисы OpenStack должны уметь их находить и применять самостоятельно.

Состав сервиса

Сервис состоит из нескольких компонентов:

- **glance-api** - реализующий API сервиса с поддержкой запросов образов;
- **База данных glance**, хранящая состояние сервиса Glance: список образов, метаданных и так далее;
- **Внешняя система хранения** для размещения файлов образов. Это может быть и обычная файловая система, и распределенное объектное хранилище;
- Функции Glance, отвечающие за обработку **метаданных**.

Установка сервиса Glance

Настройка окружения

Перед самой установкой сервиса нужно предварительно настроить некоторые компоненты инфраструктуры.

Подготовка базы данных glance

См. также: Установка и настройка СУБД MariaDB (см. стр. 22)

Всю информацию о данных образов и метаданных по умолчанию Glance хранит в базе данных SQL.

1. Войдите в окружение базы данных:

```
mysql -u root -p
```

2. Создайте базу данных glance:

```
create database glance;
```

3. Предоставьте доступ к этой базе данных пользователю glance в СУБД (для localhost и всем остальным адресам отдельно):

```
grant all privileges on glance.* to 'glance'@'localhost' identified by
'GLANCE_DBPASS';
grant all privileges on glance.* to 'glance'@'%' identified by 'GLANCE_DBPASS';
```

> Вместо GLANCE_DBPASS используйте свой пароль, он будет необходим далее.

4. Выйдите из сессии СУБД:

```
exit;
```

Создание объектов в Keystone

> См. также: Файл настройки системного окружения (см. стр. 38) и Создание объектов в Keystone (см. стр. 39).

Для сервиса Glance необходимо создать пользователя и зарегистрировать его в сервисе каталогов Keystone.

1. Настройте окружение командной строки:

```
source $HOME/admin-openrc
```

2. Создайте пользователя glance (команда интерактивно спросит пароль, далее этот пароль будет использоваться в параметрах, где указан GLANCE_PASS):

```
openstack user create --domain default --password-prompt glance
```

3. Добавьте пользователя glance в проект service (см. стр. 39) с ролью admin:

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

4. Создайте сервис image в сервисе каталогов Keystone:

```
openstack service create --name glance --description "OpenStack Image" image
```

5. Создайте три точки входа для сервиса image:
   a. публичную:

```
openstack endpoint create --region RegionOne image public http://controller:
9292
```

   b. внутреннюю:

```
openstack endpoint create --region RegionOne image internal http://
controller:9292
```

   c. административную:

```
openstack endpoint create --region RegionOne image admin http://controller:
9292
```

> controller используется в качестве примера адреса. В продуктивных средах вместо домена "controller" укажите единый DNS-адрес для сервиса Glance, который был выбран в вашей инфраструктуре. Использование IP-адресов не рекомендуется.

> Для разных типов точек входа API можно указать разные адреса. Это может быть необходимо в случае разделения, например, публичного трафика от внутреннего.

Установка сервиса Glance

> См. также: Настройка репозиториев Almalinux (см. стр. 10).

После установки всех необходимых внешних сервисов можно приступить к установке сервиса Glance.

1. Установите основной пакет сервиса:

```
dnf -y install openstack-glance
```

Стандартные пути файлов конфигурации:

- `/etc/glance` – Каталог конфигурации Glance;
- `/etc/glance/glance-api.conf` – основный файл конфигурации.

2. Очистите основной файл конфигурации, который был добавлен после установки пакета:

```
> /etc/glance/glance-api.conf
```

3. В основной файл добавьте следующую конфигурацию (описание (см. стр. 58)):

```
[DEFAULT]
debug = False
bind_host = LISTEN_ADDR
log_dir = /var/log/glance
#use_stderr = True
show_image_direct_url = True
transport_url = rabbit://openstack:RABBIT_PASS@controller:5672
cinder_catalog_info = volume:cinder:internalURL
node_staging_uri = /var/lib/glance/staging

[database]
connection = mysql+pymysql://glance:GLANCE_DBPASS@controller/glance
retry_interval = 5
connection_recycle_time = 10
max_pool_size = 1
max_retries = -1

[keystone_authtoken]
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = glance
password = GLANCE_PASS

[paste_deploy]
flavor = keystone
config_file = /usr/share/glance/glance-api-dist-paste.ini

[glance_store]
stores = file,http
default_store = file

[oslo_messaging_notifications]
driver = messagingv2

[cache]
backend = oslo_cache.memcache_pool
enabled = True
memcache_servers = controller:11211

[oslo_middleware]
enable_proxy_headers_parsing = True

[file]
filesystem_store_datadir = /var/lib/glance/images/

[task]
work_dir = /var/lib/glance/tasks_work_dir
```

> Файл по умолчанию, который появится после установки пакета, лучше всего очистить от всех строк:
>
> ```
> > /etc/glance/glance-api.conf
> ```

4. В конфигурации нужно обратить внимание на следующие параметры:
   a. Адрес прослушивания сервиса Glance API, которые зависят от метода установки;
   b. Адрес сервиса RabbitMQ, указываемый в **DEFAULT/transport_url**, в частности, пароль к пользователю openstack вместо RABBIT_PASS;
   c. Параметры журналирования сервиса (см. стр. 58), которые зависят от метода установки;
   d. Параметры подключения к СУБД в **database - connection**, в частности, пароль к БД glance вместо GLANCE_DBPASS;
   e. Параметры подключения к Keystone в **keystone_authtoken**, в частности, пароль пользователя glance вместо GLANCE_PASS;
   f. Параметры подключения к серверу memcached в **keystone_authtoken/memcached_servers** и **cache/memcache_servers**.
   g. Так же для продуктивных систем следует использовать сетевые системы хранения, а не локальную файловую систему (**file**).
5. После настройки конфигурации запустите процесс инициализации БД glance:

```
su -s /bin/sh -c "glance-manage db_sync" glance
```

Финализация установки

1. После определения конфигурации необходимо запустить сервис Glance API и добавить его в автозапуск:

```
systemctl start openstack-glance-api.service
systemctl enable openstack-glance-api.service
```

Проверка работы сервиса

1. Проверьте статус юнита Glance API:

```
systemctl status openstack-glance-api
```

> Ответ должен быть примерно таким:
>
> ```
> ● 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)
>   Drop-In: /run/systemd/system/openstack-glance-api.service.d
>            └─zzz-lxc-service.conf
>    Active: active (running) since Sun 2021-11-07 22:06:22 UTC; 1min 15s ago
>  Main PID: 13485 (glance-api)
>     Tasks: 6 (limit: 204240)
>    Memory: 106.7M
>    CGroup: /system.slice/openstack-glance-api.service
>            ├─13485 /usr/bin/python3 /usr/bin/glance-api
> ...
> Nov 07 22:06:22 tnx-mgmt-almalinux systemd[1]: Started OpenStack Image
> Service (code-named Glance) API server.
> ```

2. Проверьте наличие открытого порта 9292:

```
ss -tnlp | grep 9292
```

Ответ должен быть примерно таким:

```
LISTEN 0      128    10.236.64.162:9292       0.0.0.0:*    users:(("glance-
api",pid=32432,fd=3),...
```

3. Проверьте, что по этому порту отвечает сервис Keystone:

```
curl http://controller:9292
```

Ответ должен быть примерно таким:

```
{"versions": [{"id": "v2.9", "status": "CURRENT", "links": [{"rel": "self",
"href": "http://controller:9292/v2/"}]},...
```

Конфигурация для HAProxy

См. также: Установка балансировщика нагрузки HAProxy (см. стр. 16)

В референсной архитектуре доступ до сервиса Glance API предоставляется через балансировщика нагрузки.

1. Конфигурация для HAProxy выглядит следующим образом:

```
frontend glance_api
  bind "IP:PORT" ssl crt /usr/local/etc/haproxy/cert.pem alpn h2,http/1.1
  http-request set-header X-Forwarded-Proto https
  default_backend glance_api_backend

backend glance_api_backend
  server glance 127.0.0.1:9292
```

2. После включения этой конфигурации перезагрузите конфигурацию HAProxy:

```
systemctl reload haproxy
```

Добавление тестового образа в Glance

После первичной настройки необходимо убедиться, что базовые функции Glance работают исправно. Для этого достаточно добавить образ в хранилище Glance.

Проверка загрузки образа

См. также: Файл настройки системного окружения (см. стр. 38)

Основной проверкой работы сервиса является загрузка тестового образа в хранилище Glance.

1. Настройте окружение командной строки:

```
source $HOME/admin-openrc
```

2. Далее загрузите тестовый образ Cirros:

```
dnf -y install wget
wget http://download.cirros-cloud.net/0.5.2/cirros-0.5.2-x86_64-disk.img
```

3. Запустите команду загрузки образа:

```
glance image-create --name "cirros" --file cirros-0.5.2-x86_64-disk.img \
       --disk-format qcow2 --container-format bare --visibility=public
```

Ответ должен быть примерно таким:

```
+------------------
+--------------------------------------------------------------------------
--------+
| Property         | Value
|
+------------------
+--------------------------------------------------------------------------
--------+
| checksum         | b874c39491a2377b8490f5f1e89761a4
|
| container_format | bare
|
| created_at       | 2021-11-07T22:10:35Z
|
| direct_url       | file:///var/lib/glance/images/d8a305ac-650f-4557-
a9d9-845042a4da0f               |
| disk_format      | qcow2
|
| id               | d8a305ac-650f-4557-a9d9-845042a4da0f
|
| min_disk         | 0
|
| min_ram          | 0
|
| name             | cirros
|
| os_hash_algo     | sha512
|
| os_hash_value    |
6b813aa46bb90b4da216a4d19376593fa3f4fc7e617f03a92b7fe11e9a3981cbe8f0959dbeb
e3622 |
|                  | 5e5f53dc4492341a4863cac4ed1ee0909f3fc78ef9c3e869
|
| os_hidden        | False
|
| owner            | 891267e1ce4044a3b91ac90c41ca1603
|
| protected        | False
|
| size             | 16300544
|
| status           | active
|
| tags             | []
|
| updated_at       | 2021-11-07T22:10:36Z
|
| virtual_size     | 117440512
|
| visibility       | public
|
+------------------
+--------------------------------------------------------------------------
--------+
```

4. Через некоторое время после принятия запроса запросите список образов:

```
glance image-list
```

В ответе вы должны получить список образов с добавленным первым образом cirros:

```
+--------------------------------------+--------+
| ID                                   | Name   |
+--------------------------------------+--------+
| d8a305ac-650f-4557-a9d9-845042a4da0f | cirros |
+--------------------------------------+--------+
```

Описание файла конфигурации Glance

В описании процесса по установке сервиса Keystone предложена конфигурация для сервиса Keystone, а так же настройки входа в сервис. Это страница содержит подробное описание этих настроек.

При изменении конфигурации необходимо перезапустить веб-сервер Apache:

```
systemctl restart openstack-glance-api
```

Таблица конфигурации

Сервис Glance

Легенда таблицы доступна на этой странице[78].

Путь до конфигурации: `/etc/glance/glance-api.conf`

Имя конфигурации	Описание функции	Примечание
[DEFAULT]	Глобальные переменные сервиса.	
debug	Включение режима отладки.	Используйте только для анализа проблем. Отладочный режим может повлиять на производительность и конфиденциальность данных.
bind_host	Адрес, который будет слушать сервис Glance.	Необходимо указать один из следующих вариантов: • **localhost,** если установка *производится в рамках референсной архитектуры*; • **Адрес в mgmt-интерфейсе,** если производится *обычная установка.*
use_stderr **log_dir** **use_journal**	Параметры журналирования сервиса.	Необходимо выбрать **один из трёх** указанных параметров: • **use_stderr = True**, если установка *производится в рамках референсной архитектуры*, в этом случае журналы будут перенаправлены в виртуальное устройство /dev/stderr. Это позволяет получить логи системе управления контейнерами. • **log_dir = /var/log/glance**, если производится *обычная установка*, в этом случае журналы будут сохраняться в файлах в указанном каталоге. • **use_journal = True**, если производится *обычная установка*, в этом случае журналы будут

78 https://conf.tionix.ru/pages/viewpage.action?pageId=205881354#id-Условныеобозначения-table-format-desc

Имя конфигурации	Описание функции	Примечание
		перенаправлены в службу управления журналами systemd-journald.[79] ⚠ Режим **use_syslog** для перенаправления журналов в службу syslog/rsyslog переведён в разряд устаревших, поэтому официально не поддерживается.
workers	Количество процессов сервиса.	По умолчанию равен количеству CPU, но не больше 8, однако по некоторым причинам указывается 5 воркеров.
show_image_direct_url	Поддержка показа прямых ссылок до образа.	Используйте этот параметр только с системой хранения Ceph (для функции CoW-копирования). Документация предостерегает о проблеме безопасности[80] этого параметра.
transport_url	Список адресов до серверов RabbitMQ.	Серверы указываются через запятую с указанием префикса в каждом элементе.
cinder_catalog_info	Источник адреса точки входа в Cinder.	
[database]	Настройки доступа к базе данных.	
connection	Адрес до СУБД.	
retry_interval	Интервал между попытками подключения.	
connection_recycle_time	Время жизни одного соединения.	
max_pool_size	Максимальный размер пула соединений к СУБД.	
max_retries	Максимальное количество попыток соединений к СУБД.	-1 выключает ограничение на количество попыток.
[keystone_authtoken]	Параметры подключения к сервису Keystone.	
www_authenticate_uri **auth_url**	Адрес до сервиса Keystone.	Оба параметра должны совпадать друг с другом.
memcached_servers	Адреса до сервисов кэширования memcached.	Этот параметр действителен лишь для запросов аутентификации. Несколько адресов можно указать через запятую.
auth_type	Тип аутентификации.	Всегда должен быть равен параметру "password".

79 https://www.freedesktop.org/software/systemd/man/systemd-journald.service.html
80 https://docs.openstack.org/glance/victoria/configuration/glance_api.html#DEFAULT.show_image_direct_url

Имя конфигурации	Описание функции	Примечание
project_domain_name **user_domain_name** **project_name** **username** **password**	Данные аутентификации в сервис Keystone от имени пользователя glance в проекте service.	Оставьте эти параметры по умолчанию.
[paste_deploy]	Параметрыinterface деплоя WSGI-сервиса.	
flavor	Используемая система аутентификации.	Всегда должен быть равен "keystone".
config_file	Путь до paste-файла сервиса Glance.	
[glance_store]	Параметры системы хранения для образов.	
stores	Список доступных типов хранения.	file - хранение образов в локальной файловой системе, путь указан в секции file. http - возможность получения образов через указание HTTP-ссылки в качестве источника.
default_store	Основной тип хранения образов. Обязателен, если len(stores) > 1	Glance выберет этот тип хранения по умолчанию, если клиент явно не запросит другой.
[oslo_messaging_notifications]	Параметры оповещений событий сервиса.	Для этого параметра необходимо указание адреса RabbitMQ в transport_url.
driver	Тип драйвера оповещений.	В референсе всегда должен быть равен "messagingv2".
[cache]	Параметры кэширования запросов сервиса.	
backend	Тип внешнего сервиса кэширования.	В референсной схеме нужно использовать oslo_cache.memcache_pool.
enabled	Параметр включения кэширования.	
memcache_servers	Адреса внешних сервисов кэширования.	Этот параметр действителен для запросов Glance. Несколько адресов можно указать через запятую.
[oslo_middleware]	Параметры обработки запросов сервиса Glance.	
enable_proxy_headers_parsing	Параметр обработки заголовков прокси-сервера.	
***store**	Параметры, определяющие места хранения различных данных сервиса Glance.	

OpenStack Placement

Информация о сервисе

OpenStack Placement[81] - это отдельный сервис REST API, предназначенный для учёта данных о доступных ресурсах облачной платформы. Таким ресурсом может быть вычислительный узел, распределенная система хранения данных или пул доступных IP-адресов. Placement хранит не только информации о доступных ресурсах, но и хранит данные их потребления. Каждый инстанс, созданный в Nova, будет регистрировать и связывать различные типы ресурсов с ним в Placement.

Типы ресурсов, используемые в Placement, называются классами (classes). Имеются стандартные классы ресурсов (например, DISK_GB или VCPU), также можно создавать свои варианты классов.

Каждый ресурс, регистрируемый в Placement, называется провайдером, в котором содержится информация о свойствах этого ресурса. Например, для системы хранилища можно указать, какой тип дисков используется для хранения данных. По этим данным можно лучше определить уровень QoS для инстансов в проектах облачной платформы.

Placement содержит REST API[82] на базе протокола HTTP, сам API поддерживает микроверсионирование.

Placement в основном взаимодействует с сервисами:

- OpenStack Nova для учёта вычислительных ресурсов и их потребления инстансами;
- OpenStack Neutron для учёта потребления сетевых ресурсов и использования пулов IP-адресов;
- OpenStack Cinder для учёта потребления в хранилищах данных.

Установка сервиса Placement

Настройка окружения

Перед самой установкой сервиса нужно предварительно настроить некоторые компоненты инфраструктуры.

Установка зависимых пакетов

Placement работает как WSGI-приложение с использованием веб-сервера Apache. Поэтому перед установкой Placement в узел его необходимо установить. Если установка Placement производится в тех же узлах, где был установлен Keystone, то каких-либо дополнительных действий не требуется. Если же Placement ставится отдельно, то:

1. Установите пакет httpd:

```
dnf -y install httpd
```

2. Запустите веб-сервер и добавьте его в автозапуск:

```
systemctl start httpd
systemctl enable httpd
```

> Подробнее об этом можно узнать здесь[83].

Подготовка базы данных placement

> См. также: Установка и настройка СУБД MariaDB (см. стр. 22)

Всю информацию о данных образов и метаданных по умолчанию Placement хранит в базе данных SQL.

---

81 https://docs.openstack.org/placement/victoria/
82 https://docs.openstack.org/api-ref/placement/
83 https://docs.openstack.org/placement/victoria/install/index.html

1. Войдите в окружение базы данных:

```
mysql -u root -p
```

2. Создайте базу данных placement:

```
create database placement;
```

3. Предоставьте доступ к этой базе данных пользователю placement в СУБД (для localhost и всем остальным адресам отдельно):

```
grant all privileges on placement.* to 'placement'@'localhost' identified by
'PLACEMENT_DBPASS';
grant all privileges on placement.* to 'placement'@'%' identified by
'PLACEMENT_DBPASS';
```

> Вместо PLACEMENT_DBPASS используйте свой пароль, он будет необходим далее.

4. Выйдите из сессии СУБД:

```
exit;
```

Создание объектов в Keystone

> См. также: Файл настройки системного окружения (см. стр. 35)

Для сервиса Placement необходимо создать пользователя и зарегистрировать его в сервисе каталогов Keystone.

1. Создайте пользователя placement:

```
openstack user create --domain default --password-prompt placement
```

> Команда интерактивно спросит пароль, далее этот пароль будет использоваться там, где указан PLACEMENT_PASS.

2. Добавьте пользователя placement в проект service (см. стр. 39) с ролью admin:

```
openstack role add --project service --user placement admin
```

3. Создайте сервис placement в сервисе каталогов Keystone:

```
openstack service create --name placement --description "Placement API" placement
```

4. Создайте три точки входа для сервиса placement:
   a. публичную:

```
openstack endpoint create --region RegionOne placement public http://
controller:8778
```

   b. внутреннюю:

```
openstack endpoint create --region RegionOne placement internal http://
controller:8778
```

   c. административную:

```
openstack endpoint create --region RegionOne placement admin http://
controller:8778
```

> Вместо домена "controller" укажите единый DNS-адрес для сервиса Placement, который был выбран в вашей инфраструктуре. Использование IP-адресов не рекомендуется.

> Для разных типов точек входа API можно указать разные адреса. Это может быть необходимо в случае разделения, например, публичного трафика от внутреннего.

Установка сервиса

> См. также: Настройка репозиториев Almalinux (см. стр. 10).

После установки всех необходимых внешних сервисов можно приступить к установке сервиса Placement.

1. Установите основной пакет сервиса Placement:

```
dnf -y install openstack-placement-api
```

> Стандартные пути файлов конфигурации:
>
> - `/etc/placement` – Каталог конфигурации Placement;
> - `/etc/placement/placement.conf` – основный файл конфигурации.

2. Очистите основной файл конфигурации, который был добавлен после установки пакета:

```
> /etc/placement/placement.conf
```

3. В основной файл добавьте следующую конфигурацию (описание (см. стр. 66)):

```
[placement_database]
connection = mysql+pymysql://placement:PLACEMENT_DBPASS@controller/placement

[api]
auth_strategy = keystone

[keystone_authtoken]
auth_url = https://controller:5000/v3
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = placement
password = PLACEMENT_PASS
```

4. При конфигурации следуют обратить внимание на следующее:
   a. Данные соединения до БД placement в **placement_database/connection**, в частности, пароль к базе вместо PLACEMENT_DBPASS;
   b. Параметры соединения до пользователя placement в **keystone_authtoken**, в частности, укажите его пароль вместо PLACEMENT_PASS.
5. Инициализируйте базу данных placement:

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

Финализация установки

1. Перезапустите веб-сервер Apache:

```
systemctl restart httpd
```

2. Для проверки воспользуйтесь командой:

```
placement-status upgrade check
```

В ответ вы должны получить примерно следующий вывод:

```
+---------------------------------+
| Upgrade Check Results           |
+---------------------------------+
| Check: Missing Root Provider IDs |
| Result: Success                 |
| Details: None                   |
+---------------------------------+
| Check: Incomplete Consumers     |
| Result: Success                 |
| Details: None                   |
+---------------------------------+
```

Шифрование сервиса Placement

Placement запускается с помощью веб-сервера Apache в отдельном виртуальном узле (VirtualHost). Для включения шифрования следует дополнить настройку веб-сервера.

1. В референсной архитектуре конфигурация веб-сервера для Placement по пути /etc/httpd/httpd.conf выглядит следующим образом:

```
ServerName placement.k8s_domain_name
ServerRoot "/etc/httpd"
Include conf.modules.d/*.conf
User apache
Group apache
LogLevel warn
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
ErrorLog /dev/stderr
CustomLog /dev/stdout combined
TypesConfig /etc/mime.types
AddDefaultCharset UTF-8
EnableSendfile on

<Directory />
  AllowOverride none
  Require all denied
</Directory>

SSLProtocol all -SSLv3 -TLSv1 -TLSv1.1
SSLCipherSuite ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-
ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-
POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-
SHA384
SSLHonorCipherOrder off
SSLSessionTickets off

Listen 8778

<VirtualHost *:8778>
  WSGIDaemonProcess placement-api processes=3 threads=1 user=placement
group=placement
  WSGIProcessGroup placement-api
  WSGIScriptAlias / /usr/bin/placement-api
  WSGIApplicationGroup %{GLOBAL}
  WSGIPassAuthorization On
  ErrorLogFormat "%M"
  ErrorLog /dev/stderr
  CustomLog /dev/stdout combined
  SSLEngine on
  SSLCertificateFile certs/cert.pem
  SSLCertificateKeyFile certs/privkey.pem
  Protocols h2 http/1.1
  <Directory /usr/bin>
    Require all granted
  </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>
```

Информация о конфигурации веб-сервера описана в этой части (см. стр. 45) Руководства.

2. После включения этой конфигурации перезапустите веб-сервер:

```
systemctl restart httpd
```

3. После этого убедитесь, что шифрованные соединения доступны по порту 8778, например, командой curl:

```
curl https://placement-addr:8778
```

⚠ Если сертификат самоподписанный, то укажите:

- параметр -k, в этом случае верификация сертификата будет выключена;
- или параметр --capath до файла CA вашего центра сертификации.

❗ Функционирование продуктивных сред с самоподписанными сертификатами официально не поддерживается.

Описание файла конфигурации Placement

В описании процесса по установке сервиса Placement предложена стандартная конфигурация. Это страница содержит подробное описание настроек этой конфигурации.

При изменении конфигурации необходимо перезапустить веб-сервер Apache:

```
systemctl restart httpd
```

Таблица конфигурации

Сервис Placement

ⓘ Путь до конфигурации: `/etc/placement/placement.conf`

ⓘ Легенда таблицы доступна на этой странице[84].

Имя параметра	Описание	Примечания
[placement_database]	Параметры доступа к базе данных placement.	
connection	Параметры соединения к базе данных placement.	
[api]	Параметры работы Placement API.	
auth_strategy	Указание механизма аутентификации.	Всегда должен быть равен "keystone".
[keystone_authtoken]	Параметры подключения к сервису Keystone.	
auth_url	Адрес внутреннего API Keystone.	
memcached_servers	Адреса внешней системы кэширования Memcached.	Можно указать несколько серверов через запятую.
project_domain_name **user_domain_name** **project_name** **username** **password**	Данные для аутентификации в Keystone от имени пользователя placement.	

OpenStack Nova

Информация о сервисе Nova

- Обзор сервиса (см. стр. 67)

84 https://conf.tionix.ru/pages/viewpage.action?pageId=205881354#id-Условныеобозначения-table-format-desc

Обзор сервиса

OpenStack Nova[85] – это комплекс сервисов, которые предоставляют различные интерфейсы управления так называемыми вычислительными инстансами (экземплярами, compute instances). Обычно под "инстансом" понимается виртуальная машина, однако при наличии нужных сервисов, это может быть и железный узел, и системные контейнеры.

Фундаментально Nova разделяется на две большие части:

- **Управляющая часть** сервиса, которые предоставляют доступ к API и бизнес-логике. Устанавливаются в **управляющих узлах**.
- **Вычислительная часть** сервиса, которая занимается фактическим запуском инстансов (например, в виде виртуальных машин). Устанавливаются в **вычислительных узлах**.

Nova использует порты:

- **8774/TCP** для Nova API.
- **8775/TCP** для Nova Metadata API.
- **6082/TCP** для noVNC Proxy.

Состав сервиса

Сервис Nova состоит из нескольких компонентов:

- В управляющих узлах:
  - **nova-api** - это сервис, реализующий Nova API и сервис доступа к метаданным через Nova Metadata API;
  - **nova-scheduler** - сервис, отвечающий за планирование различных задач над инстансами;
  - **nova-conductor** - сервис, отвечающий за сбор данных с сервисов Nova в вычислительных узлах;
  - **nova-novncproxy** - сервис noVNC Proxy, перенаправляющий сессию VNC виртуальной машины в веб-интерфейс OpenStack Horizon;
  - **nova-spicehtml5proxy** - сервис SPICE Proxy, перенаправляющий сессию SPICE виртуальной машины в веб-интерфейс OpenStack Horizon. Этот сервис **официально не поддерживается**;
  - **nova-api-metadata** - сервис, предоставляющий интерфейс доступа к API метаданных Nova. Используется только в том случае, если необходимо предоставить сервис Nova Metadata API в отдельном узле без использования nova-api.
  - **База данных nova** с состоянием сервиса Nova;
  - **Очередь сообщений RabbitMQ** для взаимодействия между сервисами Nova.
- В вычислительных узлах:
  - **nova-compute** - сервис, отвечающий за обработку задач от управляющих сервисов и за фактическое их выполнение на конечных вычислительных узлах (напр., создание виртуальной машины).

TIONIX на данный момент официально поддерживает только гипервизор QEMU[86] с поддержкой аппаратного ускорения KVM[87]. Поэтому в составе вычислительных узлов будут добавлены еще два основных компонента:

- QEMU[88] – это гипервизор Type2, который поддерживает запуск виртуальных машин с эмулированием различных аппаратных платформ. Поддерживает возможность аппаратного ускорения виртуализации при помощи модуля KVM[89];
- libvirt[90] – это сервис управления виртуальными машинами и параметрами гипервизоров. Поддерживает управление гипервизором QEMU, используется сервисом nova-compute как промежуточный API к гипервизору.

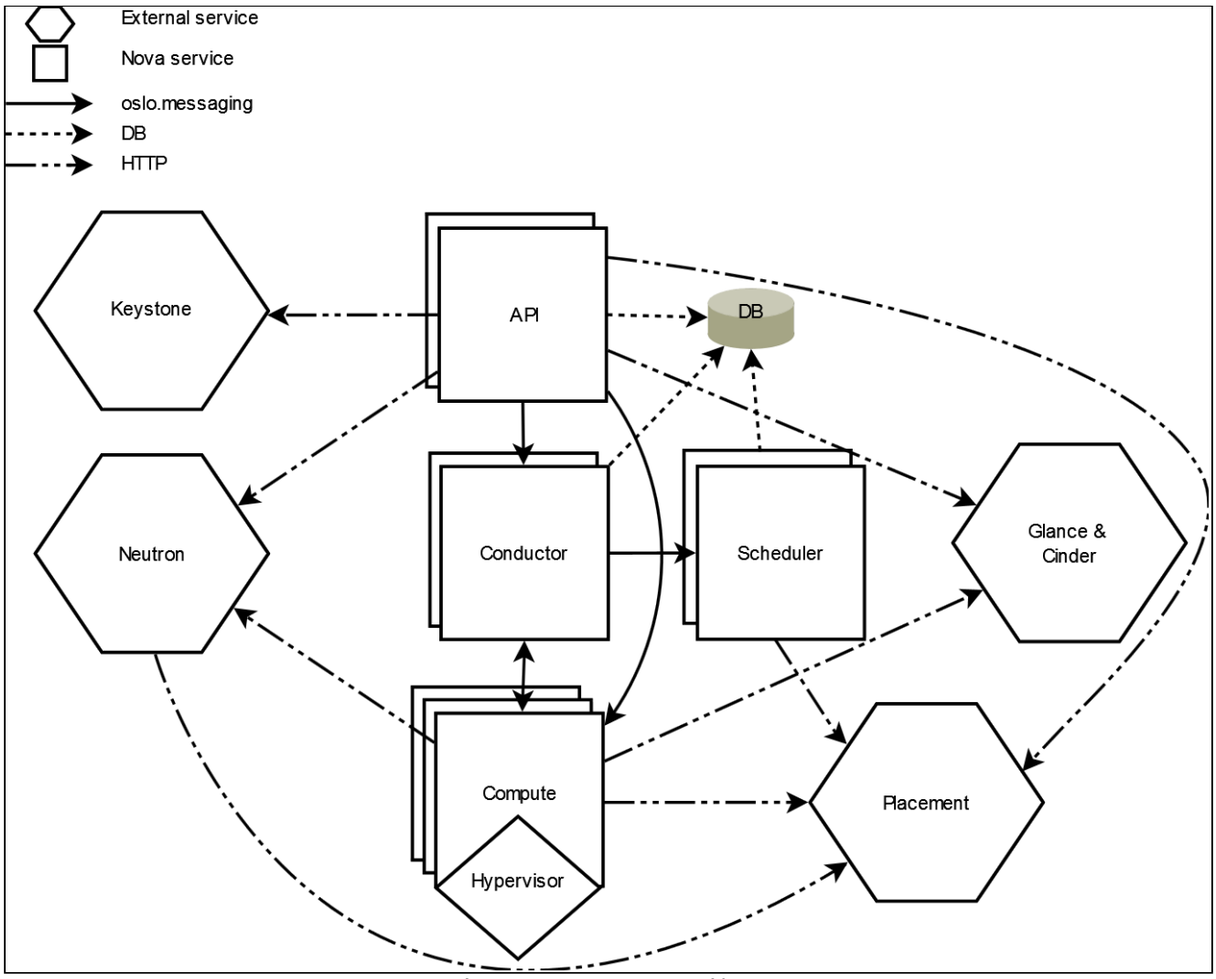
Для полноценной работы Nova требуется доступ к следующим сервисам OpenStack:

- **Glance** – Nova требуется доступ к образам и их метаданным для запуска инстансов.
- **Placement** – Nova получает информацию о доступных ресурсах с этого сервиса.
- **Neutron** – Nova требуется доступ к сетевым портам и адресам для настройки доступа до инстансов.

85 https://docs.openstack.org/nova/victoria/
86 https://www.qemu.org/
87 https://www.linux-kvm.org/page/Main_Page
88 https://www.qemu.org/
89 https://www.linux-kvm.org/page/Main_Page
90 https://libvirt.org/

- **Cinder** – Nova может использовать Cinder в качестве бэкенда для блочных устройств, подключаемых к инстансам. Технически Nova может работать без службы Cinder, однако в референсной архитектуре для продуктивных систем он обязателен.

Графически архитектура сервиса Nova выглядит следующим образом (не включена часть с noVNC Proxy):



Архитектура сервиса Nova

Основные объекты сервиса

Nova в своей работе использует несколько типов объектов.

- **Экземпляр** или **Инстанс** (Instance) – объект Nova, ассоциированный с виртуальной машиной, которая запускается в одном из гипервизоров.
- **Гипервизор** или **Узел** (Hypervisor и Host) – объект Nova, представляющий физический вычислительный узел, предназначенный для запуска виртуальных машин в виде инстансов.
- **Ячейка** (Cell) – метод изоляции объектов Nova. Позволяет разделять вычислительные ресурсы на логические группы с хранением данных в отдельной базе данных и с использованием своего сервиса обмена сообщениями. Есть два вида ячеек:
  - Так называемая **"нулевая" ячейка cell0**, который предоставлен исключительно для хранения данных инстансов, который не запустились из-за различных ошибок.
  - **Стандартная ячейка "cell1"**, куда поместятся успешно запущенные инстансы.
- **Схема ресурсов** или **Флэвор** (Flavor) – объект Nova, используемый для описания количества предоставляемых ресурсов инстансу.
- **Агрегация узлов** (Host Aggregation) – объект Nova, предназначенный для объединения гипервизоров в группы с возможностью указания общих метаданных.

Дополнительная информация

1. Официальная документация[91] проекта.
   a. Подробное описание архитектуры[92] сервиса.
   b. Информация о ячейках (cells v2)[93].

91 https://docs.openstack.org/nova/victoria/index.html
92 https://docs.openstack.org/nova/victoria/user/architecture.html
93 https://docs.openstack.org/nova/victoria/user/cellsv2-layout.html

2. Основной репозиторий[94] проекта.
3. Описание API[95] проекта.

Установка управляющих сервисов Nova

Установка управляющих сервисов Nova состоит из нескольких шагов.

Настройка окружения

Перед самой установкой сервиса нужно предварительно настроить некоторые компоненты инфраструктуры.

Подготовка баз данных nova и nova_*

См. также: Установка и настройка СУБД MariaDB (см. стр. 22)

Всю информацию о состоянии инстансов и прочую информацию Nova хранит в базе данных SQL.

1. Войдите в окружение базы данных:

```
mysql -u root -p
```

2. Nova требуется три базы данных, создайте их:

```
create database nova_api;
create database nova;
create database nova_cell0;
```

Функции этих баз данных следующие:

- **nova_api** содержит схему Nova API;
- **nova** содержит информацию обо всех инстансах, в том числе информацию стандартной ячейки.
- **nova_cell0** содержит "нулевую" ячейку.

3. Предоставьте доступ к базам данных сервиса Nova
   a. nova_api:

```
grant all privileges on nova_api.* to 'nova'@'localhost' identified by
'NOVA_DBPASS';
grant all privileges on nova_api.* to 'nova'@'%' identified by 'NOVA_DBPASS';
```

   b. nova:

```
grant all privileges on nova.* to 'nova'@'localhost' identified by
'NOVA_DBPASS';
grant all privileges on nova.* to 'nova'@'%' identified by 'NOVA_DBPASS';
```

   c. nova_cell0:

```
grant all privileges on nova_cell0.* to 'nova'@'localhost' identified by
'NOVA_DBPASS';
grant all privileges on nova_cell0.* to 'nova'@'%' identified by
'NOVA_DBPASS';
```

94 https://opendev.org/openstack/nova
95 https://docs.openstack.org/api-ref/compute/

> Вместо NOVA_DBPASS укажите свой пароль.

4. Выйдите из сессии СУБД:

```
exit;
```

Создание объектов Nova в Keystone

> См. также: Файл настройки системного окружения (см. стр. 35)

Для сервиса Nova необходимо создать пользователя и зарегистрировать его в сервисе каталогов Keystone.

1. Создайте пользователя nova (команда интерактивно спросит пароль, далее этот пароль будет использоваться в NOVA_PASS):

```
openstack user create --domain default --password-prompt nova
```

2. Добавьте пользователя nova в проект service (см. стр. 39) с ролью admin:

```
openstack role add --project service --user nova admin
```

3. Создайте сервис compute в сервисе каталогов Keystone:

```
openstack service create --name compute --description "OpenStack Compute" compute
```

4. Создайте три точки входа для сервиса compute:
   a. публичную (public):

```
openstack endpoint create --region RegionOne compute public http://
controller:8774/v2.1
```

   b. внутреннюю (internal):

```
openstack endpoint create --region RegionOne compute internal http://
controller:8774/v2.1
```

   c. административную (admin):

```
openstack endpoint create --region RegionOne compute admin http://controller:
8774/v2.1
```

> Вместо домена "controller" укажите единый DNS-адрес для сервиса Nova, который был выбран в вашей инфраструктуре. Использование IP-адресов не рекомендуется.

> Для разных типов точек входа API можно указать разные адреса. Это может быть необходимо в случае разделения, например, публичного трафика от внутреннего.

Установка сервисов Nova

Первичная настройка конфигурации

> Перед установкой сервисов Nova убедитесь, что вы установили и настроили службу Placement (см. стр. 61).

1. Установите пакеты сервисов Nova для управляющего узла:

```
dnf -y install openstack-nova-api openstack-nova-conductor \
  openstack-nova-novncproxy openstack-nova-scheduler
```

Стандартные пути конфигурации:

- Каталог конфигурационных файлов: `/etc/nova`
- Основной файл конфигурации: `/etc/nova/nova.conf`

2. Очистите основной файл конфигурации, который был добавлен после установки пакета:

```
> /etc/nova/nova.conf
```

3. Основной конфигурационный файл выглядит следующим образом (описание (см. стр. 77)):

```
[DEFAULT]
my_ip = MGMT_IP_ADDRESS
osapi_compute_listen = $my_ip
metadata_listen = $my_ip
enabled_apis = osapi_compute,metadata
transport_url = rabbit://openstack:RABBIT_PASS@controller:5672
use_stderr = true
allow_resize_to_same_host = true

[api_database]
connection = mysql+pymysql://nova:NOVA_DBPASS@controller/nova_api
connection_recycle_time = 10
max_overflow = 1000
max_pool_size = 1
max_retries = -1

[database]
connection = mysql+pymysql://nova:NOVA_DBPASS@controller/nova
connection_recycle_time = 10
max_overflow = 1000
max_pool_size = 1
max_retries = -1

[api]
auth_strategy = keystone
use_forwarded_for = true

[keystone_authtoken]
www_authenticate_uri = http://controller:5000/
auth_url = http://controller:5000/
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = nova
password = NOVA_PASS

[vnc]
enabled = true
server_listen = $my_ip
server_proxyclient_address = $my_ip

[cinder]
catalog_info = volumev3:cinderv3:internalURL
os_region_name = RegionOne
auth_url = http://controller:5000
auth_type = password
project_domain_name = default
user_domain_name = admin
project_name = service
username = cinder
password = CINDER_PASS

[placement]
region_name = RegionOne
project_domain_name = Default
project_name = service
auth_type = password
user_domain_name = Default
auth_url = http://controller:5000/v3
username = placement
password = PLACEMENT_PASS

[neutron]
url = http://controller:9696
auth_url = http://controller:5000
```

```
auth_type = password
project_domain_name = Default
user_domain_name = Default
region_name = RegionOne
project_name = service
username = neutron
password = NEUTRON_PASS
service_metadata_proxy = true
metadata_proxy_shared_secret = METADATA_PASS
valid_interfaces = internal

[oslo_concurrency]
lock_path = /var/lib/nova/tmp

[cache]
backend = oslo_cache.memcache_pool
enabled = True
memcache_servers = controller:11211

[scheduler]
max_attempts = 10
discover_hosts_in_cells_interval = 300
workers = 3

[wsgi]
secure_proxy_ssl_header = HTTP_X_FORWARDED_PROTO
api_paste_config = /etc/nova/api-paste.ini

[oslo_middleware]
enable_proxy_headers_parsing = True

[privsep]
helper_command=sudo nova-rootwrap /etc/nova/rootwrap.conf privsep-helper --config-
file /etc/nova/nova.conf

[upgrade_levels]
compute = auto
```

4. Инициализируйте базы данных сервиса Nova:
   a. nova_api со схемой API сервиса:

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

   b. nova_cell0 с данными "нулевой" ячейки:

      ```
      su -s /bin/sh -c "nova-manage cell_v2 map_cell0" nova
      ```

      > Игнорируйте ошибки, связанные с устареванием использования формата JSON в файле policy.json для сервиса Nova.

   c. данные обычной ячейки

      ```
      su -s /bin/sh -c "nova-manage cell_v2 create_cell --name=cell1 --verbose"
       nova
      ```

      > В выводе команды могут появиться сообщения, что некоторые параметры не указаны и берутся с основного файла конфигурации Nova. Эти сообщения можно игнорировать.
      >
      > В конце вывод вы должны получить идентификатор формата UUID созданной ячейки.

   d. основную базу данных nova, которая в итоге окончит инициализацию ячейки cell1:

      ```
      su -s /bin/sh -c "nova-manage db sync" nova
      ```

> Игнорируйте ошибки, связанные с устареванием использования формата JSON в файле policy.json для сервиса Nova.

5. Убедитесь, что "нулевая" и обычная ячейки появились в списке доступных:

```
su -s /bin/sh -c "nova-manage cell_v2 list_cells" nova
```

> В ответ вы должны получить примерно следующий вывод:

```
+-------+--------------------------------------
+----------------------------------------
+--------------------------------------------------+----------+
|  Name |                 UUID                 |              Transport URL
|               Database Connection               | Disabled |
+-------+--------------------------------------
+----------------------------------------
+--------------------------------------------------+----------+
| cell0 | 00000000-0000-0000-0000-000000000000 |                  none:/
| mysql+pymysql://nova:****@controller/nova_cell0 |  False   |
| cell1 | a6ae1e90-7751-4683-9c0f-de455cffce76 | rabbit://
openstack:****@controller:5672 |    mysql+pymysql://nova:****@controller/
nova    |  False   |
+-------+--------------------------------------
+----------------------------------------
+--------------------------------------------------+----------+
```

Финализация установки

1. Запустите все сервисы Nova и добавьте их в автозапуск:

```
systemctl start \
    openstack-nova-api.service \
    openstack-nova-scheduler.service \
    openstack-nova-conductor.service \
    openstack-nova-novncproxy.service
systemctl enable \
    openstack-nova-api.service \
    openstack-nova-scheduler.service \
    openstack-nova-conductor.service \
    openstack-nova-novncproxy.service
```

Проверка статуса сервиса

1. Проверьте статус компонентов сервиса Nova, например, nova-api:

```
systemctl status openstack-nova-api
```

> В ответ вы должны получить примерно следующий вывод:

```
● openstack-nova-api.service - OpenStack Nova API Server
   Loaded: loaded (/usr/lib/systemd/system/openstack-nova-api.service;
enabled; vendor preset: disabled)
  Drop-In: /run/systemd/system/openstack-nova-api.service.d
           └─zzz-lxc-service.conf
   Active: active (running) since Sun 2021-11-07 22:37:23 UTC; 1min 28s ago
 Main PID: 16644 (nova-api)
    Tasks: 9 (limit: 204240)
   Memory: 497.6M
   CGroup: /system.slice/openstack-nova-api.service
           ├─16644 /usr/bin/python3 /usr/bin/nova-api
...
```

2. Проверьте статус порта:

```
ss -tnlp | grep 8774
```

В ответ вы должны получить примерно следующий вывод:

```
LISTEN 0      128    10.236.64.231:8774       0.0.0.0:*    users:(("nova-
api",pid=16684,fd=7),...
```

Адрес LISTEN должен быть равен адресу, указанный в **DEFAULT/my_ip** основного файла конфигурации Nova.

3. Получите статус Nova API:

```
curl http://controller:8774
```

В ответ вы должны получить примерно следующий вывод:

```
{"versions": [{"id": "v2.0", "status": "SUPPORTED", "version": "",
"min_version": "", "updated": "2011-01-21T11:33:21Z",...
```

Конфигурация для HAProxy

См. также: Установка балансировщика нагрузки HAProxy (см. стр. 16)

В референсной архитектуре доступ до сервисов Nova (кроме nova-novncproxy) предоставляется через балансировщика нагрузки.

1. Конфигурация для HAProxy выглядит следующим образом:

```
frontend nova_api
  bind "MGMT_IP:8774" ssl crt /usr/local/etc/haproxy/cert.pem alpn h2,http/1.1
  http-request set-header X-Forwarded-Proto https
  default_backend nova_api_backend

backend nova_api_backend
  server nova 127.0.0.1:8774

frontend nova_metadata
  bind "MGMT_IP:8775" ssl crt /usr/local/etc/haproxy/cert.pem alpn h2,http/1.1
  http-request set-header X-Forwarded-Proto https
  default_backend nova_metadata_backend

backend nova_metadata_backend
  server nova 127.0.0.1:8775
```

2. После включения этой конфигураци перзагрузите конфигурацию HAProxy:

```
systemctl reload haproxy
```

Установка вычислительной части

Установка вычислительной части производится на вычислительных узлах.

Установка сервиса nova-compute

Установка сервиса nova-compute достаточно проста.

1. Установите пакет сервиса nova-compute:

```
dnf -y install openstack-nova-compute
```

> По умолчанию этот пакет установит пакеты гипервизора QEMU и системы управления гипервизоров libvirtd. Их отдельно устанавливать не требуется.

2. Примените конфигурационный файл, описанный в разделе установки управляющей части (см. стр. 71), по пути `/etc/nova/nova.conf`.

> На текущем шаге вычислительные узлы будут использовать стандартный драйвер (compute_driver) libvirt.

3. Запустите сервис nova-compute и сервис libvirtd и добавьте его в автозапуск:

```
systemctl start libvirtd openstack-nova-compute
systemctl enable libvirtd openstack-nova-compute
```

Проверка добавления вычислительного узла

По умолчанию вычислительный узел автоматически не добавляется в ячейку. В референсной архитектуре включен параметр, который обновит содержимое ячейки при появлении вычислительного узла раз в 5 минут. Однако этот процесс в ручном режиме для целей тестирования можно ускорить. Дальнейшие команды нужно выполнить на управляющем узле.

1. Проверьте, что сервис nova-compute зарегистрировался в облачной платформе:

```
openstack compute service list --service nova-compute
+----+-------+--------------+------+-------+---------+----------------------------
+
| ID | Host  | Binary       | Zone | State | Status  | Updated At
|
+----+-------+--------------+------+-------+---------+----------------------------
+
| 1  | node1 | nova-compute | nova | up    | enabled | 2017-04-14T15:30:44.000000
 |
+----+-------+--------------+------+-------+---------+----------------------------
+
```

2. Включите команду обнаружения новых вычислительных узлов в стандартной ячейке:

```
su -s /bin/sh -c "nova-manage cell_v2 discover_hosts --verbose" nova
```

В ответ вы должны получить примерно следующий вывод:

```
[root@tnx-mgmt-almalinux ~]# su -s /bin/sh -c "nova-manage cell_v2
discover_hosts --verbose" nova
2021-11-07 22:44:49.859 24841 WARNING oslo_policy.policy
[req-6d80d63f-6a6d-4fb5-966a-14f88f4a8066 - - - - -] JSON formatted
policy_file support is deprecated since Victoria release. You need to use
YAML format which will be default in future. You can use ``oslopolicy-
convert-json-to-yaml`` tool to convert existing JSON-formatted policy file
to YAML-formatted in backward compatible way: https://docs.openstack.org/
oslo.policy/latest/cli/oslopolicy-convert-json-to-yaml.html.
2021-11-07 22:44:49.860 24841 WARNING oslo_policy.policy
[req-6d80d63f-6a6d-4fb5-966a-14f88f4a8066 - - - - -] JSON formatted
policy_file support is deprecated since Victoria release. You need to use
YAML format which will be default in future. You can use ``oslopolicy-
convert-json-to-yaml`` tool to convert existing JSON-formatted policy file
to YAML-formatted in backward compatible way: https://docs.openstack.org/
oslo.policy/latest/cli/oslopolicy-convert-json-to-yaml.html.
Found 2 cell mappings.
Skipping cell0 since it does not contain hosts.
Getting computes from cell 'cell1': a6ae1e90-7751-4683-9c0f-de455cffce76
Checking host mapping for compute host 'tnx-mgmt-almalinux':
7f17e0f3-03fe-416b-96b8-7669a87821c7
Creating host mapping for compute host 'tnx-mgmt-almalinux':
7f17e0f3-03fe-416b-96b8-7669a87821c7
Found 1 unmapped computes in cell: a6ae1e90-7751-4683-9c0f-de455cffce76
```

Ошибки устаревания формата JSON в policy.json для Nova можно игнорировать.

Описание файла конфигурации Nova

В описании процесса по установке сервиса Nova предложена стандартная конфигурация. Это страница содержит подробное описание настроек в нём.

При изменении конфигурации необходимо все сервисы Nova:

```
systemctl restart openstack-nova-*
```

Таблица конфигурации

Управляющие сервисы Nova

Путь до конфигурации: `/etc/nova/nova.conf`

Легенда таблицы доступна на этой странице[96].

Имя параметра	Описание параметра	Примечания
[DEFAULT]	Глобальные параметры сервисов Nova.	
my_ip	IP-адрес сервиса Nova API в mgmt-сети инфраструктуры.	Необходимо выбрать **один из двух** указанных параметров: • **127.0.0.1**, если установка *производится в рамках референсной архитектуры*; • Адрес в интерфейсе mgmt-сети, если производится обычная установка.

96 https://conf.tionix.ru/pages/viewpage.action?pageId=205881354#id-Условныеобозначения-table-format-desc

Имя параметра	Описание параметра	Примечания
osapi_compute_listen	Адрес прослушивания сервиса Nova API.	По умолчанию адрес для этого параметра берётся со значения параметра my_ip. ⚠ Для прослушивания всех интерфейсов можно указать адрес 0.0.0.0, однако такая конфигурация не рекомендуется к использованию в продуктовых инсталляциях.
metadata_listen	Адрес сервиса API метаданных Nova.	По умолчанию адрес для этого параметра берётся со значения параметра my_ip. ⚠ Для прослушивания всех интерфейсов можно указать адрес 0.0.0.0, однако такая конфигурация не рекомендуется к использованию в продуктовых инсталляциях.
enabled_apis	Включение доступных типов Nova API.	
transport_url	Адреса сервисов RabbitMQ для обмена сообщениями между сервисами OpenStack.	Можно указать через запятую. ⚠ Указание префикса rabbit:// для каждого узла обязательно.
use_stderr **log_dir** **use_journal**	Параметры журналирования сервиса.	Необходимо выбрать **один из трёх** указанных параметров: • **use_stderr = True**, если установка *производится в рамках референсной архитектуры*, в этом случае журналы будут перенаправлены в виртуальное устройство /dev/stderr. Это позволяет получить логи системе управления контейнерами. • **log_dir = /var/log/keystone**, если производится *обычная установка*, в этом случае журналы будут сохраняться в файлах в указанном каталоге. • **use_journal = True**, если производится *обычная установка*, в этом случае журналы будут перенаправлены в службу управления журналами systemd-journald.[97] ⚠ Режим use_syslog для перенаправления журналов в сервис syslog переведён в разряд устаревших, поэтому официально не поддерживается.
allow_resize_to_same_host	Включение возможности изменения размера инстанса, не меняя узел, где он запущен.	Этот параметр предлагается удалить для продуктивных установок.
ssl_only	Включение шифрования для Nova.	Этот параметр действителен только для сервиса NoVNC Proxy. Шифрование для подключений к сервису Nova API и Nova Metadata API реализуется функцией SSL Termination на уровне HAProxy.

97 https://www.freedesktop.org/software/systemd/man/systemd-journald.service.html

Имя параметра	Описание параметра	Примечания
cert	Указание pem-файла с сертификатом и ключом к нему.	⚠ Убедитесь в том, что файл сертификата доступен для чтения от имени пользователя nova. ❗ Работа с самоподписанными сертификатами официально не поддерживается.
[api_database] **[database]**	Параметры соединения к базам данных Nova.	
connection	Адрес подключения к базе данных сервиса Nova.	
connection_recycle_time	Время для восстановления соединения до базы данных.	
max_overflow	Максимальное количество соединений при заполнении пула соединений.	
max_pool_size	Максимальный размер пула соединений к базе данных.	
max_retries	Максимальное количество попыток соединения до базы данных.	-1 выключает ограничение на количество попыток соединения.
[api]	Параметры работы API Nova.	
auth_strategy	Включение механизма аутентификации.	
use_forwarded_for	Включение заголовка X-Forwarder-For.	⚠ Необходимо для поддержки балансировщика нагрузки HAProxy.
[keystone_authtoken]	Параметры подключения к сервису Keystone	
www_authenticate_uri	Адрес публичного API Keystone	
auth_url	Адрес внутреннего API Keystone	
memcached_servers	Адреса внешней системы кэширования Memcached	Можно указать несколько серверов через запятую.
auth_type	Тип аутентификации.	

Имя параметра	Описание параметра	Примечания
project_domain_name **user_domain_name** **project_name** **username** **password**	Данные для аутентификации в Keystone от имени пользователя nova.	
[vnc]	Параметры для подключения к VNC-сессиям ВМ.	
enabled	Включение функции VNC	Необходимо для получения изображения виртуальной машины в Dashboard.
server_listen	Адрес прослушивания VNC-сервера	По умолчанию адрес для этого параметра берётся со значения параметра my_ip. ⚠ Для прослушивания всех интерфейсов можно указать адрес 0.0.0.0, однако такая конфигурация не рекомендуется к использованию в продуктовых инсталляциях.
server_proxyclient_address	Адрес прослушивания клиента прокси VNC-сервера	По умолчанию адрес для этого параметра берётся со значения параметра my_ip. ⚠ Для прослушивания всех интерфейсов можно указать адрес 0.0.0.0, однако такая конфигурация не рекомендуется к использованию в продуктовых инсталляциях.
[glance]	Параметры сервиса Glance.	
api_servers	Адреса сервисов API Glance.	Устарело[98], предлагается удалить из конфигурации.
[cinder]	Параметры для подключения к сервису Cinder.	
catalog_info	Параметр указания точки входа в сервис Cinder.	internal нередко включают в отдельную сеть, недоступная из публичной сети, в этом случае этот параметр повышает безопасность деплоя.
os_region_name	Имя региона, зарегистрированное в Keystone.	
auth_url	Внутренний адрес сервиса Keystone.	
auth_type	Тип аутентификации.	

98 https://docs.openstack.org/nova/victoria/configuration/config.html#glance.api_servers

Имя параметра	Описание параметра	Примечания
project_domain_name **user_domain_name** **project_name** **username** **password**	Данные для аутентификации в Keystone от имени пользователя cinder.	Адрес до сервиса Cinder Nova получит из сервиса каталогов Keystone.
[placement]	Параметр сервиса Placement.	
auth_url	Внутренний адрес сервиса Keystone.	
region_name	Имя региона, зарегистрированное в Keystone.	
auth_type	Тип аутентификации.	
project_domain_name **project_name** **user_domain_name** **username** **password**	Данные для аутентификации в Keystone от имени пользователя placement.	Адрес до сервиса Placement Nova получит из сервиса каталогов Keystone.
[neutron]	Параметры, отвечающие за подключение к сервису Neutron.	
url	Адрес подключения к сервису Neutron.	В описании конфигурации этот параметр не находится.
auth_url	Адрес подключения к сервису.	
auth_type	Тип аутентификации.	
project_domain_name **project_name** **user_domain_name** **username** **password**	Данные для аутентификации в Keystone от имени пользователя neutron.	
service_metadata_proxy	Включение поддержки сервиса метаданных Neutron.	
metadata_proxy_shared_secret	Пароль к сервису метаданных Neutron. Обязателен, если пр. пункт True.	
ovs_bridge	Имя внутреннего мост-интерфейса OVS по умолчанию.	
valid_interfaces	Допустимые типы точек входа в сервис Neutron.	
[oslo_concurrency]	Параметры обработки конкурентных задач в Oslo.	

Имя параметра	Описание параметра	Примечания
lock_path	Каталог для файлов блокировок.	
[cache]	Параметры внешней системы кэширования.	
backend	Модуль для работы с внешней системой кэширования. Обязателен, если enabled = True.	oslo_cache.memcache_pool рекомендуется использовать для высоконагруженных сервисов OpenStack с большим количеством тредов.
enabled	Включение кэширования запросов сервиса Nova.	
memcache_servers	Адреса системы кэширования memcached. Обязателен, если enabled = True.	Через запятую можно указать несколько адресов.
[scheduler]	Параметры планировщика инстансов Nova.	
max_attempts	Максимальное количество попыток сборки или перемещения инстанса.	
discover_hosts_in_cells_interval	Время между обновлениями списка узлов в ячейке Nova.	
workers	Количество процессов сервиса nova-scheduler.	
[wsgi]	Параметры WSGI-компонента.	
secure_proxy_ssl_header	Параметр наличия хедера при использовании прокси.	Внимание: используйте только в случае наличия reverse proxy, иначе не указывайте этот параметр.
api_paste_config	Путь до файла paste deploy.	
[oslo_middleware]	Параметры для запросов к сервису Nova.	
enable_proxy_headers_parsing	Включение обработки заголовков прокси-сервера.	Внимание: используйте только в случае наличия reverse proxy, иначе не указывайте этот параметр.
[privsep_entrypoint]		Этого параметра нет в официальной документации для Victoria
[guestfs]	Параметры для libguestfs[99].	
debug	Включение режима отладки для libguestfs.	Включайте этот параметр только для целей тестирования и обслуживания.
[upgrade_levels]		Этого параметра нет в официальной документации для Victoria.

99 https://libguestfs.org

OpenStack Neutron

Информация о сервисе Neutron

OpenStack Neutron[100] - это комплекс сервисов, которые призваны решать вопросы управления сетевой инфраструктурой облачной платформы. Neutron позволяет создавать сети, прикреплять порты к инстансам, управлять сетевым доступом и так далее. Плагины в составе Neutron позволяют расширить функциональность основного сервиса.

Neutron использует порт 9696/TCP.

Состав сервиса

Neutron состоит из следующих компонентов (для референсной архитектуры):

- **Neutron Server** (neutron-server) - это основной сервис, реализующий Neutron API и контролирующий передачу сообщений между сетевыми компонентами облачной платформы.
- **Плагин ML2** для поддержки OVN - это драйвер, позволяющий настраивать сетевые функции в SDN-контроллере OVN. Для этого требуется отдельная установка OVN-контроллера и баз данных Open vSwitch.
- **База данных neutron** хранит логическую схему сети облачной платформы, состояние объектов (например, портов) и данные сетевого доступа.
- **Очередь сообщений** необходим как протокол обмена сообщениями между частями сервиса Neutron.

При использовании классических схем установки Neutron в схеме добавляются дополнительные агенты (L3, DHCP и так далее), однако в случае использования OVN эти агенты не нужны, все сетевые функции реализуются SDN-контроллером.

Сетевые функции сервиса

Neutron предоставляет ряд сетевых сервисов:

- **Внутренняя сеть** (internal network) - это сети L2, предоставляемые конечным виртуальным машинам для подключения к сетевой инфраструктуре. Во внутренней сети используются **фиксированные IP-адреса**, выдаваемые сервисом **DHCP**, и создаются внутри оверлейных сетей типа VXLAN[101] или Geneve[102].
- **DHCP** для внутренней сети – сервис, необходимый для автоматического предоставления IP-адресов во внутренних сетях. Для каждой внутренней сети создаётся виртуальный сервер DHCP.
- **Внешняя сеть** (external network) – это сеть, предоставляющая сервисам внутренней сети получить доступ к сетям вне облачной платформы (например, в Интернет) и наоборот. Для этой задачи используются **маршрутизаторы**.
- **Маршрутизатор** (Router) – виртуальный сервис, предоставляющая возможность перенаправления трафика между внутренними сетями и в сети вне облачной платформы.
- **Плавающий IP-адрес** (Floating IP)– это адрес, предоставляющий возможность подключения к сервисам, расположенных во внутренних сетях, из сетей вне облачной платформы. Используется для внешних сетей. Плавающий IP-адрес использует технологию трансляции сетевых адресов (NAT[103]).
- **Служба метаданных** (Metadata Service) – специальный сетевой сервис, который предоставляет инстансам виртуальной машины доступ к его метаданным.

Дополнительные материалы

- Официальная документация[104] проекта.
  - Описание архитектуры[105] сервиса (дополнение[106]).
- Официальный репозиторий[107] проекта.
- Описание Neutron API[108].

100 https://docs.openstack.org/neutron/victoria/
101 https://ru.wikipedia.org/wiki/Virtual_Extensible_LAN
102 https://en.wikipedia.org/wiki/Generic_Network_Virtualization_Encapsulation
103 https://en.wikipedia.org/wiki/Network_address_translation
104 https://docs.openstack.org/neutron/victoria/
105 https://docs.openstack.org/neutron/victoria/install/overview.html
106 https://docs.openstack.org/security-guide/networking/architecture.html
107 https://opendev.org/openstack/neutron
108 https://docs.openstack.org/api-ref/network/v2/index.html

Установка OVN

В референсной архитектуре по умолчанию принято, что сетевые функции и интерфейс для управления ими предоставляется SDN-контроллером OVN (Open Virtual Network), который в свою очередь является оркестратором OpenFlow-правил для конечных виртуальных коммутаторов Open vSwitch[109], установленных в вычислительных узлах.

Кратко про OVN

OVN[110] - это комплекс сервисов для виртуального коммутатора Open vSwitch, позволяющие конвертировать описание виртуальной инфраструктуры в правила OpenFlow[111]. Основными конечными функциями OVN является создание сетевых функций: L3-маршрутизатора, DHCP-сервера и т. д.

Общая схема работы сервиса выглядит так:

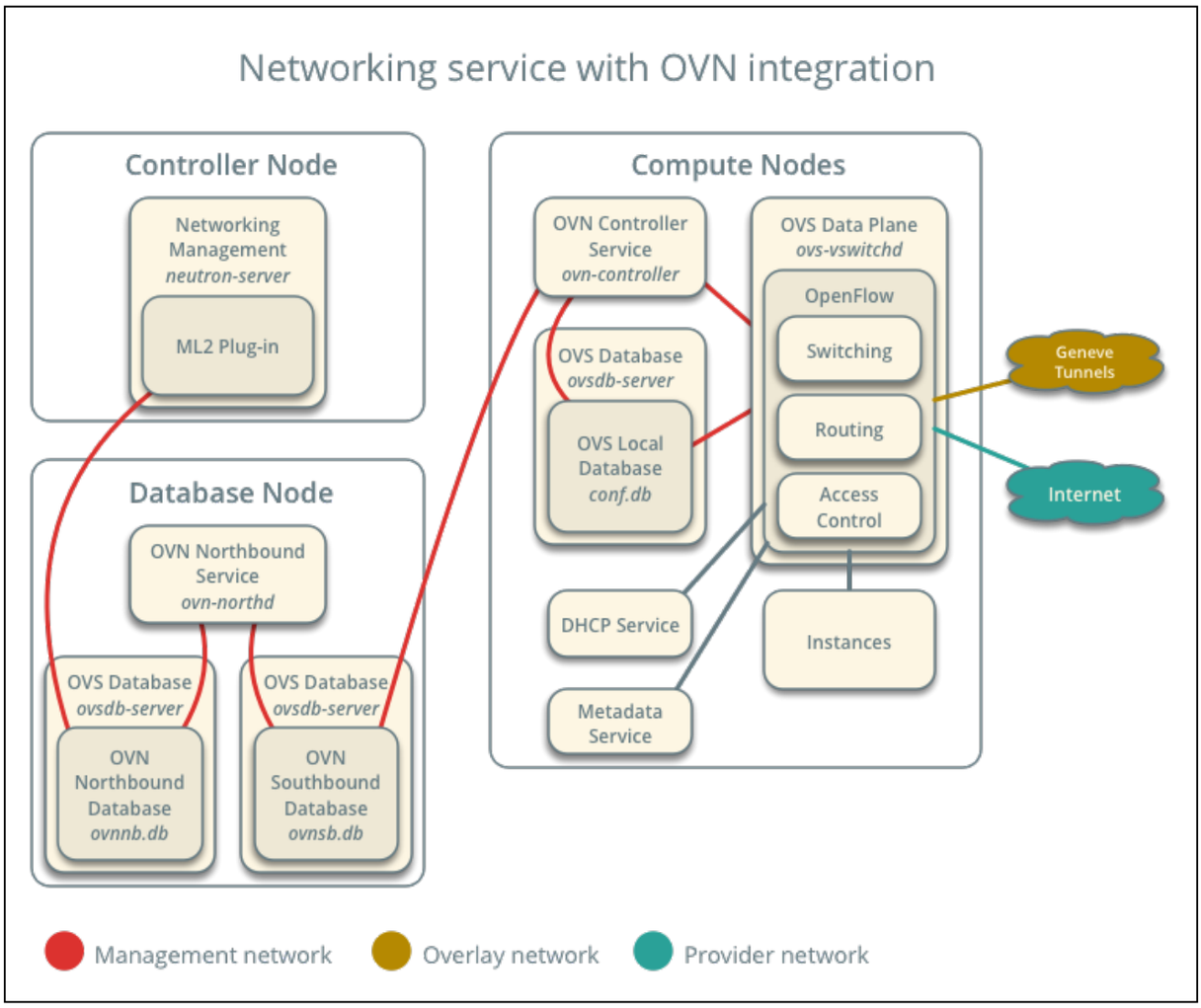


Схема работы сервиса

Комментарий по схеме:

- Neutron работает с OVN с помощью компонента ML2. Его настройка описана здесь (см. стр. 89).
- Database Node - это узел, где хранится сетевая конфигурация. В референсной архитектуре этот узел расположен в своем контейнере. При обычной установке этот узел совмещается с управляющим узлом.
- Компонент ML2 записывает логическую сетевую архитектуру облачной платформы в так называемую северную базу данных OVS (ovnnb.db).
- Южная база данных (ovnsb.db) хранит сконвертированные конечные правила OpenFlow.
- Сервис ovn-northd занимается конвертацией логической сетевой архитектуры в правила OpenFlow, которая хранится в северной базе, в южную базу данных OVS.
- В вычислительном узле устанавливается контроллер OVN, который получает правила OpenFlow с южной базы данных OVS помещает в локальную базу.
- Внутри OVS, использующий локальную базу создаются сетевые службы.

109 https://www.openvswitch.org/
110 https://www.ovn.org/en/
111 https://ru.wikipedia.org/wiki/OpenFlow

- Доступ к физической сетевой инфраструктуре осуществляется через физическое сетевое оборудование вычислительного узла.
- Управляющий узел не предоставляет сетевых функций.

Установка OVN в управляющем узле

OVN может быть установлен как с сервисом Neutron, так и отдельно, так как Neutron использует TCP-протокол для взаимодействия с SDN-контроллером. В референсной архитектуре принято, что Neutron и узел базы данных OVS хранятся в своих контейнерах. При обычной установке в управляющем узле нужно настроить два компонента: ovsdb-server от OVS и ovn-northd от OVN.

1. Установите пакеты openvswitch-ovn и networking-ovn:

```
dnf -y install rdo-ovn rdo-ovn-central rdo-ovn-host
```

2. Запустите сервис OVS и OVN, после чего включите их в список автозапуска:

```
systemctl start ovs-vswitchd openvswitch ovn-northd
systemctl enable openvswitch ovn-northd
```

3. Настройте доступ до северной и южной баз данных OVS по протоколу TCP с периодом проверки доступности:

```
ovn-nbctl set-connection ptcp:6641:0.0.0.0 -- \
          set connection . inactivity_probe=60000
ovn-sbctl set-connection ptcp:6642:0.0.0.0 -- \
          set connection . inactivity_probe=60000
```

> ⓘ При обычной установке вместо 0.0.0.0 для продуктивных систем используйте IP-адрес mgmt-интерфейса.

> ⚠ При наличии фаервола включите порты 6641 в узлах, где запущен neutron-server, и 6642, где запущен neutron-server, в шлюзовых машинах и вычислительных узлах.

> ⚠ На данный момент можно запустить лишь один активный инстанс сервиса ovn-northd. Для HA-варианта требуется использование системы кластеризации (Pacemaker и т.д.)

Установка OVN в вычислительных узлах

> ⚠ Установка OVN на данный момент предполагается только при использовании референсной архитектуры.

В вычислительных узлах нужно запустить сервис ovn-controller.

1. Установите пакеты openvswitch-ovn и networking-ovn:

```
dnf -y install rdo-ovn rdo-ovn-central rdo-ovn-host
```

2. Запустите сервис OVS и включите его в список автозапуска:

```
systemctl start ovs-vswitchd openvswitch
systemctl enable openvswitch
```

3. В OVS вычислительного узла укажите адрес южной базы данных OVS, которая запущена в управляющем узле:

```
ovs-vsctl set open . external-ids:ovn-remote=tcp:MGMT_CTRL_OVS_SN_DB_IP:6642
```

> ⓘ Вместо MGMT_CTRL_OVS_SN_DB_IP укажите адрес управляющего узла, где запущен инстанс ovn-northd.

4. Включите поддержку протокола оверлейной сети Geneve:

```
ovs-vsctl set open . external-ids:ovn-encap-type=geneve
```

5. Укажите IP-адрес сетевого интерфейса для оверлейной сети:

```
ovs-vsctl set open . external-ids:ovn-encap-ip=OVERLAY_NIC_IP_ADDRESS
```

6. Запустите сервис ovn-controller и добавьте его в автозапуск:

```
systemctl start ovn-controller
systemctl enable ovn-controller
```

7. Для верификации запуска службы запустите команду:

```
ovn-sbctl show
```

> Команда должна вернуть вывод без каких-либо сообщений.

Установка управляющих сервисов Neutron

Установка управляющих сервисов Neutron, а также драйвера ML2 состоит из нескольких шагов.

Настройка окружения

Подготовка базы данных neutron

> *См. также:* Установка и настройка СУБД MariaDB (см. стр. 22)

Состояние сервиса Neutron хранится в базе данных SQL, поэтому предварительно её надо создать.

1. Войдите в окружение базы данных:

```
mysql -u root -p
```

2. Создайте базу данных neutron:

```
create database neutron;
```

3. Предоставьте доступ к этой базе данных пользователю neutron в СУБД (для localhost и всем остальным адресам отдельно, вместо NEUTRON_DBPASS используйте свой пароль):

```
grant all privileges on neutron.* to 'neutron'@'localhost' identified by
'NEUTRON_DBPASS';
grant all privileges on neutron.* to 'neutron'@'%' identified by 'NEUTRON_DBPASS';
```

4. Выйдите из сессии СУБД:

```
exit;
```

Создание объектов в Keystone

> *См. также:* Файл настройки системного окружения (см. стр. 35)

Для сервиса Neutron необходимо создать пользователя и зарегистрировать его в сервисе каталогов Keystone.

1. Создайте пользователя neutron (команда интерактивно спросит пароль, далее этот пароль будет использоваться в NEUTRON_PASS):

```
openstack user create --domain default --password-prompt neutron
```

2. Добавьте пользователя neutron в проект service (см. стр. 39) с ролью admin:

```
openstack role add --project service --user neutron admin
```

3. Создайте сервис network в сервисе каталогов Keystone:

```
openstack service create --name neutron --description "OpenStack Network" network
```

4. Создайте три точки входа для сервиса network:
   a. публичную:

```
openstack endpoint create --region RegionOne network public http://
controller:9696
```

   b. внутреннюю:

```
openstack endpoint create --region RegionOne network internal http://
controller:9696
```

   c. административную:

```
openstack endpoint create --region RegionOne network admin http://controller:
9696
```

> Вместо домена "controller" укажите единый DNS-адрес для сервиса Neutron, который был выбран в вашей инфраструктуре. Использование IP-адресов не рекомендуется.

> Для разных типов точек входа API можно указать разные адреса. Это может быть необходимо в случае разделения, например, публичного трафика от внутреннего.

Установка сервиса Neutron

Установка сервиса Neutron API

Установка сервиса Neutron - достаточно не тривиальная операция, в которой есть чёткое разделение ролей. В этом разделе документация ограничится описанием сервиса Neutron API, остальные компоненты будут описаны в разделах далее.

1. Установите пакет сервиса Neutron и компонент ML2[112]:

```
dnf -y install openstack-neutron
```

> Стандартные пути конфигурации:
>
> - Каталог конфигурационных файлов: `/etc/neutron`
> - Основной файл конфигурации: `/etc/neutron/neutron.conf`
> - Каталог конфигурации плагинов Neutron ML2: `/etc/neutron/plugins/ml2`
> - Файл конфигурации ML2: `/etc/neutron/plugins/ml2/ml2.conf.ini`

2. Очистите основной файл конфигурации, который был добавлен после установки пакета:

```
> /etc/neutron/neutron.conf
```

3. Включите в основной файл конфигурации следующее (описание (см. стр. 91)):

112 https://docs.openstack.org/neutron/victoria/admin/config-ml2.html

```
[DEFAULT]
debug = False
bind_host = LISTEN_ADDR
log_dir = /var/log/neutron
#use_stderr = True
api_paste_config = /usr/share/neutron/api-paste.ini

# Workers
api_workers = 5
metadata_workers = 5
rpc_workers = 3
rpc_state_report_workers = 3

core_plugin = neutron.plugins.ml2.plugin.Ml2Plugin
service_plugins = neutron.services.ovn_l3.plugin.OVNL3RouterPlugin
allow_overlapping_ips = True
transport_url = rabbit://openstack:RABBIT_PASS@controller
auth_strategy = keystone
notify_nova_on_port_status_changes = True
notify_nova_on_port_data_changes = True

[database]
connection = mysql+pymysql://neutron:NEUTRON_DBPASS@controller/neutron
connection_recycle_time = 10
max_pool_size = 1
max_retries = -1

[keystone_authtoken]
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = neutron
password = NEUTRON_PASS
memcache_use_advanced_pool = True

[nova]
auth_url = http://controller:5000
auth_type = password
project_domain_name = Default
user_domain_name = Default
region_name = RegionOne
project_name = service
username = nova
password = NOVA_PASS
endpoint_type = internal

[placement]
auth_type = password
auth_url = http://controller:5000/v3
region_name = RegionOne
project_domain_name = default
project_name = service
user_domain_name = default
username = placement
password = PLACEMENT_PASS
os_interface = internal

[oslo_concurrency]
lock_path = /var/lib/neutron/tmp

[oslo_middleware]
enable_proxy_headers_parsing = True
```

```
[oslo_messaging_notifications]
driver = messagingv2
```

Настройка компонента ML2

Neutron ML2 - это фреймворк, позволяющий унифицировать реализацию сетевых функций при создании сетевых драйверов и имплементации протоколов. Подробнее об ML2 описано здесь[113].

В рамках типов установки принято следующее:

- Драйвер ML2 OVN используется при установке по схеме референсной архитектуры;
- Драйвер ML2 OVS используется при установке по схеме классической архитектуры.

OVN

1. Установите пакет компонента ML2:

```
dnf -y install openstack-neutron-ml2
```

> Стандартные пути конфигурации:
>
> - Каталог конфигурации плагинов Neutron ML2: `/etc/neutron/plugins/ml2`
> - Основной файл конфигурации ML2: `/etc/neutron/plugins/ml2/ml2_conf.ini`

2. Очистите основной файл конфигурации, который был добавлен после установки пакета:

```
> /etc/neutron/plugins/ml2/ml2.conf.ini
```

3. Включите в основной файл конфигурации следующее (описание (см. стр. 91)):

> Перед настройкой конфигурации убедитесь, что вы корректно настроили службу OVN и все его компоненты успешно запущены. Подробнее здесь (см. стр. 84).

```
[ml2]
mechanism_drivers = ovn
type_drivers = local,flat,vlan,geneve
tenant_network_types = geneve
extension_drivers = port_security
overlay_ip_version = 4

[ml2_type_geneve]
vni_ranges = 1:65536
max_header_size = 38

[ml2_type_vlan]
network_vlan_ranges = provider

[securitygroup]
enable_security_group = true

[ovn]
ovn_nb_connection = tcp:10.0.0.11:6641
ovn_sb_connection = tcp:10.0.0.11:6642
ovn_l3_scheduler = leastloaded
ovn_metadata_enabled = true
```

4. Включите эту конфигурацию ML2 созданием символической ссылки в каталоге сервиса Neutron:

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

Финализация установки

1. Инициализируйте базу данных neutron:

---

113 https://docs.openstack.org/neutron/victoria/admin/config-ml2.html

```
su -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
```

⚠ Обратите внимание на то, после инициализации базы данных содержимое конфигурации ML2 менять настоятельно не рекомендуется из-за потенциально возможных конфликтов во время будущих миграций базы данных Neutron. Включите все необходимые параметры ML2 перед инициализацией базы данных.

2. Запустите сервис Neutron API и добавьте его в автозапуск:

```
systemctl start neutron-server
systemctl enable neutron-server
```

Проверка работы сервиса

1. Проверьте статус сервиса Neutron API:

```
systemctl status neutron-server
```

ⓘ В ответ вы должны получить примерно следующий вывод:

```
● neutron-server.service - OpenStack Neutron Server
   Loaded: loaded (/usr/lib/systemd/system/neutron-server.service;
disabled; vendor preset: disabled)
  Drop-In: /run/systemd/system/neutron-server.service.d
           └─zzz-lxc-service.conf
   Active: active (running) since Mon 2021-11-08 10:39:17 UTC; 6s ago
 Main PID: 59873 (/usr/bin/python)
    Tasks: 13 (limit: 204240)
   Memory: 226.3M
   CGroup: /system.slice/neutron-server.service
           ├─59873 /usr/bin/python3 /usr/bin/neutron-server --config-file /
usr/share/neutron/neutron-dist.conf --config-dir /usr/share/neutron/server
--config-file /etc/neutron/neutro>
 ...
```

2. Проверьте статус порта:

```
ss -tnlp | grep 9696
```

ⓘ В ответ вы должны получить примерно следующий вывод:

```
LISTEN 0      4096   10.236.64.231:9696       0.0.0.0:*    users:
(("neutron-server:",pid=59890,fd=6),
```

Порт 9696 должен иметь статус LISTEN и слушать адрес, указанный в bind_host (с учетом резолвинга имён DNS).

3. Проверьте статус Neutron API:

```
curl http://controller:9696
```

ⓘ В ответ вы должны получить примерно следующий вывод:

```
{"versions": [{"id": "v2.0", "status": "CURRENT", "links": [{"rel": "self",
"href": "http://controller:9696/v2.0/"}]}]}
```

Конфигурация для HAProxy

См. также: Установка балансировщика нагрузки HAProxy (см. стр. 16)

1. В референсной архитектуре доступ до сервиса Neutron Server предоставляется через балансировщика нагрузки. Конфигурация для HAProxy выглядит следующим образом:

```
frontend neutron
  bind "MGMT_IP:9696" ssl crt /usr/local/etc/haproxy/cert.pem alpn h2,http/1.1
  http-request set-header X-Forwarded-Proto https
  default_backend neutron_backend

backend neutron_backend
  server neutron 127.0.0.1:9696
```

2. После включения этой конфигурации перезагрузите конфигурацию HAProxy:

```
systemctl reload haproxy
```

Настройка вычислительного узла

В случае референсной архитектуры отдельная настройка сервиса Neutron в вычислительном узле не требуется. Достаточно настройки сервиса контроллера OVN. Эта настройка описана здесь (см. стр. 85).

Описание файла конфигурации Neutron

В описании процесса по установке сервиса Neutron предложена стандартная конфигурация. Это страница содержит подробное описание настроек этой конфигурации.

При изменении конфигурации необходимо перезапустить компоненты сервиса Neutron:

```
systemctl restart openstack-neutron-*
```

Документация по всем параметрам сервиса Neutron доступна по этой ссылке

Таблица конфигурации

Компоненты сервиса Neutron

Путь до конфигурации: `/etc/neutron/neutron.conf`

Легенда таблицы доступна на этой странице[114].

Имя параметра	Описание	Примечание
[DEFAULT]	Глобальные параметры сервиса Neutron	
debug	Включение режима отладки для службы Neutron	Включайте только при проблемах с платформой.
use_stderr	Перенаправлять журналы в /dev/stderr	В референсной архитектуре необходимо для перенаправления журналов в систему управления контейнерами. При обычной установке нужно установить в False и указать параметр log_dir = /var/log/neutron

114 https://conf.tionix.ru/pages/viewpage.action?pageId=205881354#id-Условныеобозначения-table-format-desc

Имя параметра	Описание	Примечание
bind_host	Адрес прослушивания сервиса Neutron	В референсной архитектуре должен быть настроен на loopback. При обычной установке укажите адрес 0.0.0.0
api_paste_config	Путь до конфигурации API	
***_workers**	Количество процессов для каждого из подсервисов Neutron	
core_plugin	Основной плагин управления сетевой частью.	Всегда должен использоваться драйвер ML2.
service_plugins	Плагины для активации сетевых функций	Для референсной архитектуры нужно использовать L3 OVN Router. Для обычной установки можно использовать обычный L3 Router.
allow_overlapping_ips	Возможность использования одинаковых адресов и диапазона подсетей для виртуальных сетей.	
transport_url	Адреса сервисов RabbitMQ для обмена сообщениями между сервисов OpenStack.	Можно указать через запятую, указание префикса rabbit:// для каждого узла обязателен.
auth_strategy	Включение механизма аутентификации.	
notify_nova_on_port_status_changes	Включение оповещения сервиса Nova об изменении статуса сетевого порта.	
notify_nova_on_port_data_changes	Включение оповещения сервиса Nova об изменении данных сетевого порта (напр. фиксированного IP-адреса).	
[database]	Параметры соединения к базам данных Neutron.	
connection	Адрес подключения к базе данных neutron.	
connection_recycle_time	Время для восстановления соединения до базы данных.	
max_pool_size	Максимальный размер пула соединений к базе данных.	
max_retries	Максимальное количество попыток соединения до базы данных.	-1 выключает ограничение на количество попыток соединения
[keystone_authtoken]	Параметры подключения к сервису Keystone.	
www_authenticate_uri	Адрес публичного API Keystone.	
auth_url	Адрес внутреннего API Keystone.	

Имя параметра	Описание	Примечание
memcached_servers	Адреса внешней системы кэширования Memcached.	Можно указать несколько серверов через запятую.
auth_type	Тип аутентификации	
project_domain_name **user_domain_name** **project_name** **username** **password**	Данные для аутентификации в Keystone от имени пользователя neutron	
[nova]	Данные подключения к сервису Nova через получение точек входа в Keystone.	
auth_url	Адрес внутреннего API Keystone.	
auth_type	Тип аутентификации.	
project_domain_name **user_domain_name** **project_name** **username** **password**	Данные для аутентификации в Keystone от имени пользователя nova.	
endpoint_type	Указание типа точки входа для подключения к сервису Nova.	
[placement]	Параметры подключения к сервису Placement.	
auth_type	Тип аутентификации.	
auth_url	Адрес внутреннего API Keystone.	
region_name	Имя региона, зарегистрированное в Keystone.	
project_domain_name **user_domain_name** **project_name** **username** **password**	Данные для аутентификации в Keystone от имени пользователя placement.	
[oslo_concurrency]	Параметры обработки конкурентных задач в Oslo.	
lock_path	Каталог для файлов блокировок.	
[oslo_middleware]	Параметры для запросов к сервису Nova.	
enable_proxy_headers_parsing	Включение обработки заголовков прокси-сервера.	Внимание: используйте только в случае наличия reverse proxy, иначе не указывайте этот параметр.
[oslo_messaging_notifications]	Параметры оповещений.	Для этого параметра необходимо указание адреса RabbitMQ в transport_url.
driver	Тип драйвера оповещений.	В референсе всегда должен быть равен "messagingv2".

Настройки компонента ML2

Путь до конфигурации: `/etc/neutron/plugins/ml2/ml2_conf.ini`

OpenStack Cinder

Информация о сервисе Cinder

OpenStack Cinder[115] - это сервис, который отвечает за управление блочными системами хранения облачной платформы. Конечной функцией Cinder является создание диска (volume), который потом можно примонтировать в инстанс.

Cinder в Tionix Cloud Platfrom официально поддерживает работу со следующими типами систем хранения:

- **iSCSI/FC LUN** - обеспечивает возможность подключения LUN внешних систем хранения по протоколам iSCSI или FC. Для вендоров проприетарных систем предлагаются отдельные плагины управления.
  - В простом варианте можно использовать тома LVM с iSCSI-таргетом на базе Linux.
- **Ceph** - обеспечивает создание RBD-устройств системы хранения Ceph.

Cinder состоит из нескольких отдельных сервисов:

- **cinder-api** - это сервис, реализующий общие функции управления дисками, а так же предоставляет Cinder API[116];
- **cinder-scheduler** - сервис, отвечающий за планирование задач над дисками;
- **База данных cinder** - база данных SQL, которая хранит статус всех созданных объектов в Cinder и добавленных систем хранения.

Так же для дальнейшего понимания настройки Cinder нужно объяснить значение ещё нескольких терминов:

- **Диск (volume)** - это, как уже было сказано, является конечным объектом, который предоставляет сервис. Диск всегда является блочным устройством (даже при использовании файловой системы, тогда в качестве конечных блочных устройств будут использоваться файлы с таблицей разметки).
- **Бэкенд (backend)** - это часть конфигурации, который содержит данные подключения к внешней системе хранения. Cinder может поддерживать несколько бэкендов.
- **Тип диска (volume type)** - это объект Cinder, позволяющий разделять типы систем хранения данных. Тип диска может содержать один или несколько бэкендов.

Установка управляющей части

Настройка окружения

Перед самой установкой сервиса нужно предварительно настроить некоторые компоненты инфраструктуры.

Подготовка базы данных glance

См. также: Установка и настройка СУБД MariaDB (см. стр. 22)

Всю информацию о данных образов и метаданных по умолчанию Cinder хранит в базе данных SQL.

1. Войдите в окружение базы данных:

```
mysql -u root -p
```

2. Создайте базу данных cinder:

```
create database cinder;
```

3. Предоставьте доступ к этой базе данных пользователю cinder в СУБД (для localhost и всем остальным адресам отдельно):

115 https://docs.openstack.org/cinder/victoria/
116 https://docs.openstack.org/api-ref/block-storage/

```
grant all privileges on cinder.* to 'cinder'@'localhost' identified by
'CINDER_DBPASS';
grant all privileges on cinder.* to 'cinder'@'%' identified by 'CINDER_DBPASS';
```

Вместо CINDER_DBPASS используйте свой пароль, он будет необходим далее.

4. Выйдите из сессии СУБД:

```
exit;
```

Создание объектов в Keystone

См. также: Файл настройки системного окружения (см. стр. 35).

Для сервиса Cinder необходимо создать пользователя и зарегистрировать его в сервисе каталогов Keystone.

1. Создайте пользователя cinder (команда интерактивно спросит пароль, далее этот пароль будет использоваться в CINDER_PASS):

```
openstack user create --domain default --password-prompt cinder
```

2. Добавьте пользователя cinder в проект service с ролью admin:

```
openstack role add --project service --user cinder admin
```

3. Создайте сервис cinderv2 в сервисе каталогов Keystone:

```
openstack service create --name cinderv2 --description "OpenStack Block Storage"
 volumev2
```

4. Создайте сервис cinderv3 в сервисе каталогов Keystone:

```
openstack service create --name cinderv3 --description "OpenStack Block Storage"
 volumev3
```

5. Создайте три точки входа для каждого из созданных сервисов:
   a. публичные:

```
openstack endpoint create --region RegionOne volumev2 public http://
controller:8776/v2/%\(project_id\)s
openstack endpoint create --region RegionOne volumev3 public http://
controller:8776/v3/%\(project_id\)s
```

   b. внутреннюю:

```
openstack endpoint create --region RegionOne volumev2 internal http://
controller:8776/v2/%\(project_id\)s
openstack endpoint create --region RegionOne volumev3 internal http://
controller:8776/v3/%\(project_id\)s
```

   c. административную:

```
openstack endpoint create --region RegionOne volumev2 admin http://
controller:8776/v2/%\(project_id\)s
openstack endpoint create --region RegionOne volumev3 admin http://
controller:8776/v3/%\(project_id\)s
```

Вместо домена "controller" укажите единый DNS-адрес для сервиса Glance, который был выбран в вашей инфраструктуре. Использование IP-адресов не рекомендуется.

Установка сервиса Cinder

> *См. также:* Настройка репозиториев Almalinux (см. стр. 10).

После установки всех необходимых внешних сервисов можно приступить к установке сервиса Cinder.

1. Установите основной пакет сервиса Cinder:

   ```
   dnf -y install openstack-cinder
   ```

   > Стандартные пути конфигурации:
   >
   > - Каталог конфигурационных файлов: `/etc/cinder`
   > - Основной файл конфигурации: `/etc/cinder/cinder.conf`

2. Очистите основной файл конфигурации, который был добавлен после установки пакета:

   ```
   > /etc/cinder/cinder.conf
   ```

3. Включите в основной файл конфигурации следующее (описание (см. стр. 102)):

```
[DEFAULT]
# Общие параметры Cinder
my_ip = MGMT_IP
osapi_volume_listen = $my_ip
osapi_volume_workers = 5
transport_url = rabbit://openstack:RABBIT_PASS@controller:5672
auth_strategy = keystone
use_forwarded_for = True
volume_name_template = volume-%s
glance_api_servers = http://controller:9292
glance_api_version = 2
api_paste_config = /etc/cinder/api-paste.ini

# Параметры журналирования
debug = False
log_dir = /var/log/cinder

[oslo_middleware]
enable_proxy_headers_parsing = True

[database]
connection = mysql+pymysql://cinder:CINDER_DBPASS@controller/cinder
connection_recycle_time = 10
max_pool_size = 1
max_retries = -1

[oslo_concurrency]
lock_path = /var/lib/cinder/lock

[keystone_authtoken]
www_authenticate_uri = http://controller:5000
auth_url = http://controller:5000
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = cinder
password = CINDER_PASS

[nova]
auth_url = http://controller:5000
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = nova
password = NOVA_PASS
endpoint_type = internal
```

4. При настройке сервиса следует обратить особое внимание на следующие параметры:

> ⓘ Легенда таблицы доступна на этой странице[117].

Параметр	Значение
my_ip	Необходимо указать адрес узла в интерфейсе сети управления. Не используйте адрес DNS.
osapi_volume_listen	Необходимо указать в зависимости от типа установки: • Классическая архитектура: $my_ip • Референсная архитектура: 0.0.0.0

117 https://conf.tionix.ru/pages/viewpage.action?pageId=205881354#id-Условныеобозначения-table-format-desc

Параметр	Значение
transport_url	Укажите данные RabbitMQ, в частности: • имя пользователя в RabbitMQ (в примере: openstack) • пароль пользователя в RabbitMQ вместо RABBIT_PASS • адрес до сервиса RabbitMQ (в примере: controller)
glance_api_servers	Адреса до сервисов Glance.
Параметры журналирования	Эти параметры зависят от типа установки: • Классическая архитектура: нужно использовать предложенный в конфигурации параметр **log_dir** или **use_journal = True** • Референсная архитектура: нужно использовать параметр **use_stderr = True** Параметр debug можно указать во всех типах установки.
database/ connection	Укажите данные доступа к базе данных, в частности, • имя пользователя в MariaDB (в примере: cinder) • пароль пользователя в MariaDB вместо CINDER_DBPASS • адрес сервиса MariaDB (в примере: controller)
keystone_authtoken	Укажите корректные адреса до сервиса Keystone, в частности, пароль для пользователя cinder вместо CINDER_PASS. • Отдельно корректно укажите адреса серверов memcached в memcached_servers.
nova	Укажите корректные адреса до сервиса Keystone, в частности, пароль для пользователя nova вместо NOVA_PASS.

Все параметры, указанные в конфигурации, описаны в этой странице (см. стр. 102).

Финализация установки

1. Инициализируйте базу данных cinder:

```
su -s /bin/sh -c "cinder-manage db sync" cinder
```

2. Запустите сервисы Cinder и включите их в автозапуск:

```
systemctl start openstack-cinder-api openstack-cinder-scheduler
systemctl enable openstack-cinder-api openstack-cinder-scheduler
```

Проверка работы сервиса

1. Проверьте статус сервиса Cinder API и планировщика:

```
systemctl status openstack-cinder-api openstack-cinder-scheduler
```

Ответ должен быть примерно таким:

```
● openstack-cinder-api.service - OpenStack Cinder API Server
   Loaded: loaded (/usr/lib/systemd/system/openstack-cinder-api.service;
enabled; vendor preset: disabled)
  Drop-In: /run/systemd/system/openstack-cinder-api.service.d
           └─zzz-lxc-service.conf
   Active: active (running) since Tue 2021-11-09 19:36:29 UTC; 3s ago
 Main PID: 153801 (cinder-api)
    Tasks: 6 (limit: 204240)
   Memory: 145.3M
   CGroup: /system.slice/openstack-cinder-api.service
           ├─153801 /usr/bin/python3 /usr/bin/cinder-api --config-file /
usr/share/cinder/cinder-dist.conf --config-file /etc/cinder/cinder.conf --
logfile /var/log/cinder/api.log
...
Nov 09 19:36:29 tnx-mgmt-almalinux systemd[1]: Started OpenStack Cinder API
Server.

● openstack-cinder-scheduler.service - OpenStack Cinder Scheduler Server
   Loaded: loaded (/usr/lib/systemd/system/openstack-cinder-
scheduler.service; enabled; vendor preset: disabled)
  Drop-In: /run/systemd/system/openstack-cinder-scheduler.service.d
           └─zzz-lxc-service.conf
   Active: active (running) since Tue 2021-11-09 19:36:29 UTC; 3s ago
 Main PID: 153802 (cinder-schedule)
    Tasks: 4 (limit: 204240)
   Memory: 131.2M
   CGroup: /system.slice/openstack-cinder-scheduler.service
           └─153802 /usr/bin/python3 /usr/bin/cinder-scheduler --config-
file /usr/share/cinder/cinder-dist.conf --config-file /etc/cinder/
cinder.conf --logfile /var/log/cinder/schedul>

Nov 09 19:36:29 tnx-mgmt-almalinux systemd[1]: Started OpenStack Cinder
Scheduler Server.
```

2. Проверьте статус порта Cinder API:

```
ss -tnlp | grep 8776
```

Ответ должен быть примерно таким:

```
LISTEN 0      128    10.236.64.231:8776       0.0.0.0:*    users:(("cinder-
api",pid=153822,fd=8),...
```

Порт 8776 должен иметь статус LISTEN и слушаться IP-адрес интерфейса сети управления (mgmt).

3. Получите статус API:

```
curl http://controller:8776
```

Ответ должен быть примерно таким:

```
{"versions": [{"id": "v2.0", "status": "DEPRECATED", "version": "",
"min_version": "", "updated": "2017-02-25T12:00:00Z",...
```

4. Получите список дисков:

```
openstack volume list
```

> При успешном выполнении команда вернёт пустой вывод, так как на этот момент установки в Cinder не был добавлен ни один диск.

5. Также можно получить список зарегистрированных сервисов Cinder:

```
openstack volume service list
```

6. Команда должна вернуть информацию, что cinder-scheduler успешно запущен:

```
+------------------+------------+------+---------+-------
+----------------------------+
| Binary           | Host       | Zone | Status  | State | Updated_at
|
+------------------+------------+------+---------+-------
+----------------------------+
| cinder-scheduler | controller | nova | enabled | up    | 2016-09-30T02:27:
41.000000 |
+------------------+------------+------+---------+-------
+----------------------------+
```

Конфигурация для HAProxy

> *См. также:* Установка балансировщика нагрузки HAProxy (см. стр. 16).

В референсной архитектуре доступ до сервиса Cinder API предоставляется через балансировщика нагрузки.

1. Конфигурация для HAProxy выглядит следующим образом:

```
frontend cinder
  bind "MGMT_IP:8776" ssl crt /usr/local/etc/haproxy/cert.pem alpn h2,http/1.1
  http-request set-header X-Forwarded-Proto https
  default_backend cinder_backend

backend cinder_backend
  server cinder 127.0.0.1:8776
```

2. После включения этой конфигурации перезагрузите конфигурацию HAProxy:

```
systemctl reload haproxy
```

Настройка бэкенда Cinder

Cinder поддерживает различные системы хранения и их настройка немного отличается друг от друга. В этом разделе приведён пример настройки простого бэкенда, который содержит данные подключения к системе хранения Ceph. Про саму настройку бэкенда описание будет предложено в другом разделе, в данном разделе рассматривается просто пример того, как добавляется условная система хранения в облачную платформу.

Бэкенды обрабатываются сервисом cinder-volume.

Установка сервиса cinder-volume

cinder-volume по своей сути является промежуточным сервисом между облачной платформой и системами хранения и фактически занимается их управлением. В зависимости от задачи cinder-volume может быть установлен:

- в отдельном узле (обычно это касается систем хранения на базе LVM);
- в управляющих узлах (если API системы хранения работает по сетевым протоколам);
- отдельно в своём контейнере.

В референсной архитектуре по умолчанию используется последний вариант, при обычной установке - в основном второй. В любом случае нужно установить этот сервис и настроить его.

1. В выбранном узле установите пакет сервиса cinder-volume:

```
dnf -y install openstack-cinder
```

2. Если cinder-volume был установлен в управляющем узле, отдельной настройки файла конфигурации `/etc/cinder/cinder.conf` на данном шаге не требуется, иначе необходимо использовать конфигурацию Cinder, описанном в этом разделе (см. стр. 96).
3. Перезапустите сервис cinder-volume и добавьте его в автозапуск:

```
systemctl start openstack-cinder-volume
systemctl enable openstack-cinder-volume
```

Создание нового бэкенда

Каждая система хранения должна быть добавлена в Cinder в качестве отдельного бэкенда.

1. Для этого в конфигурацию `/etc/cinder/cinder.conf` в узле с cinder-volume нужно добавить отдельную секцию с именем этого бэкенда:

```
[blk]
volume_driver = cinder.volume.drivers.rbd.RBDDriver
volume_backend_name = backend_blk
rbd_pool = volumes
rbd_ceph_conf = /etc/ceph/ceph.conf
rbd_flatten_volume_from_snapshot = false
rbd_max_clone_depth = 5
rbd_store_chunk_size = 4
rados_connect_timeout = -1
```

2. Далее в секции [DEFAULT] нужно явно включить добавленный бэкенд:

```
[DEFAULT]
...
enabled_backends = blk
...
```

> В enabled_backends указывается имя секции файла конфигурации, указанный в квадратных скобках.

3. После чего перезапустите службу cinder-volume:

```
systemctl restart cinder-volume
```

Проверка добавления бэкенда

1. Убедитесь, что бэкенд, добавленный в cinder-volume, отображается в списке сервисов Cinder:

```
openstack volume service list
```

2. В ответ на это вы должны получить примерно следующее:

```
+------------------+------------+------+---------+-------
+----------------------------+
| Binary           | Host       | Zone | Status  | State | Updated_at
|
+------------------+------------+------+---------+-------
+----------------------------+
| cinder-scheduler | controller | nova | enabled | up    | 2016-09-30T02:27:
41.000000 |
| cinder-volume    | vlm01@blk  | nova | enabled | up    | 2016-09-30T02:27:
46.000000 |
+------------------+------------+------+---------+-------
+----------------------------+
```

Создание типа диска

По умолчанию один cinder-volume рассчитан на работу с одним бэкендом. Однако имеется возможность включения нескольких бэкендов в одном инстансе cinder-volume. Для этого дополнительно нужны так

называемые *типы дисков*, которые позволяют разделять бэкенды между собой при работе с блочными устройствами.

1. Вначале создайте ещё один бэкенд. Для этого просто добавьте ещё одну секцию в конфигурацию, как при настройке первого бэкенда:

```
[blk2]
volume_driver = cinder.volume.drivers.rbd.RBDDriver
volume_backend_name = backend_blk2
rbd_pool = volumes2
rbd_ceph_conf = /etc/ceph/ceph2.conf
rbd_flatten_volume_from_snapshot = false
rbd_max_clone_depth = 5
rbd_store_chunk_size = 4
rados_connect_timeout = -1
```

2. Добавьте этот бэкенд в список включенных:

```
[DEFAULT]
...
enabled_backends = blk, blk2
...
```

3. Перезапустите сервис cinder-volume:

```
systemctl restart cinder-volume
```

4. Создайте типы диска:

```
openstack volume type create ceph-blk
openstack volume type create ceph-blk2
```

5. Теперь необходимо связать между собой бэкенд и тип диска:

```
openstack volume type set ceph-blk --property volume_backend_name=blk
openstack volume type set ceph-blk2 --property volume_backend_name=blk2
```

6. После этого при создании диска можно выбрать нужный тип диска для того, чтобы блочное устройство создалось в нужном хранилище:

```
openstack volume create --size 1 --type ceph-blk2 pretty_volume_name
```

7. Отдельно необходимо отметить, что для двух и более бэкендов можно указать одно имя бэкенда в параметре **volume_backend_name** (имена секций при этом должны различаться). В этом случае при создании блочного устройства с указанием типа диска планировщик сам выберет нужный бэкенд в зависимости от параметров фильтрации планировщика, например, в зависимости от заполненности бэкендов хранения.

Описание файла конфигурации Cinder

В описании процесса по установке сервиса Cinder предложена стандартная конфигурация. Это страница содержит подробное описание настроек этой конфигурации.

При изменении конфигурации необходимо перезапустить компоненты сервиса Cinder:

```
systemctl restart openstack-cinder-*
```

Таблица конфигурации

Компоненты сервиса Cinder

ⓘ Путь до конфигурации: `/etc/cinder/cinder.conf`

ⓘ Легенда таблицы доступна на этой странице[118].

118 https://conf.tionix.ru/pages/viewpage.action?pageId=205881354#id-Условныеобозначения-table-format-desc

Имя параметра	Описание	Примечания
[DEFAULT]	Глобальные настройки сервиса Cinder.	
debug	Включение режима отладки.	Используйте только в режиме обслуживания или тестирования.
use_stderr	Перенаправление журнала в /dev/stderr.	Этот параметр используется в референсной архитектуре для перенаправления журнала в систему управления контейнерами.
use_forwarded_for	Включение поддержки заголовка X-Forwarded-For.	Этот параметр нужно включать при наличии прокси-сервера или балансировщика нагрузки.
osapi_volume_workers	Количество процессов для Cinder API.	
volume_name_template	Шаблон именования дисков.	
osapi_volume_listen	Адрес прослушивания сервиса Cinder.	В референсной архитектуре нужно указать 127.0.0.1. Для обычной установки укажите адрес mgmt-интерфейса.
auth_strategy	Указание механизма аутентификации.	Всегда должен быть равен значению "keystone".
my_ip	Адрес сервиса Cinder.	Необходимо указать адрес mgmt-интерфейса.
glance_api_servers	Адреса сервисов Glance API.	Можно указать несколько адресов Glance через запятую.
glance_api_version	Указание версии Glance API.	Должен быть равен "2".
transport_url	Адрес сервиса RabbitMQ.	Можно указать несколько адресов RabbitMQ через запятую, указание префикса rabbit:// обязательно для каждого адреса.
api_paste_config	Путь до файла конфигурации Cinder API.	
[oslo_middleware]	Параметры обработки запросов в WSGI.	
enable_proxy_headers_parsing	Включение обработки заголовков прокси-сервера.	Этот параметр нужно включать при наличии прокси-сервера или балансировщика нагрузки.
[database]	Параметры базы данных cinder.	
connection	Параметры подключения к базе данных cinder.	
connection_recycle_time	Время для восстановления подключения к базе данных.	
max_pool_size	Максимальный размер пула соединений.	

Имя параметра	Описание	Примечания
max_retries	Максимальное количество попыток соединения к базе данных.	"-1" выключает ограничение по количеству попыток.
[oslo_concurrency]	Параметры обработки конкурентных задач.	
lock_path	Каталог для хранения файлов блокировок.	
[keystone_authtoken]	Параметры подключения к сервису Keystone.	
www_authenticate_uri	Внешний адрес сервиса Keystone	
auth_url	Внутренний адрес сервиса Keystone	
memcached_servers	Адрес сервисов кэширования Memcached	Можно указать несколько адресов Memcached через запятую.
auth_type	Тип аутентификации.	
project_domain_name **user_domain_name** **project_name** **username** **password**	Данные подключения к сервису Keystone от имени cinder.	
[nova]	Данные подключения к Keystone для получения адреса сервиса Nova.	
auth_url	Внутренний адрес сервиса Keystone.	
auth_type	Тип аутентификации.	
project_domain_name **user_domain_name** **project_name** **username** **password**	Данные подключения к сервису Keystone от имени nova.	

OpenStack Ceilometer

Информация о сервисе Ceilometer

Обзор сервиса

OpenStack Telemetry (Ceilometer)[119] - это служба сбора телеметрических данных, которая предоставляет возможность собирать данные мониторинга с основных компонентов OpenStack, производить нормализацию и хранение собранных данных в одном из поддерживаемых систем хранения. По умолчанию в качестве системы хранения используется Gnocchi[120]. Проект Gnocchi позволяет хранить собираемые измерения в агрегированном виде и предоставляет API для работы с ними.

119 https://docs.openstack.org/ceilometer/victoria/
120 https://gnocchi.osci.io

Основные компоненты

OpenStack Ceilometer состоит из следующих агентов:

- **ceilometer-agent-compute** - запускается на вычислительных узлах и опрашивает статистику использования ресурсов;
- **ceilometer-agent-central** - запускается на управляющих узлах, опрашивает статистику использования ресурсов для объектов, не привязанных к экземплярам или вычислительным узлам. Допускается запуск нескольких экземпляров.
- **ceilometer-agent-notification** - запускается на управляющих узлах, обрабатывает данные из очередей сообщений для создания данных о событиях и измерениях. По умолчанию обработанные данные отправляются в Gnocchi[121].

Дополнительные материалы

1. Официальная документация[122] проекта.
2. Официальный репозиторий[123] проекта.
3. Ceilometer CLI[124].

Установка управляющих сервисов Ceilometer

Установка сервисов Gnocchi

Настройка окружения

Перед самой установкой сервиса нужно предварительно настроить некоторые компоненты инфраструктуры.

Подготовка базы данных gnocchi

См. также: Установка и настройка СУБД MariaDB (см. стр. 22)

Информацию индекса метрик по умолчанию Gnocchi хранит в базе данных SQL.

1. Войдите в окружение базы данных:

```
mysql -u root -p
```

2. Создайте базу данных gnocchi:

```
create database gnocchi;
```

3. Предоставьте доступ к этой базе данных пользователю gnocchi в СУБД (для localhost и всем остальным адресам отдельно):

```
grant all privileges on gnocchi.* to 'gnocchi'@'localhost' identified by
'GNOCCHI_DBPASS';
grant all privileges on gnocchi.* to 'gnocchi'@'%' identified by 'GNOCCHI_DBPASS';
```

Вместо GNOCCHI_DBPASS используйте свой пароль, он будет необходим далее.

4. Выйдите из сессии СУБД:

```
exit;
```

Создание объектов в Keystone

См. также: Файл настройки системного окружения (см. стр. 38) и Создание объектов в Keystone (см. стр. 39).

Для сервиса Glance необходимо создать пользователя и зарегистрировать его в сервисе каталогов Keystone.

---

121 https://gnocchi.osci.io
122 https://docs.openstack.org/ceilometer/victoria/
123 https://opendev.org/openstack/ceilometer
124 https://docs.openstack.org/ceilometer/victoria/cli/index.html

1. Настройте окружение командной строки:

```
source $HOME/admin-openrc
```

2. Создайте пользователя gnocchi (команда интерактивно спросит пароль, далее этот пароль будет использоваться в параметрах, где указан GNOCCHI _PASS):

```
openstack user create --domain default --password-prompt gnocchi
```

3. Добавьте пользователя ceilometer в проект service (см. стр. 39) с ролью admin:

```
openstack role add --project service --user gnocchi admin
```

4. Создайте сервис metric в сервисе каталогов Keystone:

```
openstack service create --name gnocchi --description "Metric Service" metric
```

5. Создайте три точки входа для сервиса gnocchi:
   a. публичную:

```
openstack endpoint create --region RegionOne metric public http://controller:
8041
```

   b. внутреннюю:

```
openstack endpoint create --region RegionOne metric internal http://
controller:8041
```

   c. административную:

```
openstack endpoint create --region RegionOne metric admin http://controller:
8041
```

> controller используется в качестве примера адреса. В продуктивных средах вместо домена "controller" укажите единый DNS-адрес для сервиса Glance, который был выбран в вашей инфраструктуре. Использование IP-адресов не рекомендуется.

> Для разных типов точек входа API можно указать разные адреса. Это может быть необходимо в случае разделения, например, публичного трафика от внутреннего.

Установка сервиса Gnocchi

> См. также: Настройка репозиториев Almalinux (см. стр. 10).

После установки всех необходимых внешних сервисов можно приступить к установке сервиса Glance.

1. Установите основной пакет сервиса:

```
dnf -y install gnocchi-api gnocchi-metricd python3-gnocchiclient
```

> Стандартные пути файлов конфигурации:
>
> - `/etc/gnocchi` – Каталог конфигурации Glance;
> - `/etc/gnocchi/gnocchi.conf` – основный файл конфигурации.

2. Очистите основной файл конфигурации, который был добавлен после установки пакета:

```
> /etc/gnocchi/gnocchi.conf
```

3. В основной файл добавьте следующую конфигурацию (описание (см. стр. 58)):

```
[api]
auth_mode = keystone

[keystone_authtoken]
auth_type = password
auth_url = http://controller:5000/v3
project_domain_name = Default
user_domain_name = Default
project_name = service
username = gnocchi
password = GNOCCHI_PASS
interface = internalURL
region_name = RegionOne

[indexer]
url = mysql+pymysql://gnocchi:GNOCCHI_DBPASS@controller/gnocchi

[storage]
# coordination_url is not required but specifying one will improve
# performance with better workload division across workers.
coordination_url = redis://controller:6379
file_basepath = /var/lib/gnocchi
driver = file
```

4. В конфигурации нужно обратить внимание на следующие параметры (переделать в таблицу):
   a. Адрес прослушивания сервиса Gnocchi API, которые зависят от метода установки (добавить в конфиг);
   b. Параметры журналирования сервиса (см. стр. 58), которые зависят от метода установки (добавить в конфиг);
   c. Параметры подключения к СУБД в **database - connection**, в частности, пароль к БД glance вместо GNOCCHI_DBPASS;
   d. Параметры подключения к Keystone в **keystone_authtoken**, в частности, пароль пользователя gnocchi вместо GNOCCHI_PASS;
   e. Параметры подключения к серверу memcached в **keystone_authtoken/memcached_servers** и **cache/memcache_servers** (добавить в конфиг).
5. После настройки конфигурации запустите процесс инициализации gnocchi:

```
gnocchi-upgrade
```

Финализация установки

1. После определения конфигурации необходимо запустить сервис Glance API и добавить его в автозапуск:

```
systemctl start openstack-gnocchi-api
systemctl start openstack-gnocchi-metricd
systemctl enable openstack-gnocchi-api
systemctl enable openstack-gnocchi-metricd
```

Проверка работы сервиса

1. Проверьте статус юнита Gnocchi API:

```
systemctl status openstack-gnocchi-api
```

Ответ должен быть примерно таким:

```
● gnocchi-api.service - Gnocchi API service
   Loaded: loaded (/usr/lib/systemd/system/gnocchi-api.service; enabled;
vendor preset: disabled)
  Drop-In: /run/systemd/system/gnocchi-api.service.d
           └─zzz-lxc-service.conf
   Active: active (running) since Sun 2021-11-07 22:06:22 UTC; 1min 15s ago
 Main PID: 13485 (gnocchi-api)
    Tasks: 6 (limit: 204240)
   Memory: 106.7M
   CGroup: /system.slice/gnocchi-api.service
           ├─13485 /usr/bin/python3 /usr/bin/gnocchi-api
```

2. Проверьте наличие открытого порта 8401:

```
ss -tnlp | grep 8401
```

Ответ должен быть примерно таким:

```
LISTEN 0      128    10.236.64.162:8401       0.0.0.0:*    users:
(("gnocchi-api",pid=32432,fd=3),...
```

Конфигурация для HAProxy

См. также: Установка балансировщика нагрузки HAProxy (см. стр. 16)

В референсной архитектуре доступ до сервиса Gnocchi API предоставляется через балансировщика нагрузки.

1. Конфигурация для HAProxy выглядит следующим образом:

```
frontend gnocchi_api
  bind "IP:PORT" ssl crt /usr/local/etc/haproxy/cert.pem alpn h2,http/1.1
  http-request set-header X-Forwarded-Proto https
  default_backend gnocchi_api_backend

backend gnocchi_api_backend
  server glance 127.0.0.1:8401
```

2. После включения этой конфигурации перезагрузите конфигурацию HAProxy:

```
systemctl reload haproxy
```

Установка сервисов Ceilometer

Установка сервиса

1. Установите сервис Ceilometer:

```
dnf -y install openstack-ceilometer-notification \
  openstack-ceilometer-central
```

2. Настройте конфигурационный файл /etc/ceilometer/pipeline.yaml:

```
publishers:
    - gnocchi://?filter_project=service&archive_policy=low
```

3. Настройте конфигурационный файл /etc/ceilometer/ceilometer.conf:

```
[DEFAULT]
...
transport_url = rabbit://openstack:RABBIT_PASS@controller
...
[service_credentials]
...
auth_type = password
auth_url = http://controller:5000/v3
project_domain_id = default
user_domain_id = default
project_name = service
username = ceilometer
password = CEILOMETER_PASS
interface = internalURL
region_name = RegionOne
```

Примечание

В параметрах RABBIT_PASS и CEILOMETER_PASS необходимо указать пароль, который был установлен для пользователя ceilometer. Подробнее параметры конфигурационного файла описаны в соответствующем разделе (см. стр. 114).

4. Обновите конфигурацию сервиса:

```
ceilometer-upgrade
```

Финализация установки

1. Запустите сервис Ceilometer и добавить его в автозапуск:

```
systemctl enable openstack-ceilometer-notification.service \
  openstack-ceilometer-central.service
systemctl start openstack-ceilometer-notification.service \
  openstack-ceilometer-central.service
```

Настройка сбора метрик в сервисах OpenStack

Сбор метрик Cinder

Примечание

Необходимо предварительно установить и настроить службу Cinder (см. стр. 94).

Для сбора метрик службы блочного хранилища Cinder необходимо выполнить следующие шаги:

1. Настройте конфигурационный файл /etc/cinder/cinder.conf:

```
[oslo_messaging_notifications]
...
driver = messagingv2
```

2. Включите периодический опрос статистических данных:

```
cinder-volume-usage-audit  --start_time='YYYY-MM-DD HH:MM:SS' \
  --end_time='YYYY-MM-DD HH:MM:SS' --send_actions
```

Данный сценарий будет фиксировать все действия над объектами начиная с даты, установленной в параметре --start_time и заканчивая датой, установленной в параметре --end_time.

3. Задать периодичность фиксации событий можно в /etc/cron.d/cinder-metrics:

```
*/5 * * * * /usr/bin/cinder-volume-usage-audit --send_actions
```

Это файл для системы периодических задач cron. Кратко по синтаксису:
- ***/5 * * * *** – период фиксации событий в формате cron,[125] в данном случае – каждые пять минут;
- **/usr/bin/cinder-volume-usage-audit** - путь до скрипта сбора данных (этот скрипт содержится в пакете openstack-cinder).

4. Перезапустите службы блочного хранилища на управляющих узлах и сервис периодических задач crond:

```
systemctl restart openstack-cinder-api.service openstack-cinder-scheduler.service
crond.service
```

5. Перезапустите службы блочного хранилища на узлах, где настроены бэкенды Cinder:

```
systemctl restart openstack-cinder-volume.service
```

Сбор метрик Glance

Примечание

Необходимо предварительно установить и настроить службу Glance (см. стр. 51).

Примечание

Все команды выполняются на управляющем узле.

Для сбора метрик службы образов Glance необходимо выполнить следующие шаги:

1. Настройте конфигурационный файл `/etc/glance/glance-api.conf`:

```
[DEFAULT]
...
transport_url = rabbit://openstack:RABBIT_PASS@controller

[oslo_messaging_notifications]
...
driver = messagingv2
```

В параметре RABBIT_PASS укажите пароль для пользователя openstack в RabbitMQ.

2. Перезапустите службу Glance:

```
systemctl restart openstack-glance-api.service
```

Сбор метрик Keystone

На текущий момент поддерживается регистрация событий Keystone только для сервиса Nova.

Примечание

Необходимо предварительно установить и настроить службу Keystone (см. стр. 33).

Для сбора метрик по обращению сервиса Nova к функциям аутентификации сервиса Keystone необходимо выполнить следующие шаги:

1. Включите обработку аудита фиксации событий в `/etc/nova/api-paste.ini`:
   - укажите путь до аудита в секции [filter:audit]:

```
[filter:audit]
paste.filter_factory = keystonemiddleware.audit:filter_factory
audit_map_file = /etc/nova/api_audit_map.conf
```

125 http://www.nncron.ru/nncronlt/help/RU/working/cron-format.htm

- добавьте параметр audit в секцию [composite:openstack_compute_api_v21] строго перед параметром osapi_compute_app_v2:

```
[composite:openstack_compute_api_v21]
keystone = faultwrap sizelimit authtoken keystonecontext ratelimit audit
osapi_compute_app_v2
```

2. Создайте файл аудита `/etc/nova/api_audit_map.conf`. Образец файла можно взять из данного репозитория[126].
3. Перезапустите службу Nova:

```
systemctl restart openstack-nova-api.service
```

Сбор метрик Neutron

Примечание

Необходимо предварительно установить и настроить службу Neutron (см. стр. 83).

Примечание

Все команды выполняются на управляющем узле.

Для сбора метрик службы сетевой инфраструктурой Neutron необходимо выполнить следующие шаги:

1. Настройте конфигурационный файл `/etc/neutron/neutron.conf`:

```
[oslo_messaging_notifications]
...
driver = messagingv2
```

2. Перезапустите службу Neutron:

```
systemctl restart neutron-server.service
```

Установка сервиса Redis

Redis - это сервис хранения и кэширования данных, которые хранится в оперативной памяти в виде базы данных в возможностью их долговременного хранения на диске. Redis необходим для хранения данных метрик сервиса Gnocchi.

Установка сервиса

1. Установите пакет redis:

```
dnf -y install redis
```

Стандартные пути конфигурации:

- Основной файл конфигурации: `/etc/redis.conf`

2. В основном конфигурационном файле укажите следующую конфигурацию:

126 https://github.com/openstack/pycadf/blob/master/etc/pycadf/nova_api_audit_map.conf

```
# Настройки сети
bind controller
protected-mode no
port 6379
tcp-backlog 511
timeout 0
tcp-keepalive 60

# Настройки запуска сервиса
daemonize yes
supervised auto
pidfile /var/run/redis_6379.pid
loglevel notice
logfile /var/log/redis/redis.log
databases 16
always-show-logo no

# Настройки хранения
save 900 1
save 300 10
save 60 10000
stop-writes-on-bgsave-error yes
rdbcompression yes
rdbchecksum yes
dbfilename dump.rdb
dir /var/lib/redis

# Настройки режима Append Only
appendonly no
appendfilename "appendonly.aof"
appendfsync everysec
no-appendfsync-on-rewrite no
auto-aof-rewrite-percentage 100
auto-aof-rewrite-min-size 64mb
aof-load-truncated yes
aof-use-rdb-preamble yes

# Параметры функции Slow Log
slowlog-log-slower-than 10000
slowlog-max-len 128

# Параметры Latency Monitor
latency-monitor-threshold 0

# Параметры оповещений
notify-keyspace-events ""

# Дополнительные параметры сервиса
hash-max-ziplist-entries 512
hash-max-ziplist-value 64
list-max-ziplist-size -2
list-compress-depth 0
set-max-intset-entries 512
zset-max-ziplist-entries 128
zset-max-ziplist-value 64
hll-sparse-max-bytes 3000
stream-node-max-bytes 4096
stream-node-max-entries 100
activerehashing yes
client-output-buffer-limit normal 0 0 0
client-output-buffer-limit replica 256mb 64mb 60
client-output-buffer-limit pubsub 32mb 8mb 60
hz 10
dynamic-hz yes
aof-rewrite-incremental-fsync yes
rdb-save-incremental-fsync yes
```

Настройка вычислительного узла для Ceilometer

Сбор метрик Nova

Установка сервиса

1. Установите сервис Ceilometer:

```
yum install openstack-ceilometer-compute
```

2. Настройте конфигурационный файл /etc/ceilometer/ceilometer.conf:

```
[DEFAULT]
...
transport_url = rabbit://openstack:RABBIT_PASS@controller
...
[service_credentials]
...
auth_url = http://controller:5000
project_domain_id = default
user_domain_id = default
auth_type = password
username = ceilometer
project_name = service
password = CEILOMETER_PASS
interface = internalURL
region_name = RegionOne
```

В параметрах RABBIT_PASS и CEILOMETER_PASS необходимо указать пароль, который был установлен для пользователя ceilometer. Подробнее параметры конфигурационного файла описаны в соответствующем разделе (см. стр. 114).

3. Настройте конфигурационный файл /etc/nova/nova.conf:

```
[DEFAULT]
...
instance_usage_audit = True
instance_usage_audit_period = hour

[notifications]
...
notify_on_state_change = vm_and_task_state

[oslo_messaging_notifications]
...
driver = messagingv2
```

Настройка сбора IPMI метрик

1. Установите сервис openstack-ceilometer-ipmi:

```
yum install openstack-ceilometer-ipmi
```

2. Настройте конфигурационный файл /etc/sudoers:

```
ceilometer ALL = (root) NOPASSWD: /usr/bin/ceilometer-rootwrap /etc/ceilometer/
rootwrap.conf *
```

3. Настройте конфигурационный файл /etc/ceilometer/polling.yaml:

```
- name: ipmi
  interval: 300
  meters:
    - hardware.ipmi.temperature
```

4. Запустите сервис Ceilometer и добавить его в автозапуск:

```
systemctl enable openstack-ceilometer-ipmi.service
systemctl start openstack-ceilometer-ipmi.service
```

Финализация установки

1. Запустите сервис Ceilometer и добавить его в автозапуск:

```
systemctl enable openstack-ceilometer-compute.service
systemctl start openstack-ceilometer-compute.service
```

2. Перезапустите службу Nova:

```
systemctl restart openstack-nova-compute.service
```

Описание файла конфигурации Ceilometer

В описании процесса по установке сервиса Ceilometer предложена стандартная конфигурация. Это страница содержит подробное описание настроек этой конфигурации.

При изменении конфигурации необходимо перезапустить веб-сервер Apache:

```
systemctl restart httpd
```

Таблица конфигурации

Сервис Ceilometer

ⓘ Путь до конфигурации: `/etc/ceilometer/ceilometer.conf`.

ⓘ Легенда таблицы доступна на этой странице[127].

Имя параметра	Описание	Примечания
[DEFAULT]	Глобальные переменные сервиса.	
transport_url	Список адресов до серверов RabbitMQ.	Серверы указываются через запятую с указанием префикса в каждом элементе.
auth_type	Тип аутентификации.	Всегда должен быть равен параметру "password".
auth_url	Адрес до сервиса Keystone.	
memcached_servers	Адреса до сервисов кэширования memcached.	
project_domain_name **user_domain_name** **project_name** **username** **password**	Данные аутентификации в сервис Keystone от имени пользователя ceilometer в проекте service.	
interface	Тип точки доступа в сервисе Keystone.	
region_name	Имя региона, которое будет использоваться для точек доступа.	
[oslo_messaging_rabbit]	Параметры RabbitMQ.	

127 https://conf.tionix.ru/pages/viewpage.action?pageId=205881354#id-Условныеобозначения-table-format-desc

Имя параметра	Описание	Примечания
rabbit_retry_interval	Интервал подключения к RabbitMQ.	
rabbit_retry_backoff	Интервал между повторными попытками подключения к RabbitMQ.	

Установка и настройка модулей TIONIX

Предварительная настройка

Перед установкой всех модулей TIONIX вначале необходимо создать некоторые объекты:

- пользователь и виртуальный хост tionix в RabbitMQ;
- пользователь в OpenStack Keystone в проекте service с ролью admin.

Эти объекты нужны для всех основных модулей (если не указано иное).

Подготовка объектов в RabbitMQ

См. также: Установка сервиса RabbitMQ (см. стр. 29).

Для функций межсервисного взаимодействия сервисы TIONIX используют функции сервиса RabbitMQ.

1. Создайте пользователя tionix, где TIONIX_RABBIT_PASS - это пароль для него:

```
rabbitmqctl add_user tionix TIONIX_RABBIT_PASS
```

2. Добавьте виртуальный хост[128] (vhost):

```
rabbitmqctl add_vhost tionix
```

3. Укажите все типы прав tionix в виртуальном хосте tionix (последний указан через параметр -p):

```
rabbitmqctl set_permissions -p tionix tionix ".*" ".*" ".*"
```

4. Эти же права укажите в стандартном виртуальном хосте "/":

```
rabbitmqctl set_permissions tionix "." "." ".*"
```

Создание объектов в OpenStack Keystone

См. также: Раздел по установке и настройке OpenStack Keystone (см. стр. 33)

Модулям TIONIX требуется зарегистрировать пользователя tionix в OpenStack Keystone (см. стр. 33) для взаимодействия с другими сервисами платформы.

1. Создайте пользователя tionix в проекте service, расположенный в домене default (пароль будет запрошен в интерактивном режиме):

```
openstack user create --domain default --project service --project-domain default
 --password-prompt tionix
```

2. Укажите этому пользователю роль admin для получения административных прав в проекте service:

```
openstack role add --user tionix --user-domain default --project service --
project-domain default admin
```

Остальные объекты (например, точки входа/endpoints) будут созданы индивидуально для отдельных модулей TIONIX.

Поддержка сервиса Sentry

Некоторые модули Tionix поддерживают отправку журналов сервису Sentry[129]. Для его поддержки нужно установить библиотеку Raven:

```
dnf -y install python3-raven
```

128 https://www.rabbitmq.com/vhosts.html
129 https://sentry.io/welcome/

Client

Информация о модуле Client

Обзор

TIONIX Client – служебный модуль, необходимый для предоставления доступа к функциональности модулей TIONIX.

Основные функции клиента TIONIX:

- Реализация ClientAPI - REST API, предоставляющий дополнительные возможности по взаимодействию с объектами облачной платформы:
  - JournalAPI - это часть ClientAPI, предоставляющий доступ к журналам, содержащий информацию обо всех операциях над объектами облачной платформы;
- Расширение консольного клиента openstackclient функциями модулей TIONIX;
- Включение драйвера tnx_ldap[130], решающий некоторые проблемы со стандартным драйвером LDAP в сервисе Keystone;
- Включение исправления для сервиса Cinder, решающий вопрос живой миграции виртуальной машины при наличии блочных устройств на базе протоколов iSCSI[131] и Fibre Channel[132].

Описание функций

Драйвер tnx_ldap

В составе модуля содержится вариант identity-драйвера для сервисов LDAP, который называется **tnx_ldap**. Этот драйвер основан на оригинальном драйвере ldap сервиса Keystone и выполняет несколько дополнительных функций:

- Устанавливает параметр *keystone.identity.backends.ldap.common.WRITABLE = True* для отключения ошибки при вызове методов create/update для объектов, которые хранятся в базе данных LDAP.
- При аутентификации обрабатывает ответы серверов LDAP и ищет признаки истечения времени жизни токена (password expired). Эта функция требуется для вызова функции обновления пароля со стороны пользователя.
- Меняет типы операций в зависимости от типа сервера LDAP при записи нового пароля, например, в случае с Active Directory при отправке поля и значения требуется указать тип операции replace, для остальных типов LDAP-серверов достаточно указать add.

Установка модуля Client

Настройка окружения

Перед самой установкой сервиса нужно предварительно настроить некоторые компоненты инфраструктуры.

См. также: Раздел с предварительной настройкой окружения для модулей TIONIX (см. стр. 116).

Подготовка базы данных tionix

См. также: Установка и настройка СУБД MariaDB (см. стр. 22)

Всю информацию о данных для аутентификации и авторизации по умолчанию Tionix Client хранит в базе данных MariaDB.

1. Войдите в окружение базы данных:

```
mysql -u root -p
```

2. Создайте базу данных tionix:

130 https://conf.tionix.ru/pages/viewpage.action?pageId=164102237#id-Использованиесвязанныхсдоменомсистемхраненияданныхпользователей-tnx-ldap-desc
131 https://ru.wikipedia.org/wiki/ISCSI
132 https://ru.wikipedia.org/wiki/Fibre_Channel

```
create database tionix;
```

3. Предоставьте доступ к этой базе данных пользователю tionix в СУБД (для localhost и всем остальным адресам отдельно, вместо TIONIX_DBPASS используйте свой пароль):

```
grant all privileges on tionix.* to 'tionix'@'localhost' identified by
'TIONIX_DBPASS';
grant all privileges on tionix.* to 'tionix'@'%' identified by 'TIONIX_DBPASS';
```

4. Выйдите из сессии СУБД

```
exit;
```

Установка модуля

1. Установите пакет модуля:

```
dnf -y install python3-tionix_client
```

> **Информация**
>
> Пути конфигурации модуля:
>
> - `/etc/tionix` — основной каталог конфигурации.
> - `/etc/tionix/tionix.yaml` — основной файл конфигурации.
>
> Основной файл конфигурации может отсутствовать после установки пакетов, в этом случае достаточно создать его.

2. Добавьте следующие параметры в основной файл конфигурации (описание (см. стр. 172)):

```
# Общие параметры TIONIX
CINDER_VERSION: '3.50'
TRACEBACK_ENABLED: False

# Параметры сервиса Journal API
LOG_LEVEL: 'INFO'
JOURNAL_API_LISTEN: 'LISTEN_IP'
JOURNAL_API_LISTEN_PORT: 9360
JOURNAL_API_LOGFILE: '/var/log/tionix/journal/api.log'
JOURNAL_LISTENER_LOGFILE: '/var/log/tionix/journal/listener.log'
JOURNAL_NOVA_LISTENER_LOGFILE: '/var/log/tionix/journal/nova-listener.log'
JOURNAL_KEYSTONE_LISTENER_LOGFILE: '/var/log/tionix/journal/keystone-listener.log'

# Общие параметры Keystone
KEYSTONE:
    auth_url: 'https://controller:5000'
    auth_version: '3'
    auth_user: 'admin'
    auth_password: 'ADMIN_PASS'
    auth_tenant: 'admin'
    user_domain_name: 'default'
    project_domain_name: 'default'
    service_user: 'tionix'
    service_password: 'TIONIX_PASS'
    service_project: 'service'
    service_user_domain_name: 'default'
    service_project_domain_name: 'default'
    compute_service_name: 'compute'
    volume_service_name: 'volumev3'
    gnocchi_service_name: 'metric'
    journal_service_type: 'tnx-journal'
    nc_service_type: 'tnx-nc'
    monitor_service_type: 'tnx-monitor'
    vdi_service_type: 'tnx-vdi'
    scheduler_service_type: 'tnx-scheduler'
    memcached_servers: 'controller:11211'

# Общие параметры доступа к базам данных TIONIX
DB:
    ENGINE: 'mysql+pymysql'
    HOST: 'controller'
    PORT: '3306'
    NAME: 'tionix'
    USER: 'tionix'
    PASSWORD: 'TIONIX_CLIENT_DBPASS'
    MAX_POOL_SIZE: 5
    MAX_OVERFLOW: 30
    POOL_RECYCLE: 3600
    POOL_TIMEOUT: 30

# Общие параметра доступа к RabbitMQ
RABBIT_QUEUES:
    broker_type: 'amqp'
    host: 'controller'
    port: '5672'
    vhost: 'tionix'
    username: 'tionix'
    password: 'TIONIX_RABBIT_PASS'
    durable: True

NOVA_RABBIT_VHOST: '/'
KEYSTONE_RABBIT_VHOST: '/'
```

3. При конфигурировании обратите особое внимание на следующие настройки:

Легенда таблицы доступна на этой странице[133].

Параметр	Значение
JOURNAL_API_LISTEN	Укажите слушаемый адрес сервиса Journal API в зависимости от типа установки: • классическая архитектура: адрес интерфейса в сети управления (mgmt); • референсная архитектура: 0.0.0.0.
Параметры LOGFILE	Укажите пути файлов журналирования в зависимости от типа установки: • классическая архитектура: пути, предложенные к конфигурации; • референсная архитектура: укажите **/dev/stdout** или **/dev/stderr** во всех параметрах.
KEYSTONE	Укажите данные подключения к сервису Keystone, в частности: • пароль пользователя admin в сервисе Keystone вместо ADMIN_PASS; • пароль пользователя tionix в сервисе Keystone вместо TIONIX_PASS; • отдельно укажите адреса серверов memcached в **memcached_servers**.
DB	Укажите параметры подключения к СУБД MariaDB, в частности: • пароль для пользователя tionix для базы данных tionix вместо TIONIX_CLIENT_DBPASS.
RABBIT_QUEUES	Укажите параметры подключения к сервису RabbtMQ, в частности: • пароль к пользователю tionix в RabbitMQ вместо TIONIX_RABBIT_PASS; • включение режима durable для сохранения состояния очередей при перезапуске RabbitMQ.

Все параметры предложенной конфигурации доступны на этой странице (см. стр. 172).

4. Создайте каталоги для журналирования Journal API:

```
mkdir -p /var/log/tionix/journal
chown tionix:tionix /var/log/tionix/journal
```

5. Выполните первичную инициализацию модуля:

```
openstack tnx configure -n tnx_client
```

6. Выполните миграцию базы данных:

```
openstack tnx db migrate -n tnx_client
```

Запуск сервиса Journal

Одним из основных функций клиента является реализация сервиса Journal, которые регистрирует события над объектами некоторых служб OpenStack.

1. Создайте сервис Journal API в Keystone:

```
openstack service create --name tnx-journal --description "TIONIX Journal Service"
 tnx-journal
```

2. Создайте точки входа (endpoint):
   a. публичную:

133 https://conf.tionix.ru/pages/viewpage.action?pageId=205881354#id-Условныеобозначения-table-format-desc

```
openstack endpoint create --region RegionOne tnx-journal public http://
controller:9360
```

b. внутреннюю:

```
openstack endpoint create --region RegionOne tnx-journal internal http://
controller:9360
```

c. административную:

```
openstack endpoint create --region RegionOne tnx-journal admin http://
controller:9360
```

3. Включите и запустите сервисы Journal, реализующие функциональность доступа к зарегистрированным событиям для объектов облачной платформы:

```
systemctl start tionix-journal-api
systemctl start tionix-journal-keystone-listener
systemctl start tionix-journal-listener
systemctl start tionix-journal-nova-listener
systemctl enable tionix-journal-api
systemctl enable tionix-journal-keystone-listener
systemctl enable tionix-journal-listener
systemctl enable tionix-journal-nova-listener
```

Проверка работы сервиса

1. Проверьте статус сервисов, например, tionix-journal-api:

```
systemctl status tionix-journal-api
```

Ответ должен быть примерно таким:

```
● tionix-journal-api.service - TIONIX Journal API service
   Loaded: loaded (/usr/lib/systemd/system/tionix-journal-api.service;
enabled; vendor preset: disabled)
  Drop-In: /run/systemd/system/tionix-journal-api.service.d
           └─zzz-lxc-service.conf
   Active: active (running) since Tue 2021-11-09 20:25:21 UTC; 3min 54s ago
 Main PID: 157515 (tnx-journal-api)
    Tasks: 1 (limit: 204240)
   Memory: 63.7M
   CGroup: /system.slice/tionix-journal-api.service
           └─157515 /usr/libexec/platform-python /usr/bin/tnx-journal-api
...
```

2. Проверьте статус порта:

```
ss -tnlp | grep 9360
```

Ответ должен быть примерно таким:

```
LISTEN 0      50     10.236.64.231:9360       0.0.0.0:*    users:(("tnx-
journal-api",pid=157515,fd=4))
```

Порт должен иметь статус LISTEN и слушать адрес сети управления (или 0.0.0.0).

3. Настройте поддержку уведомлений о событиях в поддерживаемых сервисах OpenStack. (см. стр. 121)

Настройка сервиса Journal

Сервис Journal - это компонент модуля TIONIX Client, регистрирующий и возвращающий события от действий над объектами в OpenStack. Поддерживаются следующие сервисы:

- OpenStack Nova - события по виртуальным машинам.

- OpenStack Keystone - события по аутентификации пользователей.

В этой статье будет описана настройка указанных сервисов для включения этой функции.

> Не следует путать сервис Journal в составе Client и сервис Journal[134] в составе Systemd[135], который занимается сбором системных журналов ОС.

OpenStack Nova

Для включения журналирования событий над объектами в сервисе OpenStack Nova нужно включить дополнительную настройку этого сервиса.

1. В файл `/etc/nova/api-paste.ini` в управляющем узле нужно добавить следующую конфигурацию для включения фильтрации событий и их отправку в сервисы Journal:

```
[filter:tnx_audit]
paste.filter_factory = tionix_client.journal.api_filter:filter_factory
```

2. В том же файле в управляющем узле добавьте фильтр *tnx_audit* в список доступных при авторизации в OpenStack Keystone (*auth_strategy = keystone* в `/etc/nova/nova.conf`):

```
[composite:openstack_compute_api_v21]
use = call:nova.api.auth:pipeline_factory_v21
noauth2 = cors compute_req_id faultwrap sizelimit noauth2 osapi_compute_app_v21
keystone = cors compute_req_id faultwrap sizelimit authtoken keystonecontext
tnx_audit osapi_compute_app_v21
```

3. В файле `/etc/nova/nova.conf` в управляющем и вычислительном узлах включите возможность отправки уведомлений (notifications)[136] о событиях через RabbitMQ:

```
[oslo_messaging_notifications]
driver = messagingv2
```

4. Перезапустите службы OpenStack Nova в управляющем и вычислительном узлах:

```
systemctl restart openstack-nova-*
```

OpenStack Keystone

Для включения журналирования событий над объектами в сервисе OpenStack Keystone нужно включить дополнительную настройку этого сервиса.

1. В файле `/etc/keystone/keystone.conf` укажите поддержку уведомлений о событиях[137] в формате CADF[138]:

```
[DEFAULT]
notification_format = cadf
```

2. Там же включите возможность отправки уведомлений (notifications)[139] о событиях через RabbitMQ:

```
[oslo_messaging_notifications]
driver = messagingv2
```

> В конфигурации OpenStack Keystone должен быть указан адрес сервера RabbitMQ, который включен в основной файл конфигурации модулей TIONIX. Конфигурация предоставлена в статье с установкой сервиса OpenStack Keystone (см. стр. 35).

3. Перезапустите веб-сервер, где запущено WSGI-приложение сервиса OpenStack Keystone:

```
systemctl restart httpd
```

134 https://www.freedesktop.org/software/systemd/man/systemd-journald.service.html
135 https://ru.wikipedia.org/wiki/Systemd
136 https://docs.openstack.org/nova/victoria/reference/notifications.html
137 https://docs.openstack.org/keystone/victoria/advanced-topics/event_notifications.html
138 https://www.dmtf.org/standards/cadf
139 https://docs.openstack.org/nova/victoria/reference/notifications.html

Настройка драйвера LDAP

Client содержит модифицированный драйвер (см. стр. 117) для работы с серверами LDAP. Эта статья кратко описывает его настройку.

1. В основном файле конфигурации сервиса OpenStack Keystone, который находится по пути `/etc/keystone/keystone.conf`, измените используемый драйвер:

```
[identity]
...
driver = tnx_ldap
```

2. В каждом файле конфигурации для доменов Keystone[140], которые расположены в каталоге `/etc/keystone/domains`, необходимо выключить пулинг подключений при аутентификации[141] к серверу LDAP:

```
[ldap]
...
use_auth_pool = False
```

3. Перезапустите сервис веб-сервера Apache:

```
systemctl restart httpd
```

NodeControl

Информация о сервисе NodeControl

Обзор

OpenStack содержит очень большое количество функций, связанных с управлением вычислительных узлов, однако они, в основном, связаны с регистрацией этих узлов как гипервизоров в облачной платформе и запуске виртуальных машин в них. Сервис NodeControl расширяет функциональность управления вычислительными узлами новыми функциями.

Список основных функций, реализуемые сервисом NodeControl можно разделить на две части:

1. Вычислительные узлы (связаны с сервисом OpenStack Nova):
   - расширение списка метаданных вычислительного узла информацией об их сведениях инвентаризации и локации (напр., физическое положение в стойке и серийные номера оборудования);
   - предоставление функции управления образами корневой ОС для запуска вычислительных узлов с использованием сетевых функции, в частности, протокола PXE;
   - управление питанием вычислительного узла через протоколы IPMI и Intel AMT;
   - автоматическая эвакуация виртуальных машин при наличии проблем с соединением до вычислительного узла;
   - функция резервных вычислительных узлов, ожидающих включения в работу в при отказе активного вычислительного узла;
   - улучшение проверки на доступность сервисов nova-compute в вычислительных узлах и виртуальных машин с использованием модуля TIONIX Agent;
2. Блочные устройства (связаны с сервисом OpenStack Cinder):
   - расширенный сбор данных о блочных хранилищах;
   - реализация универсального доступа к блочным системам хранения без использования отдельных вендорных драйверов.

NodeControl поддерживает особые функции:

- реализация концепции виртуальных облачных инфраструктур со своими вычислительными узлами и сетевой изоляцией, которые запускаются в центральной облачной платформе, расположенная в физической инфраструктуре.
- поддержка внешних систем восстановления инфраструктуры после катастрофической ситуации или, кратко, DRS[142].

Более подробно обо всех этих функциях будет рассказано в "Управлении сервисом NodeControl". Этот раздел будет содержать установку и первичный запуск сервиса.

---

140 https://conf.tionix.ru/pages/viewpage.action?pageId=164102237#id-Использованиесвязанныхсдоменомсистемхраненияданныхпользователей-ks-domain-specific-drivers
141 https://docs.oracle.com/javase/jndi/tutorial/ldap/connect/pool.html
142 https://en.wikipedia.org/wiki/Disaster_recovery

Состав компонентов сервиса NodeControl

Сервис NodeControl состоит из нескольких компонентов, работающие как службы в systemd:

- **tionix-node-control-api** - это сетевой сервис реализует и даёт доступ к NodeControl API;
- **tionix-node-control-node-syncer** - отвечает за обновление состояния вычислительных узлов через механизмы Nova;
- **tionix-node-control-node-tracker** - отслеживает задачи (tasks), передаваемые вычислительным узлам сервисом Nova через RabbitMQ;
- **tionix-node-control-worker** - исполняет задачи самого сервиса NodeControl;
- **tionix-node-control-agent** - этот сервис взаимодействует с агентом Tionix;
- **tionix-node-control-drs-trigger** - следит за событиями с внешней системы DRS;
- **tionix-node-control-nova-listener** - следит за событиями виртуальных машин через механизмы Nova;
- **tionix-node-control-storage-syncer** - отвечает за обновление состояния дисков и систем хранения через механизмы Cinder.

Установка сервиса NodeControl

Настройка окружения

Перед самой установкой сервиса нужно предварительно настроить некоторые компоненты инфраструктуры.

См. также: Раздел с предварительной настройкой окружения для модулей TIONIX (см. стр. 116).

Подготовка базы данных tionix_node_control

См. также: Установка и настройка СУБД MariaDB (см. стр. 22)

Всю информацию о данных для аутентификации и авторизации по умолчанию Tionix NodeControl хранит в базе данных MariaDB.

1. Войдите в окружение базы данных:

```
mysql -u root -p
```

2. Создайте базу данных tionix:

```
create database tionix_node_control;
```

3. Предоставьте доступ к этой базе данных пользователю tionix в СУБД (для localhost и всем остальным адресам отдельно, вместо TIONIX_DBPASS используйте свой пароль):

```
grant all privileges on tionix_node_control.* to 'tionix'@'localhost' identified
by 'TIONIX_DBPASS';
grant all privileges on tionix_node_control.* to 'tionix'@'%' identified by
'TIONIX_DBPASS';
```

4. Выйдите из сессии СУБД

```
exit;
```

Установка модуля

Процесс установки

1. Установите пакет модуля и пакет net-tools:

```
dnf -y install python3-tionix_node_control net-tools
```

Стандартные пути до файлов конфигурации:

- `/etc/tionix` — основной каталог конфигурации.
- `/etc/tionix/tionix.yaml` — общий файл конфигурации TIONIX.
- `/etc/tionix/node_control.yaml` - основной файл конфигурации сервиса NodeControl.

Основной файл конфигурации сервиса NodeControl может отсутствовать после установки пакета, в этом случае достаточно создать файл.

2. Добавьте эту конфигурацию в основной файл конфигурации сервиса NodeControl (описание (см. стр. 128)):

```
# Общие параметры сервиса NodeControl
NODE_CONTROL_API_LISTEN: 'LISTEN_IP'
NODE_CONTROL_API_LISTEN_PORT: 9362
NODE_CONTROL_API_AUDIT_ENABLED: True
ENABLE_CEPH_INTEGRATION: False

PXE:
    conf_dir: '/var/lib/tftpboot/pxelinux.cfg/'

SETTINGS_TRACKER:
    mutex: 3
    mutex_up: 1
    loop_time: 30

DHCP_LEASES_FILEPATHS:
    - '/var/lib/dhcp/dhcpd/state/dhcpd.leases'
    - '/var/lib/dhcp/dhcpd.leases'

# Параметры журналирования
NODE_CONTROL_API_LOGFILE: '/var/log/tionix/node-control/api.log'
NODE_CONTROL_NODE_SYNCER_LOGFILE: '/var/log/tionix/node-control/node-syncer.log'
NODE_CONTROL_NODE_TRACKER_LOGFILE: '/var/log/tionix/node-control/node-tracker.log'
NODE_CONTROL_WORKER_LOGFILE: '/var/log/tionix/node-control/worker.log'
NODE_CONTROL_NOVA_LISTENER_LOGFILE: '/var/log/tionix/node-control/nova-
listener.log'

# Параметры базы данных
DB:
    NAME: 'tionix_node_control'
DB_CONNECTION_MAX_RETRIES: 2

# Параметры планировщика NodeControl
SYNC_NOVA_NODES_TIME: 60
RETRIES_WAIT_FOR_VM_STATUS: 60
RETRIES_WAIT_FOR_NODE_STATE: 240
TIMEOUT_RESERV_NODE_UP: 15
MAX_TICK_COUNT: 10
SLEEP_TIME: 30
MAX_DOWN_HOSTS: 1
ALLOW_HOST_AUTO_POWER_OFF: False
HOST_RESTART_TIMEOUT: 600
HOST_ATTACH_RETRY_DELAY: 120
HOST_ATTACH_MAX_RETRIES: 10
HOST_ATTACH_NETWORK_TAG: ""
ALLOW_EVACUATE_HOST: True

# Параметры для виртуальных контроллеров OpenStack
KEY_PATH: '/etc/tionix/hybrid/tionix.crt'
CONTROLLER_AUTH_PATH: '/etc/tionix/hybrid/admin-openrc'
CONTROLLER_USERNAME: 'tionix'
ENABLE_NETWORK_ISOLATION: False
NETWORK_ISOLATION_API_HOST: ""
NETWORK_ISOLATION_API_PORT: 5549

# Параметры сервиса Sentry
SENTRY:
    ENABLED: False
    LOG_LEVEL: INFO
    DSN: http://PUBLIC_KEY:SECRET_KEY@SENTRY_ADDR/PROJECT_ID
```

Подробное описание конфигурации для этого сервиса предоставлено в этом подразделе (см. стр. 128). Параметры подключения к СУБД и RabbitMQ и прочие параметры NodeControl получает с общего файла конфигурации TIONIX.

3. Выполните первичную инициализацию модуля (вместе с модулем TIONIX Client):

```
openstack tnx configure -n tnx_node_control tnx_client
```

4. Выполните миграцию базы данных:

```
openstack tnx db migrate -n tnx_node_control
```

Создание сервиса NodeControl

1. Создайте сервис tnx-nc в OpenStack Keystone:

```
openstack service create --name tnx-nc --description "TIONIX Node Control Service"
 tnx-nc
```

2. Создайте точки входа (endpoint):
   a. публичную:

```
openstack endpoint create --region RegionOne tnx-nc public http://controller:
9362
```

   b. внутреннюю:

```
openstack endpoint create --region RegionOne tnx-nc internal http://
controller:9362
```

   c. административную:

```
openstack endpoint create --region RegionOne tnx-nc admin http://controller:
9362
```

Финализация установки

1. Включите и запустите службы, реализующие функциональность доступа к зарегистрированным событиям для объектов облачной платформы:

```
systemctl start tionix-node-control-api
systemctl start tionix-node-control-node-syncer
systemctl start tionix-node-control-node-tracker
systemctl start tionix-node-control-worker
systemctl start tionix-node-control-agent
systemctl start tionix-node-control-drs-trigger
systemctl start tionix-node-control-nova-listener
systemctl start tionix-node-control-storage-syncer
systemctl enable tionix-node-control-api
systemctl enable tionix-node-control-node-syncer
systemctl enable tionix-node-control-node-tracker
systemctl enable tionix-node-control-worker
systemctl enable tionix-node-control-agent
systemctl enable tionix-node-control-drs-trigger
systemctl enable tionix-node-control-nova-listener
systemctl enable tionix-node-control-storage-syncer
```

2. Перезапустите все службы TIONIX для регистрации сервиса NodeControl:

```
systemctl restart tionix-*
```

Проверка работы сервиса

1. Проверьте статус сервисов, например, tionix-node-control-api:

```
systemctl status tionix-node-control-api.service
```

В ответ вы должны получить примерно следующее:

```
● tionix-node-control-api.service - TIONIX NodeControl API service
   Loaded: loaded (/usr/lib/systemd/system/tionix-node-control-api.service;
enabled; vendor preset: disabled)
  Drop-In: /run/systemd/system/tionix-node-control-api.service.d
           └─zzz-lxc-service.conf
   Active: active (running) since Tue 2021-11-09 21:13:33 UTC; 2min 44s ago
 Main PID: 171547 (/usr/libexec/pl)
    Tasks: 1 (limit: 204240)
   Memory: 83.5M
   CGroup: /system.slice/tionix-node-control-api.service
           └─171547 /usr/libexec/platform-python /usr/bin/tnx-node-control-
api
...
```

2. Проверьте статус порта сервиса:

```
ss -tnlp | grep 9362
```

Ответ должен выглядеть примерно так:

```
LISTEN 0      50     10.236.64.231:9362       0.0.0.0:*    users:(("/usr/
libexec/pl",pid=171547,fd=5))
```

Порт должен быть в статусе LISTEN и должен прослушиваться адрес сети управления (mgmt).

Описание файла конфигурации сервиса NodeControl

В описании процесса по установке сервиса NodeControl (см. стр. 124) предложена стандартная конфигурация для настройки. Это страница содержит подробное описание этой конфигурации.

При изменении конфигурации необходимо перезапустить все компоненты сервиса NodeControl:

```
systemctl restart tionix-node-control-*
```

Таблица конфигурации

Легенда таблицы доступна на этой странице[143].

Немного о формате yaml

Файл формата yaml[144] критичен к отступам текста[145] при описании параметров. Стрелкой в имени параметра указан отступ в 4 пробела от начала строки, а между стрелками указана связь между родительским и конечным параметром.

Имя параметра	Описание	Примечания
DB	Параметры подключения к базе данных.	

143 https://conf.tionix.ru/pages/viewpage.action?pageId=205881354#id-Условныеобозначения-table-format-desc
144 https://yaml.org/
145 https://yaml.org/spec/1.2.2/#61-indentation-spaces

Имя параметра	Описание	Примечания
DB → NAME	Имя базы данных сервиса NodeControl.	Остальные параметры подключения к базе данных берутся с основного файла конфигурации TIONIX.
PXE	Параметры управления сетевым сервисом PXE[146].	
PXE → conf_dir	Путь до файлов ядра Linux для запуска ОС через протокол PXE и доступные через протоколы TFTP или HTTP.	
SETTINGS_TRACKER	Параметры, определяющие параметры отслеживания состояния гипервизора.	
SETTINGS_TRACKER → mutex	Количество попыток определения статуса гипервизора при переходе в статус *down* перед запуском обработчика.	
SETTINGS_TRACKER → mutex_up	Количество попыток, при котором возвращается статус *up*, перед запуском обработчика.	
SETTINGS_TRACKER → loop_time	Интервал времени между проверками.	

DHCP_LEASES_FILEPATHS	Список файлов .leases DHCP-сервера, содержащие данные о выделенных IP-адресах и времени их аренды.	Внутри параметра должен быть список с отступом в 4 проблема, формат которого принят в формате yaml, например:
		`- '/var/lib/dhcp/dhcpd/state/dhcpd.leases'`
		`- '/var/lib/dhcp/dhcpd.leases'`

SYNC_NOVA_NODES_TIME	Интервал обновления статуса вычислительных узлов	Указывается в секундах.
DB_CONNECTION_MAX_RETRIES	Максимальное количество попыток соединения к базе данных.	Этот параметр будет использоваться только для базы данных, указанных в параметре DB этой конфигурации.
TIMEOUT_RESERV_NODE_UP	Время ожидания, при котором резервный узел должен получить статус *up*.	
MAX_TICK_COUNT	Максимальное количество попыток проверить статус вычислительного узла после выключения питания гипервизора и последующей автоэвакуации узла в платформе.	

146 https://en.wikipedia.org/wiki/Preboot_Execution_Environment

Имя параметра	Описание	Примечания
SLEEP_TIME	Интервал между проверками статуса вычислительного узла после выключения питания.	Указывается в секундах.
HOST_RESTART_TIMEOUT	Время ожидания, при котором вычислительный узел должен перезапуститься.	Указывается в секундах.
MAX_DOWN_HOSTS	Максимальное количество вычислительных узлов, которые могут иметь статус *down*.	Если параметр был превышен, то выключается функция автоэвакуации.
ALLOW_HOST_AUTO_POWER_OFF	Разрешение на автоматический перезапуск вычислительного узла после получения им статуса *down*.	
HOST_RESTART_TIMEOUT	Время ожидания на перезапуск вычислительного узла.	Указывается в секундах.
HOST_ATTACH_RETRY_DELAY	Интервал между попытками добавления вычислительного узла к виртуальному контроллеру OpenStack.	Указывается в секундах.
HOST_ATTACH_MAX_RETRIES	Максимальное количество попыток добавления вычислительного узла к виртуальному контроллеру OpenStack.	
HOST_ATTACH_NETWORK_TAG	Тэг для фильтрации сетей при выборе сетевого интерфейса виртуального контроллера.	
KEY_PATH	Путь до файла с ключом, необходимый для входа в виртуальный контроллер OpenStack.	Используется для входа через протокол SSH.
CONTROLLER_AUTH_PATH	Путь до файла с параметрами окружения, содержащие данные аутентификации для виртуального контроллера OpenStack.	
CONTROLLER_USERNAME	Имя пользователя, необходимое для входа в окружение виртуального контроллера OpenStack.	Используется для входа через протокол SSH.
SENTRY	Параметры логирования событий, происходящие в сервисе, с использованием сервиса Sentry.	В основном, необходимо для сбора данных мониторинга и сообщений об ошибках в удаленный сервис Sentry.

Имя параметра	Описание	Примечания
SENTRY → ENABLED	Включение отправки данных мониторинга и сообщений об ошибках удаленному серверу Sentry.	
SENTRY → LOG_LEVEL	Уровень логирования событий для отправки.	
SENTRY → DSN	Название источника данных (DSN[147]), зарегистрированный в удаленном сервере Sentry.	DSN оформляется по следующей схеме: `http(s)://public_key:secret_key@domain/project_id`
ENABLE_CEPH_INTEGRATION	Включения функции выделенных пулов с независимыми дисковыми устройствами в системе хранения Ceph.	Эта функция создавала отдельные изолированные группы дисков. Было предназначено для изоляции данных репликации между пулами.
ENABLE_NETWORK_ISOLATION	Включение изоляции сетей, предназначенные для виртуальных контроллеров OpenStack.	
NETWORK_ISOLATION_API_HOST	Адрес API сервиса, реализующий изоляцию сети.	
NETWORK_ISOLATION_API_PORT	Порт API сервиса, реализующий изоляцию сети.	
RETRIES_WAIT_FOR_VM_STATUS	Максимальное количество попыток опроса состояния гипервизора.	
RETRIES_WAIT_FOR_NODE_STATE	Максимальное количество попыток опроса состояния виртуальной машины.	
ALLOW_EVACUATE_HOST	Разрешение на эвакуацию виртуальных машин из недоступного вычислительного узла.	
NODE_CONTROL_API_LISTEN	Адрес прослушивания для API сервиса NodeControl.	По умолчанию используется адрес 0.0.0.0. Рекомендуется использование только адреса сети управления.
NODE_CONTROL_API_LISTEN_PORT	Порт прослушивания для API сервиса NodeControl.	
NODE_CONTROL_API_AUDIT_ENABLED	Включение журналирования всех запросов, которые выполняются для NodeControl API.	

147 https://docs.sentry.io/product/sentry-basics/dsn-explainer/

Имя параметра	Описание	Примечания
NODE_CONTROL_API_LOGFILE	Путь до файла журналов для запросов к NodeControl API. Используется сервисом **tionix-node-control-api.**	Необходимо выбрать **один из двух** указанных параметров: • **/dev/stdout**, если установка *производится в рамках референсной архитектуры***,** в этом случае журналы будут перенаправлены в виртуальное устройство /dev/stdout. Это позволит получить журналы системой управления контейнерами; • **путь до файла**, если производится *обычная установка*, в этом случае журналы будут сохранены в указанном по пути файле. Нативная поддержка сервиса Journald на данный момент отсутствует.
NODE_CONTROL_NODE_SYNCER_LOGFILE	Путь до файла журналов сервиса по синхронизации данных состояния вычислительных узлов. Используется сервисом **tionix-node-control-node-syncer.**	Необходимо выбрать **один из двух** указанных параметров: • **/dev/stdout**, если установка *производится в рамках референсной архитектуры***,** в этом случае журналы будут перенаправлены в виртуальное устройство /dev/stdout. Это позволит получить журналы системой управления контейнерами; • **путь до файла**, если производится *обычная установка*, в этом случае журналы будут сохранены в указанном по пути файле. Нативная поддержка сервиса Journald на данный момент отсутствует.
NODE_CONTROL_NODE_TRACKER_LOGFILE	Путь до файла журналов сервиса по отслеживанию задач для вычислительных узлов. Используется сервисом **tionix-node-control-node-tracker.**	Необходимо выбрать **один из двух** указанных параметров: • **/dev/stdout**, если установка *производится в рамках референсной архитектуры***,** в этом случае журналы будут перенаправлены в виртуальное устройство /dev/stdout. Это позволит получить журналы системой управления контейнерами; • **путь до файла**, если производится *обычная установка*, в этом случае журналы будут сохранены в указанном по пути файле. Нативная поддержка сервиса Journald на данный момент отсутствует.

Имя параметра	Описание	Примечания
NODE_CONTROL_WORKER_LOGFILE	Путь до файла журналов сервиса, выполняемые основные задачи сервиса NodeControl. Используется сервисом **tionix-node-control-worker.**	Необходимо выбрать **один из двух** указанных параметров: • **/dev/stdout**, если установка *производится в рамках референсной архитектуры*, в этом случае журналы будут перенаправлены в виртуальное устройство /dev/stdout. Это позволит получить журналы системой управления контейнерами; • **путь до файла**, если производится *обычная установка*, в этом случае журналы будут сохранены в указанном по пути файле. Нативная поддержка сервиса Journald на данный момент отсутствует.
NODE_CONTROL_NOVA_LISTENER_LOGFILE	Путь до файла журналов сервиса, отслеживающие задачи над виртуальными машинами. Используется сервисом **tionix-node-control-nova-listener.**	Необходимо выбрать **один из двух** указанных параметров: • **/dev/stdout**, если установка *производится в рамках референсной архитектуры*, в этом случае журналы будут перенаправлены в виртуальное устройство /dev/stdout. Это позволит получить журналы системой управления контейнерами; • **путь до файла**, если производится *обычная установка*, в этом случае журналы будут сохранены в указанном по пути файле. Нативная поддержка сервиса Journald на данный момент отсутствует.

Дополнительные параметры

Эти параметры не включены в предлагаемую конфигурацию, однако при необходимости их можно добавить.

Имя параметра	Описание	Примечания
LOG_LEVEL	Уровень детализации для всех типов журналов. По умолчанию: **INFO.**	
ENABLE_AGENT	Включение функций управления вычислительным узлом, предоставляемые модулем Tionix Agent. По умолчанию: **False.**	
RABBIT_QUEUES	Указание имени виртуального хоста в сервисе RabbitMQ. По умолчанию: **"vhost: tionix".**	
NODE_CONTROL_AGENT_LOGFILE	Путь до файла журналов, регистрирующий взаимодействие NodeControl с агентом TIONIX: • используется сервисом **tionix-node-control-agent** • зависит от ENABLE_AGENT	
NODE_CONTROL_DRS_TRIGGER_LOGFILE	Путь до файла журналов, регистрирующий срабатывание триггером системе	

Имя параметра	Описание	Примечания
	восстановления после катастрофической ситуации (DRS[148]): • используется сервисом **tionix-node-control-drs-trigger;** • зависит от: **DRS**.	
DRS	Параметры подключения к системе восстановления после катастрофической ситуации (DRS[149]).	
DRS → DRS_HOSTNAME	Адрес системы восстановления после катастрофической ситуации (DRS[150]): • зависит от: **DRS.**	
DRS → DRS_PORT	Порт системы восстановления после катастрофической ситуации (DRS[151]): • по умолчанию: 80; • зависит от: **DRS.**	
DRS → DRS_USER	Имя пользователя к системе восстановления после катастрофической ситуации (DRS[152]): • зависит от: **DRS.**	
DRS → DRS_PASSWORD	Пароль для пользователя в системе восстановления после катастрофической ситуации (DRS[153]): • зависит от: **DRS**.	
STORAGE_SYNC_INTERVAL	Интервал между запросами на получение данных о дисках в сервисе Cinder. По умолчанию: 60.	Указывается в секундах.
RECOVERY_PRIORITY	Указывает уровень приоритета узлов при автоэвакуации виртуальных машин.	Выбирается целое число в диапазоне от 1 до 10.
PRIORITIZED_EVACUATION_TIMEOUT	Время ожидания автоэвакуации между различными группами приоритезации автоэвакуации.	
EXTRA_AVAILABILITY_CHECK	Параметры для дополнительной проверки доступности вычислительных узлов.	Для этой проверки используется общее хранилище, куда записываются данные проверки по доступности вычислительного узла.
EXTRA_AVAILABILITY_CHECK → CHECK_ENABLED	Включение дополнительной проверки: • по умолчанию: **False;** • зависит от: **EXTRA_AVAILABILITY_CHECK.**	

148 https://en.wikipedia.org/wiki/Disaster_recovery
149 https://en.wikipedia.org/wiki/Disaster_recovery
150 https://en.wikipedia.org/wiki/Disaster_recovery
151 https://en.wikipedia.org/wiki/Disaster_recovery
152 https://en.wikipedia.org/wiki/Disaster_recovery
153 https://en.wikipedia.org/wiki/Disaster_recovery

Имя параметра	Описание	Примечания
EXTRA_AVAILABILITY_CHECK → DELAY	Интервал между попытками доступа к файлу, содержащий информацию о состоянии вычислительных узлов: • по умолчанию: 60; • зависит от: **CHECK_ENABLED.**	Указывается в секундах.
EXTRA_AVAILABILITY_CHECK → ATTEMPTS	Количество попыток доступа к файлу, содержащий информацию о состоянии вычислительных узлов: • по умолчанию: 2; • зависит от: **CHECK_ENABLED.**	
EXTRA_AVAILABILITY_CHECK → INSTANCE_RATE	Количество виртуальных машин (инстансов), которые должны быть активны в вычислительном узле: • по умолчанию: 100; • зависит от: **CHECK_ENABLED.**	Указывается в процентах от 0 до 100.

Отладка

Для максимального подробного журналирования ошибок при выполнении работы сервиса можнно включить поддержку трассировок:

Имя параметра	Описание	Примечания
TRACEBACK_ENABLED	Включение трассирки ошибок в файлах журналов. По умолчанию: **False.**	

Настройка хранилища статусов узлов NodeControl

Введение

NodeControl при использовании функции автоэвакуации может воспользоваться дополнительным механизмом проверки узлов, а не только полагаться на информацию, которую предоставляет сервис Nova. Для этого в NodeControl можно настроить хранилище статусов NodeControl, затем привязать его к вычислительному узлу.

Для своей работы требует настроенные сервисы:

- NodeControl (УУ)
- Agent (ВУ)

Перед включением этой функции убедитесь, что требуемые сервисы настроены и корректно работают.

Логика работы хранилища

Хранилища проверки доступности предназначены для дополнительной проверки корректности изменения статуса сервиса Nova Compute и проверки процента запущенных виртуальных машин на вычислительном узле при его переходе из статуса up в статус down. После создания и назначения хранилища проверки доступности, средствами модуля Tionix Agent производится сбор данных о состоянии виртуальных машин на узле и запись данных в файл хранилища. Хранилище имеет директорию для вычислительного узла, а также директорию для хоста с установленным NodeControl. Если в конфигурационном файле этого модуля настроена дополнительная проверка доступности, то в случае, когда статус вычислительного узла меняется с up на down, и над узлом[154] не выполнялось операций с помощью средств управления питанием, будет запущена дополнительная проверка доступности через хранилища, общие для вычислительного узла и хоста, на котором установлен NodeControl. Проверка будет производиться поэтапно:

- Средствами модуля NodeControl осуществляется попытка считать данные из файла в хранилище, которое также подключено к вычислительному узлу и осуществляет сбор данных о статусе виртуальных машин с помощью модуля Agent;
- Если попытка считывания данных оказалась неуспешной, то NodeControl будет пытаться получить данные от других хранилищ, подключенных к виртуальному узлу;

154 https://docs.tionix.ru/3.0.52/glossary/index.html#term-5

- Если не были считаны данные ни с одного хранилища доступности, то выполняется задержка при повторном считывания данных со всех хранилищ доступности, подключенных к вычислительным узлам, время задержки определяется значением параметра `DELAY` из конфигурационного файла. Когда количество попыток считывания данных превысило значение параметра `ATTEMPTS`, выставленное в конфигурационном файле, вычислительный узел считается выключенным некорректно и обработка выполняется по стандартному сценарию;
- Если считывание данных прошло успешно, и запись в файл произошла раньше перехода узла из `up` в `down`, то осуществляются попытки считывания данных с других хранилищ доступности, подключенных к вычислительному узлу. Если считывание данных прошло успешно и время записи в файл более позднее, чем переход вычислительных узлов из `up` в `down`, то определяется процент запущенных виртуальных машин на узле;
- Если процент запущенных машин на узле больше или равен указанному в конфигурационном файле значению, то изменение статуса вычислительного узла с `up` на `down` считается корректным. Узел не включается в список потерянных, и дополнительные действия над ним не требуются;
- Если процент запущенных машин на узле меньше значения из конфигурационного файла, то изменение статуса узла считается некорректным. Вычислительный узел помечается как потерянный и учитывается при расчете количества узлов в статусе `down` для сравнения со значением параметра `MAX_DOWN_HOSTS` конфигурационного файла.

Предварительная настройка

Данное хранилище создается на всех контроллерах и вычислительных узлах. Пути могут быть произвольными, однако в качестве умолчания рекомендуем использовать путь /etc/tionix/statestore. Если хранилищ несколько, то создайте каталоги внутри statestore.

На всех УУ и ВУ создайте каталог с этим путем и укажите tionix в качестве имени пользователя системы и группы:

```
mkdir /etc/tionix/statestore
chown tionix:tionix /etc/tionix/statestore
```

На этом предварительная настройка завершена.

Настройка хранилища и привязка к ВУ

Вход в платформу

Зайдите в окружение контроллера облачной платформы по SSH:

```
ssh root@controller
```

Настройте параметры окружения для возможности входа в облачную платформу:

```
source ${HOME}/admin-openrc.sh
```

Создание хранилища

Первым делом необходимо создать само хранилище, где используется следующий синтаксис:

```
openstack tnx storage create NAME /path/to/statestore/dir/on/compute /path/to/
statestore/dir/on/controller
```

Где:

- **NAME** - имя хранилища состояния.
- **/path/to/statestore/dir/on/compute** - это абсолютный путь в файловой системе ОС, запущенная в ВУ и где предполагается хранить данные состояния узлов. Для всех ВУ этот путь должен быть одинаковым.
- **/path/to/statestore/dir/on/controller** - это абсолютный путь в файловой системе ОС, запущенная в УУ и где предполагается хранить данные состояния узлов. Для всех УУ этот путь должен быть одинаковым.

Для нашего примера пути команда будет выглядеть так:

```
openstack tnx storage create default /etc/tionix/statestore /etc/tionix/statestore
```

Вы должны получить примерно такой вывод:

```
+------------------+----------------------+
|Field             |Value                 |
+------------------+----------------------+
|Storage ID        |2                     |
+------------------+----------------------+
|Storage Name      |default               |
+------------------+----------------------+
|Path for compute  |/etc/tionix/statestore|
+------------------+----------------------+
|Path for controller|/etc/tionix/statestore|
+------------------+----------------------+
```

Привязка хранилища к ВУ

Для привязки хранилища к ВУ используется отдельная команда assign со следующим синтаксисом:

```
openstack tnx storage assign STORAGE_ID --nodes NODE_ID
```

Где:

- **STORAGE_ID** - ID хранилища статусов (в выводе создания - значение Storage ID).
- **NODE_ID** - ID гипервизора вычислительного узла, можно указать несколько через пробел. Не используйте ID сервиса вычислений nova-compute.

⚠ Важно не менять порядок синтаксиса: вначале всегда нужно указывать ID хранилища, а только потом ID гипервизоров.

Пример команды:

```
openstack tnx storage assign 2 --nodes 1 2 3
```

При успехе вы должны следующее сообщение:

```
Nodes have been assigned.
```

Отвязка хранилища от ВУ

Для привязки хранилища к ВУ используется отдельная команда assign со следующим синтаксисом:

```
openstack tnx storage unassign STORAGE_ID --node NODE_ID
```

Где:

- **STORAGE_ID** - ID хранилища статусов (в выводе создания - значение Storage ID).
- **NODE_ID** - ID гипервизора вычислительного узла, к которому привязано хранилище, можно указать только один узел. Не используйте ID сервиса вычислений nova-compute.

Пример команды:

```
openstack tnx storage unassign 2 --node 2
```

При успехе вы должны следующее сообщение:

```
Node has been unassigned.
```

Удаление хранилища

Для привязки хранилища к ВУ используется отдельная команда assign со следующим синтаксисом:

```
openstack tnx storage delete STORAGE_ID
```

Где:

- **STORAGE_ID** - ID хранилища статусов (в выводе создания - значение Storage ID).

Перед удалением убедитесь, что вы отвязали хранилище от всех вычислительных узлов.

Пример команды:

```
openstack tnx storage delete 2
```

При успехе вы получите сообщение:

```
Storage with id "1" has been deleted
```

Настройка устройств управления питанием узлов

Введение

NodeControl при использовании функции автоэвакуации может воспользоваться возможностями управления питанием вычислительных узлов для их временного вывода из эксплуатации. Для этого NodeControl может регистрировать устройства питания, которые затем можно привязать в вычислительному узлу.

Для своей работы требует настроенные сервисы:

- NodeControl (УУ)
- Соответствующие протоколы для управления питанием (IPMI, SSH и так далее).

Перед включением этой функции убедитесь, что требуемые сервисы настроены и корректно работают.

Возможности

Утилиты пакета управления питанием вычислительных узлов позволяют:

- Определить и предоставить по запросу информацию о соответствии вычислительного узла и контактной площадки устройства управления питанием;
- Назначить резервные вычислительные узлы (включаются в случае выхода из строя сопоставленных с ними основных, заменяя собой нерабочий основной вычислительный узел);
- Управлять питанием по адресу контактных площадок и устройств управления питанием;
- Управлять питанием по именам вычислительных узлов.

Поддерживаемые типы устройств

Для управления питанием используется аппаратно-программный комплекс. Аппаратная часть состоит из устройств управления питанием, например: ICPDAS, DAEnetIP2 и др. Программная же часть состоит из клиента, позволяющего удаленно управлять устройством.

Реализована поддержка следующих устройств управления питанием:

- Устройства на основе платы DAEnetIP2, использующей протокол SNMP;
- Устройства ICP DAS ET-7067, использующие протокол MODBUS;
- Устройства с поддержкой технологии AMT;
- Устройства с поддержкой интерфейса IPMI;
- Виртуальные устройства, использующие протокол SSH для управления гипервизором.

Модуль предоставляет следующие возможности:

1. Получать информацию о состоянии питания портов устройства;
2. Управлять состоянием портов устройства - включать, выключать;
3. Для устройств, реализующих ACPI управление устройствами, запускать "мягкое" выключение.

Настройка устройств управления

Основные команды

Инициализация устройства питания

Основной командой инициализации устройства является следующая:

```
openstack tnx power init
```

Команда работает в интерактивном режиме и состоит из следующих вопросов:

- Тип устройства.
- Тип коммуникационного (сетевого) протокола.
- IP-адрес или доменное имя устройства управления питанием.
- Сетевой порт устройства (от 1 до 65535).

- Имя пользователя для устройства управления.
- Пароль пользователя для устройства управления.
- Имя устройства управления в NodeControl.

В зависимости от типа добавляемого устройства настройка будет несколько отличаться.

Получение списка устройств

Для получения списка устройств питания выполните команду:

```
openstack tnx power list
```

Управление устройствами питания

Команда manage позволяет управлять устройствами питания:

```
openstack tnx power manage
```

Настройка IPMI (для реализации Supermicro)

Запустите команду инициализации и ответьте утвердительно на первый вопрос:

```
openstack tnx power init
```

Далее команда выведет список типов устройств:

```
Select control name:
1: DaenetIP2
2: DaenetIP2_ACPI
3: ET7067
4: IntelAMT
5: SshDevice
6: SupermicroRackDevice
```

Выберите пункт 6.

Следующий вопрос связан с типом протокола подключения:

```
Select protocol name:
1: intel_amt
2: ipmi
3: modbus
4: snmp
5: ssh
```

Здесь необходимо выбрать 2.

Далее команда очередно спросит параметры:

- IP-адреса или доменного имени. Здесь нужно указать адрес IPMI/BMC-устройства.
- Порт устройства. Укажите 623, если порт менялся, то на соответствующий.
- Имя пользователя и пароль. Укажите те, что используются для входа в сессию IPMI. Желательно использовать ограниченную учетную запись.
- Имя устройства. Любое, этот параметр используется для регистрации в NodeControl.

После окончания добавления команда придет в изначальное состояние и заново спросить первый вопрос о добавлении устройства:

```
Do you want to add a new control ([y]/n)
```

Если вы более не хотите ничего добавлять, то укажите n. После чего команда сообщит об успешном добавлении устройства:

```
Added 1 power controls.
```

Далее команда спросит о дальнейших шагах:

```
Do you want to (a)dd control, (d)elete or (e)dit existing (e/a/d/[n])?
```

Для выхода нажмите n.

В случае отсутствия соединения с IPMI-устройством или, например, неверных данных входа команда вернет ошибку:

```
2022-02-28 08:26:21.107 1148480 ERROR openstack [-] None
```

⚠ Необходимо отметить, что добавление устройство питания пока не будет отображаться в списке настроенных устройств, которые получаются при команде list

SSHDevice

Запустите команду инициализации и ответьте утвердительно на первый вопрос:

```
openstack tnx power init
```

Далее команда выведет список типов устройств:

```
Select control name:
1: DaenetIP2
2: DaenetIP2_ACPI
3: ET7067
4: IntelAMT
5: SshDevice
6: SupermicroRackDevice
```

Выберите пункт 5.

Следующий вопрос связан с типом протокола подключения:

```
Select protocol name:
1: intel_amt
2: ipmi
3: modbus
4: snmp
5: ssh
```

Здесь необходимо выбрать 5.

Далее команда очередно спросит параметры:

- IP-адреса или доменного имени. Здесь нужно указать адрес IPMI/BMC-устройства.
- Порт устройства. Укажите 22, если порт менялся, то на соответствующий.
- Имя пользователя и пароль. Укажите те, что используются для входа в сессию SSH Желательно использовать ограниченную учетную запись с поддержкой команды poweroff.
- Имя устройства. Любое, этот параметр используется для регистрации в NodeControl.

После окончания добавления команда придет в изначальное состояние и заново спросить первый вопрос о добавлении устройства:

```
Do you want to add a new control ([y]/n)
```

Если вы более не хотите ничего добавлять, то укажите n. После чего команда сообщит об успешном добавлении устройства:

```
Added 1 power controls.
```

Далее команда спросит о дальнейших шагах:

```
Do you want to (a)dd control, (d)elete or (e)dit existing (e/a/d/[n])?
```

Для выхода нажмите n.

В случае отсутствия соединения с IPMI-устройством или, например, неверных данных входа команда вернет ошибку:

```
2022-02-28 08:26:21.107 1148480 ERROR openstack [-] None
```

Необходимо отметить, что добавление устройство питания пока не будет отображаться в списке настроенных устройств, которые получаются при команде list.

На момент написания документации при попытке добавить устройство SSH появляется ошибка:

```
('Device could not be added!', PowerDeviceNotAvailable('Control "SshDevice" is
not available.',))
```

Scheduler

Информация о сервисе Scheduler

TIONIX Scheduler - это сервис, исполняющий функции планировщика различных событий, которые должны выполняться к какому-то моменту времени или по определенному периоду времени. Для функции планирования задач используется библиотека Celery[155].

Компоненты сервиса Scheduler

Scheduler состоит из нескольких компонентов, оформленных как сервисы в systemd:

- **tionix-scheduler-beat** - определяет, что та или иная задача готова к исполнению через периодическую проверку их статуса;
- **tnx-scheduler-worker** - выполняет задачи, выбранные сервисом beat как готовые к выполнению.

Установка сервиса Scheduler

Настройка окружения

Перед самой установкой сервиса нужно предварительно настроить некоторые компоненты инфраструктуры.

См. также: Раздел с предварительной настройкой окружения для модулей TIONIX (см. стр. 116).

Подготовка базы данных tionix_scheduler

См. также: Установка и настройка СУБД MariaDB (см. стр. 22)

Всю информацию о данных для аутентификации и авторизации по умолчанию TIONIX Scheduler хранит в базе данных MariaDB.

1. Войдите в окружение базы данных:

```
mysql -u root -p
```

2. Создайте базу данных tionix_scheduler:

```
create database tionix_scheduler;
```

3. Предоставьте доступ к этой базе данных пользователю tionix в СУБД (для localhost и всем остальным адресам отдельно, вместо TIONIX_SCHED_DBPASS используйте свой пароль):

```
grant all privileges on tionix_scheduler.* to 'tionix'@'localhost' identified by
'TIONIX_DBPASS';
grant all privileges on tionix_scheduler.* to 'tionix'@'%' identified by
'TIONIX_DBPASS';
```

4. Выйдите из сессии СУБД

155 https://docs.celeryproject.org/en/stable/getting-started/introduction.html

```
exit;
```

Установка модуля

Процесс установки

1. Установите пакет модуля:

```
dnf -y install python3-tionix_scheduler
```

> **Информация**
>
> Пути конфигурации модуля:
>
> - `/etc/tionix` – основной каталог конфигурации;
> - `/etc/tionix/tionix.yaml` – общий файл конфигурации TIONIX;
> - `/etc/tionix/scheduler.yaml` – основной файл конфигурации сервиса Scheduler.
>
> Основной файл конфигурации может отсутствовать после установки пакета, в этом случае достаточно его создать.

2. В основной файл конфигурации сервиса Scheduler включите следующую конфигурацию:

```
# Общие параметры Scheduler
CELERYBEAT_SYNC_EVERY: 60
CELERYBEAT_MAX_LOOP_INTERVAL: 30

ENTRY_GROUPS:
    - tionix_tasks

# Параметры журналирования
SCHEDULER_WORKER_LOGFILE: '/var/log/tionix/scheduler/worker.log'
SCHEDULER_BEAT_LOGFILE: '/var/log/tionix/scheduler/beat.log'

# Параметры базы данных
DB:
    NAME: 'tionix_scheduler'

# Параметры сервиса Sentry
SENTRY:
    ENABLED: False
    LOG_LEVEL: INFO
    DSN: http://PUBLIC_KEY:SECRET_KEY@SENTRY_ADDR/PROJECT_ID
```

3. Выполните первичную инициализацию модуля (вместе с модулем TIONIX Client):

```
openstack tnx configure -n tnx_scheduler tnx_client
```

4. Выполните миграцию базы данных:

```
openstack tnx db migrate -n tnx_scheduler
```

Создание сервиса Scheduler

1. Создайте сервис Scheduler:

```
openstack service create --name tnx-scheduler --description "TIONIX Scheduler
Service" tnx-scheduler
```

2. Создайте точки входа (endpoint):
   a. публичную:

```
openstack endpoint create --region RegionOne tnx-scheduler public http://
controller:10001
```

   b. внутреннюю:

```
openstack endpoint create --region RegionOne tnx-scheduler internal http://
controller:10001
```

c. административную:

```
openstack endpoint create --region RegionOne tnx-scheduler admin http://
controller:10001
```

Финализация установки

1. Включите и запустите службы, реализующие функциональность доступа к зарегистрированным событиям для объектов облачной платформы:

```
systemctl start tionix-scheduler-beat.service
systemctl start tionix-scheduler-worker.service
systemctl enable tionix-scheduler-beat.service
systemctl enable tionix-scheduler-worker.service
```

ⓘ Сервис Tionix Scheduler API запускается как wsgi-приложение в веб-сервере Apache.

2. Перезапустите все службы TIONIX для завершения регистрации сервиса Scheduler, а также веб-сервер httpd для перезапуска сервиса Scheduler API:

```
systemctl restart tionix-* httpd
```

Проверка работы сервиса

1. Проверьте статус сервисов, например, tionix-scheduler-beat:

```
systemctl status tionix-scheduler-beat
```

ⓘ Ответ должен быть примерно таким:

```
● tionix-scheduler-beat.service - TIONIX Scheduler beat
   Loaded: loaded (/usr/lib/systemd/system/tionix-scheduler-beat.service;
enabled; vendor preset: disabled)
  Drop-In: /run/systemd/system/tionix-scheduler-beat.service.d
           └─zzz-lxc-service.conf
   Active: active (running) since Tue 2021-11-09 21:45:14 UTC; 1min 44s ago
 Main PID: 175808 ([celery beat])
    Tasks: 1 (limit: 204240)
   Memory: 84.3M
   CGroup: /system.slice/tionix-scheduler-beat.service
           └─175808 [celery beat]
...
```

Обратите внимание на наличие процесса celery beat в CGroup. Это знак успешного запуска сервиса Celery.

2. Проверьте статус порта сервиса:

```
ss -tnlp | grep 10001
```

ⓘ Ответ должен быть примерно таким:

```
LISTEN 0      511                *:10001            *:*    users:(("httpd",
pid=175976,fd=10),...
```

Описание файла конфигурации Scheduler

В описании процесса по установке сервиса Scheduler предложено содержимое конфигурации, которое предложено сохранить в управляющем узле по пути `/etc/tionix/scheduler.yaml`. Это страница содержит подробное описание этой конфигурации.

При изменении конфигурации необходимо перезапустить все компоненты сервиса Scheduler:

```
systemctl restart tionix-scheduler-*
```

Таблица конфигурации

> **Немного о формате yaml**
>
> Файл формата yaml[156] критичен к отступам текста[157] при описании параметров. Стрелкой в имени параметра указан отступ в 4 пробела от начала строки.

> Легенда таблицы доступна на этой странице[158].

Имя параметра	Описание	Примечания
DB	Параметры подключения к базе данных.	
→ **NAME**	Имя базы данных сервиса Scheduler.	Остальные параметры подключения к базе данных берутся с основного файла конфигурации TIONIX.
LOG_LEVEL	Уровень детализации для всех типов журналов. По умолчанию: **INFO**.	
CELERYBEAT_SYNC_EVERY	Количество задач, после выполнения которых требуется обновить список задач (beats).	
CELERYBEAT_MAX_LOOP_INTERVAL	Максимальный интервал между повторяемыми задачами.	
SENTRY	Параметры логирования событий, происходящие в сервисе, с использованием сервиса Sentry.	В основном, необходимо для сбора данных мониторинга и сообщений об ошибках в удаленный сервис Sentry.
→ **ENABLED**	Включение отправки данных мониторинга и сообщений об ошибках удаленному серверу Sentry. Обязателен, если выставлен SENTRY.	
→ **LOG_LEVEL**	Уровень логирования событий для отправки. Обязателен, если выставлен SENTRY.	

156 https://yaml.org/
157 https://yaml.org/spec/1.2.2/#61-indentation-spaces
158 https://conf.tionix.ru/pages/viewpage.action?pageId=205881354#id-Условныеобозначения-table-format-desc

Имя параметра	Описание	Примечания
→ **DSN**	Название источника данных (DSN[159]), зарегистрированный в удаленном сервере Sentry. Обязателен, если выставлен SENTRY.	DSN оформляется по следующей схеме: `http(s)://public_key:secret_key@domain/project_id`
SCHEDULER_WORKER_LOGFILE	Путь до файла журналов исполнителя задач Scheduler. Используется сервисом **tionix-scheduler-worker.**	Необходимо выбрать **один из двух** указанных параметров: • **/dev/stdout**, если установка *производится в рамках референсной архитектуры*, в этом случае журналы будут перенаправлены в виртуальное устройство /dev/stdout. Это позволит получить журналы системой управления контейнерами; • **путь до файла**, если производится *обычная установка*, в этом случае журналы будут сохранены в указанном по пути файле. Нативная поддержка сервиса Journald на данный момент отсутствует.
SCHEDULER_BEAT_LOGFILE	Путь до файла журналов обновления списка задач. Используется сервисом **tionix-scheduler-beat.**	Необходимо выбрать **один из двух** указанных параметров: • **/dev/stdout**, если установка *производится в рамках референсной архитектуры*, в этом случае журналы будут перенаправлены в виртуальное устройство /dev/stdout. Это позволит получить журналы системой управления контейнерами; • **путь до файла**, если производится *обычная установка*, в этом случае журналы будут сохранены в указанном по пути файле. Нативная поддержка сервиса Journald на данный момент отсутствует.

Дебаггинг

Для максимального подробного журналирования ошибок при выполнении работы сервиса можно включить поддержку трассировок:

Имя параметра	Описание	Примечания
TRACEBACK_ENABLED	Включение трассировки ошибок в файлах журналов. По умолчанию: **False.**	

Monitor

Информация о сервисе Monitor

TIONIX Monitor - это сервис, который интегрируется с системами мониторинга для получения статистических данных по потреблению ресурсов и производительности виртуальных машин, а также их отображения в TIONIX Dashboard. Поддерживает следующие системы мониторинга:

159 https://docs.sentry.io/product/sentry-basics/dsn-explainer/

- Ceilometer[160] совместно с Gnocchi[161] - основной сервис, предоставляющие данные статистики по виртуальным машинам;
- Zabbix[162] - получение данных статистики, собранные Zabbix по инфраструктуре.

Дополнительной функцией сервиса является наблюдение за состоянием сервиса резервного копирования Bareos[163].

Список компонентов сервиса

Monitor состоит из нескольких компонентов в виде сервисов в systemd:

- **tionix-monitor-api** - сервис, который реализует и даёт доступ к Monitor API;
- **tionix-monitor-nova-listener** - сервис, регистрирующий события по виртуальным машинам с использованием механизмов Nova;
- **tionix-monitor-tionix-listener** - сервис, регистрирующие события в самих модулях TIONIX.

Установка сервиса TIONIX Monitor

Настройка окружения

Перед самой установкой сервиса нужно предварительно настроить некоторые компоненты инфраструктуры.

См. также: Раздел с предварительной настройкой окружения для модулей TIONIX (см. стр. 116).

Подготовка базы данных tionix_monitor

См. также: Установка и настройка СУБД MariaDB (см. стр. 22)

Всю информацию о данных для аутентификации и авторизации по умолчанию TIONIX Monitor хранит в базе данных MariaDB.

1. Войдите в окружение базы данных:

```
mysql -u root -p
```

2. Создайте базу данных tionix_monitor:

```
create database tionix_monitor;
```

3. Предоставьте доступ к этой базе данных пользователю tionix в СУБД (для localhost и всем остальным адресам отдельно, вместо TIONIX_MON_DBPASS используйте свой пароль):

```
grant all privileges on tionix_monitor.* to 'tionix'@'localhost' identified by
'TIONIX_DBPASS';
grant all privileges on tionix_monitor.* to 'tionix'@'%' identified by
'TIONIX_DBPASS';
```

4. Выйдите из сессии СУБД:

```
exit;
```

Установка модуля

Перед установкой модуля убедитесь в установке и корректной настройке сервиса OpenStack Ceilometer (см. стр. 104).

1. Установите пакет модуля:

160 https://docs.openstack.org/ceilometer/latest/
161 https://specs.openstack.org/openstack/telemetry-specs/specs/juno/gnocchi.html
162 https://www.zabbix.com/ru
163 https://www.bareos.com/

```
dnf -y install python3-tionix_monitor
```

Стандартные пути до файлов конфигурации:

- `/etc/tionix` – основной каталог конфигурации;
- `/etc/tionix/tionix.yaml` – общий файл конфигурации TIONIX;
- `/etc/tionix/monitor.yaml` – основной файл конфигурации сервиса Monitor.

2. Выполните первичную инициализацию модуля (вместе с модулем TIONIX Client):

```
openstack tnx configure -n tnx_monitor tnx_client
```

3. Добавьте эту конфигурацию сервиса в файл `/etc/tionix/monitor.yaml` (описание (см. стр. 149)):

```
# Основные параметры сервиса
DB:
    NAME: 'tionix_monitor'
MONITOR_API_LISTEN: 'LISTEN_IP'
MONITOR_API_LISTEN_PORT: 9363
MONITOR_API_AUDIT_ENABLED: True

# Параметры журналирования сервиса
LOG_LEVEL: 'INFO'
MONITOR_API_LOGFILE: '/var/log/monitor/api.log'
MONITOR_NOVA_LISTENER_LOGFILE: '/var/log/monitor/nova-listener.log'

# Параметры сбора данных
ENABLE_CEILOMETER_MONITORING: True
CEILOMETER_METERS:
    - 'memory.usage'
    - 'cpu_util'
    - 'disk.device.read.requests.rate'
    - 'disk.device.write.requests.rate'
    - 'disk.device.read.bytes.rate'
    - 'disk.device.write.bytes.rate'
    - 'disk.device.latency'
    - 'disk.device.iops'
    - 'disk.read.requests.rate'
    - 'disk.write.requests.rate'
    - 'disk.read.bytes.rate'
    - 'disk.write.bytes.rate'
    - 'network.incoming.bytes.rate'
    - 'network.outgoing.bytes.rate'
    - 'network.incoming.packets.rate'
    - 'network.outgoing.packets.rate'

ENABLE_ZABBIX_MONITORING: False
ENABLE_BACKUP: False

# Включение поддержки Sentry
SENTRY:
    ENABLED: False
    LOG_LEVEL: INFO
    DSN: http://SET_PUBLIC_KEY:SET_SECRET_KEY@SENTRY_IP/PROJECT_ID
```

4. При выполнении конфигурирования следует обратить особое внимание на следующие параметры:

Параметр	Значение
DB → NAME	Укажите корректное имя базы, указанное при при настройке БД для Monitor. По умолчанию: **tionix_monitor**.
MONITOR_API_LISTEN	Укажите корректный слушаемый адрес в зависимости от типа установки: • классическая архитектура: Адрес сети управления (mgmt) облачной платформой; • референсная архитектура: 0.0.0.0.

Параметр	Значение
Параметры LOGFILE	Укажите пути файлов журналирования в зависимости от типа установки: • классическая архитектура: как предложено в конфигурации; • референсная архитектура: **/dev/stdout** или **/dev/stderr**.
CEILOMETER_METERS	При включении сбора данных мониторинга с использованием Ceilometer (**ENABLE_CEILOMETER_MONITORING = True**) в этом списке укажите нужные метрики.

> Информацию об остальных параметрах можно получить в описании файла конфигурации Monitor (см. стр. 149).

5. Выполните миграцию базы данных:

```
openstack tnx db migrate -n tnx_monitor
```

Создание сервиса Monitor API

1. Создайте сервис Monitor API:

```
openstack service create --name tnx-monitor --description "TIONIX Monitor Service"
 tnx-monitor
```

2. Создайте точки входа (endpoint):

```
openstack endpoint create --region RegionOne tnx-monitor internal http://
controller:9363
openstack endpoint create --region RegionOne tnx-monitor admin http://controller:
9363
openstack endpoint create --region RegionOne tnx-monitor public http://controller:
9363
```

Финализация установки

1. Включите и запустите службы, реализующие функциональность доступа к зарегистрированным событиям для объектов облачной платформы:

```
systemctl start tionix-monitor-api.service
systemctl start tionix-monitor-tionix-listener.service
systemctl start tionix-monitor-nova-listener.service
systemctl enable tionix-monitor-api.service
systemctl enable tionix-monitor-tionix-listener.service
systemctl enable tionix-monitor-nova-listener.service
```

2. Перезапустите все службы TIONIX для завершения регистрации сервиса Monitor, а также веб-сервер httpd:

```
systemctl restart tionix-* httpd
```

Проверка работы сервиса

1. Проверьте статус сервисов, например, tionix-node-control-api:

```
systemctl status tionix-monitor-api.service
```

В ответ вы должны получить примерно следующее:

```
● tionix-monitor-api.service - TIONIX Monitor API service
   Loaded: loaded (/usr/lib/systemd/system/tionix-monitor-api.service;
enabled; vendor preset: disabled)
  Drop-In: /run/systemd/system/tionix-monitor-api.service.d
           └─zzz-lxc-service.conf
   Active: active (running) since Tue 2021-11-09 21:13:33 UTC; 2min 44s ago
 Main PID: 171547 (/usr/libexec/pl)
    Tasks: 1 (limit: 204240)
   Memory: 83.5M
   CGroup: /system.slice/tionix-monitor-api.service
           └─171547 /usr/libexec/platform-python /usr/bin/tnx-monitor-api
...
```

2. Проверьте статус порта сервиса:

```
ss -tnlp | grep 9363
```

Ответ должен выглядеть примерно так:

```
LISTEN 0      50     10.236.64.231:9363       0.0.0.0:*    users:(("/usr/
libexec/pl",pid=171547,fd=5))
```

Порт должен быть в статусе LISTEN и должен прослушиваться адрес сети управления (mgmt).

Описание файла конфигурации Monitor

В описании процесса по установке сервиса Monitor предложено содержимое конфигурации, которое предложено сохранить в управляющем узле по пути `/etc/tionix/monitor.yaml`. Это страница содержит подробное описание этой конфигурации.

При изменении конфигурации необходимо перезапустить все компоненты сервиса Monitor:

```
systemctl restart tionix-monitor-*
```

Таблица конфигурации

Немного о формате yaml

Файл формата yaml[164] критичен к отступам текста[165] при описании параметров. Стрелкой в имени параметра указан отступ в 4 пробела от начала строки.

Легенда таблицы доступна на этой странице[166].

Имя параметра	Описание	Примечания
DB	Параметры подключения к базе данных.	
→ **NAME**	Имя базы данных сервиса Monitor.	Остальные параметры подключения к базе данных берутся с основного файла конфигурации TIONIX.

164 https://yaml.org/
165 https://yaml.org/spec/1.2.2/#61-indentation-spaces
166 https://conf.tionix.ru/pages/viewpage.action?pageId=205881354#id-Условныеобозначения-table-format-desc

Имя параметра	Описание	Примечания
LOG_LEVEL	Уровень детализации для всех типов журналов. По умолчанию: **INFO**	
CEILOMETER_METERS	Метрики, информацию от которых необходимо получать для хранения и визуализации сервисом Monitor. Обязателен, если выставлен ENABLE_CEILOMETER_MONITORING.	
ENABLE_CEILOMETER_MONITORING	Включение получения данных мониторинга с сервиса Ceilometer.	
ENABLE_ZABBIX_MONITORING	Включение получения данных мониторинга с сервиса Zabbix.	
ENABLE_BACKUP	Включение получения данных по резервным копиям с сервиса BareOS.	
SENTRY	Параметры логирования событий, происходящие в сервисе, с использованием сервиса Sentry.	В основном, необходимо для сбора данных мониторинга и сообщений об ошибках в удаленный сервис Sentry.
→ **ENABLED**	Включение отправки данных мониторинга и сообщений об ошибках удаленному серверу Sentry. Обязателен, если выставлен SENTRY.	
→ **LOG_LEVEL**	Уровень логирования событий для отправки. Обязателен, если выставлен SENTRY.	
→ **DSN**	Название источника данных (DSN[167]), зарегистрированный в удаленном сервере Sentry. Обязателен, если выставлен SENTRY.	DSN оформляется по следующей схеме: `http(s)://public_key:secret_key@domain/project_id`
MONITOR_API_LISTEN	Адрес прослушивания для API сервиса Monitor.	ⓘ Рекомендуется использование только адреса сети управления.
NODE_CONTROL_API_LISTEN_PORT	Порт прослушивания для API сервиса Monitor.	
MONITOR_API_AUDIT_ENABLED	Включение журналирования всех запросов, которые выполняются для Monitor API.	

167 https://docs.sentry.io/product/sentry-basics/dsn-explainer/

Имя параметра	Описание	Примечания
MONITOR_API_LOGFILE	Путь до файла журналов для запросов к Monitor API. Используется сервисом **tionix-node-monitor-api.**	Необходимо выбрать **один из двух** указанных параметров: • **/dev/stdout**, если установка *производится в рамках референсной архитектуры*, в этом случае журналы будут перенаправлены в виртуальное устройство /dev/stdout. Это позволит получить журналы системой управления контейнерами; • **путь до файла**, если производится *обычная установка*, в этом случае журналы будут сохранены в указанном по пути файле. Нативная поддержка сервиса Journald на данный момент отсутствует.
MONITOR_NOVA_LISTENER_LOGFILE	Путь до файла журналов сервиса по отслеживанию события в сервисе Nova. Используется сервисом **tionix-monitor-nova-listener.**	Необходимо выбрать **один из двух** указанных параметров: • **/dev/stdout**, если установка *производится в рамках референсной архитектуры*, в этом случае журналы будут перенаправлены в виртуальное устройство /dev/stdout. Это позволит получить журналы системой управления контейнерами; • **путь до файла**, если производится *обычная установка*, в этом случае журналы будут сохранены в указанном по пути файле. Нативная поддержка сервиса Journald на данный момент отсутствует.

Отладка

Для максимального подробного журналирования ошибок при выполнении работы сервиса можнно включить поддержку трассировок:

Имя параметра	Описание	Примечания
TRACEBACK_ENABLED	Включение трассирки ошибок в файлах журналов. По умолчанию: **False.**	

Dashboard

Информация о модуле Dashboard

OpenStack Horizon

В составе проектов OpenStack имеется компонент OpenStack[168] Hor[169]izon[170], который предоставляет функции панели управления облачной инфраструктурой с помощью веб-приложения. TIONIX расширил функциональность веб-интерфейса для поддержки всех возможностей своих модулей.

OpenStack Horizon содержит в себе основные функции управления облачной платформой:

- управление виртуальными машинами;
- управление сетями и маршрутизаторами;
- управление блочными устройствами;
- управления образами виртуальными машинам;

168 https://docs.openstack.org/horizon/victoria/
169 https://docs.openstack.org/horizon/victoria/
170 https://docs.openstack.org/horizon/victoria/

- управление пользователями и проектами.

Сервисы OpenStack могут содержать дополнительные плагины для OpenStack Horizon, расширяющий список возможностей по управлению платформой.

OpenStack Horizon использует порт 443/TCP и является веб-приложением.

TIONIX Dasboard

TIONIX Dashboard добавляет следующие функции в веб-панель:

- расширенные функции управления гипервизорами;
- управление инфраструктурой VDI;
- планирование задач над виртуальными машинами;
- поддержка систем управления питанием вычислительных узлов;
- исправление большого количества проблем, которые имеются в оригинальном Horizon.
- прочие функции, предоставляемые модулями TIONIX.

Для полноценной работы всех функций Dashboard требуется установка следующих модулей TIONIX:

1. NodeControl (см. стр. 123) – для расширения функций гипервизоров;
2. Scheduler (см. стр. 141) – для задач планирования;
3. Monitor (см. стр. 145) – для визуализации данных потребления ресурсов виртуальными машинами;
4. Модуль лицензирования TIONIX – для показа данных по лицензии для платформы.

Описание файла конфигурации сервиса Dashboard

В описании процесса по установке сервиса Dashboard[171] предложена стандартная конфигурация для настройки. Это страница содержит подробное описание этой конфигурации.

При изменении конфигурации необходимо перезапустить веб-сервер и службу кэширования:

```
systemctl restart httpd
systemctl restart memcached
```

Таблица конфигурации

Легенда таблицы доступна на этой странице[172].

Немного о формате yaml

Файл формата yaml[173] критичен к отступам текста[174] при описании параметров. Стрелкой в имени параметра указан отступ в 4 пробела от начала строки, а между стрелками указана связь между родительским и конечным параметром.

Параметр	Описание	Значение по умолчанию
LOG_LEVEL	Уровень логирования. Доступные значения: • DEBUG; • INFO; • WARNING; • ERROR; • CRITICAL. Значения являются регистронезависимыми.	INFO

171 https://conf.tionix.ru/x/KoCBCg
172 https://conf.tionix.ru/pages/viewpage.action?pageId=205881354#id-Условныеобозначения-table-format-desc
173 https://yaml.org/
174 https://yaml.org/spec/1.2.2/#61-indentation-spaces

Параметр	Описание	Значение по умолчанию
KEYSTONE	Настройки для авторизации в службе Keystone, где: • auth_url - адрес сервиса Keystone; • auth_version - версия Keystone: 2 или 3; • auth_user - логин пользователя; • auth_password - пароль пользователя; • auth_tenant - название проекта; • compute_service_name - название службы Compute; • volume_service_name - название службы Volume; • network_service_name - название службы Neutron; • identity_service_name - название службы Keystone.	• auth_url - http://localhost:5000; • auth_version - 3; • auth_user - admin; • auth_password - admin; • auth_tenant - admin; • compute_service_name - compute; • volume_service_name - volumev2; • network_service_name - network; • identity_service_name - identity.
NEUTRON_VERSION	Версия клиента: 2.	
DB	Настройки базы данных, где: • ENGINE - тип базы данных; • USER - пользователь базы данных; • PASSWORD - пароль базы данных; • HOST - хост, на котором запущена база данных; • PORT — порт сервера с базой данных; • NAME - название базы данных.	• ENGINE - django.db.backends.mysql; • USER - tionix; • PASSWORD - password; • HOST - localhost; • PORT — 3306; • NAME - tionix_dash.

SENTRY	Настройки логирования Sentry, где:
	• ENABLED - Флаг, отвечающий за отправку сообщений об ошибках в Sentry. Возможные значения:
	  • True;
	  • False.
	Значения являются регистронезависимыми.
	• DSN - Адрес сервера Sentry, содержит ключ пользователя и идентификатор проекта;
	• LOG_LEVEL - Уровень логирования в Sentry. Значения являются регистронезависимыми.
		• False;
		• — *Адрес внутреннего сервера* Sentry;
		• CRITICAL.

ENABLE_QOS	Активация вкладки «Сетевые политики QoS». Возможные значения: • True; • False.	False
MIN_RESERVE_VM	Значение по умолчанию для минимального количества резервных виртуальных машин при создании VDI проекта.	null
POINT_METER_API_URL	Настройка доступа до модуля лицензирования по поинтам. Указывается адрес узла с установленным TIONIX.PointMeter. Пример: http://127.0.0.1:9367/.	null

Параметр	Описание	Значение по умолчанию
`VOLUME_ATTACH_MAX_RETRIES`	Пороговое значение. Устанавливает количество попыток подключения диска к VDI-машине.	`40`

Отладка

Для максимального подробного журналирования ошибок при выполнении работы сервиса можнно включить поддержку трассировок:

Имя параметра	Описание	Примечания
TRACEBACK_ENABLED	Включение трассирки ошибок в файлах журналов. По умолчанию: **False.**	

Установка веб-панели Horizon

Важно

Перед установкой TIONIX Dashboard необходимо установить панель управления Horizon (Dashboard).

Установка

1. Для работы OpenStack Horizon требуется настроенный веб-сервер Apache. Если Horizon запускается вместе с сервисами, которые уже используют веб-сервер Apache, то дополнительной настройки самого веб-сервера не требуется. При запуске в отдельном узле установите веб-сервер Apache:

   ```
   dnf -y install httpd
   ```

2. Установите пакет с веб-панелью Horizon:

   ```
   dnf -y install openstack-dashboard
   ```

   Пути конфигурации:

   - `/etc/openstack-dashboard` – общий каталог конфигурации;
   - `/etc/openstack-dashboard/local_settings` – основной файл конфигурации Horizon.

3. Включите следующие настройки в основной файл конфигурации (описание (см. стр. 160)).
   Конфигурация Horizon:

```
import os
from django.utils.translation import ugettext_lazy as _
from openstack_dashboard.settings import HORIZON_CONFIG

# Параметры Django
WEBROOT = "/dashboard"
ALLOWED_HOSTS = ['DOMAIN1', 'DOMAIN2']
LOCAL_PATH = '/tmp'
SECRET_KEY='$SECRET_KEY'
EMAIL_BACKEND = 'django.core.mail.backends.console.EmailBackend'

# Параметры подключения к OpenStack
OPENSTACK_HOST = "controller"
OPENSTACK_KEYSTONE_URL = "http://controller:5000/v3"

OPENSTACK_API_VERSIONS = {
  "identity": 3,
  "image": 2,
  "volume": 3,
}

OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = True
OPENSTACK_KEYSTONE_DEFAULT_DOMAIN = "Default"
OPENSTACK_KEYSTONE_DEFAULT_ROLE = "user"
POLICY_FILES_PATH = "/etc/openstack-dashboard"

# TODO: Нужно перенести в раздел с шифрованием.
# CSRF_COOKIE_SECURE = True
# SESSION_COOKIE_SECURE = True

# Параметры журналирования
DEBUG = False
LOGGING = {
    'version': 1,
    'formatters': {
        'console': {
            'format': '%(levelname)s %(name)s %(message)s'
        },
        'operation': {
            # The format of "%(message)s" is defined by
            # OPERATION_LOG_OPTIONS['format']
            'format': '%(message)s'
        },
    },
    'handlers': {
        'null': {
            'level': 'DEBUG',
            'class': 'logging.NullHandler',
        },
        'console': {
            # Set the level to "DEBUG" for verbose output logging.
            'level': 'DEBUG' if DEBUG else 'INFO',
            'class': 'logging.StreamHandler',
            'formatter': 'console',
        },
        'operation': {
            'level': 'INFO',
            'class': 'logging.StreamHandler',
            'formatter': 'operation',
        },
    },
    'loggers': {
        'horizon': {
            'handlers': ['console'],
            'level': 'DEBUG',
            'propagate': False,
        },
```

```
        'horizon.operation_log': {
            'handlers': ['operation'],
            'level': 'INFO',
            'propagate': False,
        },
        'openstack_dashboard': {
            'handlers': ['console'],
            'level': 'DEBUG',
            'propagate': False,
        },
        'novaclient': {
            'handlers': ['console'],
            'level': 'DEBUG',
            'propagate': False,
        },
        'cinderclient': {
            'handlers': ['console'],
            'level': 'DEBUG',
            'propagate': False,
        },
        'keystoneauth': {
            'handlers': ['console'],
            'level': 'DEBUG',
            'propagate': False,
        },
        'keystoneclient': {
            'handlers': ['console'],
            'level': 'DEBUG',
            'propagate': False,
        },
        'glanceclient': {
            'handlers': ['console'],
            'level': 'DEBUG',
            'propagate': False,
        },
        'neutronclient': {
            'handlers': ['console'],
            'level': 'DEBUG',
            'propagate': False,
        },
        'swiftclient': {
            'handlers': ['console'],
            'level': 'DEBUG',
            'propagate': False,
        },
        'oslo_policy': {
            'handlers': ['console'],
            'level': 'DEBUG',
            'propagate': False,
        },
        'openstack_auth': {
            'handlers': ['console'],
            'level': 'DEBUG',
            'propagate': False,
        },
        'django': {
            'handlers': ['console'],
            'level': 'DEBUG',
            'propagate': False,
        },
        'django.db.backends': {
            'handlers': ['null'],
            'propagate': False,
        },
        'requests': {
            'handlers': ['null'],
            'propagate': False,
        },
```

```
        'urllib3': {
            'handlers': ['null'],
            'propagate': False,
        },
        'chardet.charsetprober': {
            'handlers': ['null'],
            'propagate': False,
        },
        'iso8601': {
            'handlers': ['null'],
            'propagate': False,
        },
        'scss': {
            'handlers': ['null'],
            'propagate': False,
        },
    },
}

# Список стандартных групп безопасности в Horizon
SECURITY_GROUP_RULES = {
    'all_tcp': {
        'name': _('All TCP'),
        'ip_protocol': 'tcp',
        'from_port': '1',
        'to_port': '65535',
    },
    'all_udp': {
        'name': _('All UDP'),
        'ip_protocol': 'udp',
        'from_port': '1',
        'to_port': '65535',
    },
    'all_icmp': {
        'name': _('All ICMP'),
        'ip_protocol': 'icmp',
        'from_port': '-1',
        'to_port': '-1',
    },
    'ssh': {
        'name': 'SSH',
        'ip_protocol': 'tcp',
        'from_port': '22',
        'to_port': '22',
    },
    'smtp': {
        'name': 'SMTP',
        'ip_protocol': 'tcp',
        'from_port': '25',
        'to_port': '25',
    },
    'dns': {
        'name': 'DNS',
        'ip_protocol': 'tcp',
        'from_port': '53',
        'to_port': '53',
    },
    'http': {
        'name': 'HTTP',
        'ip_protocol': 'tcp',
        'from_port': '80',
        'to_port': '80',
    },
    'pop3': {
        'name': 'POP3',
        'ip_protocol': 'tcp',
        'from_port': '110',
        'to_port': '110',
```

```
    },
    'imap': {
        'name': 'IMAP',
        'ip_protocol': 'tcp',
        'from_port': '143',
        'to_port': '143',
    },
    'ldap': {
        'name': 'LDAP',
        'ip_protocol': 'tcp',
        'from_port': '389',
        'to_port': '389',
    },
    'https': {
        'name': 'HTTPS',
        'ip_protocol': 'tcp',
        'from_port': '443',
        'to_port': '443',
    },
    'smtps': {
        'name': 'SMTPS',
        'ip_protocol': 'tcp',
        'from_port': '465',
        'to_port': '465',
    },
    'imaps': {
        'name': 'IMAPS',
        'ip_protocol': 'tcp',
        'from_port': '993',
        'to_port': '993',
    },
    'pop3s': {
        'name': 'POP3S',
        'ip_protocol': 'tcp',
        'from_port': '995',
        'to_port': '995',
    },
    'ms_sql': {
        'name': 'MS SQL',
        'ip_protocol': 'tcp',
        'from_port': '1433',
        'to_port': '1433',
    },
    'mysql': {
        'name': 'MYSQL',
        'ip_protocol': 'tcp',
        'from_port': '3306',
        'to_port': '3306',
    },
    'rdp': {
        'name': 'RDP',
        'ip_protocol': 'tcp',
        'from_port': '3389',
        'to_port': '3389',
    },
}
```

4. Перезапустите веб-сервер Apache:

```
systemctl restart httpd
```

5. Веб-панель по умолчанию будет доступна по пути, откройте его через веб-браузер:

```
http://controller/dashboard
```

> При указании доменного имени убедитесь, что эта имя резолвится в указанном в клиенте DNS. Иначе воспользуйтесь подключением через IP-адрес.

Включение шифрования веб-панели

1. Укажите конфигурацию для веб-сервера Apache по пути `/etc/httpd/conf.d/openstack-horizon.conf`.
   Конфигурация Apache:

```
ServerName controller
ServerRoot "/etc/httpd"
Include conf.modules.d/*.conf
User apache
Group apache
LogLevel warn
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\""
 combined
ErrorLog /dev/stderr
CustomLog /dev/stdout combined
TypesConfig /etc/mime.types
AddDefaultCharset UTF-8
EnableSendfile on

<Directory />
  AllowOverride none
  Require all denied
</Directory>

SSLProtocol all -SSLv3 -TLSv1 -TLSv1.1
SSLCipherSuite ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-
ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-
POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-
GCM-SHA384
SSLHonorCipherOrder off
SSLSessionTickets off
AddOutputFilterByType DEFLATE text/plain text/html text/xml text/css text/
javascript application/xml application/javascript application/json image/
svg+xml

Listen 80
Listen 443

<VirtualHost *:80>
  Redirect permanent / https://controller/
</VirtualHost>

<VirtualHost *:443>
  WSGIDaemonProcess dashboard
  WSGIProcessGroup dashboard
  WSGIScriptAlias / /usr/share/openstack-dashboard/openstack_dashboard/wsgi.py
  WSGIApplicationGroup %{GLOBAL}
  Alias /static /usr/share/openstack-dashboard/static
  SSLEngine on
  SSLCertificateFile certs/cert.pem
  SSLCertificateKeyFile certs/privkey.pem
  Protocols h2 http/1.1
  <Directory /usr/share/openstack-dashboard/openstack_dashboard>
    Options All
    AllowOverride All
    Require all granted
  </Directory>
  <Directory /usr/share/openstack-dashboard/static>
    Options All
    AllowOverride All
    Require all granted
  </Directory>
</VirtualHost>
```

   Информация о конфигурации веб-сервера описана в этой части (см. стр. 45) Руководства.

2. После включения этой конфигурации перезапустите веб-сервер:

```
systemctl restart httpd
```

Перезапуск юнита httpd может длиться некоторое время из-за работы с статическими файлами веб-панели.

3. Убедитесь, что шифрованные соединения доступны по порту 443, например, командой curl:

```
curl -v https://controller:443
```

Если сертификат самоподписанный, то укажите:

- параметр -k, в этом случае верификация сертификата будет выключена;
- или параметр --capath до файла CA вашего центра сертификации.

Функционирование продуктивных сред с самоподписанными сертификатами официально не поддерживается.

4. С помощью веб-браузера загрузите страницу веб-панели.

Таблица конфигурации

Horizon

Путь до конфигурации: `/etc/openstack-dashboard/local_settings`

Легенда таблицы доступна на этой странице[175].

Имя параметра	Описание	Примечания
import **from**	Встроенные механизмы загрузки модулей Python.	Этот файл фактически является кодом на языке Python. Здесь загружаются модули, необходимые для корректной обработки файла конфигурации компонентом Horizon.
WEBROOT	Корневой путь до объектов веб-панели.	
DEBUG	Включение режима отладки для Horizon.	Хранение журналов Horizon зависят от настроек журналирования веб-сервера. Является параметром фреймворка Django.
ALLOWED_HOSTS	Разрешенные домены для подключения к Horizon.	В этот список можно добавить имена доменных имён, которые смогут получить доступ к веб-панели. Можно добавить несколько имён, оформленных как список в Python. Аргумент '*' (вместе с кавычками) позволит подключиться со всех доменных имён.
LOCAL_PATH	Путь до локальных ресурсов модуля.	Является параметром фреймворка Django.

175 https://conf.tionix.ru/pages/viewpage.action?pageId=205881354#id-Условныеобозначения-table-format-desc

Имя параметра	Описание	Примечания
SECRET_KEY	Уникальная комбинация инсталляции компонента.	Используется для криптографической подписи[176], параметр фреймворка Django.
EMAIL_BACKEND	Включение бэкенда для отправки писем электронной почты.	По умолчанию письма, генерируемые веб-панелью, отправляются в стандартной ввод терминала.
OPENSTACK_HOST	Переменная для адреса сервиса Keystone.	Эта переменная используется другими переменными в конфигурации, например, **OPENSTACK_KEYSTONE_URL.** Сама переменная при подключении к сервису Keystone не используется. Параметр **OPENSTACK_HOST** нужно использовать при наличии только одного региона в облачной платформе. Если регионов больше, то воспользуйтесь параметром AVAILABLE_REGIONS[177].
OPENSTACK_KEYSTONE_URL	Полный URL для подключения к Keystone.	
OPENSTACK_API_VERSIONS	Указание версий различных API сервисов OpenStack.	
OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT	Включение поддержки нескольких доменов, созданных в сервисе Keystone.	Необходимо для домен-специфичных драйверов[178], настраиваемых в сервисе Keystone.
OPENSTACK_KEYSTONE_DEFAULT_DOMAIN	Используемый домен Keystone по умолчанию.	
OPENSTACK_KEYSTONE_DEFAULT_ROLE	Используемая для пользователей роль по умолчанию.	Используется при добавлении пользователя в проект через веб-панель без указания роли.
CSRF_COOKIE_SECURE	Передача данных CSRF только через шифрованный канал. Обязателен, если включен HTTPS.	Ограничивает передачу CSRF[179] cookie только через протокол HTTPS. Является параметром фреймворка[180] Django.

176 https://docs.djangoproject.com/en/3.2/topics/signing/
177https://github.com/openstack/horizon/blob/5e4ca1a9fdec04db08552e9e93fe372b8b8b45ae/doc/source/configuration/settings.rst#available-regions
178 https://conf.tionix.ru/pages/viewpage.action?pageId=164102237#id-Использованиесвязанныхсдоменомсистемхраненияданныхпользователей-ks-domain-specific-drivers
179 https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B6%D1%81%D0%B0%D0%B9%D1%82%D0%BE%D0%B2%D0%B0%D1%8F_%D0%BF%D0%BE%D0%B4%D0%B4%D0%B5%D0%BB%D0%BA%D0%B0_%D0%B7%D0%B0%D0%BF%D1%80%D0%BE%D1%81%D0%B0
180 https://docs.djangoproject.com/en/3.2/ref/csrf/

Имя параметра	Описание	Примечания
SESSION_COOKIE_SECURE	Передача данных сессий только через шифрованный канал. Обязателен, если включен HTTPS.	Ограничивает передачу session cookie[181] только через протокол HTTPS. Является параметром фреймворка Django.
POLICY_FILES_PATH	Путь до файлов ролевой модели для Horizon.	У веб-панели имеются свои настройки для различных ролей (файлы policy.yml) и они могут не совпадать с тем, что предоставляет Keystone. В основном они используются для более тонкого управления функциями в веб-панели, доступных обычным пользователям.
SESSION_ENGINE	Метод хранения данных сессии на стороне сервера.	Предлагаемая настройка хранит данные сессии в системе кэширования. ⚠ При очистке данных в системе кэширования все установленные сессии будут завершены.
CACHES	Данные системы кэширования.	
TIME_ZONE	Часовой пояс по умолчанию.	Влияет на показываемое время для объектов в веб-панели. Пользователь может поменять часовой пояс в настройках аккаунта.
LOGGING	Расширенные параметры логгирования частей веб-панели.	Является параметром фреймворка Django.
SECURITY_GROUP_RULES	Предлагаемые для добавления значения правил безопасности.	Влияет на список доступных условий при добавлении правила в группе безопасности.

Установка модуля Dashboard

1. Установите пакета модуля:

   ```
   dnf -y install python3-tionix_dashboard
   ```

   ⓘ Стандартные пути конфигурации:

   - `/etc/tionix/dashboard.yaml` – основной файл конфигурации.

2. В файл `/etc/openstack-dashboard/local_settings` добавьте загрузку модуля настроек Dashboard:

   ```
   try:
       from tionix_dashboard.settings import *
   except ImportError:
       pass
   ```

3. В референсной архитектуре используется следующая конфигурация в формате YAML[182]:

181 https://docs.djangoproject.com/en/3.2/topics/http/sessions/#using-cookie-based-sessions
182 https://en.wikipedia.org/wiki/YAML

Это только часть конфигурации, используемый только для компонента TIONIX Dashboard. Глобальные настройки хранятся в файле `/etc/tionix/tionix.yaml`.

```
DB:
    ENGINE: 'django.db.backends.mysql'
    NAME: 'tionix_dash'

NEUTRON_VERSION: 2

KEYSTONE:
    network_service_name: 'network'
    identity_service_name: 'identity'

MIN_RESERVE_VM: null

VOLUME_ATTACH_MAX_RETRIES: 40

POINT_METER_API_URL: http://tnx-pm.k8s_domain_name:9367/
```

Описание параметров:

Легенда таблицы доступна на этой странице[183].

Имя параметра	Описание	Примечания
DB	Глобальные параметры базы данных компонента.	
→ **ENGINE**	Движок к базе данных.	django.db.backends.mysql является универсальным драйвером, который подходит и для СУБД MariaDB.
→ **NAME**	Имя базы данных компонента.	
KEYSTONE	Параметры подключения к сервису Keystone.	
→ **network_service_name**	Название точки входа для сервиса Neutron.	
→ **identity_service_name**	Название точки входа для сервиса Keystone.	
MIN_RESERVE_VM	Минимальное количество резервируемых ВМ.	Настройка резервирования ВМ выполняется в сервисе NodeControl.
VOLUME_ATTACH_MAX_RETRIES	Максимальное количество попыток подключения диска к VDI-машине.	
POINT_METER_API_URL	Адрес к сервису Pointmeter (см. стр. 164).	

4. Запустите команду сбора статичных ресурсов для обновления данных по графическим элементам:

```
python3 /usr/share/openstack-dashboard/manage.py collectstatic
```

5. Войдите в окружение СУБД MariaDB:

```
mysql -u root -p
```

6. Создайте базу данных tionix_dash:

183 https://conf.tionix.ru/pages/viewpage.action?pageId=205881354#id-Условныеобозначения-table-format-desc

```
CREATE DATABASE tionix_dash;
```

7. Предоставьте права доступа к этой базе данных пользователю tionix:

```
GRANT ALL PRIVILEGES ON tionix_dash.* TO 'tionix'@'localhost' IDENTIFIED BY
'TIONIX_PASS';
GRANT ALL PRIVILEGES ON tionix_dash.* TO 'tionix'@'%' IDENTIFIED BY 'TIONIX_PASS';
```

8. Выйдите из сессии СУБД:

```
exit;
```

9. Создайте структуру базы данных:

```
openstack tnx db migrate -n tnx_dashboard
```

10. Перезапустите все сервисы TIONIX:

```
systemctl restart tionix-*
```

11. Перезапустите сервисы веб-сервера Apache и системы кэширования memcached:

```
systemctl restart httpd
systemctl restart memcached
```

12. Войдите в веб-панель, пройдите аутентификацию и проверьте наличие пункта меню "ТИОНИКС" в левом боковом меню.

На этом первичная настройка компонента TIONIX Dashboard завершена.

Установка темы TIONIX

Установка темы не является обязательным шагом настройки TIONIX Dashboard. Основной задачей темы является брендинг веб-панели и использование фирменного цвета в интерфейсе, функциональных изменений нет.

1. Установите пакет с темой:

```
dnf -y install python3-tionix_dashboard_theme
```

2. В файл `/etc/openstack-dashboard/local_settings` добавьте загрузку темы:

```
try:
    from tionix_dashboard_theme import *
except ImportError:
    pass
```

3. Запустите команду сбора статичных ресурсов для обновления данных по графическим элементам:

```
python3 /usr/share/openstack-dashboard/manage.py collectstatic
```

4. Перезапустите сервисы веб-сервера Apache и системы кэширования memcached:

```
systemctl restart httpd
systemctl restart memcached
```

5. Войдите в веб-панель и убедитесь в смене цветовой схемы интерфейса приглашения на вход и изменения логотипа OpenStack на логотип TIONIX.

Pointmeter

Информация о сервисе PointMeter

TIONIX Pointmeter - это сервис, который генерирует отчетную информацию по потреблению ресурсов облачной платформы в рамках конкретного проекта и передаёт его поставщику услуг или заказчику через средства электронной почты. Данные Pointmeter шифруются с использованием протокола GPG и применением ассиметричного шифрования (по умолчанию, на базе алгоритмов AES).

Компоненты сервиса Pointmeter

Pointmeter реализован в виде единого сервиса, запускаемый через systemd:

- **tionix-point-meter-api** - реализует и предоставляет сетевой доступ к Pointmeter API.

Краткая информация о работе сервиса

Pointmeter вкратце работает следующим образом:

1. Заказчику облачных услуг, если это необходимо, предоставляется открытый ключ GPG, с помощью которого будут зашифрованы данные отчета. Закрытая часть ключа остаётся у поставщика услуг.
2. Сервис обращается в сервис OpenStack Nova для получения отчета о потреблении ресурсов в формате CSV.
3. Это файл CSV шифруется открытым ключом.
4. Полученный зашифрованный файл отправляется по средствам электронной почты поставщику услуг.
5. Поставщик услуг с помощью закрытого ключа расшифровывает полученные данные и получает информацию о потреблении. Без закрытого ключа расшифровка данных невозможна.

Установка сервиса TIONIX Pointmeter

Настройка окружения

Перед самой установкой сервиса нужно предварительно настроить некоторые компоненты инфраструктуры.

См. также: Раздел с предварительной настройкой окружения для модулей TIONIX (см. стр. 116).

Установка модуля

1. Установите пакет модуля:

   ```
   dnf -y install python3-tionix_point_meter
   ```

   Информация

   Пути конфигурации:

   - `/etc/tionix` — основной каталог конфигурации;
   - `/etc/tionix/tionix.yaml` — основной файл конфигурации;
   - `/etc/tionix/point_meter.yaml` - файл конфигурации сервиса PointMeter.

2. Выполните первичную инициализацию модуля:

   ```
   openstack tnx configure -n tnx_point_meter
   ```

3. Используйте эту конфигурацию по пути `/etc/tionix/point_meter.yaml`:

   ```
   DEBUG: True
   MAIL_SERVER: smtp.yandex.ru
   MAIL_PORT: 587
   MAIL_USE_TLS: True
   MAIL_USE_SSL: False
   MAIL_USERNAME: test@yandex.ru
   MAIL_PASSWORD: '*******'
   MAIL_ASCII_ATTACHMENTS: True
   TIONIX_MAIL: 'points@tionix.ru'
   CRON_SCHEDULE: '0 3 1 */1 *'
   TIME_ZONE: 'Europe/Moscow'
   ```

4. В файл конфигурации TIONIX Dashboard по пути `/etc/tionix/dashboard.yaml` укажите адрес сервиса PointMeter добавлением следующей конфигурации:

   ```
   POINT_METER_API_URL: http://127.0.0.1:9367/
   ```

Финализация установки

1. Запустите сервис PointMeter:

```
systemctl enable tionix-point-meter-api.service
systemctl start tionix-point-meter-api.service
```

2. Перезапустите все службы TIONIX для регистрации сервиса PointMeter:

```
systemctl restart tionix-*
```

Описание файла конфигурации Pointmeter

В описании процесса по установке сервиса Pointmeter предложено содержимое конфигурации, которое предложено сохранить в управляющем узле по пути `/etc/tionix/pointmeter.yaml`. Это страница содержит подробное описание этой конфигурации.

При изменении конфигурации необходимо перезапустить все компоненты сервиса Pointmeter:

```
systemctl restart tionix-pointmeter-*
```

Таблица конфигурации

Немного о формате yaml

Файл формата yaml[184] критичен к отступам текста[185] при описании параметров. Стрелкой в имени параметра указан отступ в 4 пробела от начала строки.

По умолчанию параметры логирования Pointmeter берёт из основного файла конфигурации `/etc/tionix/tionix.yaml`.

Легенда таблицы доступна на этой странице[186].

Имя параметра	Описание	Примечания
DEBUG	Включение режима подобного журналирования.	
MAIL_SERVER	Адрес SMTP-сервера, с помощью которого будет отправлено письмо.	Для корректной отправки отчета необходимо, чтобы на почтовом сервере был включен доступ по протоколу IMAP или POP3.
MAIL_PORT	Порт SMTP-сервера.	
MAIL_USE_TLS	Включение шифрования с помощью протокола TLS.	
MAIL_USE_SSL	Включение шифрования с помощью протокола SSL.	
MAIL_USERNAME	Имя пользователя для аутентификации в SMTP-сервере.	В основном, необходимо для сбора данных мониторинга и сообщений об ошибках в удаленный сервис Sentry.
MAIL_DEFAULT_SENDER	Имя или адрес отправителя, который будет указан в поле Sender в письме с отчётом.	

184 https://yaml.org/
185 https://yaml.org/spec/1.2.2/#61-indentation-spaces
186 https://conf.tionix.ru/pages/viewpage.action?pageId=205881354#id-Условныеобозначения-table-format-desc

Имя параметра	Описание	Примечания
MAIL_PASSWORD	Пароль пользователя для аутентификации в SMTP-сервере.	
MAIL_ASCII_ATTACHMENTS	Конвертация путей файлов, которые будут добавлены как вложение в письмо, из UTF-8 в кодировку ASCII.	Используйте этот параметр только в том случае, если в путях встречаются только латинские символы из кодировки ASCII.
TIONIX_MAIL	Адрес электронной почты, куда будут отправлены отчёты сервиса.	Не меняйте этот параметр без крайней необходимости.
TIONIX_CLIENTS	Адрес или адреса электронной почты, которые будут включены в поле CC отправляемого письма.	Можно указать несколько адресов через символ ;.
CRON_SCHEDULE	Правило cron для периодической отправки отчётов.	
TIME_ZONE	Информация о часовом поясе, которая будет влиять на показываемое время отправки письма.	

Compute Agent

Информация об агенте TIONIX

Agent - это компонент TIONIX, который может управлять узлами облачной платформы с использованием RPC-запросов. Обычно агент используется на вычислительных узлах, однако имеется возможность настройки коммутаторов компании Mellanox.

В качестве RPC-транспорта используется сервис RabbitMQ.

Описание функций

Agent предоставляет следующие возможности:

- включает и выключает режим динамического конфигурирования компонентов вычислительных узлов (DCC) и использованием сервиса Consul;
- управляет состоянием сервиса SNMP на вычислительных и иных узлах;
- управляет состоянием сервиса SSH;
- предоставляет возможность "горячей" замены выделенных ресурсов для виртуальных машин;
- управляет некоторыми функциями коммутаторов Mellanox, ОС который основана на Linux;
- предоставляет функции настройки проксирования протокола SPICE для VDI-функций.

Описание типов установки

Каждая описанная функция в Agent включается методом указания типа конфигурирования:

- **compute** - настраивает вычислительные узлы, которые используют изолированную сеть;
- **control** - предоставляет общую возможность настройки вычислительных узлов;
- **selfdiscovery** - включает режим саморегистрации вычислительного узла;
- **mlx** - предназначен для настройки некоторых коммутаторов Mellanox;
- **consul** - необходимо для изменения параметров вычислительных узлов на лету (DCC);
- **spice_proxy** - настраивает проксирование SPICE путем генерирования конфигурации для балансировщика нагрузки HAProxy.

Эти типы указываются во время конфигурирования сервиса.

Установка сервиса Agent

Установка сервиса

Установка должна выполняться на вычислительных узлах.

1. Установите пакет компонента:

```
dnf -y install python3-tionix_agent
```

Стандартные пути конфигурации:

- `/etc/tionix` — основной каталог конфигурации.
- `/etc/tionix/tionix.yaml` — основной файл конфигурации TIONIX.
- `/etc/tionix/agent.conf` - файл конфигурации для сервиса Agent.

2. Запустите команду конфигурирования агента, где вместо type следует указать тип настройки (см. стр. 167):

```
openstack tnx agent configure --type <type>
```

Можно указать несколько типов через запятую.

3. Включите следующую конфигурацию агента по пути `/etc/tionix/agent.yaml` (описание (см. стр. 168)):

```
[DEFAULT]
transport_url = amqp://user:RABBIT_PASS@controller:5672/vhost
durable = false
agent_type = control, spice_proxy
```

Список типов агента должен совпадать со списком, указанный во время конфигурирования компонента.

Финализация установки

1. После настройки агента необходимо запустить сервис с агентом:

```
systemctl enable tionix-agent.service
systemctl start tionix-agent.service
```

Для полноценного функционирования агента требуется явно указать параметр *ENABLE_AGENT = True* в конфигурационном файле сервиса NodeControl.

Описание файла конфигурации Agent

В описании процесса по установке Agent предложено содержимое конфигурации, которое предложено сохранить в управляющем узле по пути `/etc/tionix/agent.conf`. Это страница содержит подробное описание этой конфигурации.

При изменении конфигурации необходимо перезапустить все компоненты компонента Agent:

```
systemctl restart tionix-agent-*
```

Таблица конфигурации

Легенда таблицы доступна на этой странице[187].

Имя параметра	Описание	Примечания
[DEFAULT]	Глобальные параметры компонента.	
transport_url	Адреса сервисов RabbitMQ для RPC-функций.	Можно указать только один сервер RabbitMQ. При наличии в имени пользователя и пароле символов, которые для формата INI являются специальными (например, знак комментирования "#"), то можно использовать его URL-код. • Пример: `passw#ord` → `passw%23ord`
durable	Параметр для подключения к очереди сообщений AMQP. Возможные значения: • `true` - очередь сообщений будет сохранять свое состояние и восстанавливаться после перезапуска брокера; • `false` - очередь сообщений будет удаляться после перезапуска брокера.	
agent_type	Типы агентов, которые необходимо активировать. Возможные значения: • `compute` - для изолированного вычислительного узла; • `control`, `selfdiscovery` - для любого вычислительного узла; • `mlx`- для коммутаторов; • `consul` - для службы синхронизации конфигурационных файлов с хранилищем Consul; • `spice_proxy` - для запуска сервиса, отвечающего за корректную работу виртуальных машин через SPICE-сессии.	
[consul]	Параметры подключения к сервису Consul. Обязателен, если consul в agent_type.	
host	Доменное имя или IP-адрес сервиса Consul. Обязателен, если consul в agent_type.	
port	Порт сервиса Consul. Обязателен, если consul в agent_type.	
token	Токен аутентификации к сервису Consul. Обязателен, если consul в agent_type.	

Дебаггинг

Для подробного журналирования событий можно указать следующие параметры:

187 https://conf.tionix.ru/pages/viewpage.action?pageId=205881354#id-Условныеобозначения-table-format-desc

Таблица конфигурации

Имя параметра	Описание	Примечания
[DEFAULT]	Глобальные параметры компонента.	
debug	Включение подробного журналирования.	
log_file	Путь до файла журнала. Обязателен, если выставлен debug.	Journald на данный момент не поддерживается.

Drivers

Информация о драйверах TIONIX

Драйверы TIONIX - это специализированные модули для взаимодействия с некоторыми типами оборудования. На данный момент включает в себя следующие компоненты:

- Драйвер для работы с общим LVM, предназначенный для сервиса Cinder. Его основной задачей является унификация использования дискового пространства для всех вендоров систем хранения.

Установка и настройка драйвера Cinder

Установка компонента

> Драйвер Cinder в комплекте Drivers должен быть установлен на узлах, где запущен сервис cinder-volume (см. стр. 100).

1. Установите пакет Drivers:

```
dnf -y install python3-tionix_drivers
```

2. Убедитесь, что tionix-agent запущен на узлах с сервисами *cinder-volume*, *cinder-backup* и *nova-compute*, и в нём настроен тип "*control*".
3. В файле `/etc/cinder/cinder.conf` включите драйвер общего LVM с использованием системы блокировок sanlock как бэкенд:

```
[DEFAULT]
rpc_response_timeout = 600
volume_manager=tionix_client.block_storage.manager.TnxVolumeManager
enabled_backends = sanlock

[sanlock-backend]
volume_driver = tionix_drivers.cinder.volume.drivers.sharedlvm.SharedLVMDriver
agent_transport_url = amqp://tionix:password@sanlock.stand.loc/tionix
volume_group = vol
lvm_type = default
lvm_mirrors = 0
volume_backend_name = sanlock
agent_response_timeout = 60

[nova]
token_auth_url = http://sanlock.stand.loc:5000
auth_section = keystone_authtoken
auth_type = password
```

> Укажите *lvm_type = thin*, если необходима поддержка тонких томов.

4. Перезапустите сервис cinder-volume:

```
systemctl restart openstack-cinder-volume
```

Параметры настройки бэкенда с общим LVM

Для настройки бэкенда с общим драйвером применяются дополнительные параметры. Основные параметры сервиса Cinder описаны здесь (см. стр. 102).

Легенда таблицы доступна на этой странице[188].

Имя параметра	Описание	Примечания
[DEFAULT]	Глобальные параметры сервиса Cinder.	
rpc_response_timeout	Время ожидания ответа на запрос RPC.	
volume_manager	Указание модуля управления дисками.	
enabled_backends	Включенные бэкенды (системы хранения), которые описаны в конфигурации Cinder.	Можно указать несколько бэкендов через запятую.
[sanlock-backend]	Раздел с настройками для бэкенда с именем внутри квадратных скобок.	Эти имена используются для указания включенных бэкендов.
volume_driver	Имя драйвера управления блочными устройствами.	
agent_transport_url	Адрес до сервиса RabbitMQ.	
volume_group	Имя группы дисков (VG) в LVM, в котором будут создаваться LV-устройства.	
lvm_type	Тип дисков в VG-группе.	Может иметь два значения: • **default** – блоки будет выданы на весь размер диска; • **thin** – блоки будут выдаваться по мере необходимости для записи ("тонкие диски"). В этом случае возможно использование переподписки по дискам.
lvm_mirrors	Создание зеркальных LV в VG-группе.	При включении этого параметра в VG-группе будут созданы копии LV.
volume_backend_name	Имя бэкенда для платформы.	Это имя используется при регистрации бэкенда в группе дисков Cinder.
agent_response_timeout	Время ожидания ответа от агента.	
[nova]	Параметры взаимодействия с сервисом Nova	
token_auth_url	Данные сервиса Keystone для получения данных о сервисе Nova и генерации токена авторизации.	
auth_section	Имя сервиса аутентификации.	
auth_type	Тип аутентификации.	

188 https://conf.tionix.ru/pages/viewpage.action?pageId=205881354#id-Условныеобозначения-table-format-desc

Описание основного файла конфигурации модулей TIONIX

Все модули TIONIX используют путь до файла `/etc/tionix/tionix.yaml` как основной файл конфигурации. При установке любого из модулей устанавливаются оба вида конфигурационных файлов: модульный и общий, с расширением `.yaml.example` в директорию `/etc/tionix/`. С описанием формата можно ознакомиться в соответствующем разделе официальной документации[189]. Настройки отдельных функций модулей хранятся в отдельных файлах, они будут описаны отдельно в разделах установки этих модулей.

Файлы конфигурации содержат минимальный набор секций, необходимых для работы модулей. После внесения изменений в файл `yaml.example`, в целях сохранения образца настроек следует сохранить их под другим именем. Файлы могут содержать одинаковые секции и параметры, при разных параметрах используются настройки модульного файла конфигурации. В случае отсутствия файлов конфигурации будут использоваться параметры по умолчанию из файлов `yaml.example`.

Файл конфигурации

По умолчанию файл по пути `/etc/tionix/tionix.yaml` выглядит так:

189 http://yaml.org

```
CINDER_VERSION: '3.50'

KEYSTONE:
    auth_url: 'https://keystone.k8s_domain_name:5000'
    auth_version: '3'
    auth_user: 'admin'
    auth_password: 'GJOPlnp7WHO3NB'
    auth_tenant: 'admin'
    compute_service_name: 'compute'
    volume_service_name: 'volumev3'
    gnocchi_service_name: 'metric'
    journal_service_type: 'tnx-journal'
    nc_service_type: 'tnx-nc'
    monitor_service_type: 'tnx-monitor'
    vdi_service_type: 'tnx-vdi'
    scheduler_service_type: 'tnx-scheduler'
    user_domain_name: 'default'
    project_domain_name: 'default'
    memcached_servers: 'memcached.default.svc.k8s.k8s_domain_name:11211'
    service_user: 'tionix'
    service_password: '0opvfFoURXj7c1'
    service_project: 'service'
    service_user_domain_name: 'default'
    service_project_domain_name: 'default'

DB:
    ENGINE: 'mysql+pymysql'
    USER: 'tionix'
    PASSWORD: 'RErSymQdykAt7X'
    HOST: 'mysql.default.svc.k8s.k8s_domain_name'
    PORT: '3306'
    NAME: 'tionix'
    MAX_POOL_SIZE: 5
    MAX_OVERFLOW: 30
    POOL_RECYCLE: 3600
    POOL_TIMEOUT: 30

RABBIT_QUEUES:
    vhost: 'tionix'
    broker_type: 'amqp'
    host: 'rabbitmq.default.svc.k8s.k8s_domain_name'
    port: '5672'
    username: 'tionix'
    password: 'eBlwpYNgph8meT'
    durable: False

LOG_LEVEL: 'INFO'
NOVA_RABBIT_VHOST: '/'
KEYSTONE_RABBIT_VHOST: '/'
TRACEBACK_ENABLED: True

JOURNAL_API_LISTEN: '0.0.0.0'
JOURNAL_API_LISTEN_PORT: 9360
JOURNAL_API_LOGFILE: '/dev/stdout'
JOURNAL_LISTENER_LOGFILE: '/dev/stdout'
JOURNAL_NOVA_LISTENER_LOGFILE: '/dev/stdout'
JOURNAL_KEYSTONE_LISTENER_LOGFILE: '/dev/stdout'

SENTRY:
    ENABLED: True
    DSN: http://1d91324a511a54741a396f4fadca925ec:
1a35b43635bf4dce9d0d49ae08d8bf99@my.sentry.loc/2
```

Таблица конфигурации

Немного о формате yaml

Файл формата yaml[190] критичен к отступам текста[191] при описании параметров. Стрелкой в имени параметра указан отступ в 4 пробела от начала строки.

Важно

При изменении файла конфигурации необходимо перезапустить веб-сервер и службы модулей:

```
systemctl restart httpd
systemctl restart tionix-*
```

Легенда таблицы доступна на этой странице[192].

Имя параметра	Описание
CINDER_VERSION	Версия CinderClient для подключения к службе Cinder.
JOURNAL_API_LISTEN	IP-адрес, на котором будет запущена служба Journal API.
JOURNAL_API_LISTEN_PORT	Порт, на котором будет запущена служба Journal API.
JOURNAL_API_LOGFILE	Путь до лог файла службы Journal API.
JOURNAL_KEYSTONE_LISTENER_LOG FILE	Путь до лог файла службы Keystone listener.
JOURNAL_LISTENER_LOGFILE	Путь до лог файла службы Journal listener.
JOURNAL_NOVA_LISTENER_LOGFILE	Путь до лог файла службы Nova listener.
NEUTRON_VERSION	Поддерживаемая версия Neutron API.
SENTRY	Настройки системы Sentry.
RABBIT_QUEUES	Настройки сервиса выполнения асинхронных задач.
LOG_LEVEL	Уровень журналирования модулей. Доступные значения: • `DEBUG`; • `INFO` (по умолчанию); • `WARNING`; • `ERROR`; • `CRITICAL`. Значения являются регистронезависимыми. По умолчанию INFO.
NOVA_RABBIT_VHOST	Виртуальный хост RabbitMQ, используемый сервисом Nova.
KEYSTONE_RABBIT_VHOST	Виртуальный хост RabbitMQ используемый сервисом Keystone.
TRACEBACK_ENABLED	Параметр для вывода трассировки ошибки интерпретатором в журнал сервисов. Возможные значения: • `True`; • `False` (по умолчанию). Параметры являются регистронезависимыми.

190 https://yaml.org/
191 https://yaml.org/spec/1.2.2/#61-indentation-spaces
192 https://conf.tionix.ru/pages/viewpage.action?pageId=205881354#id-Условныеобозначения-table-format-desc

Имя параметра	Описание
KEYSTONE → auth_url	Адрес внутренней точки доступа к сервису Keystone.
KEYSTONE → auth_version	Версия протокола сервиса Keystone.
KEYSTONE → auth_user	Имя пользователя в Keystone для сервисов Tionix.
KEYSTONE → auth_password	Пароль пользователя в Keystone для сервисов Tionix.
KEYSTONE → auth_tenant	Название проекта Keystone, используемый сервисами Tionix.
KEYSTONE → compute_service_name	Имя сервиса в Keystone, используемый Nova (по умолчанию compute).
KEYSTONE → volume_service_name	Тип сервиса в Keysone, используемый Cinder (по умолчанию volume).
KEYSTONE → ceilometer_service_name	Тип сервиса Ceilometer, используемый Ceilometer (по умолчанию metering).
KEYSTONE → user_domain_name	Название домена пользователя.
KEYSTONE → project_domain_name	Название домена проекта.
KEYSTONE → service_user	Имя пользователя, добавленный в проект service.
KEYSTONE → service_password	Пароль пользователя, добавленный в проект service.
KEYSTONE → service_project	Название проекта, добавленный в проект service.
KEYSTONE → service_user_domain_name	Название домена Keystone пользователя, добавленный в проект service.
KEYSTONE → service_project_domain_name	Название домена Keystone проекта, добавленный в проект service.
DB → ENGINE	Тип базы данных.
DB → USER	Пользователь базы данных.
DB → PASSWORD	Пароль к базе данных.
DB → HOST	Узел, на котором запущена СУБД.
DB → PORT	Порт, на котором запущена СУБД.
DB → MAX_POOL_SIZE	Максимальное размер пула соединений к СУБД.
DB → MAX_OVERFLOW	Максимальный размер переполнения пула.
DB → POOL_RECYCLE	Время жизни соединения в пуле. При достижении времени соединение в пуле будет закрыто.
DB → POOL_TIMEOUT	Время таймаута соединения в пуле.
RABBIT_QUEUES → broker_type	Тип брокера сообщений для передачи сообщений.
RABBIT_QUEUES → host	Узел, на котором расположен брокер сообщений.
RABBIT_QUEUES → port	Порт, на котором расположен брокер сообщений.
RABBIT_QUEUES → username	Имя пользователя.
RABBIT_QUEUES → password	Пароль пользователя.

Имя параметра	Описание
RABBIT_QUEUES → durable	Режим сохранения состояния очередей при перезапуске RabbitMQ. Возможные значения: • `True`; • `False`.
SENTRY → DSN	Адрес, на который отправляются сообщения о событиях системой мониторинга ошибок Sentry.
SENTRY → ENABLED	Параметр, включающий систему мониторинга ошибок Sentry. Допустимые значения: • `True`; • `False`. По умолчанию `False`.
SENTRY → LOG_LEVEL	Уровень логирования системы мониторинга ошибок Sentry.