



# Установка и первичная настройка облачной платформы

## Содержание

<b>Требования к облачной платформе Tionix.....</b>	<b>6</b>
Требования к построению кластера.....	6
Введение.....	6
Требования к вычислительным ресурсам.....	7
Требования к сети.....	8
Сегментация сети.....	8
Требования к дисковому пространству.....	8
Требования к блочным устройствам Cinder.....	8
Требования к хостовым ОС.....	9
Системные пакеты.....	9
Модули TIONIX.....	9
<b>Настройка окружения.....</b>	<b>10</b>
Миграция с CentOS 8 на Almalinux.....	10
Введение.....	10
Предварительная настройка.....	10
Миграция дистрибутива.....	10
Настройка сетевых интерфейсов.....	11
Минимальная конфигурация.....	11
Смена имени машины.....	11
Регистрация доменных имён узлов.....	11
Настройка репозитория Almalinux.....	11
Введение.....	11
Включение репозитория.....	12
Установка системных пакетов.....	13
Настройка сервиса NTP.....	14
Введение.....	14
Установка сервера NTP.....	15
Установка клиента NTP.....	15
Проверка установки.....	16
Установка балансировщика нагрузки HAProxy.....	16
Установка HAProxy.....	16
Настройка фронтенда и бэкенда.....	20
Настройка SSL Termination в HAProxy.....	21
Установка и настройка СУБД MariaDB.....	22
Введение.....	22
Установка MariaDB.....	23
Настройка кластера Galera.....	23
Таблицы конфигурации.....	25
Установка сервиса memcached.....	28
Установка сервиса.....	28
Проверка работы сервиса.....	28
Установка сервиса RabbitMQ.....	29

Введение.....	29
Установка сервиса.....	30
Проверка работы сервиса.....	30
Создание начальных объектов в RabbitMQ .....	31
<b>Установка и настройка служб OpenStack.....</b>	<b>33</b>
OpenStack Keystone.....	33
Информация о сервисе Keystone .....	33
Установка сервиса Keystone .....	35
Создание первых объектов .....	39
Верификация работы сервиса.....	41
Шифрование сервиса.....	42
Описание файла конфигурации Keystone .....	47
OpenStack Glance .....	51
Информация о сервисе Glance.....	51
Установка сервиса Glance .....	51
Добавление тестового образа в Glance .....	55
Описание файла конфигурации Glance .....	56
OpenStack Placement .....	59
Информация о сервисе .....	59
Установка сервиса Placement.....	60
Шифрование сервиса Placement .....	63
Описание файла конфигурации Placement.....	64
OpenStack Nova.....	65
Информация о сервисе Nova .....	65
Установка управляющих сервисов Nova .....	67
Установка вычислительной части .....	73
Описание файла конфигурации Nova .....	74
OpenStack Neutron.....	80
Информация о сервисе Neutron .....	80
Установка OVN .....	81
Установка управляющих сервисов Neutron .....	84
Настройка вычислительного узла .....	89
Описание файла конфигурации Neutron.....	89
OpenStack Cinder .....	92
Информация о сервисе Cinder .....	92
Установка управляющей части.....	92
Настройка бэкенда Cinder.....	97
Описание файла конфигурации Cinder .....	99
OpenStack Ceilometer.....	101
Информация о сервисе Ceilometer .....	101
Установка управляющих сервисов Ceilometer .....	102
Настройка вычислительного узла для Ceilometer .....	109
Описание файла конфигурации Ceilometer.....	110
<b>Установка и настройка модулей TIONIX.....</b>	<b>112</b>

Предварительная настройка.....	112
Подготовка объектов в RabbitMQ.....	112
Создание объектов в OpenStack Keystone.....	112
Поддержка сервиса Sentry.....	112
Client.....	113
Информация о модуле Client.....	113
Установка модуля Client.....	113
Настройка сервиса Journal.....	117
Настройка драйвера LDAP.....	118
NodeControl.....	118
Информация о сервисе NodeControl.....	118
Установка сервиса NodeControl.....	119
Описание файла конфигурации сервиса NodeControl.....	122
Настройка хранилища статусов узлов NodeControl.....	130
Настройка устройств управления питанием узлов.....	133
Scheduler.....	136
Информация о сервисе Scheduler.....	136
Установка сервиса Scheduler.....	136
Описание файла конфигурации Scheduler.....	139
Monitor.....	141
Информация о сервисе Monitor.....	141
Установка сервиса TIONIX Monitor.....	141
Описание файла конфигурации Monitor.....	144
Dashboard.....	147
Информация о модуле Dashboard.....	147
Описание файла конфигурации сервиса Dashboard.....	147
Установка веб-панели Horizon.....	149
Установка модуля Dashboard.....	157
Pointmeter.....	160
Информация о сервисе PointMeter.....	160
Установка сервиса TIONIX Pointmeter.....	160
Описание файла конфигурации Pointmeter.....	161
Compute Agent.....	162
Информация об агенте TIONIX.....	162
Установка сервиса Agent.....	163
Описание файла конфигурации Agent.....	164
Drivers.....	165
Информация о драйверах TIONIX.....	165
Установка и настройка драйвера Cinder.....	166
Approvie.....	167
Информация о сервисе Approvie.....	167
Установка сервиса Approvie.....	168
Описание файла конфигурации Approvie.....	171
Описание основного файла конфигурации модулей TIONIX.....	173

Файл конфигурации.....	173
Таблица конфигурации .....	174

# Требования к облачной платформе Tionix

## Требования к построению кластера

- [Введение](#) (см. стр. 6)
  - [Управляющие узлы](#) (см. стр. 6)
    - [Расemaker](#) (см. стр. 6)
    - [MariaDB](#) (см. стр. 6)
    - [RabbitMQ](#) (см. стр. 6)
    - [memcached](#) (см. стр. 7)
    - [Сервисы OpenStack](#) (см. стр. 7)
    - [Модули TIONIX](#) (см. стр. 7)
  - [Вычислительные узлы](#) (см. стр. 7)

## Введение

Платформа TSP 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.

### Расemaker

Расemaker является основным средством запуска сервисов в режиме отказоустойчивости. В качестве средства синхронизации состояния сервисов используется [Corosync](#)<sup>1</sup>.

Для корректной работы Расemaker требуется:

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

### MariaDB

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

<sup>1</sup> <http://corosync.github.io/corosync/>

<sup>2</sup> <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-powpcproху 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 назначением:

- 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 (DWRPD >= 3) RAID1	2 x 300GB SSD (DWRPD >= 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: 10Gbit/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 [TIOX Driver \(см. стр. 165\)](#), 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 репозитория 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](#)<sup>3</sup>.

### 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-протокол) 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.

<sup>3</sup> <https://en.wikipedia.org/wiki/NetworkManager>

**i** Если вы нам необходимо произвести миграцию ОС с CentOS 8 на Almalinux 8, то используйте [эту инструкцию](#). (см. стр. 10)

## 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 содержанием:

```
[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 репозитория закончена.

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

## Установка системных пакетов

После добавления репозитория 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<sup>4</sup>](#) для OpenStack:

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


#### **Примечание**

Для продуктивных систем крайне нежелательно выключать систему мандатного доступа [SELinux<sup>5</sup>](#).

## TIONIX

Для модулей TIONIX необходимы следующие пакеты:

1. Установите пакет лицензирования модулей:

 **Важно**

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

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

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

- [Введение \(см. стр. 14\)](#)
- [Установка сервера NTP \(см. стр. 15\)](#)
- [Установка клиента NTP \(см. стр. 15\)](#)
- [Проверка установки \(см. стр. 16\)](#)

### 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<sup>6</sup>](#) (Network Time Protocol), а в качестве реализации этого протокола – [chrony<sup>7</sup>](#). 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

<sup>6</sup> <https://ru.wikipedia.org/wiki/NTP>

<sup>7</sup> <https://chrony.tuxfamily.org/>

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](#)<sup>8</sup>.

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

NTP-сервер использует порт 123/UDP.

**i** 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
```

**i** 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
```

**i** 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).

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
```

**i** 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
```

**i** 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):

<sup>8</sup> <https://habr.com/ru/post/79461/>

```
server controller iburst
```

**i** 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
```

**i** 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<sup>9</sup> – 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:

<sup>9</sup> <http://www.haproxy.org/>



```
dnf -y install haproxy
```

**i** Стандартные пути конфигурации:

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

2. Приведите конфигурацию HAProxy к следующему:

```
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

**i** Легенда таблицы доступна [на этой странице](#)<sup>10</sup>.

Имя параметра	Описание	Примечания
global	Глобальные настройки сервиса HAProxy.	

<sup>10</sup> <https://conf.tionix.ru/pages/viewpage.action?pageId=205881354#id-Условныеобозначения-table-format-desc>

Имя параметра	Описание	Примечания
log	Параметры журналирования.	<p>Указанное значение отправляет журнал сервиса в stderr, что позволит перенаправить его в систему управления контейнера.</p> <p>Для обычной установки <a href="#">требуется</a><sup>11</sup> установка syslog-сервиса и указание его адреса:</p> <ul style="list-style-type: none"> <li>log 127.0.0.1 local0</li> </ul> <p>Параметр global - это имя канала журналирования. Можно указать несколько параметров log с разными каналами, если требуется разделение журналов для фронтэндов.</p>
maxconn	Максимальное количество соединений к сервису.	<p>По умолчанию значение равно 2000.</p> <div style="border: 1px solid #ffc107; padding: 5px;"> <p> Не забудьте повысить лимиты на количество открытых файлов (nofile) для сервиса до значения maxconn.</p> </div>
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).	<p>В этом параметре явно выключаются все небезопасные версии протокола TLS для фронтэндов.</p> <p>Тикеты TLS необходимо выключить из-за <a href="#">их проблем с безопасностью</a><sup>12</sup> в версии TLS v1.2 (для TLS v1.3+ этот параметр не применим).</p>
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).	<p>В этом параметре явно выключаются все небезопасные версии протокола TLS для фронтэндов.</p> <p>Тикеты TLS необходимо выключить из-за <a href="#">их проблем с безопасностью</a><sup>13</sup> в версии TLS</p>

11 <https://www.haproxy.com/blog/introduction-to-haproxy-logging/>

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
		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.
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.

**i** В списке ciphers 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 зашифрованного 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.

**i** 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

**i** Легенда таблицы доступна [на этой странице](#)<sup>14</sup>.

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
```

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

## Таблица конфигурации

 Легенда таблицы доступна [на этой странице](#)<sup>15</sup>.


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 фронта/бэкенда вначале проверьте правильность конфигурации:

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

4. Перезагрузите конфигурацию сервиса HAProxy

```
systemctl reload haproxy
```

 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](#)<sup>16</sup> на уровне балансировщика нагрузки [HAProxy](#)<sup>17</sup>. Эта статья кратко опишет эту часть настройки балансировщика.

Если кратко, SSL Termination позволяет устанавливать зашифрованное 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 терминования 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 терминования SSL в балансировщике выглядит так:

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

<sup>16</sup> <https://www.haproxy.com/blog/haproxy-ssl-termination/>

<sup>17</sup> <https://www.haproxy.com/>

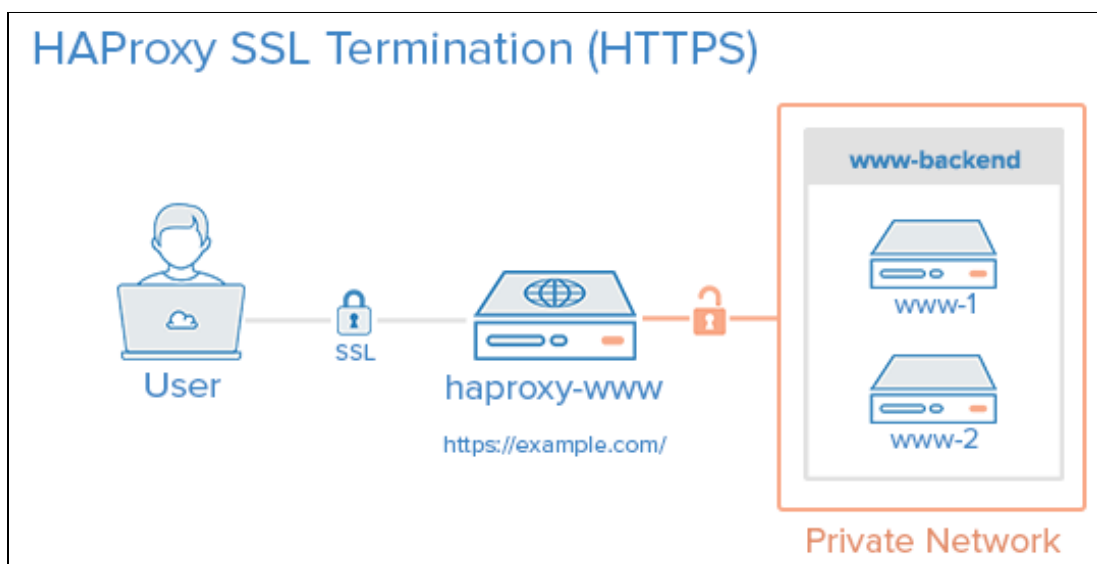


Схема терминирования SSL

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:

```
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<sup>18</sup> — это СУБД с открытым исходным кодом, которая является форком проекта MySQL<sup>19</sup> после перехода прав на этот продукт компании Oracle<sup>20</sup>. На данный момент [продолжает активно развиваться](#)<sup>21</sup> 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<sup>22</sup>. 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<sup>23</sup>, с 10.6 появилась поддержка движка хранения Aria<sup>24</sup> (в экспериментальном режиме, как и MyISAM<sup>25</sup>), в котором хранятся

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/>

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](#)<sup>26</sup>, Galera – [порты](#)<sup>27</sup> 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
```

**i** 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. 25)):

```
[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](#)<sup>28</sup>.

**i** 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

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>

28 <https://galeracluster.com/library/documentation/architecture.html>

чение) будет заблокирован ([подробнее](#)<sup>29</sup>). Возможен вариант с использованием [арбитра](#)<sup>30</sup>, 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 получают реплику. Выберите такой узел и в нём запустите команду:

```
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 *референтной архитектуре* доступ до сервисов Galera предоставляется через балансировщика нагрузки. Конфигурация для HAProxy выглядит следующим образом ([описание](#) (см. стр. 27)):

```
global
```

<sup>29</sup> <https://galeracluster.com/library/documentation/crash-recovery.html>

<sup>30</sup> <https://galeracluster.com/library/documentation/arbitrator.html>



```

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](#)<sup>31</sup>.

## Общие параметры MariaDB

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

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 <a href="#">здесь</a> <sup>32</sup> .
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 <a href="#">здесь</a> <sup>33</sup> .

## Параметры кластеризации Galera

Имя параметра	Описание	Примечания
[galera]	Параметры репликации Galera	
wsrep_on	Включение протокола wsrep API	wsrep API является внутренним механизмом 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 <b>бинарных журналов</b> <sup>34</sup> состояния БД.	На момент написания Galera официально <b>поддерживает только формат ROW</b> <sup>35</sup> 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 <sup>36</sup>	<p>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. <a href="#">Это безопасно при использовании ROW-формата для бинарных журналов и позволяет повысить быстродействие.</a><sup>37</sup></p> <p>Также нужно отметить, что при работе автоинкремента ID объектов будут увеличиваться на шаг, равный числу узлов кластера Galera.</p>
innodb_doublewrite	Включение буфера двойной записи.	<p>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 <a href="#">здесь</a><sup>38</sup>.</p> <p>Отрицательно влияет на производительность кластера из-за двойной записи на диск.</p>
innodb_flush_log_at_trx_commit	Политика записи данных транзакции в буферном журнале.	<p>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).</p> <p>Очень сильно положительно влияет на производительность кластера.</p>
innodb_buffer_pool_size	Размера буферного пула в ОЗУ для промежуточной записи транзакций к БД.	<p>Это параметр для оптимизации быстродействия, определяющий, сколько ОЗУ может занять буферный пул MariaDB. Разработчики <a href="#">рекомендуют указывать</a><sup>39</sup> 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).</p>

## Настройки HAProxy

 Легенда таблицы доступна [на этой странице](#)<sup>40</sup>.


<sup>36</sup> [https://mariadb.com/kb/en/auto\\_increment/](https://mariadb.com/kb/en/auto_increment/)

<sup>37</sup> [https://www.percona.com/blog/2017/07/26/what-is-innodb\\_autoinc\\_lock\\_mode-and-why-should-i-care/](https://www.percona.com/blog/2017/07/26/what-is-innodb_autoinc_lock_mode-and-why-should-i-care/)

<sup>38</sup> <https://mariadb.com/kb/en/innodb-doublewrite-buffer/>

<sup>39</sup> <https://mariadb.com/kb/en/innodb-buffer-pool/>

<sup>40</sup> <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 <a href="#">использовать внутреннюю реализацию резолвинга DNS</a> <sup>41</sup> 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 <a href="#">EDNS</a> <sup>42</sup> .

## Установка сервиса memcached

[memcached](#)<sup>43</sup> – это простой сетевой сервис кэширования данных. Многие сервисы OpenStack могут кэшировать часть данных в этом сервисе, в частности, проекты Keystone и Nova. memcached хранит информацию в оперативной памяти в виде [хеш-таблицы](#)<sup>44</sup>. 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 конфигурациикажите 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 обеспечиваются созданием нового экземпляра контейнера с сервисом при проблемах со старым.

## Проверка работы сервиса

1. Проверьте статус юнита memcached:

<sup>41</sup> <https://www.haproxy.com/blog/dns-service-discovery-haproxy/>

<sup>42</sup> <https://ru.wikipedia.org/wiki/EDNS>

<sup>43</sup> <https://memcached.org/>

<sup>44</sup> <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
```

**i** 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
```

**i** 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
```

**i** 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](https://www.rabbitmq.com/)<sup>45</sup> – это проект с открытым исходным кодом, реализацию сетевой сервис обмена сообщениями в режиме [Publisher-Subscriber \(Pub/Sub\)](https://ru.wikipedia.org/wiki/Publisher-Subscriber_(Pub/Sub))<sup>46</sup>. Выделяется хорошей производительностью, достаточно простой конфигурацией и доступен во всех известных дистрибутивах.

RabbitMQ использует порт [5672/TCP](https://ru.adminsub.net/tcp-udp-port-finder/5672)<sup>47</sup>.

## Установка сервиса

RabbitMQ доступен из официальных репозиториях

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 `metcached`:

```
systemctl status rabbitmq-server
```

### 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)
```

<sup>45</sup> <https://www.rabbitmq.com/>

<sup>46</sup> [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\\_\(%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD\\_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F\)](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))

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

```

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
```

**i** 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](#)<sup>48</sup>.

## 3. Проверьте статус порта сервиса epmd:

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

**i** В ответ на команду вы должны получить примерно следующий вывод:

```

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

```

epmd по умолчанию прослушивает все интерфейсы.

Подробнее о сетевых настройках RabbitMQ можно [узнать здесь](#)<sup>49</sup>.

## 4. Получите данные состояния RabbitMQ:

```
rabbitmqctl status
```

**i** 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

48 <https://www.rabbitmq.com/networking.html>

49 <https://www.rabbitmq.com/networking.html>

**i** Более подробно об аутентификации в RabbitMQ можно узнать [по этой ссылке](#)<sup>50</sup>.

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](#)<sup>51</sup> tionix:

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

**i** 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.

<sup>50</sup> <https://www.rabbitmq.com/access-control.html>

<sup>51</sup> <https://www.rabbitmq.com/vhosts.html>



# Установка и настройка служб OpenStack

## OpenStack Keystone

### Информация о сервисе Keystone

- [Обзор сервиса \(см. стр. 33\)](#)
- [Краткий список терминов \(см. стр. 33\)](#)
- [Работа основной функции сервиса \(см. стр. 34\)](#)
- [Дополнительные материалы \(см. стр. 34\)](#)

### Обзор сервиса

[OpenStack Keystone](#)<sup>52</sup> (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 пользователю 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.

<sup>52</sup> <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](#)<sup>53</sup>.
- **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 передает 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](#)<sup>54</sup> официальной документации проекта.
  - [Подробная архитектура сервиса](#)<sup>55</sup>.
- [Официальный репозиторий](#)<sup>56</sup> проекта.

53 <https://conf.tionix.ru/pages/viewpage.action?pagelId=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<sup>57</sup> сервиса.](#)

## Установка сервиса Keystone

- [Настройка окружения \(см. стр. 35\)](#)
  - [Подготовка базы данных keystone \(см. стр. 35\)](#)
  - [Установка зависимых пакетов \(см. стр. 35\)](#)
    - [Apache и модули \(см. стр. 35\)](#)
- [Установка сервиса Keystone \(см. стр. 36\)](#)
- [Финализация установки \(см. стр. 37\)](#)
- [Проверка работы сервиса \(см. стр. 37\)](#)
- [Файл настройки системного окружения \(см. стр. 38\)](#)

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<sup>58</sup>](#) – 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<sup>59</sup>](#) – 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
```

- 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

**i** См. также: [Настройка репозитория Almalinux](#) (см. стр. 11).

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

- Установите пакет сервиса:

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

**i** Стандартные пути файлов конфигурации:

- `/etc/keystone` – Каталог конфигурации Keystone;
- `/etc/keystone/keystone.conf` – основной файл конфигурации.

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

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

- В основной файл добавьте следующую конфигурацию ([ОПИСАНИЕ](#) (см. стр. 47)):

```
[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
```

- 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. 48), 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
```

**i** 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
```

**i** 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.

**w** 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
```

**i** 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
```


**i** 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 содержанием ([описание](#) (см. стр. 50)):

```
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
```

- i** В ответ вы должны получить список пользователя, который должен содержать пользователя admin:

```
+-----+-----+
| ID | Name |
+-----+-----+
| e8f0a2078a9548029f7f9025beeec983 | admin |
+-----+-----+
```

## Создание первых объектов

- [Создание проекта service \(см. стр. 39\)](#)
- [Создание нового домена \(см. стр. 39\)](#)
- [Создание нового проекта \(см. стр. 40\)](#)
- [Создание обычного пользователя и указание роли в проекте \(см. стр. 40\)](#)

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
```

- i** 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 | [] |
+-----+-----+
```

- i** 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
```

- i** 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](#)<sup>60</sup> 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
```

ℹ 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
```

ℹ При выполнении команды интерактивно будет запрошен пароль для пользователя

ℹ При успешном выполнении команды вы должны получить следующий вывод:

```

+-----+-----+
| Field | Value |
+-----+-----+
| domain_id | default |
| enabled | True |
| id | 65d58683983a49d0979652be4ae6d4a2 |
| name | user |

```

<sup>60</sup><https://conf.tionix.ru/pages/viewpage.action?pageId=164102237#id->

Использование связанных с доменом систем хранения данных пользователей-ks-domain-specific-drivers



```
| options | {} |
| password_expires_at | None |
+-----+-----+
```

## 2. Создайте новую роль:

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

**i** 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
```

**i** Для предоставления административных прав пользователю можете указать роль admin.

**i** При успешном выполнении команда должна вернуть пустой вывод.

**i** 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](#)<sup>61</sup>.

## 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
```

**i** В ответ вы должны получить данные сгенерированного токена:

```
+-----+
+-----+-----+
| Field | Value |
+-----+-----+
| expires | 2016-02-12T20:14:07.056119Z |
| id | gAAAAABWvi7_B8kKQD9wdXac8MoZiQldmjE0643d-e_j-XXq9AmIegIbA7UHGPv |
| | atnN21qtOMjCFwX7BREJEQnVOAj3ncLRQgAYRsfSU_MrsuWb4EDtnjU7HEpoBb4 |
| | o6ozsA_NmFWEpLeKy0uNn_WeKbAhYygrsmQGA49dcLHVnz-OMVLIyM9ws |
+-----+-----+
```

<sup>61</sup> <https://conf.tionix.ru/pages/viewpage.action?pagelD=164102171>

```
| project_id | 343d245e850143a096806dfaefa9afdc |
| user_id   | ac3377633149401296f6c0d92d79dc16 |
+-----+
+-----+
```

## 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_B8kKQD9wdXac8MoZiQldmjE0643d-e_j-XXq9AmIegIbA7UHGPv |
| |      | atnN21qt0MjCFWX7BReJEQnV0Aj3ncLRQgAYRsfSU_MrsuWb4EDtnjU7HEpoBb4 |
| |      | o6ozsA_NmFWepLeKy0uNn_WeKbAhYgrsmQGA49dcLHVnz-0MVLiyM9ws |
| project_id | 343d245e850143a096806dfaefa9afdc |
| user_id   | ac3377633149401296f6c0d92d79dc16 |
+-----+
+-----+
```

## 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](#)<sup>62</sup> 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. 44)):

```
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
```

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

```

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 зашифрованные 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](#)<sup>63</sup>.

Имя параметра	Описание	Примечания
<b>ServerName</b>	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.
<b>ServerRoot</b>	Путь до корневого каталога веб-сервера.	 Не меняйте этот параметр без необходимости.
<b>Include</b>	Включение дополнительных файлов конфигурации, расположенных по указанному пути.	
<b>User</b>	Пользователь операционной системы, от имени которого будут запущены процессы веб-сервера Apache.	
<b>Group</b>	Группа операционной системы, от имени которой будут запущены процессы веб-сервера Apache.	
<b>LogFormat</b>	Формат журналирования событий.	
<b>ErrorLog</b>	Путь для хранения журналов ошибок.	Есть три варианта использования: <ul style="list-style-type: none"> <li>• <b>/dev/stderr</b> – для референсной архитектуры, перенаправит журналы в систему управления контейнерами.</li> <li>• <b>/var/log/httpd/keystone-error.log</b> – для обычной установки, перенаправит журнал по указанному пути.</li> <li>• <b>journald</b> – для обычной установки, перенаправит журнал в сервис journald. Требуется включение модуля <a href="#">mod_journald</a><sup>64</sup>.</li> </ul>
<b>CustomLog</b>	Путь для хранения журналов запросов к веб-серверу.	Есть три варианта использования: <ul style="list-style-type: none"> <li>• <b>/dev/stdout</b> – для референсной архитектуры, перенаправит журналы в систему управления контейнерами.</li> <li>• <b>/var/log/httpd/keystone-custom.log</b> – для обычной установки, перенаправит журнал по указанному пути.</li> <li>• <b>journald</b> – для обычной установки, перенаправит журнал в сервис journald (журналы будут доступны юниту httpd). Требуется включение модуля <a href="#">mod_journald</a><sup>65</sup>.</li> </ul>

63 <https://conf.tionix.ru/pages/viewpage.action?pagelId=205881354#id-Условныеобозначения-table-format-desc>

64 [https://httpd.apache.org/docs/trunk/mod/mod\\_journald.html](https://httpd.apache.org/docs/trunk/mod/mod_journald.html)

65 [https://httpd.apache.org/docs/trunk/mod/mod\\_journald.html](https://httpd.apache.org/docs/trunk/mod/mod_journald.html)

Имя параметра	Описание	Примечания
<b>SSLProtocol</b>	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+.
<b>SSLCipherSuite</b>	Список доступных алгоритмов шифрования.	Данный список был составлен по определенным критериям, в основном, по итоговой производительности и уровню безопасности в целом. Вначале используются EC <sup>66</sup> -варианты AES <sup>67</sup> в режиме GCM <sup>68</sup> 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 <sup>69</sup> в качестве legacy-варианта.
<b>SSLHonorCipherOrder</b>	Ограничение поддержки алгоритмов шифрования списков SSLCipherSuite.	При включении этого параметра клиент может использовать только те алгоритмы, которые указаны в <b>SSLCipherSuite</b> .
<b>SSLSessionTickets</b>	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 <sup>70</sup> эту директиву следует выключить.
<b>Listen</b>	Адрес и порт прослушивания для виртуального узла Apache.	<p>Через указанный порт будет доступно WSGI-приложение с API сервиса Keystone.</p> <p>По умолчанию порт равен 5000.</p> <div style="border: 1px solid orange; padding: 5px;"> <p> 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:</p> <ul style="list-style-type: none"> <li>Listen controller:5000</li> </ul> </div>
<b>VirtualHost</b>	Директива описания виртуального узла Apache.	В этой секции описываются параметры процесса, реализующий WSGI-приложение.
<b>WSGIDaemonProcess</b>	Имя запускаемого процесса	В этом параметре указываются параметры запуска сервиса Keystone как WSGI-приложения.
<b>WSGIProcessGroup</b>	Группа процессов WSGI, к которому относится WSGI-приложение с сервисом Keystone.	




<sup>6</sup> [https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%BB%D0%B8%D0%BF%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BO%D1%8F\\_%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%BO%D1%84%D0%B8%D1%8F](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%BB%D0%B8%D0%BF%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BO%D1%8F_%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%BO%D1%84%D0%B8%D1%8F)

<sup>67</sup> [https://ru.wikipedia.org/wiki/AES\\_\(%D1%81%D1%82%D0%BO%D0%BD%D0%B4%D0%BO%D1%80%D1%82\\_%D1%88%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%BO%D0%BD%D0%B8%D1%8F\)](https://ru.wikipedia.org/wiki/AES_(%D1%81%D1%82%D0%BO%D0%BD%D0%B4%D0%BO%D1%80%D1%82_%D1%88%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%BO%D0%BD%D0%B8%D1%8F))

<sup>68</sup> [https://ru.wikipedia.org/wiki/Galois/Counter\\_Mode](https://ru.wikipedia.org/wiki/Galois/Counter_Mode)

<sup>69</sup> [https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB\\_%D0%94%D0%B8%D1%84%D1%84%D0%B8\\_%E2%80%94%D0%A5%D0%B5%D0%BB%D0%BB%D0%BC%D0%BO%D0%BD%D0%BO](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB_%D0%94%D0%B8%D1%84%D1%84%D0%B8_%E2%80%94%D0%A5%D0%B5%D0%BB%D0%BB%D0%BC%D0%BO%D0%BD%D0%BO)

<sup>70</sup> <https://blog.filippo.io/we-need-to-talk-about-session-tickets/>

Имя параметра	Описание	Примечания
<b>WSGIScriptAlias</b>	Ссылка на путь до WSGI-приложения.	В данном случае указывается, по какому пути URL Apache должен будет обратиться к WSGI-приложению.
<b>WSGIApplicationGroup</b>	Группа приложений WSGI, к которому относится WSGI-приложение с сервисом Keystone.	
<b>WSGIPassAuthorization</b>	Включение обработки заголовков авторизации в запросах HTTP.	
<b>LimitRequestBody</b>	Максимальный размер запроса HTTP (в байтах).	
<b>ErrorLogFormat</b>	Формат журнала ошибок.	
<b>SSLEngine</b>	Включение режима шифрования соединений для виртуального узла Apache.	
<b>SSLCertificateFile</b>	Путь до файла сертификата для шифрования соединения. Обязателен, если указан <b>SSLEngine on</b> .	<p>Можно указать:</p> <ul style="list-style-type: none"> <li>абсолютный путь с указанием "/" в начале пути.</li> <li>относительный путь, который начинается с пути <b>ServerRoot</b>. В этом случае "/" в начале пути не указывается.</li> </ul> <div style="border: 1px solid #ffc107; padding: 5px; margin-top: 10px;"> <p> Файл сертификата должен быть доступен для чтения процессам Apache.</p> </div>
<b>SSLCertificateKeyFile</b>	Путь до файла ключа к сертификату для шифрования соединения. Обязателен, если указан <b>SSLEngine on</b> .	<p>Можно указать:</p> <ul style="list-style-type: none"> <li>абсолютный путь с указанием "/" в начале пути.</li> <li>относительный путь, который начинается с пути <b>ServerRoot</b>. В этом случае "/" в начале пути не указывается.</li> </ul> <div style="border: 1px solid #ffc107; padding: 5px; margin-top: 10px;"> <p> Файл сертификата должен быть доступен для чтения процессам Apache.</p> </div>
<b>Protocols</b>	Доступные версии протокола HTTP.	<div style="border: 1px solid #6c757d; padding: 5px; margin-top: 10px;"> <p> При указании протокола h2 автоматически включается протокол APLN, позволяющий клиенту узнать, какие версии протокола HTTP поддерживаются сервером.</p> </div>
<b>Directory</b>	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.
<b>Require</b>	Права доступа к указанному в директиве пути.	В данном случае предоставляются все права к каталогу <code>/usr/bin</code> .

Имя параметра	Описание	Примечания
<b>Alias</b>	Глобальный алиас на ресурс при запросе ссылки HTTP.	В случае сервиса Keystone необходимо создать алиас <code>/identity</code> , откуда по умолчанию должен быть доступен его API.
<b>Location</b>	Параметры ссылки HTTP.	
<b>SetHandler</b>	Указание обработчика при вызове ссылки.	<code>wsgi-script</code> включает обработчик для исполнения WSGI-приложения.
<b>Options</b>	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](#)<sup>71</sup>.

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




Имя параметра	Описание	Примечания
<b>DEFAULT</b>	Глобальные параметры сервиса Keystone.	
<b>debug</b>	Включение отладочного режима журналирования.	<p>В продуктивных средах используйте только для тестирования и выявления проблем.</p> <div style="border: 1px solid #ffc107; padding: 5px; background-color: #fff3cd;"> <p> Отладочный режим может повлиять на производительность и конфиденциальность данных.</p> </div>
<b>[oslo_messaging_notifications] driver transport_url</b>	Включение поддержки уведомлений о событиях в сервисе с использованием RabbitMQ.	<p>Эти параметры необходимы <a href="#">для сервиса Journal</a> (см. стр. 117), входящий в состав модуля <a href="#">TIONIX Client</a> (см. стр. 113).</p> <p>Несколько адресов указываются в <code>transport_url</code> через запятую. Подключения к ним будут происходить последовательно.</p>

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

Имя параметра	Описание	Примечания
<b>use_stderr</b> <b>log_dir</b> <b>use_journal</b>	Параметры журналирования сервиса.	<p>Необходимо выбрать <b>один из трёх</b> указанных параметров:</p> <ul style="list-style-type: none"> <li>• <b>use_stderr = True</b>, если установка производится в рамках <i>референсной архитектуры</i>, в этом случае журналы будут перенаправлены в виртуальное устройство /dev/stderr. Это позволяет получить логи системе управления контейнерами.</li> <li>• <b>log_dir = /var/log/keystone</b>, если производится <i>обычная установка</i>, в этом случае журналы будут сохраняться в файлах в указанном каталоге.</li> <li>• <b>use_journal = True</b>, если производится <i>обычная установка</i>, в этом случае журналы будут перенаправлены в службу управления журналами <a href="#">systemd-journald</a>.<sup>72</sup></li> </ul> <div style="border: 1px solid #ffc107; padding: 5px; margin-top: 10px;"> <p> Режим use_syslog для перенаправления журналов в службу syslog переведён в разряд устаревших, поэтому официально не поддерживается.</p> </div>
<b>oslo_middleware</b>	Параметры обработки запросов к API сервиса Keystone.	
<b>enable_proxy_headers_parsing</b>	Обработка заголовков прокси-сервера.	<div style="border: 1px solid #ffc107; padding: 5px;"> <p> Этот параметр обязателен при использовании балансировщика нагрузки HAProxy.</p> </div>
<b>database</b>	Параметры подключения к базе данных keystone.	
<b>connection</b>	Адрес базы данных keystone.	Можно указать только один адрес.
<b>connection_recycle_time</b>	Время жизни соединений к базе данных, которые имеются в пуле соединений.	
<b>max_pool_size</b>	Максимальный размер пула соединений к базе данных.	
<b>max_retries</b>	Максимальное количество попыток соединений в пуле.	Значение "-1" выключает ограничение по количеству попыток.
<b>identity</b>	Основные параметры обработки данных аутентификации в Keystone.	

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



Имя параметра	Описание	Примечания
<b>driver</b>	Метод хранения и доступа к данным аутентификации.	<p>driver может иметь два значения для трёх ситуаций:</p> <ul style="list-style-type: none"> <li>"<b>sql</b>" – в этом случае данные аутентификации будут сохранены в базе данных Keystone в СУБД MariaDB для всех доменов OpenStack.</li> <li>"<b>ldap</b>" – в этом случае данные аутентификации пользователей и групп облачной платформы будут получены с LDAP-сервера в рамках домена "default" по умолчанию.</li> <li>"<b>sql</b>" и <a href="#">включение домен-специфичных драйверов</a><sup>73</sup> – 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.</li> <li>"<b>tnx_ldap</b>" – это расширенный компанией TIONIX <a href="#">вариант драйвера (см. стр. 118) "ldap"</a>. Может работать совместно с <a href="#">домен-специфичными драйверами</a><sup>74</sup>.</li> </ul>
<b>domain_specific_drivers_enabled</b>	<a href="#">Включение домен-специфичных драйверов</a> <sup>75</sup> .	Эта функция позволяет для каждого домена OpenStack указать свои данные подключения в LDAP-серверам.
<b>domain_config_dir</b>	Каталог конфигурации домен-специфичных драйверов. Обязателен, если включены домен-специфичные драйверы.	
<b>caching</b>	Включение кэширования для создания и валидации токена.	<div style="border: 1px solid #ccc; padding: 5px;"> <p> Требуется включения глобальной функции кэширования в секции cache.</p> </div>
<b>token</b>	Параметры генерирования токенов авторизации.	
<b>revoke_by_id</b>	Включение отзыва токена по его ID.	
<b>provider</b>	Алгоритм генерирования токена.	<div style="border: 1px solid #ffc107; padding: 5px; margin-bottom: 5px;"> <p> Для продуктивных систем всегда используйте алгоритм fernet.</p> </div> <div style="border: 1px solid #dc3545; padding: 5px;"> <p> Работа с другими типами токенов в продуктивных системах официально не поддерживается.</p> </div>

<sup>73</sup> <https://conf.tionix.ru/pages/viewpage.action?pagelId=164102237#id->



Использование связанных доменов систем хранения данных пользователей-ks-domain-specific-drivers

<sup>74</sup> <https://conf.tionix.ru/pages/viewpage.action?pagelId=164102237#id->


Использование связанных доменов систем хранения данных пользователей-ks-domain-specific-drivers

<sup>75</sup> <https://conf.tionix.ru/pages/viewpage.action?pagelId=164102237#id->

Использование связанных доменов систем хранения данных пользователей-ks-domain-specific-drivers

Имя параметра	Описание	Примечания
<b>expiration</b>	Время жизни токена.	<div style="border: 1px solid #ffc107; padding: 5px;"> <p> В продуктивных системах не используйте слишком большой интервал жизни токена (не больше 24 часов).</p> </div>
<b>cache</b>	Глобальные параметры кэширования сервиса Keystone	
<b>backend</b>	Параметр реализации кэширования. Обязателен, если указан <code>enabled</code> в секции <code>cache</code> .	<p>Может иметь два параметра:</p> <ul style="list-style-type: none"> <li>• <b>dogpile.cache.memcached</b> – рекомендуется для небольших инсталляций.</li> <li>• <b>oslo_cache.memcache_pool</b> – рекомендуется для высоконагруженных систем из-за поддержки пулинга соединений.</li> </ul> <div style="border: 1px solid #ffc107; padding: 5px;"> <p> По умолчанию для продуктивных систем используйте <b>oslo_cache.memcache_pool</b>.</p> </div>
<b>enabled</b>	Включения режима кэширования.	
<b>servers</b>	Список серверов Memcached.	Настраивается в виде списка <code>addr:port</code> , разделенные запятыми.

### Файл экспорта параметров входа

 Легенда таблицы доступна [на этой странице](#)<sup>76</sup>.

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

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

<sup>76</sup> <https://conf.tionix.ru/pages/viewpage.action?pageId=205881354#id-Условныеобозначения-table-format-desc>

# OpenStack Glance

## Информация о сервисе Glance

[OpenStack Glance](#)<sup>77</sup> – это служба предоставления и хранения наборов данных, необходимых конечным пользователям для создания вычислительных объектов облачной платформы. Иными словами, в Glance 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 VM 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](#) (см. стр. 51)
  - [Подготовка базы данных glance](#) (см. стр. 51)
  - [Создание объектов в Keystone](#) (см. стр. 52)
- [Установка сервиса Glance](#) (см. стр. 53)
  - [Финализация установки](#) (см. стр. 54)
- [Проверка работы сервиса](#) (см. стр. 54)
  - [Конфигурация для HAProxy](#) (см. стр. 55)

## 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:

<sup>77</sup> <https://docs.openstack.org/glance/victoria/>

```
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';
```

**i** Вместо GLANCE\_DBPASS используйте свой пароль, он будет необходим далее.

4. Выйдите из сессии СУБД:

```
exit;
```

## Создание объектов в Keystone

**i** 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
```

**i** 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.

**i** 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 репозитория Almalinux](#) (см. стр. 11).

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` – основной 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. 57)):

```
[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
```

- i** 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. 57), 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
```

- i** 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
```

**i** 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
```

**i** Ответ должен быть примерно таким:

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

## Конфигурация для HAProxy

**i** 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

**i** 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
```

**i** 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
```

**i** 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](#)<sup>78</sup>.

 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: <ul style="list-style-type: none"> <li>• <b>localhost</b>, если установка производится в рамках референсной архитектуры;</li> <li>• <b>Адрес в mgmt-интерфейсе</b>, если производится обычная установка.</li> </ul>
use_stderr log_dir use_journal	Параметры журналирования сервиса.	Необходимо выбрать <b>один из трёх</b> указанных параметров: <ul style="list-style-type: none"> <li>• <b>use_stderr = True</b>, если установка производится в рамках референсной архитектуры, в этом случае журналы будут перенаправлены в виртуальное устройство /dev/stderr. Это позволяет получить логи системы управления контейнерами.</li> <li>• <b>log_dir = /var/log/glance</b>, если производится обычная установка, в этом случае журналы будут сохраняться в файлах в указанном каталоге.</li> <li>• <b>use_journal = True</b>, если производится обычная установка, в этом случае журналы будут перенаправлены в службу управления журналами <a href="#">systemd-journald</a>.<sup>79</sup></li> </ul>  Режим <b>use_syslog</b> 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.

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

79 <https://www.freedesktop.org/software/systemd/man/systemd-journald.service.html>

Имя конфигурации	Описание функции	Примечание
<b>show_image_direct_url</b>	Поддержка показа прямых ссылок до образа.	Используйте этот параметр только с системой хранения Ceph (для функции CoW-копирования). Документация предупреждает о <a href="#">проблеме безопасности</a> <sup>80</sup> этого параметра.
<b>transport_url</b>	Список адресов до серверов RabbitMQ.	Серверы указываются через запятую с указанием префикса в каждом элементе.
<b>cinder_catalog_info</b>	Источник адреса точки входа в Cinder.	
<b>[database]</b>	Настройки доступа к базе данных.	
<b>connection</b>	Адрес до СУБД.	
<b>retry_interval</b>	Интервал между попытками подключения.	
<b>connection_recycle_time</b>	Время жизни одного соединения.	
<b>max_pool_size</b>	Максимальный размер пула соединений к СУБД.	
<b>max_retries</b>	Максимальное количество попыток соединений к СУБД.	-1 выключает ограничение на количество попыток.
<b>[keystone_authtoken]</b>	Параметры подключения к сервису Keystone.	
<b>www_authenticate_uri</b> <b>auth_url</b>	Адрес до сервиса Keystone.	Оба параметра должны совпадать друг с другом.
<b>memcached_servers</b>	Адреса до сервисов кэширования memcached.	Этот параметр действителен лишь для запросов аутентификации. Несколько адресов можно указать через запятую.
<b>auth_type</b>	Тип аутентификации.	Всегда должен быть равен параметру "password".
<b>project_domain_name</b> <b>user_domain_name</b> <b>project_name</b> <b>username</b> <b>password</b>	Данные аутентификации в сервис Keystone от имени пользователя glance в проекте service.	Оставьте эти параметры по умолчанию.
<b>[paste_deploy]</b>	Параметры interface деплоя WSGI-сервиса.	
<b>flavor</b>	Используемая система аутентификации.	Всегда должен быть равен "keystone".

80 [https://docs.openstack.org/glance/victoria/configuration/glance\\_api.html#DEFAULT.show\\_image\\_direct\\_url](https://docs.openstack.org/glance/victoria/configuration/glance_api.html#DEFAULT.show_image_direct_url)

Имя конфигурации	Описание функции	Примечание
<b>config_file</b>	Путь до paste-файла сервиса Glance.	
<b>[glance_store]</b>	Параметры системы хранения для образов.	
<b>stores</b>	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.
<b>default_store</b>	Основной тип хранения образов. Обязателен, если len(stores) > 1	Glance выберет этот тип хранения по умолчанию, если клиент явно не запросит другой.
<b>[oslo_messaging_notifications]</b>	Параметры оповещений событий сервиса.	Для этого параметра необходимо указание адреса RabbitMQ в transport_url.
<b>driver</b>	Тип драйвера оповещений.	В референсе всегда должен быть равен "messagingv2".
<b>[cache]</b>	Параметры кэширования запросов сервиса.	
<b>backend</b>	Тип внешнего сервиса кэширования.	В референсной схеме нужно использовать oslo_cache.memcache_pool.
<b>enabled</b>	Параметр включения кэширования.	
<b>memcache_servers</b>	Адреса внешних сервисов кэширования.	Этот параметр действителен для запросов Glance. Несколько адресов можно указать через запятую.
<b>[oslo_middleware]</b>	Параметры обработки запросов сервиса Glance.	
<b>enable_proxy_headers_parsing</b>	Параметр обработки заголовков прокси-сервера.	
<b>*store</b>	Параметры, определяющие места хранения различных данных сервиса Glance.	

## OpenStack Placement

### Информация о сервисе

[OpenStack Placement](https://docs.openstack.org/placement/victoria/)<sup>81</sup> - 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.

<sup>81</sup> <https://docs.openstack.org/placement/victoria/>

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](#)<sup>82</sup> 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](#) (см. стр. 60)
  - [Установка зависимых пакетов](#) (см. стр. 60)
  - [Подготовка базы данных placement](#) (см. стр. 60)
  - [Создание объектов в Keystone](#) (см. стр. 61)
- [Установка сервиса](#) (см. стр. 61)
  - [Финализация установки](#) (см. стр. 62)

### 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](#)<sup>83</sup>.

#### Подготовка базы данных placement

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

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

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';
```

<sup>82</sup> <https://docs.openstack.org/api-ref/placement/>

<sup>83</sup> <https://docs.openstack.org/placement/victoria/install/index.html>

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

**i** Вместо PLACEMENT\_DBPASS используйте свой пароль, он будет необходим далее.

#### 4. Выйдите из сессии СУБД:

```
exit;
```

## Создание объектов в Keystone

**i** 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
```

**i** 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
```

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

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

## Установка сервиса

**i** См. также: [Настройка репозитория Almalinux](#) (см. стр. 11).

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

#### 1. Установите основной пакет сервиса Placement:

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

**i** Стандартные пути файлов конфигурации:

- `/etc/placement` – Каталог конфигурации Placement;
- `/etc/placement/placement.conf` – основной 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 \(см. стр. 64\)](#)):

```
[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:
- Данные соединения до БД placement в **placement\_database/connection**, в частности, пароль к базе вместо PLACEMENT\_DBPASS;
  - 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
```

**i** 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. 44) Руководства.

2. После включения этой конфигурации перезапустите веб-сервер:

```
systemctl restart httpd
```

3. После этого убедитесь, что зашифрованные 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 СА 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. Эта 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](#)<sup>84</sup>.

Имя параметра	Описание	Примечания
<b>[placement_database]</b>	Параметры доступа к базе данных placement.	
<b>connection</b>	Параметры соединения к базе данных placement.	
<b>[api]</b>	Параметры работы Placement API.	
<b>auth_strategy</b>	Указание механизма аутентификации.	Всегда должен быть равен "keystone".
<b>[keystone_authtoken]</b>	Параметры подключения к сервису Keystone.	
<b>auth_url</b>	Адрес внутреннего API Keystone.	
<b>memcached_servers</b>	Адреса внешней системы кэширования Memcached.	Можно указать несколько серверов через запятую.

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



Имя параметра	Описание	Примечания
<b>project_domain_name</b> <b>user_domain_name</b> <b>project_name</b> <b>username</b> <b>password</b>	Данные для аутентификации в Keystone от имени пользователя placement.	

## OpenStack Nova

### Информация о сервисе Nova

- [Обзор сервиса \(см. стр. 65\)](#)
- [Состав сервиса \(см. стр. 65\)](#)
- [Основные объекты сервиса \(см. стр. 66\)](#)
- [Дополнительная информация \(см. стр. 67\)](#)

### Обзор сервиса

[OpenStack Nova](#)<sup>85</sup> – 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](#)<sup>86</sup> с поддержкой аппаратного ускорения [KVM](#)<sup>87</sup>. Поэтому в составе вычислительных узлов будут добавлены еще два основных компонента:

85 <https://docs.openstack.org/nova/victoria/>

86 <https://www.qemu.org/>

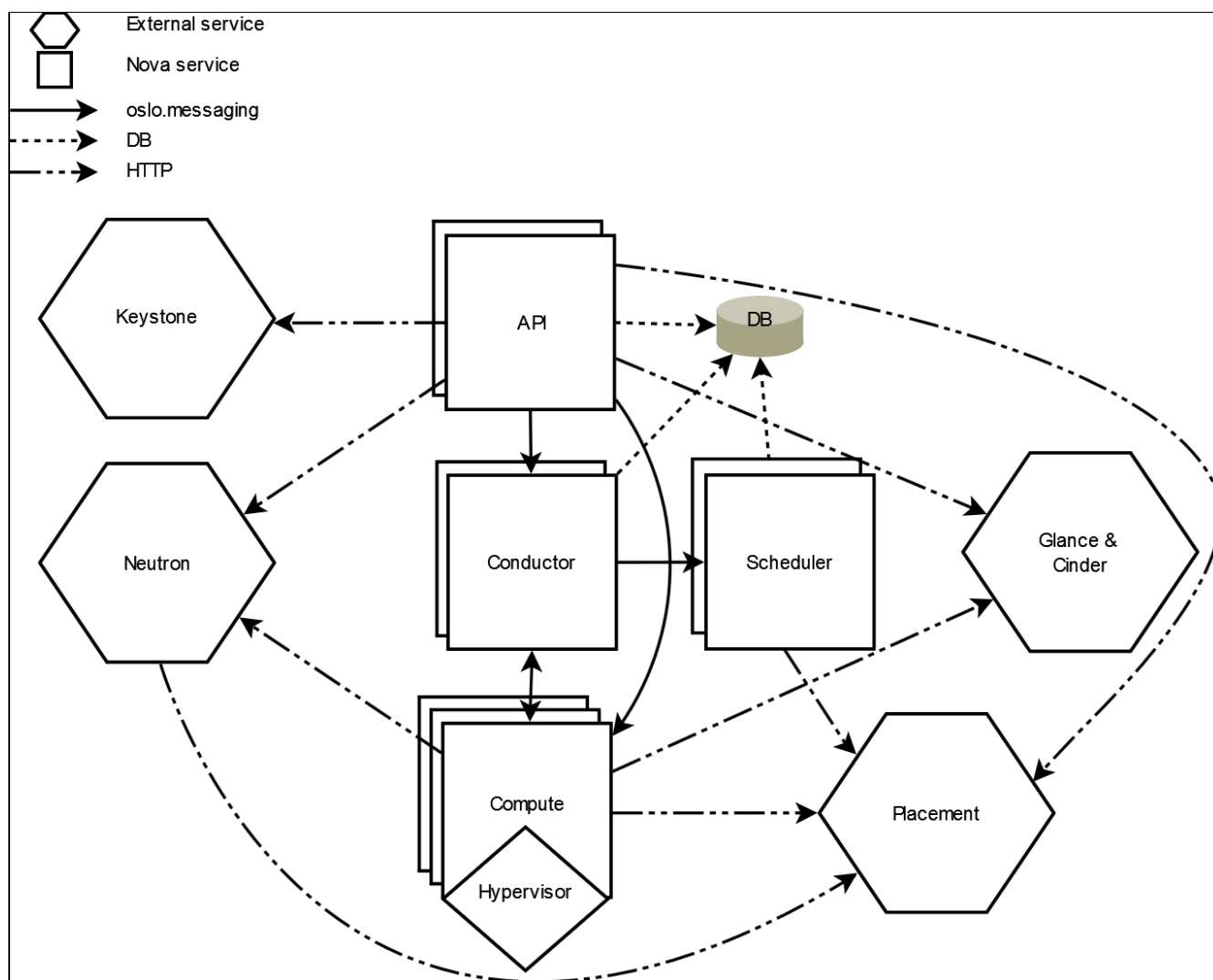
87 [https://www.linux-kvm.org/page/Main\\_Page](https://www.linux-kvm.org/page/Main_Page)

- **QEMU**<sup>88</sup> – это гипервизор Type2, который поддерживает запуск виртуальных машин с эмулированием различных аппаратных платформ. Поддерживает возможность аппаратного ускорения виртуализации при помощи модуля **KVM**<sup>89</sup>;
- **libvirt**<sup>90</sup> – 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.
- **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.

88 <https://www.qemu.org/>

89 [https://www.linux-kvm.org/page/Main\\_Page](https://www.linux-kvm.org/page/Main_Page)

90 <https://libvirt.org/>

- **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](#)<sup>91</sup> проекта.
  - a. [Подробное описание архитектуры](#)<sup>92</sup> сервиса.
  - b. [Информация о ячейках \(cells v2\)](#)<sup>93</sup>.
2. [Основной репозиторий](#)<sup>94</sup> проекта.
3. [Описание API](#)<sup>95</sup> проекта.

## Установка управляющих сервисов Nova

- [Настройка окружения](#) (см. стр. 67)
  - [Подготовка баз данных nova и nova\\_\\*](#) (см. стр. 67)
  - [Создание объектов Nova в Keystone](#) (см. стр. 68)
- [Установка сервисов Nova](#) (см. стр. 69)
  - [Первичная настройка конфигурации](#) (см. стр. 69)
  - [Финализация установки](#) (см. стр. 71)
  - [Проверка статуса сервиса](#) (см. стр. 71)
  - [Конфигурация для HAProxy](#) (см. стр. 72)

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:

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>

94 <https://opendev.org/openstack/nova>

95 <https://docs.openstack.org/api-ref/compute/>

```
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';
```

 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. 59).

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. 74)):

```
[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
```

**i** 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
```

**i** 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
```

**i** 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
```

**i** 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
```

**i** 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
```

**i** 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
```

**i** 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

**i** См. также: [Установка балансировщика нагрузки HAProxy \(см. стр. 16\)](#)

В референсной архитектуре доступ до сервисов Nova (кроме nova-повнспроху) 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 конфигурации перезагрузите 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
```

**i** По умолчанию этот пакет установит пакеты гипервизора QEMU и системы управления гипервизоров libvirt. Их отдельно устанавливать не требуется.

2. Примените конфигурационный файл, [описанный в разделе установки управляющей части \(см. стр. 69\)](#), по пути `/etc/nova/nova.conf`.

**i** На текущем шаге вычислительные узлы будут использовать стандартный драйвер (compute\_driver) libvirt.

3. Запустите сервис nova-compute и сервис libvirt 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
```

**i** 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](#)<sup>96</sup>.

Имя параметра	Описание параметра	Примечания
<b>[DEFAULT]</b>	Глобальные параметры сервисов Nova.	
<b>my_ip</b>	IP-адрес сервиса Nova API в mgmt-сети инфраструктуры.	Необходимо выбрать <b>один из двух</b> указанных параметров: <ul style="list-style-type: none"> <li><b>127.0.0.1</b>, если установка производится в рамках референсной архитектуры;</li> <li>Адрес в интерфейсе mgmt-сети, если производится обычная установка.</li> </ul>
<b>osapi_compute_listen</b>	Адрес прослушивания сервиса Nova API.	По умолчанию адрес для этого параметра берётся со значения параметра <code>my_ip</code> . <div style="border: 1px solid #ffc107; padding: 5px; margin-top: 10px;">  Для прослушивания всех интерфейсов можно указать адрес 0.0.0.0, однако такая конфигурация не рекомендуется к использованию в продуктовых инсталляциях. </div>

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

Имя параметра	Описание параметра	Примечания
<b>metadata_listen</b>	Адрес сервиса API метаданных Nova.	<p>По умолчанию адрес для этого параметра берётся со значения параметра <code>my_ip</code>.</p> <p>⚠ Для прослушивания всех интерфейсов можно указать адрес <code>0.0.0.0</code>, однако такая конфигурация не рекомендуется к использованию в продуктовых инсталляциях.</p>
<b>enabled_apis</b>	Включение доступных типов Nova API.	
<b>transport_url</b>	Адреса сервисов RabbitMQ для обмена сообщениями между сервисами OpenStack.	<p>Можно указать через запятую.</p> <p>⚠ Указание префикса <code>rabbit://</code> для каждого узла обязательно.</p>
<b>use_stderr</b> <b>log_dir</b> <b>use_journal</b>	Параметры журналирования сервиса.	<p>Необходимо выбрать <b>один из трёх</b> указанных параметров:</p> <ul style="list-style-type: none"> <li>• <b>use_stderr = True</b>, если установка производится в рамках референсной архитектуры, в этом случае журналы будут перенаправлены в виртуальное устройство <code>/dev/stderr</code>. Это позволяет получить логи системе управления контейнерами.</li> <li>• <b>log_dir = /var/log/keystone</b>, если производится обычная установка, в этом случае журналы будут сохраняться в файлах в указанном каталоге.</li> <li>• <b>use_journal = True</b>, если производится обычная установка, в этом случае журналы будут перенаправлены в службу управления журналами <a href="https://www.freedesktop.org/software/systemd/man/systemd-journald.service.html">systemd-journald</a>.<sup>97</sup></li> </ul> <p>⚠ Режим <code>use_syslog</code> для перенаправления журналов в сервис <code>syslog</code> переведён в разряд устаревших, поэтому официально не поддерживается.</p>
<b>allow_resize_to_same_host</b>	Включение возможности изменения размера инстанса, не меняя узел, где он запущен.	Этот параметр предлагается удалить для продуктивных установок.
<b>ssl_only</b>	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
<b>cert</b>	Указание pem-файла с сертификатом и ключом к нему.	<div style="border: 1px solid yellow; padding: 5px; margin-bottom: 5px;">  Убедитесь в том, что файл сертификата доступен для чтения от имени пользователя nova.         </div> <div style="border: 1px solid red; padding: 5px;">  Работа с самоподписанными сертификатами официально не поддерживается.         </div>
<b>[api_database]</b> <b>[database]</b>	Параметры соединения к базам данных Nova.	
<b>connection</b>	Адрес подключения к базе данных сервиса Nova.	
<b>connection_recycle_time</b>	Время для восстановления соединения до базы данных.	
<b>max_overflow</b>	Максимальное количество соединений при заполнении пула соединений.	
<b>max_pool_size</b>	Максимальный размер пула соединений к базе данных.	
<b>max_retries</b>	Максимальное количество попыток соединения до базы данных.	-1 выключает ограничение на количество попыток соединения.
<b>[api]</b>	Параметры работы API Nova.	
<b>auth_strategy</b>	Включение механизма аутентификации.	
<b>use_forwarded_for</b>	Включение заголовка X-Forwarder-For.	<div style="border: 1px solid yellow; padding: 5px;">  Необходимо для поддержки балансировщика нагрузки HAProxy.         </div>
<b>[keystone_authtoken]</b>	Параметры подключения к сервису Keystone	
<b>www_authenticate_uri</b>	Адрес публичного API Keystone	
<b>auth_url</b>	Адрес внутреннего API Keystone	
<b>memcached_servers</b>	Адреса внешней системы кэширования Memcached	Можно указать несколько серверов через запятую.
<b>auth_type</b>	Тип аутентификации.	

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

98 [https://docs.openstack.org/nova/victoria/configuration/config.html#glance.api\\_servers](https://docs.openstack.org/nova/victoria/configuration/config.html#glance.api_servers)

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

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

Имя параметра	Описание параметра	Примечания
[guestfs]	Параметры для <a href="#">libguestfs</a> <sup>99</sup> .	
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.

## OpenStack Neutron

### Информация о сервисе Neutron

- [Состав сервиса \(см. стр. 80\)](#)
- [Сетевые функции сервиса \(см. стр. 80\)](#)
- [Дополнительные материалы \(см. стр. 81\)](#)

[OpenStack Neutron](#)<sup>100</sup> – это комплекс сервисов, которые призваны решать вопросы управления сетевой инфраструктурой облачной платформы. Neutron позволяет создавать сети, прикреплять порты к экземплярам, 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](#)<sup>101</sup> или [Geneve](#)<sup>102</sup>.
- **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

99 <https://libguestfs.org>

100 <https://docs.openstack.org/neutron/victoria/>

101 [https://ru.wikipedia.org/wiki/Virtual\\_Extensible\\_LAN](https://ru.wikipedia.org/wiki/Virtual_Extensible_LAN)

102 [https://en.wikipedia.org/wiki/Generic\\_Network\\_Virtualization\\_Encapsulation](https://en.wikipedia.org/wiki/Generic_Network_Virtualization_Encapsulation)



для внешних сетей. Плавающий IP-адрес использует технологию трансляции сетевых адресов (NAT<sup>103</sup>).

- **Служба метаданных** (Metadata Service) – специальный сетевой сервис, который предоставляет инстансам виртуальной машины доступ к его метаданным.

## Дополнительные материалы

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

## 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](#)<sup>109</sup>, установленных в вычислительных узлах.

## Кратко про OVN

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

Общая схема работы сервиса выглядит так:

---

103 [https://en.wikipedia.org/wiki/Network\\_address\\_translation](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>

109 <https://www.openvswitch.org/>

110 <https://www.ovn.org/en/>

111 <https://ru.wikipedia.org/wiki/OpenFlow>

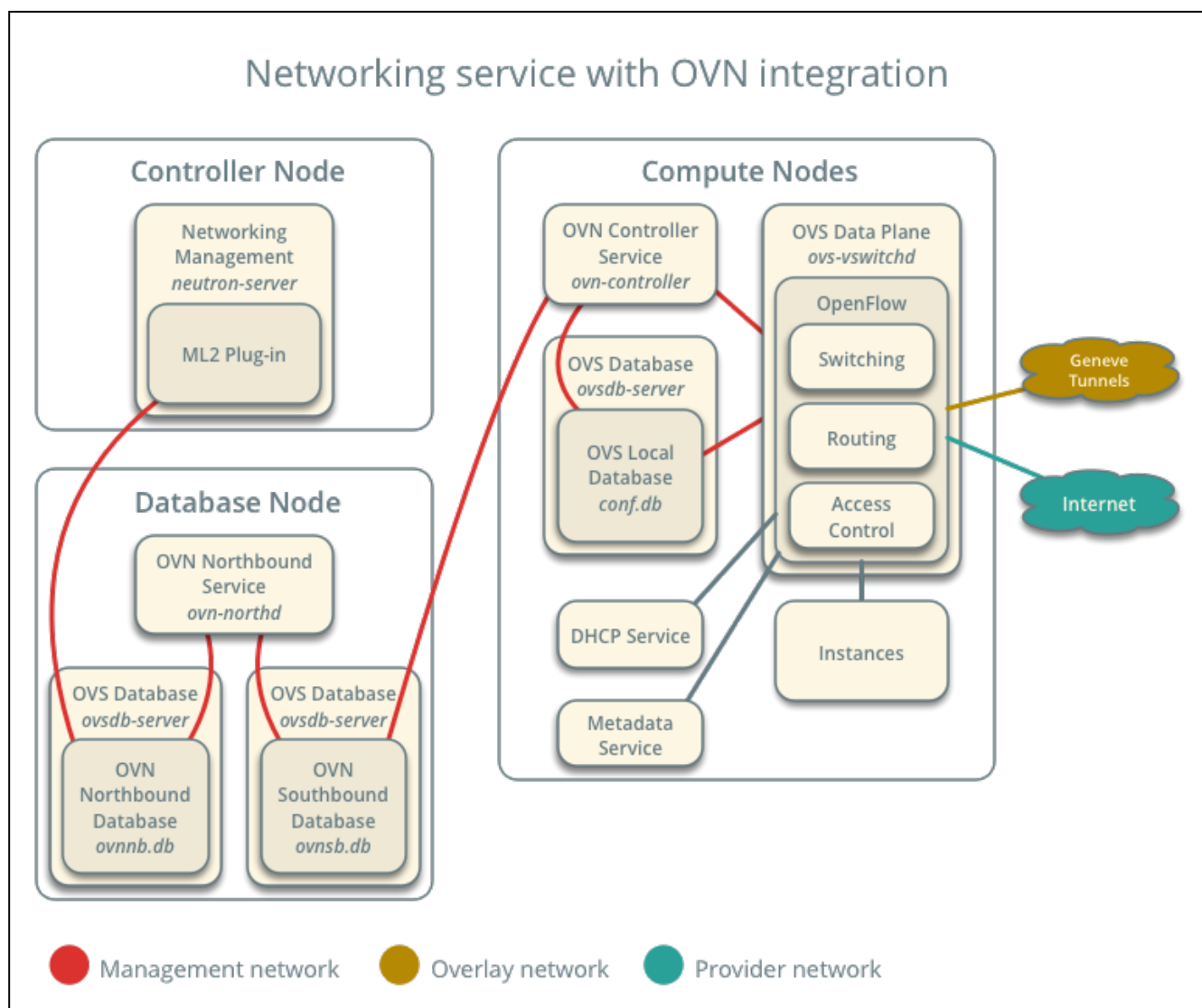


Схема работы сервиса

Комментарий по схеме:

- Neutron работает с OVN с помощью компонента ML2. Его настройка описана [здесь](#) (см. стр. 86).
- 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.
- 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 tcp:6641:0.0.0.0 -- \
set connection . inactivity_probe=60000
ovn-sbctl set-connection tcp: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

- [Настройка окружения](#) (см. стр. 84)
  - [Подготовка базы данных neutron](#) (см. стр. 84)
  - [Создание объектов в Keystone](#) (см. стр. 84)
- [Установка сервиса Neutron](#) (см. стр. 85)
  - [Установка сервиса Neutron API](#) (см. стр. 85)
  - [Настройка компонента ML2](#) (см. стр. 86)
    - [OVN](#) (см. стр. 86)
- [Финализация установки](#) (см. стр. 87)
- [Проверка работы сервиса](#) (см. стр. 88)
  - [Конфигурация для HAProxy](#) (см. стр. 88)

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:
  - а. 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
```

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

**i** 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](#)<sup>112</sup>:

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

**i** 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 \(см. стр. 89\)](#)):

```
[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
```

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

```

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_auth_token]
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](#)<sup>113</sup>.

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:

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

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

**i** 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 \(см. стр. 89\)](#)):

**⚠** Перед настройкой конфигурации убедитесь, что вы корректно настроили службу OVN и все его компоненты успешно запущены. Подробнее [здесь \(см. стр. 81\)](#).

```
[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:

```
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
```

**i** 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
```

**i** 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
```

**i** 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

**i** 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 \(см. стр. 83\)](#).

## 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](#)<sup>114</sup>.

Имя параметра	Описание	Примечание
<b>[DEFAULT]</b>	Глобальные параметры сервиса Neutron	
<b>debug</b>	Включение режима отладки для службы Neutron	Включайте только при проблемах с платформой.
<b>use_stderr</b>	Перенаправлять журналы в <code>/dev/stderr</code>	В референсной архитектуре необходимо для перенаправления журналов в систему управления контейнерами. При обычной установке нужно установить в <code>False</code> и указать параметр <code>log_dir = /var/log/neutron</code>
<b>bind_host</b>	Адрес прослушивания сервиса Neutron	В референсной архитектуре должен быть настроен на <code>loopback</code> . При обычной установке укажите адрес <code>0.0.0.0</code>
<b>api_paste_config</b>	Путь до конфигурации API	
<b>*_workers</b>	Количество процессов для каждого из подсервисов Neutron	
<b>core_plugin</b>	Основной плагин управления сетевой частью.	Всегда должен использоваться драйвер ML2.
<b>service_plugins</b>	Плагины для активации сетевых функций	Для референсной архитектуры нужно использовать L3 OVN Router. Для обычной установки можно использовать обычный L3 Router.

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

Имя параметра	Описание	Примечание
<b>allow_overlapping_ips</b>	Возможность использования одинаковых адресов и диапазона подсетей для виртуальных сетей.	
<b>transport_url</b>	Адреса сервисов RabbitMQ для обмена сообщениями между сервисов OpenStack.	Можно указать через запятую, указание префикса rabbit:// для каждого узла обязателен.
<b>auth_strategy</b>	Включение механизма аутентификации.	
<b>notify_nova_on_port_status_changes</b>	Включение оповещения сервиса Nova об изменении статуса сетевого порта.	
<b>notify_nova_on_port_data_changes</b>	Включение оповещения сервиса Nova об изменении данных сетевого порта (напр. фиксированного IP-адреса).	
<b>[database]</b>	Параметры соединения к базам данных Neutron.	
<b>connection</b>	Адрес подключения к базе данных neutron.	
<b>connection_recycle_time</b>	Время для восстановления соединения до базы данных.	
<b>max_pool_size</b>	Максимальный размер пула соединений к базе данных.	
<b>max_retries</b>	Максимальное количество попыток соединения до базы данных.	-1 выключает ограничение на количество попыток соединения
<b>[keystone_authtoken]</b>	Параметры подключения к сервису Keystone.	
<b>www_authenticate_uri</b>	Адрес публичного API Keystone.	
<b>auth_url</b>	Адрес внутреннего API Keystone.	
<b>memcached_servers</b>	Адреса внешней системы кэширования Memcached.	Можно указать несколько серверов через запятую.
<b>auth_type</b>	Тип аутентификации	
<b>project_domain_name</b> <b>user_domain_name</b> <b>project_name</b> <b>username</b> <b>password</b>	Данные для аутентификации в Keystone от имени пользователя neutron	

Имя параметра	Описание	Примечание
[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 указать 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](#)<sup>115</sup> – 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](#)<sup>116</sup>;
- **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 образам 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):

<sup>115</sup> <https://docs.openstack.org/cinder/victoria/>

<sup>116</sup> <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';
```

**i** Вместо CINDER\_DBPASS используйте свой пароль, он будет необходим далее.

4. Выйдите из сессии СУБД:

```
exit;
```

## Создание объектов в Keystone

**i** 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
```

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

## Установка сервиса Cinder

**i** См. также: [Настройка репозитория Almalinux](#) (см. стр. 11).

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

1. Установите основной пакет сервиса Cinder:

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

**i** 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. 99)):

```
[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_auth]
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](#)<sup>117</sup>.

Параметр	Значение
<b>my_ip</b>	Необходимо указать адрес узла в интерфейсе сети управления. Не используйте адрес DNS.
<b>osapi_volume_listen</b>	Необходимо указать в зависимости от типа установки: <ul style="list-style-type: none"> <li>Классическая архитектура: \$my_ip</li> <li>Референсная архитектура: 0.0.0.0</li> </ul>
<b>transport_url</b>	Укажите данные RabbitMQ, в частности: <ul style="list-style-type: none"> <li>имя пользователя в RabbitMQ (в примере: openstack)</li> <li>пароль пользователя в RabbitMQ вместо RABBIT_PASS</li> <li>адрес до сервиса RabbitMQ (в примере: controller)</li> </ul>
<b>glance_api_servers</b>	Адреса до сервисов Glance.
<b>Параметры журналирования</b>	Эти параметры зависят от типа установки: <ul style="list-style-type: none"> <li>Классическая архитектура: нужно использовать предложенный в конфигурации параметр <b>log_dir</b> или <b>use_journal = True</b></li> <li>Референсная архитектура: нужно использовать параметр <b>use_stderr = True</b></li> </ul> Параметр debug можно указать во всех типах установки.
<b>database/connection</b>	Укажите данные доступа к базе данных, в частности, <ul style="list-style-type: none"> <li>имя пользователя в MariaDB (в примере: cinder)</li> <li>пароль пользователя в MariaDB вместо CINDER_DBPASS</li> <li>адрес сервиса MariaDB (в примере: controller)</li> </ul>
<b>keystone_auth_token</b>	Укажите корректные адреса до сервиса Keystone, в частности, пароль для пользователя cinder вместо CINDER_PASS. <ul style="list-style-type: none"> <li>Отдельно корректно укажите адреса серверов memcached в memcached_servers.</li> </ul>
<b>nova</b>	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. 99).

## 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

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

```

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
```

**i** 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
```

**i** 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
```

**i** 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

**i** 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. 94).
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 создавалось 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](#)<sup>118</sup>.

Имя параметра	Описание	Примечания
<b>[DEFAULT]</b>	Глобальные настройки сервиса Cinder.	
<b>debug</b>	Включение режима отладки.	Используйте только в режиме обслуживания или тестирования.

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

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

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

## OpenStack Ceilometer

### Информация о сервисе Ceilometer

- [Обзор сервиса](#) (см. стр. 102)
  - [Основные компоненты](#) (см. стр. 102)
- [Дополнительные материалы](#) (см. стр. 102)

## Обзор сервиса

[OpenStack Telemetry \(Ceilometer\)](#)<sup>119</sup> – 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](#)<sup>120</sup>. Проект Gnocchi позволяет хранить собираемые измерения в агрегированном виде и предоставляет API для работы с ними.

### 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](#)<sup>121</sup>.

### Дополнительные материалы

1. [Официальная документация](#)<sup>122</sup> проекта.
2. [Официальный репозиторий](#)<sup>123</sup> проекта.
3. [Ceilometer CLI](#)<sup>124</sup>.

## 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;
```

119 <https://docs.openstack.org/ceilometer/victoria/>

120 <https://gnocchi.osci.io>

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>

## Создание объектов в Keystone

**i** 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 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
```

**i** 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.

**i** 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

**i** См. также: [Настройка репозитория Almalinux \(см. стр. 11\)](#).

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
```

**i** Стандартные пути файлов конфигурации:

- `/etc/gnocchi` – Каталог конфигурации Glance;
- `/etc/gnocchi/gnocchi.conf` – основной 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. 57)):

```
[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):
- Адрес прослушивания сервиса Gnocchi API, которые зависят от метода установки (добавить в конфиг);
  - Параметры [журналирования сервиса](#) (см. стр. 57), которые зависят от метода установки (добавить в конфиг);
  - Параметры подключения к СУБД в **database - connection**, в частности, пароль к БД glance вместо GNOCCHI\_DBPASS;
  - Параметры подключения к Keystone в **keystone\_authtoken**, в частности, пароль пользователя gnocchi вместо GNOCCHI\_PASS;
  - 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
```

- i** 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

- i** 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 \(см. стр. 105\)](#)
- [Финализация установки \(см. стр. 106\)](#)

### 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. 110).

4. Обновите конфигурацию сервиса:

```
ceilometer-upgrade
```

## Финализация установки

1. Запустите сервис Ceilometer и добавьте 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. 92).

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,<sup>125</sup> 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`:

<sup>125</sup> <http://www.nncron.ru/nncronIt/help/RU/working/cron-format.htm>

```
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
```

- 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](#)<sup>126</sup>.
3. Перезапустите службу Nova:

<sup>126</sup> [https://github.com/openstack/pycadf/blob/master/etc/pycadf/nova\\_api\\_audit\\_map.conf](https://github.com/openstack/pycadf/blob/master/etc/pycadf/nova_api_audit_map.conf)

```
systemctl restart openstack-nova-api.service
```

## Сбор метрик Neutron

### ✓ Примечание

Необходимо предварительно установить и настроить службу [Neutron](#) (см. стр. 80).

### ✓ 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 хранятся в оперативной памяти в виде базы данных в возможности 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:

```
# 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 \(см. стр. 109\)](#)
- [Настройка сбора IPMI метрик \(см. стр. 110\)](#)
- [Финализация установки \(см. стр. 110\)](#)

### 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 \(см. стр. 110\)](#).

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 добавьте его в автозапуск:

```
systemctl enable openstack-ceilometer-ipmi.service
systemctl start openstack-ceilometer-ipmi.service
```

## Финализация установки

1. Запустите сервис Ceilometer и добавьте 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. Эта страница содержит подробное описание настроек этой конфигурации.

При изменении конфигурации необходимо перезапустить веб-сервер Apache:

```
systemctl restart httpd
```

## Таблица конфигурации

### Сервис Ceilometer

**i** Путь до конфигурации: `/etc/ceilometer/ceilometer.conf`.

**i** Легенда таблицы доступна [на этой странице](#)<sup>127</sup>.

Имя параметра	Описание	Примечания
<b>[DEFAULT]</b>	Глобальные переменные сервиса.	
<b>transport_url</b>	Список адресов до серверов RabbitMQ.	Серверы указываются через запятую с указанием префикса в каждом элементе.
<b>auth_type</b>	Тип аутентификации.	Всегда должен быть равен параметру "password".
<b>auth_url</b>	Адрес до сервиса Keystone.	
<b>memcached_servers</b>	Адреса до сервисов кэширования memcached.	
<b>project_domain_name</b> <b>user_domain_name</b> <b>project_name</b> <b>username</b> <b>password</b>	Данные аутентификации в сервис Keystone от имени пользователя ceilometer в проекте service.	
<b>interface</b>	Тип точки доступа в сервисе Keystone.	
<b>region_name</b>	Имя региона, которое будет использоваться для точек доступа.	
<b>[oslo_messaging_rabbit]</b>	Параметры RabbitMQ.	
<b>rabbit_retry_interval</b>	Интервал подключения к RabbitMQ.	
<b>rabbit_retry_backoff</b>	Интервал между повторными попытками подключения к RabbitMQ.	

<sup>127</sup> <https://conf.tionix.ru/pages/viewpage.action?pageId=205881354#id-Условныеобозначения-table-format-desc>

# 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](#)<sup>128</sup> (`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](#)<sup>129</sup>. Для его поддержки нужно установить библиотеку Raven:

```
dnf -y install python3-raven
```

<sup>128</sup> <https://www.rabbitmq.com/vhosts.html>

<sup>129</sup> <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](#)<sup>130</sup>, 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](#)<sup>131</sup> и [Fibre Channel](#)<sup>132</sup>.

### 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 \(см. стр. 112\)](#).

### 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:

```
create database tionix;
```

<sup>130</sup><https://conf.tionix.ru/pages/viewpage.action?pageId=164102237#id->

Использование связанных доменов систем хранения данных пользователей-tnx-ldap-desc

<sup>131</sup> <https://ru.wikipedia.org/wiki/ISCSI>

<sup>132</sup> [https://ru.wikipedia.org/wiki/Fibre\\_Channel](https://ru.wikipedia.org/wiki/Fibre_Channel)

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 `/etc/tionix/tionix.yaml` ([описание](#) (см. стр. 173)):

```
# 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: '/'

BLOCKED_ROLES: ['vdi-user']

```

3. При конфигурировании обратите особое внимание на следующие настройки:

 Легенда таблицы доступна [на этой странице](#)<sup>133</sup>.

Параметр	Значение
<b>JOURNAL_API_LISTEN</b>	Укажите слушаемый адрес сервиса Journal API в зависимости от типа установки: <ul style="list-style-type: none"> <li>классическая архитектура: адрес интерфейса в сети управления (mgmt);</li> <li>референсная архитектура: O.O.O.O.</li> </ul>
<b>Параметры LOGFILE</b>	Укажите пути файлов журналирования в зависимости от типа установки: <ul style="list-style-type: none"> <li>классическая архитектура: пути, предложенные к конфигурации;</li> <li>референсная архитектура: укажите <b>/dev/stdout</b> или <b>/dev/stderr</b> во всех параметрах.</li> </ul>
<b>KEYSTONE</b>	Укажите данные подключения к сервису Keystone, в частности: <ul style="list-style-type: none"> <li>пароль пользователя admin в сервисе Keystone вместо ADMIN_PASS;</li> <li>пароль пользователя tionix в сервисе Keystone вместо TIONIX_PASS;</li> <li>отдельно укажите адреса серверов memcached в <b>memcached_servers</b>.</li> </ul>
<b>DB</b>	Укажите параметры подключения к СУБД MariaDB, в частности: <ul style="list-style-type: none"> <li>пароль для пользователя tionix для базы данных tionix вместо TIONIX_CLIENT_DBPASS.</li> </ul>
<b>RABBIT_QUEUES</b>	Укажите параметры подключения к сервису RabbitMQ, в частности: <ul style="list-style-type: none"> <li>пароль к пользователю tionix в RabbitMQ вместо TIONIX_RABBIT_PASS;</li> <li>включение режима durable для сохранения состояния очередей при перезапуске RabbitMQ.</li> </ul>

 Все параметры предложенной конфигурации доступны на этой [странице](#) (см. стр. 173).

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

- Создайте каталоги для журналирования Journal API:

```
mkdir -p /var/log/tionix/journal
chown tionix:tionix /var/log/tionix/journal
```

- Выполните первичную инициализацию модуля:

```
openstack tnx configure -n tnx_client
```

- 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.

- Создайте сервис Journal API в Keystone:

```
openstack service create --name tnx-journal --description "TIONIX Journal Service"
tnx-journal
```

- Создайте точки входа (endpoint):

- публичную:

```
openstack endpoint create --region RegionOne tnx-journal public http://
controller:9360
```

- внутреннюю:

```
openstack endpoint create --region RegionOne tnx-journal internal http://
controller:9360
```

- административную:

```
openstack endpoint create --region RegionOne tnx-journal admin http://
controller:9360
```


- 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

- 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
```

**i** 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. 117)

## 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.

**i** Не следует путать сервис Journal в составе Client и [сервис Journal](#)<sup>134</sup> в составе [Systemd](#)<sup>135</sup>, 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 \(notifications\)](#)<sup>136</sup> о событиях через RabbitMQ:

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

4. Перезапустите службы OpenStack Nova в управляющем и вычислительном узлах:

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

<sup>134</sup> <https://www.freedesktop.org/software/systemd/man/systemd-journald.service.html>

<sup>135</sup> <https://ru.wikipedia.org/wiki/Systemd>

<sup>136</sup> <https://docs.openstack.org/nova/victoria/reference/notifications.html>

## 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](#)<sup>137</sup> в формате [CADF](#)<sup>138</sup>.

```
[DEFAULT]
notification_format = cadf
```

2. Там же включите [возможность отправки уведомлений \(notifications\)](#)<sup>139</sup> о событиях через RabbitMQ:

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

**i** 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
```

## Настройка драйвера LDAP

Client содержит [модифицированный драйвер](#) (см. стр. 113) 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](#)<sup>140</sup>, которые расположены в каталоге `/etc/keystone/domains`, необходимо выключить [пулинг подключений при аутентификации](#)<sup>141</sup> 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

<sup>137</sup> [https://docs.openstack.org/keystone/victoria/advanced-topics/event\\_notifications.html](https://docs.openstack.org/keystone/victoria/advanced-topics/event_notifications.html)

<sup>138</sup> <https://www.dmtf.org/standards/cadf>

<sup>139</sup> <https://docs.openstack.org/nova/victoria/reference/notifications.html>

<sup>140</sup> <https://conf.tionix.ru/pages/viewpage.action?pageId=164102237#id->

Использование связанных доменов систем хранения данных пользователей-ks-domain-specific-drivers

<sup>141</sup> <https://docs.oracle.com/javase/jndi/tutorial/ldap/connect/pool.html>

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](#)<sup>142</sup>.

Более подробно обо всех этих функциях будет рассказано в "Управлении сервисом NodeControl". Этот раздел будет содержать установку и первичный запуск сервиса.

## 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 \(см. стр. 112\)](#).

### 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:

<sup>142</sup> [https://en.wikipedia.org/wiki/Disaster\\_recovery](https://en.wikipedia.org/wiki/Disaster_recovery)



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

- Создайте базу данных `tionix`:

```
create database tionix_node_control;
```

- 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';
```

- Выйдите из сессии СУБД

```
exit;
```

## Установка модуля

### Процесс установки

- Установите пакет модуля и пакет `net-tools`:

```
dnf -y install python3-tionix_node_control net-tools
```

**i** 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.

- Добавьте эту конфигурацию в основной файл конфигурации сервиса NodeControl ([описание \(см. стр. 122\)](#)):

```
# 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. 122). 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
```

**i** 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
```

**i** 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. 119) 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](#)<sup>143</sup>.

### Немного о формате yaml

Файл формата [yaml](#)<sup>144</sup> критичен к отступам текста<sup>145</sup> 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
<b>DB</b>	Параметры подключения к базе данных.	
<b>DB → NAME</b>	Имя базы данных сервиса NodeControl.	Остальные параметры подключения к базе данных берутся с основного файла конфигурации TIONIX.
<b>PXE</b>	Параметры управления сетевым сервисом PXE <sup>146</sup> .	
<b>PXE → conf_dir</b>	Путь до файлов ядра Linux для запуска ОС через протокол PXE и доступные через протоколы TFTP или HTTP.	
<b>SETTINGS_TRACKER</b>	Параметры, определяющие параметры отслеживания состояния гипервизора.	
<b>SETTINGS_TRACKER → mutex</b>	Количество попыток определения статуса гипервизора при переходе в статус <i>down</i> перед запуском обработчика.	
<b>SETTINGS_TRACKER mutex_up</b>	→ Количество попыток, при котором возвращается статус <i>up</i> , перед запуском обработчика.	
<b>SETTINGS_TRACKER loop_time</b>	→ Интервал времени между проверками.	

<sup>143</sup> <https://conf.tionix.ru/pages/viewpage.action?pagelId=205881354#id-Условныеобозначения-table-format-desc>

<sup>144</sup> <https://yaml.org/>

<sup>145</sup> <https://yaml.org/spec/1.2.2/#61-indentation-spaces>

<sup>146</sup> [https://en.wikipedia.org/wiki/Preboot\\_Execution\\_Environment](https://en.wikipedia.org/wiki/Preboot_Execution_Environment)

Имя параметра	Описание	Примечания
<b>DHCP_LEASES_FILEPATHS</b>	Список файлов <code>.leases</code> DHCP-сервера, содержащие данные о выделенных IP-адресах и времени их аренды.	Внутри параметра должен быть список с отступом в 4 пробела, формат которого принят в формате <code>yaml</code> , например: <pre> - '/var/lib/dhcp/dhcpd/state/dhcpd.leases' - '/var/lib/dhcp/dhcpd.leases'</pre>
<b>SYNC_NOVA_NODES_TIME</b>	Интервал обновления статуса вычислительных узлов	Указывается в секундах.
<b>DB_CONNECTION_MAX_RETRIES</b>	Максимальное количество попыток соединения к базе данных.	Этот параметр будет использоваться только для базы данных, указанных в параметре <code>DB</code> этой конфигурации.
<b>TIMEOUT_RESERV_NODE_UP</b>	Время ожидания, при котором резервный узел должен получить статус <code>up</code> .	
<b>MAX_TICK_COUNT</b>	Максимальное количество попыток проверить статус вычислительного узла после выключения питания гипервизора и последующей автоэвакуации узла в платформе.	
<b>SLEEP_TIME</b>	Интервал между проверками статуса вычислительного узла после выключения питания.	Указывается в секундах.
<b>HOST_RESTART_TIMEOUT</b>	Время ожидания, при котором вычислительный узел должен перезапуститься.	Указывается в секундах.
<b>MAX_DOWN_HOSTS</b>	Максимальное количество вычислительных узлов, которые могут иметь статус <code>down</code> .	Если параметр был превышен, то выключается функция автоэвакуации.
<b>ALLOW_HOST_AUTO_POWER_OFF</b>	Разрешение на автоматический перезапуск вычислительного узла после получения им статуса <code>down</code> .	
<b>HOST_RESTART_TIMEOUT</b>	Время ожидания на перезапуск вычислительного узла.	Указывается в секундах.

Имя параметра	Описание	Примечания
<b>HOST_ATTACH_RETRY_DELAY</b>	Интервал между попытками добавления вычислительного узла к виртуальному контроллеру OpenStack.	Указывается в секундах.
<b>HOST_ATTACH_MAX_RETRIES</b>	Максимальное количество попыток добавления вычислительного узла к виртуальному контроллеру OpenStack.	
<b>HOST_ATTACH_NETWORK_TAG</b>	Тэг для фильтрации сетей при выборе сетевого интерфейса виртуального контроллера.	
<b>KEY_PATH</b>	Путь до файла с ключом, необходимый для входа в виртуальный контроллер OpenStack.	Используется для входа через протокол SSH.
<b>CONTROLLER_AUTH_PATH</b>	Путь до файла с параметрами окружения, содержащие данные аутентификации для виртуального контроллера OpenStack.	
<b>CONTROLLER_USERNAME</b>	Имя пользователя, необходимое для входа в окружение виртуального контроллера OpenStack.	Используется для входа через протокол SSH.
<b>SENTRY</b>	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.
<b>SENTRY → ENABLED</b>	Включение отправки данных мониторинга и сообщений об ошибках удаленному серверу Sentry.	
<b>SENTRY → LOG_LEVEL</b>	Уровень логирования событий для отправки.	
<b>SENTRY → DSN</b>	Название источника данных (DSN <sup>147</sup> ), зарегистрированный в удаленном сервере Sentry.	DSN оформляется по следующей схеме: <pre>http(s):// public_key:secret_key@domain/ project_id</pre>

<sup>147</sup> <https://docs.sentry.io/product/sentry-basics/dsn-explainer/>

Имя параметра	Описание	Примечания
<b>ENABLE_CEPH_INTEGRATION</b>	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.
<b>ENABLE_NETWORK_ISOLATION</b>	Включение изоляции сетей, предназначенные для виртуальных контроллеров OpenStack.	
<b>NETWORK_ISOLATION_API_HOST</b>	Адрес API сервиса, реализующий изоляцию сети.	
<b>NETWORK_ISOLATION_API_PORT</b>	Порт API сервиса, реализующий изоляцию сети.	
<b>RETRIES_WAIT_FOR_VM_STATUS</b>	Максимальное количество попыток опроса состояния гипервизора.	
<b>RETRIES_WAIT_FOR_NODE_STATE</b>	Максимальное количество попыток опроса состояния виртуальной машины.	
<b>ALLOW_EVACUATE_HOST</b>	Разрешение на эвакуацию виртуальных машин из недоступного вычислительного узла.	
<b>NODE_CONTROL_API_LISTEN</b>	Адрес прослушивания для API сервиса NodeControl.	По умолчанию используется адрес 0.0.0.0. Рекомендуется использование только адреса сети управления.
<b>NODE_CONTROL_API_LISTEN_PORT</b>	Порт прослушивания для API сервиса NodeControl.	
<b>NODE_CONTROL_API_AUDIT_ENABLED</b>	Включение журналирования всех запросов, которые выполняются для NodeControl API.	

Имя параметра	Описание	Примечания
<b>NODE_CONTROL_API_LOGFILE</b>	Путь до файла журналов для запросов к NodeControl API. Используется сервисом <b>tionix-node-control-api</b> .	<p>Необходимо выбрать <b>один из двух</b> указанных параметров:</p> <ul style="list-style-type: none"> <li>• <b>/dev/stdout</b>, если установка производится в рамках <i>референсной архитектуры</i>, в этом случае журналы будут перенаправлены в виртуальное устройство /dev/stdout. Это позволит получить журналы системой управления контейнерами;</li> <li>• <b>путь до файла</b>, если производится <i>обычная установка</i>, в этом случае журналы будут сохранены в указанном по пути файле.</li> </ul> <p>Нативная поддержка сервиса Journald на данный момент отсутствует.</p>
<b>NODE_CONTROL_NODE_SYNCER_LOGFILE</b>	Путь до файла журналов сервиса по синхронизации данных состояния вычислительных узлов. Используется сервисом <b>tionix-node-control-node-syncer</b> .	<p>Необходимо выбрать <b>один из двух</b> указанных параметров:</p> <ul style="list-style-type: none"> <li>• <b>/dev/stdout</b>, если установка производится в рамках <i>референсной архитектуры</i>, в этом случае журналы будут перенаправлены в виртуальное устройство /dev/stdout. Это позволит получить журналы системой управления контейнерами;</li> <li>• <b>путь до файла</b>, если производится <i>обычная установка</i>, в этом случае журналы будут сохранены в указанном по пути файле.</li> </ul> <p>Нативная поддержка сервиса Journald на данный момент отсутствует.</p>
<b>NODE_CONTROL_NODE_TRACKER_LOGFILE</b>	Путь до файла журналов сервиса по отслеживанию задач для вычислительных узлов. Используется сервисом <b>tionix-node-control-node-tracker</b> .	<p>Необходимо выбрать <b>один из двух</b> указанных параметров:</p> <ul style="list-style-type: none"> <li>• <b>/dev/stdout</b>, если установка производится в рамках <i>референсной архитектуры</i>, в этом случае журналы будут перенаправлены в виртуальное устройство /dev/stdout. Это позволит получить журналы системой управления контейнерами;</li> <li>• <b>путь до файла</b>, если производится <i>обычная установка</i>, в этом случае журналы будут сохранены в указанном по пути файле.</li> </ul> <p>Нативная поддержка сервиса Journald на данный момент отсутствует.</p>

Имя параметра	Описание	Примечания
<b>NODE_CONTROL_WORKER_LOGFILE</b>	Путь до файла журналов сервиса, выполняемые основные задачи сервиса NodeControl. Используется сервисом <b>tionix-node-control-worker</b> .	<p>Необходимо выбрать <b>один из двух</b> указанных параметров:</p> <ul style="list-style-type: none"> <li>• <b>/dev/stdout</b>, если установка производится в рамках <i>референсной архитектуры</i>, в этом случае журналы будут перенаправлены в виртуальное устройство /dev/stdout. Это позволит получить журналы системой управления контейнерами;</li> <li>• <b>путь до файла</b>, если производится <i>обычная установка</i>, в этом случае журналы будут сохранены в указанном по пути файле.</li> </ul> <p>Нативная поддержка сервиса Journald на данный момент отсутствует.</p>
<b>NODE_CONTROL_NOVA_LISTENER_LOGFILE</b>	Путь до файла журналов сервиса, отслеживающие задачи над виртуальными машинами. Используется сервисом <b>tionix-node-control-nova-listener</b> .	<p>Необходимо выбрать <b>один из двух</b> указанных параметров:</p> <ul style="list-style-type: none"> <li>• <b>/dev/stdout</b>, если установка производится в рамках <i>референсной архитектуры</i>, в этом случае журналы будут перенаправлены в виртуальное устройство /dev/stdout. Это позволит получить журналы системой управления контейнерами;</li> <li>• <b>путь до файла</b>, если производится <i>обычная установка</i>, в этом случае журналы будут сохранены в указанном по пути файле.</li> </ul> <p>Нативная поддержка сервиса Journald на данный момент отсутствует.</p>

## Дополнительные параметры

Эти параметры не включены в предлагаемую конфигурацию, однако при необходимости их можно добавить.

Имя параметра	Описание	Примечания
<b>LOG_LEVEL</b>	Уровень детализации для всех типов журналов. По умолчанию: <b>INFO</b> .	
<b>ENABLE_AGENT</b>	Включение функций управления вычислительным узлом, предоставляемые модулем Tionix Agent. По умолчанию: <b>False</b> .	
<b>RABBIT_QUEUES</b>	Указание имени виртуального хоста в сервисе RabbitMQ. По умолчанию: <b>"vhost: tionix"</b> .	



Имя параметра	Описание	Примечания
<b>NODE_CONTROL_AGENT_LOGFILE</b>	Путь до файла журналов, регистрирующий взаимодействие NodeControl с агентом TIONIX: <ul style="list-style-type: none"> <li>используется сервисом <b>tionix-node-control-agent</b></li> <li>зависит от ENABLE_AGENT</li> </ul>	
<b>NODE_CONTROL_DRS_TRIGGER_LOGFILE</b>	Путь до файла журналов, регистрирующий срабатывание триггером системе восстановления после катастрофической ситуации ( <a href="#">DRS<sup>148</sup></a> ): <ul style="list-style-type: none"> <li>используется сервисом <b>tionix-node-control-drs-trigger</b>;</li> <li>зависит от: <b>DRS</b>.</li> </ul>	
<b>DRS</b>	Параметры подключения к системе восстановления после катастрофической ситуации ( <a href="#">DRS<sup>149</sup></a> ).	
<b>DRS → DRS_HOSTNAME</b>	Адрес системы восстановления после катастрофической ситуации ( <a href="#">DRS<sup>150</sup></a> ): <ul style="list-style-type: none"> <li>зависит от: <b>DRS</b>.</li> </ul>	
<b>DRS → DRS_PORT</b>	Порт системы восстановления после катастрофической ситуации ( <a href="#">DRS<sup>151</sup></a> ): <ul style="list-style-type: none"> <li>по умолчанию: 80;</li> <li>зависит от: <b>DRS</b>.</li> </ul>	
<b>DRS → DRS_USER</b>	Имя пользователя к системе восстановления после катастрофической ситуации ( <a href="#">DRS<sup>152</sup></a> ): <ul style="list-style-type: none"> <li>зависит от: <b>DRS</b>.</li> </ul>	
<b>DRS → DRS_PASSWORD</b>	Пароль для пользователя в системе восстановления после катастрофической ситуации ( <a href="#">DRS<sup>153</sup></a> ): <ul style="list-style-type: none"> <li>зависит от: <b>DRS</b>.</li> </ul>	
<b>STORAGE_SYNC_INTERVAL</b>	Интервал между запросами на получение данных о дисках в сервисе Cinder. По умолчанию: 60.	Указывается в секундах.
<b>RECOVERY_PRIORITY</b>	Указывает уровень приоритета узлов при автоэвакуации виртуальных машин.	Выбирается целое число в диапазоне от 1 до 10.
<b>PRIORITIZED_EVACUATION_TIMEOUT</b>	Время ожидания автоэвакуации между различными группами приоритизации автоэвакуации.	

148 [https://en.wikipedia.org/wiki/Disaster\\_recovery](https://en.wikipedia.org/wiki/Disaster_recovery)

149 [https://en.wikipedia.org/wiki/Disaster\\_recovery](https://en.wikipedia.org/wiki/Disaster_recovery)

150 [https://en.wikipedia.org/wiki/Disaster\\_recovery](https://en.wikipedia.org/wiki/Disaster_recovery)

151 [https://en.wikipedia.org/wiki/Disaster\\_recovery](https://en.wikipedia.org/wiki/Disaster_recovery)

152 [https://en.wikipedia.org/wiki/Disaster\\_recovery](https://en.wikipedia.org/wiki/Disaster_recovery)

153 [https://en.wikipedia.org/wiki/Disaster\\_recovery](https://en.wikipedia.org/wiki/Disaster_recovery)

Имя параметра	Описание	Примечания
<b>EXTRA_AVAILABILITY_CHECK</b>	Параметры для дополнительной проверки доступности вычислительных узлов.	Для этой проверки используется общее хранилище, куда записываются данные проверки по доступности вычислительного узла.
<b>EXTRA_AVAILABILITY_CHECK</b> → <b>CHECK_ENABLED</b>	Включение дополнительной проверки: <ul style="list-style-type: none"> <li>по умолчанию: <b>False</b>;</li> <li>зависит от: <b>EXTRA_AVAILABILITY_CHECK</b>.</li> </ul>	
<b>EXTRA_AVAILABILITY_CHECK</b> → <b>DELAY</b>	Интервал между попытками доступа к файлу, содержащий информацию о состоянии вычислительных узлов: <ul style="list-style-type: none"> <li>по умолчанию: 60;</li> <li>зависит от: <b>CHECK_ENABLED</b>.</li> </ul>	Указывается в секундах.
<b>EXTRA_AVAILABILITY_CHECK</b> → <b>ATTEMPTS</b>	Количество попыток доступа к файлу, содержащий информацию о состоянии вычислительных узлов: <ul style="list-style-type: none"> <li>по умолчанию: 2;</li> <li>зависит от: <b>CHECK_ENABLED</b>.</li> </ul>	
<b>EXTRA_AVAILABILITY_CHECK</b> → <b>INSTANCE_RATE</b>	Количество виртуальных машин (инстансов), которые должны быть активны в вычислительном узле: <ul style="list-style-type: none"> <li>по умолчанию: 100;</li> <li>зависит от: <b>CHECK_ENABLED</b>.</li> </ul>	Указывается в процентах от 0 до 100.

## Отладка

Для максимального подробного журналирования ошибок при выполнении работы сервиса можно включить поддержку трассировок:

Имя параметра	Описание	Примечания
<b>TRACEBACK_ENABLED</b>	Включение трассировки ошибок в файлах журналов. По умолчанию: <b>False</b> .	

## 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 (YY)
- Agent (BY)

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](#)<sup>154</sup> 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;
- Если не были считаны данные ни с одного хранилища доступности, то выполняется задержка при повторном считывании 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
```

<sup>154</sup> <https://docs.tionix.ru/3.0.52/glossary/index.html#term-5>

## 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 `unassign` 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 спросит 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 получаютс 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 спросит 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 получаютс 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](#)<sup>155</sup>.

### 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 \(см. стр. 112\)](#).

### Подготовка базы данных tionix\_scheduler

 См. также: [Установка и настройка СУБД MariaDB \(см. стр. 22\)](#)

<sup>155</sup> <https://docs.celeryproject.org/en/stable/getting-started/introduction.html>



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

```
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
```

**i** 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

### **i** Немного о формате yaml

Файл формата [yaml](#)<sup>156</sup> критичен к отступам текста<sup>157</sup> при описании параметров. Стрелкой в имени параметра указан отступ в 4 пробела от начала строки.

**i** Легенда таблицы доступна [на этой странице](#)<sup>158</sup>.

Имя параметра	Описание	Примечания
<b>DB</b>	Параметры подключения к базе данных.	
→ <b>NAME</b>	Имя базы данных сервиса Scheduler.	Остальные параметры подключения к базе данных берутся с основного файла конфигурации TIONIX.
<b>LOG_LEVEL</b>	Уровень детализации для всех типов журналов. По умолчанию: <b>INFO</b> .	
<b>CELERYBEAT_SYNC_EVERY</b>	Количество задач, после выполнения которых требуется обновить список задач (beats).	
<b>CELERYBEAT_MAX_LOOP_INTERVAL</b>	Максимальный интервал между повторяемыми задачами.	
<b>SENTRY</b>	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.

<sup>156</sup> <https://yaml.org/>

<sup>157</sup> <https://yaml.org/spec/1.2.2/#61-indentation-spaces>

<sup>158</sup> <https://conf.tionix.ru/pages/viewpage.action?pageId=205881354#id-Условныеобозначения-table-format-desc>

Имя параметра	Описание	Примечания
→ <b>ENABLED</b>	Включение отправки данных мониторинга и сообщений об ошибках удаленному серверу Sentry. Обязателен, если выставлен SENTRY.	
→ <b>LOG_LEVEL</b>	Уровень логирования событий для отправки. Обязателен, если выставлен SENTRY.	
→ <b>DSN</b>	Название источника данных (DSN <sup>159</sup> ), зарегистрированный в удаленном сервере Sentry. Обязателен, если выставлен SENTRY.	<p>DSN оформляется по следующей схеме:</p> <pre>http(s):// public_key:secret_key@domain/ project_id</pre>
<b>SCHEDULER_WORKER_LOG_FILE</b>	Путь до файла журналов исполнителя задач Scheduler. Используется сервисом <b>tionix-scheduler-worker</b> .	<p>Необходимо выбрать <b>один из двух</b> указанных параметров:</p> <ul style="list-style-type: none"> <li>• <b>/dev/stdout</b>, 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;</li> <li>• <b>путь до файла</b>, если производится обычная установка, в этом случае журналы будут сохранены в указанном по пути файле.</li> </ul> <p>Нативная поддержка сервиса Journald на данный момент отсутствует.</p>
<b>SCHEDULER_BEAT_LOGFILE</b>	Путь до файла журналов обновления списка задач. Используется сервисом <b>tionix-scheduler-beat</b> .	<p>Необходимо выбрать <b>один из двух</b> указанных параметров:</p> <ul style="list-style-type: none"> <li>• <b>/dev/stdout</b>, 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;</li> <li>• <b>путь до файла</b>, если производится обычная установка, в этом случае журналы будут сохранены в указанном по пути файле.</li> </ul> <p>Нативная поддержка сервиса Journald на данный момент отсутствует.</p>

## Дебаггинг

Для максимального подробного журналирования ошибок при выполнении работы сервиса можно включить поддержку трассировок:

<sup>159</sup> <https://docs.sentry.io/product/sentry-basics/dsn-explainer/>

Имя параметра	Описание	Примечания
<b>TRACEBACK_ENABLED</b>	Включение трассировки ошибок в файлах журналов. По умолчанию: <b>False</b> .	

## 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:

- [Ceilometer](#)<sup>160</sup> совместно с [Gnocchi](#)<sup>161</sup> - основной сервис, предоставляющие данные статистики по виртуальным машинам;
- [Zabbix](#)<sup>162</sup> - получение данных статистики, собранные Zabbix по инфраструктуре.

Дополнительной функцией сервиса является наблюдение за состоянием сервиса резервного копирования [Bareos](#)<sup>163</sup>.

### 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 \(см. стр. 112\)](#).

### 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';
```

<sup>160</sup> <https://docs.openstack.org/ceilometer/latest/>

<sup>161</sup> <https://specs.openstack.org/openstack/telemetry-specs/specs/juno/gnocchi.html>


<sup>162</sup> <https://www.zabbix.com/ru>

<sup>163</sup> <https://www.bareos.com/>

## 4. Выйдите из сессии СУБД:


```
exit;
```

## Установка модуля

 Перед установкой модуля убедитесь в установке и корректной настройке сервиса [OpenStack Ceilometer](#) (см. стр. 101).

## 1. Установите пакет модуля:

```
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. 144)):

```
# 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
<b>DB → NAME</b>	Укажите корректное имя базы, указанное при настройке БД для Monitor. По умолчанию: <b>tionix_monitor</b> .
<b>MONITOR_API_LISTEN</b>	Укажите корректный слушаемый адрес в зависимости от типа установки: <ul style="list-style-type: none"> <li>классическая архитектура: Адрес сети управления (mgmt) облачной платформой;</li> <li>референсная архитектура: O.O.O.O.</li> </ul>
<b>Параметры LOGFILE</b>	Укажите пути файлов журналирования в зависимости от типа установки: <ul style="list-style-type: none"> <li>классическая архитектура: как предложено в конфигурации;</li> <li>референсная архитектура: <b>/dev/stdout</b> или <b>/dev/stderr</b>.</li> </ul>
<b>CEILOMETER_METERS</b>	При включении сбора данных мониторинга с использованием Ceilometer ( <b>ENABLE_CEILOMETER_MONITORING = True</b> ) в этом списке укажите нужные метрики.

 Информацию об остальных параметрах можно получить в [описании файла конфигурации Monitor](#) (см. стр. 144).

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
```

**i** 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
```

**i** 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

### **i** Немного о формате yaml

Файл формата [yaml](https://yaml.org/)<sup>164</sup> критичен к отступам текста<sup>165</sup> при описании параметров. Стрелкой в имени параметра указан отступ в 4 пробела от начала строки.

**i** Легенда таблицы доступна [на этой странице](#)<sup>166</sup>.


Имя параметра	Описание	Примечания
<b>DB</b>	Параметры подключения к базе данных.	

<sup>164</sup> <https://yaml.org/>

<sup>165</sup> <https://yaml.org/spec/1.2.2/#61-indentation-spaces>

<sup>166</sup> <https://conf.tionix.ru/pages/viewpage.action?pageId=205881354#id-Условныеобозначения-table-format-desc>



Имя параметра	Описание	Примечания
→ <b>NAME</b>	Имя базы данных сервиса Monitor.	Остальные параметры подключения к базе данных берутся с основного файла конфигурации TIONIX.
<b>LOG_LEVEL</b>	Уровень детализации для всех типов журналов. По умолчанию: <b>INFO</b>	
<b>CEILOMETER_METERS</b>	Метрики, информацию от которых необходимо получать для хранения и визуализации сервисом Monitor. Обязателен, если выставлен <b>ENABLE_CEILOMETER_MONITORING</b> .	
<b>ENABLE_CEILOMETER_MONITORING</b>	Включение получения данных мониторинга с сервиса Ceilometer.	
<b>ENABLE_ZABBIX_MONITORING</b>	Включение получения данных мониторинга с сервиса Zabbix.	
<b>ENABLE_BACKUP</b>	Включение получения данных по резервным копиям с сервиса BareOS.	
<b>SENTRY</b>	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.
→ <b>ENABLED</b>	Включение отправки данных мониторинга и сообщений об ошибках удаленному серверу Sentry. Обязателен, если выставлен <b>SENTRY</b> .	
→ <b>LOG_LEVEL</b>	Уровень логирования событий для отправки. Обязателен, если выставлен <b>SENTRY</b> .	
→ <b>DSN</b>	Название источника данных ( <b>DSN</b> <sup>167</sup> ), зарегистрированный в удаленном сервере Sentry. Обязателен, если выставлен <b>SENTRY</b> .	DSN оформляется по следующей схеме: <pre>http(s):// public_key:secret_key@domain/ project_id</pre>
<b>MONITOR_API_LISTEN</b>	Адрес прослушивания для API сервиса Monitor.	 Рекомендуется использование только адреса сети управления.

<sup>167</sup> <https://docs.sentry.io/product/sentry-basics/dsn-explainer/>

Имя параметра	Описание	Примечания
<b>NODE_CONTROL_API_LISTEN_PORT</b>	Порт прослушивания для API сервиса Monitor.	
<b>MONITOR_API_AUDIT_ENABLED</b>	Включение журналирования всех запросов, которые выполняются для Monitor API.	
<b>MONITOR_API_LOGFILE</b>	Путь до файла журналов для запросов к Monitor API. Используется сервисом <b>tionix-node-monitor-api</b> .	<p>Необходимо выбрать <b>один из двух</b> указанных параметров:</p> <ul style="list-style-type: none"> <li>• <b>/dev/stdout</b>, 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;</li> <li>• <b>путь до файла</b>, если производится обычная установка, в этом случае журналы будут сохранены в указанном по пути файле.</li> </ul> <p>Нативная поддержка сервиса Journald на данный момент отсутствует.</p>
<b>MONITOR_NOVA_LISTENER_LOGFILE</b>	Путь до файла журналов сервиса по отслеживанию события в сервисе Nova. Используется сервисом <b>tionix-monitor-nova-listener</b> .	<p>Необходимо выбрать <b>один из двух</b> указанных параметров:</p> <ul style="list-style-type: none"> <li>• <b>/dev/stdout</b>, 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;</li> <li>• <b>путь до файла</b>, если производится обычная установка, в этом случае журналы будут сохранены в указанном по пути файле.</li> </ul> <p>Нативная поддержка сервиса Journald на данный момент отсутствует.</p>

## Отладка

Для максимального подробного журналирования ошибок при выполнении работы сервиса можно включить поддержку трассировок:

Имя параметра	Описание	Примечания
<b>TRACEBACK_ENABLED</b>	Включение трассировки ошибок в файлах журналов. По умолчанию: <b>False</b> .	

## Dashboard

### Информация о модуле Dashboard

#### OpenStack Horizon

В составе проектов OpenStack имеется компонент [OpenStack<sup>168</sup> Horizon<sup>169</sup>](#)<sup>170</sup>, 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;
- 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 Dashboard

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. 118) – для расширения функций гипервизоров;
2. [Scheduler](#) (см. стр. 136) – для задач планирования;
3. [Monitor](#) (см. стр. 141) – 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<sup>171</sup>](#) 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<sup>172</sup>](#).

 Немного о формате yaml

<sup>168</sup> <https://docs.openstack.org/horizon/victoria/>

<sup>169</sup> <https://docs.openstack.org/horizon/victoria/>

<sup>170</sup> <https://docs.openstack.org/horizon/victoria/>

<sup>171</sup> <https://conf.tionix.ru/x/KoCBCg>

<sup>172</sup> <https://conf.tionix.ru/pages/viewpage.action?pageId=205881354#id-Условныеобозначения-table-format-desc>

Файл формата [yaml](https://yaml.org/)<sup>173</sup> критичен к отступам текста<sup>174</sup> 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	<p>Уровень логирования. Доступные значения:</p> <ul style="list-style-type: none"> <li>• DEBUG;</li> <li>• INFO;</li> <li>• WARNING;</li> <li>• ERROR;</li> <li>• CRITICAL.</li> </ul> <p>Значения являются регистронезависимыми.</p>	INFO
KEYSTONE	<p>Настройки для авторизации в службе Keystone, где:</p> <ul style="list-style-type: none"> <li>• auth_url - адрес сервиса Keystone;</li> <li>• auth_version - версия Keystone: 2 или 3;</li> <li>• auth_user - логин пользователя;</li> <li>• auth_password - пароль пользователя;</li> <li>• auth_tenant - название проекта;</li> <li>• compute_service_name - название службы Compute;</li> <li>• volume_service_name - название службы Volume;</li> <li>• network_service_name - название службы Neutron;</li> <li>• identity_service_name - название службы Keystone.</li> </ul>	<ul style="list-style-type: none"> <li>• auth_url - http://localhost:5000;</li> <li>• auth_version - 3;</li> <li>• auth_user - admin;</li> <li>• auth_password - admin;</li> <li>• auth_tenant - admin;</li> <li>• compute_service_name - compute;</li> <li>• volume_service_name - volumev2;</li> <li>• network_service_name - network;</li> <li>• identity_service_name - identity.</li> </ul>
NEUTRON_VERSION	Версия клиента: 2.	
DB	<p>Настройки базы данных, где:</p> <ul style="list-style-type: none"> <li>• ENGINE - тип базы данных;</li> <li>• USER - пользователь базы данных;</li> <li>• PASSWORD - пароль базы данных;</li> <li>• HOST - хост, на котором запущена база данных;</li> <li>• PORT - порт сервера с базой данных;</li> <li>• NAME - название базы данных.</li> </ul>	<ul style="list-style-type: none"> <li>• ENGINE - django.db.backends.mysql;</li> <li>• USER - tionix;</li> <li>• PASSWORD - password;</li> <li>• HOST - localhost;</li> <li>• PORT - 3306;</li> <li>• NAME - tionix_dash.</li> </ul>
SENTRY	<p>Настройки логирования Sentry, где:</p> <ul style="list-style-type: none"> <li>• ENABLED - Флаг, отвечающий за отправку сообщений об ошибках в Sentry. Возможные значения: <ul style="list-style-type: none"> <li>• True;</li> <li>• False.</li> </ul> Значения являются регистронезависимыми.</li> <li>• DSN - Адрес сервера Sentry, содержит ключ пользователя и идентификатор проекта;</li> <li>• LOG_LEVEL - Уровень логирования в Sentry. Значения являются регистронезависимыми.</li> </ul>	<ul style="list-style-type: none"> <li>• False;</li> <li>• - Адрес внутреннего сервера Sentry;</li> <li>• CRITICAL.</li> </ul>

173 <https://yaml.org/>

174 <https://yaml.org/spec/1.2.2/#61-indentation-spaces>

Параметр	Описание	Значение по умолчанию
ENABLE_QOS	Активация вкладки «Сетевые политики QoS». Возможные значения: <ul style="list-style-type: none"> <li>• True;</li> <li>• False.</li> </ul>	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
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 можно включить поддержку трассировок:

Имя параметра	Описание	Примечания
<b>TRACEBACK_ENABLED</b>	Включение трассировки ошибок в файлах журналов. По умолчанию: <b>False</b> .	

## Установка веб-панели Horizon

- [Установка](#) (см. стр. 149)
- [Включение шифрования веб-панели](#) (см. стр. 154)
- [Таблица конфигурации](#) (см. стр. 155)
  - [Horizon](#) (см. стр. 155)

### 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. 155)).

#### 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
```

**i** 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>
```

**i** Информация о конфигурации веб-сервера описана в [этой части](#) (см. стр. 44) Руководства.

2. После включения этой конфигурации перезапустите веб-сервер:

```
systemctl restart httpd
```

**i** 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 СА вашего центра сертификации.

**⚠** Функционирование продуктивных сред с самоподписанными сертификатами официально не поддерживается.

4. С помощью веб-браузера загрузите страницу веб-панели.

## Таблица конфигурации

### Horizon

**i** Путь до конфигурации: `/etc/openstack-dashboard/local_settings`

**i** Легенда таблицы доступна [на этой странице](#)<sup>175</sup>.

Имя параметра	Описание	Примечания
<b>import from</b>	Встроенные механизмы загрузки модулей Python.	Этот файл фактически является кодом на языке Python. Здесь загружаются модули, необходимые для корректной обработки файла конфигурации компонентом Horizon.
<b>WEBROOT</b>	Корневой путь до объектов веб-панели.	
<b>DEBUG</b>	Включение режима отладки для Horizon.	Хранение журналов Horizon зависят от настроек журналирования веб-сервера. Является параметром фреймворка Django.

<sup>175</sup> <https://conf.tionix.ru/pages/viewpage.action?pageId=205881354#id-Условныеобозначения-table-format-desc>

Имя параметра	Описание	Примечания
<b>ALLOWED_HOSTS</b>	Разрешенные домены для подключения к Horizon.	<p>В этот список можно добавить имена доменных имён, которые смогут получить доступ к веб-панели. Можно добавить несколько имён, оформленных как список в Python.</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> <p> Аргумент '*' (вместе с кавычками) позволит подключиться со всех доменных имён.</p> </div>
<b>LOCAL_PATH</b>	Путь до локальных ресурсов модуля.	Является параметром фреймворка Django.
<b>SECRET_KEY</b>	Уникальная комбинация инсталляции компонента.	Используется для криптографической подписи <sup>176</sup> , параметр фреймворка Django.
<b>EMAIL_BACKEND</b>	Включение бэкенда для отправки писем электронной почты.	По умолчанию письма, генерируемые веб-панелью, отправляются в стандартной ввод терминала.
<b>OPENSTACK_HOST</b>	Переменная для адреса сервиса Keystone.	<p>Эта переменная используется другими переменными в конфигурации, например, <b>OPENSTACK_KEYSTONE_URL</b>. Сама переменная при подключении к сервису Keystone не используется.</p> <p>Параметр <b>OPENSTACK_HOST</b> нужно использовать при наличии только одного региона в облачной платформе. Если регионов больше, то воспользуйтесь параметром <a href="#">AVAILABLE_REGIONS</a><sup>177</sup>.</p>
<b>OPENSTACK_KEYSTONE_URL</b>	Полный URL для подключения к Keystone.	
<b>OPENSTACK_API_VERSIONS</b>	Указание версий различных API сервисов OpenStack.	
<b>OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT</b>	Включение поддержки нескольких доменов, созданных в сервисе Keystone.	Необходимо для <a href="#">домен-специфичных драйверов</a> <sup>178</sup> , настраиваемых в сервисе Keystone.
<b>OPENSTACK_KEYSTONE_DEFAULT_DOMAIN</b>	Используемый домен Keystone по умолчанию.	
<b>OPENSTACK_KEYSTONE_DEFAULT_ROLE</b>	Используемая для пользователей роль по умолчанию.	Используется при добавлении пользователя в проект через веб-панель без указания роли.

<sup>176</sup> <https://docs.djangoproject.com/en/3.2/topics/signing/>

<sup>177</sup> <https://github.com/openstack/horizon/blob/5e4ca1a9fdec04db08552e9e93fe372b8b8b45ae/doc/source/configuration/settings.rst#available-regions>

<sup>178</sup> <https://conf.tionix.ru/pages/viewpage.action?pageId=164102237#id->

Использование связанных доменов систем хранения данных пользователей-ks-domain-specific-drivers

Имя параметра	Описание	Примечания
<b>CSRF_COOKIE_SECURE</b>	Передача данных CSRF только через шифрованный канал. Обязателен, если включен HTTPS.	Ограничивает передачу <a href="#">CSRF</a> <sup>179</sup> cookie только через протокол HTTPS. Является <a href="#">параметром фреймворка</a> <sup>180</sup> Django.
<b>SESSION_COOKIE_SECURE</b>	Передача данных сессий только через шифрованный канал. Обязателен, если включен HTTPS.	Ограничивает передачу <a href="#">session cookie</a> <sup>181</sup> только через протокол HTTPS. Является параметром фреймворка Django.
<b>POLICY_FILES_PATH</b>	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.
<b>SESSION_ENGINE</b>	Метод хранения данных сессии на стороне сервера.	Предлагаемая настройка хранит данные сессии в системе кэширования. <div style="border: 1px solid orange; padding: 5px; margin-top: 10px;"> При очистке данных в системе кэширования все установленные сессии будут завершены.</div>
<b>CACHES</b>	Данные системы кэширования.	
<b>TIME_ZONE</b>	Часовой пояс по умолчанию.	Влияет на показываемое время для объектов в веб-панели. Пользователь может поменять часовой пояс в настройках аккаунта.
<b>LOGGING</b>	Расширенные параметры логгирования частей веб-панели.	Является параметром фреймворка Django.
<b>SECURITY_GROUP_RULES</b>	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.

<sup>179</sup> [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](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)

<sup>180</sup> <https://docs.djangoproject.com/en/3.2/ref/csrf/>


<sup>181</sup> <https://docs.djangoproject.com/en/3.2/topics/http/sessions/#using-cookie-based-sessions>

2. В файл `/etc/openstack-dashboard/local_settings` добавьте необходимые системные параметры:

```
try:
    from tionix_dashboard.settings import *
except ImportError:
    pass

AUTHENTICATION_BACKENDS = [
    'tionix_dashboard.auth.backend.TionixKeystoneBackend'
]
```

3. Настройте конфигурационный файл `/etc/tionix/dashboard.yaml`. В референсной архитектуре используется следующая конфигурация в формате [YAML](#)<sup>182</sup>:

 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/

BLOCKED_ROLES: ['openstack.roles.vdi-user']
```

Описание параметров:

 Легенда таблицы доступна [на этой странице](#)<sup>183</sup>.

Имя параметра	Описание	Примечания
<b>DB</b>	Глобальные параметры базы данных компонента.	
→ <b>ENGINE</b>	Движок к базе данных.	<code>django.db.backends.mysql</code> является универсальным драйвером, который подходит и для СУБД MariaDB.
→ <b>NAME</b>	Имя базы данных компонента.	
<b>KEYSTONE</b>	Параметры подключения к сервису Keystone.	
→ <b>network_service_name</b>	Название точки входа для сервиса Neutron.	
→ <b>identity_service_name</b>	Название точки входа для сервиса Keystone.	
<b>MIN_RESERVE_VM</b>	Минимальное количество резервируемых VM.	Настройка резервирования VM выполняется в сервисе NodeControl.

<sup>182</sup> <https://en.wikipedia.org/wiki/YAML>

<sup>183</sup> <https://conf.tionix.ru/pages/viewpage.action?pagelid=205881354#id-Условныеобозначения-table-format-desc>

Имя параметра	Описание	Примечания
<b>VOLUME_ATTACH_MAX_RETRIES</b>	Максимальное количество попыток подключения диска к VDI-машине.	
<b>POINT_METER_API_URL</b>	Адрес к сервису <a href="#">Pointmeter</a> (см. стр. 160).	

4. Запустите команду сбора статических 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`:

```
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 статических 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 передает его поставщику услуг или заказчику через средства электронной почты. Данные Pointmeter шифруются с использованием протокола GPG и применением асимметричного 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 \(см. стр. 112\)](#).

### 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
```

- 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'
```

- 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

- Запустите сервис PointMeter:

```
systemctl enable tionix-point-meter-api.service
systemctl start tionix-point-meter-api.service
```

- 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](https://yaml.org/)<sup>184</sup> критичен к отступам текста<sup>185</sup> 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](#)<sup>186</sup>.

<sup>184</sup> <https://yaml.org/>

<sup>185</sup> <https://yaml.org/spec/1.2.2/#61-indentation-spaces>

<sup>186</sup> <https://conf.tionix.ru/pages/viewpage.action?pageId=205881354#id-Условныеобозначения-table-format-desc>

Имя параметра	Описание	Примечания
<b>DEBUG</b>	Включение режима подобного журналирования.	
<b>MAIL_SERVER</b>	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.
<b>MAIL_PORT</b>	Порт SMTP-сервера.	
<b>MAIL_USE_TLS</b>	Включение шифрования с помощью протокола TLS.	
<b>MAIL_USE_SSL</b>	Включение шифрования с помощью протокола SSL.	
<b>MAIL_USERNAME</b>	Имя пользователя для аутентификации в SMTP-сервере.	В основном, необходимо для сбора данных мониторинга и сообщений об ошибках в удаленный сервис Sentry.
<b>MAIL_DEFAULT_SENDER</b>	Имя или адрес отправителя, который будет указан в поле Sender в письме с отчетом.	
<b>MAIL_PASSWORD</b>	Пароль пользователя для аутентификации в SMTP-сервере.	
<b>MAIL_ASCII_ATTACHMENTS</b>	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.
<b>TIONIX_MAIL</b>	Адрес электронной почты, куда будут отправлены отчеты сервиса.	Не меняйте этот параметр без крайней необходимости.
<b>TIONIX_CLIENTS</b>	Адрес или адреса электронной почты, которые будут включены в поле CC отправляемого письма.	Можно указать несколько адресов через символ ;
<b>CRON_SCHEDULE</b>	Правило cron для периодической отправки отчетов.	
<b>TIME_ZONE</b>	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

**i** Установка должна выполняться на вычислительных узлах.

1. Установите пакет компонента:

```
dnf -y install python3-tionix_agent
```

**i** 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. 163](#)):

```
openstack tnx agent configure --type <type>
```

**i** Можно указать несколько типов через запятую.

3. Включите следующую конфигурацию агента по пути `/etc/tionix/agent.yaml` ([описание](#) (см. [стр. 164](#))):

```
[DEFAULT]
transport_url = amqp://user:RABBIT_PASS@controller:5672/vhost
durable = false
agent_type = control, spice_proxy
```

**i** 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
```

**i** 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

**i** Легенда таблицы доступна [на этой странице](#)<sup>187</sup>.

Имя параметра	Описание	Примечания
<b>[DEFAULT]</b>	Глобальные параметры компонента.	
<b>transport_url</b>	Адреса сервисов RabbitMQ для RPC-функций.	<p>Можно указать только один сервер RabbitMQ.</p> <p>При наличии в имени пользователя и пароле символов, которые для формата INI являются специальными (например, знак комментирования "#"), то можно использовать его URL-код.</p> <ul style="list-style-type: none"> <li>Пример: <code>passw#ord</code> → <code>passw%23ord</code></li> </ul>
<b>durable</b>	<p>Параметр для подключения к очереди сообщений AMQP. Возможные значения:</p> <ul style="list-style-type: none"> <li><code>true</code> - очередь сообщений будет сохранять свое состояние и восстанавливаться после перезапуска брокера;</li> <li><code>false</code> - очередь сообщений будет удаляться после перезапуска брокера.</li> </ul>	

<sup>187</sup> <https://conf.tionix.ru/pages/viewpage.action?pageId=205881354#id-Условныеобозначения-table-format-desc>

Имя параметра	Описание	Примечания
<b>agent_type</b>	<p>Типы агентов, которые необходимо активировать.</p> <p>Возможные значения:</p> <ul style="list-style-type: none"> <li>• <code>compute</code> – для изолированного вычислительного узла;</li> <li>• <code>control, selfdiscovery</code> – для любого вычислительного узла;</li> <li>• <code>mlx</code> – для коммутаторов;</li> <li>• <code>consul</code> – для службы синхронизации конфигурационных файлов с хранилищем Consul;</li> <li>• <code>spice_proxy</code> – для запуска сервиса, отвечающего за корректную работу виртуальных машин через SPICE-сессии.</li> </ul>	
<b>[consul]</b>	Параметры подключения к сервису Consul. Обязателен, если <code>consul</code> в <code>agent_type</code> .	
<b>host</b>	Доменное имя или IP-адрес сервиса Consul. Обязателен, если <code>consul</code> в <code>agent_type</code> .	
<b>port</b>	Порт сервиса Consul. Обязателен, если <code>consul</code> в <code>agent_type</code> .	
<b>token</b>	Токен аутентификации к сервису Consul. Обязателен, если <code>consul</code> в <code>agent_type</code> .	

## Дебаггинг

Для подробного журналирования событий можно указать следующие параметры:

## Таблица конфигурации

Имя параметра	Описание	Примечания
<b>[DEFAULT]</b>	Глобальные параметры компонента.	
<b>debug</b>	Включение подробного журналирования.	
<b>log_file</b>	Путь до файла журнала. Обязателен, если выставлен <code>debug</code> .	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. 97).

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_auth_token
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. 99).

❗ Легенда таблицы доступна [на этой странице](#)<sup>188</sup>.

Имя параметра	Описание	Примечания
<b>[DEFAULT]</b>	Глобальные параметры сервиса Cinder.	
<b>rpc_response_timeout</b>	Время ожидания ответа на запрос RPC.	
<b>volume_manager</b>	Указание модуля управления дисками.	

<sup>188</sup> <https://conf.tionix.ru/pages/viewpage.action?pageId=205881354#id-Условныеобозначения-table-format-desc>

Имя параметра	Описание	Примечания
<b>enabled_backends</b>	Включенные бэкенды (системы хранения), которые описаны в конфигурации Cinder.	Можно указать несколько бэкендов через запятую.
<b>[sanlock-backend]</b>	Раздел с настройками для бэкенда с именем внутри квадратных скобок.	Эти имена используются для указания включенных бэкендов.
<b>volume_driver</b>	Имя драйвера управления блочными устройствами.	
<b>agent_transport_url</b>	Адрес до сервиса RabbitMQ.	
<b>volume_group</b>	Имя группы дисков (VG) в LVM, в котором будут создаваться LV-устройства.	
<b>lvm_type</b>	Тип дисков в VG-группе.	Может иметь два значения: <ul style="list-style-type: none"> <li>• <b>default</b> – блоки будут выданы на весь размер диска;</li> <li>• <b>thin</b> – блоки будут выдаваться по мере необходимости для записи ("тонкие диски"). В этом случае возможно использование переподписки по дискам.</li> </ul>
<b>lvm_mirrors</b>	Создание зеркальных LV в VG-группе.	При включении этого параметра в VG-группе будут созданы копии LV.
<b>volume_backend_name</b>	Имя бэкенда для платформы.	Это имя используется при регистрации бэкенда в группе дисков Cinder.
<b>agent_response_timeout</b>	Время ожидания ответа от агента.	
<b>[nova]</b>	Параметры взаимодействия с сервисом Nova	
<b>token_auth_url</b>	Данные сервиса Keystone для получения данных о сервисе Nova и генерации токена авторизации.	
<b>auth_section</b>	Имя сервиса аутентификации.	
<b>auth_type</b>	Тип аутентификации.	

## Approvie

### Информация о сервисе Approvie

TIONIX.Approvie – модуль, расширяющий функции [RBAC](#)<sup>189</sup> платформ OpenStack. Модуль позволяет использовать сторонний сервис для обработки разрешений доступа к ресурсам платформы OpenStack.

<sup>189</sup> <https://conf.tionix.ru/pages/viewpage.action?pageId=174030874#id-Глоссарий-rbac-term>

## Компоненты сервиса Approvie

Сервис Approvie состоит из нескольких компонентов, работающие как службы в systemd:

- **tionix-keystone-rbac.service** - Сервис управления правилами [HttpCheck](#)<sup>190</sup> для службы Keystone.
- **tionix-cinder-rbac.service** - Сервис управления правилами [HttpCheck](#)<sup>191</sup> для службы Cinder.
- **tionix-glance-rbac.service** - Сервис управления правилами [HttpCheck](#)<sup>192</sup> для службы Glance.
- **tionix-nova-rbac.service** - Сервис управления правилами [HttpCheck](#)<sup>193</sup> для службы Nova.
- **tionix-neutron-rbac.service** - Сервис управления правилами [HttpCheck](#)<sup>194</sup> для службы Neutron.

## Установка сервиса Approvie

- [Настройка окружения](#) (см. стр. 168)
- [Установка модуля](#) (см. стр. 168)
  - [Процесс установки](#) (см. стр. 168)
  - [Настройка сервиса Cinder](#) (см. стр. 169)
  - [Настройка сервиса Glance](#) (см. стр. 169)
  - [Настройка сервиса Nova](#) (см. стр. 169)
  - [Настройка сервиса Neutron](#) (см. стр. 170)
  - [Настройка сервиса Keystone](#) (см. стр. 170)
  - [Настройка RBAC-сервиса](#) (см. стр. 171)
  - [Пример файла политик с правилом HttpCheck](#) (см. стр. 171)

## Настройка окружения

Перед самой установкой сервиса нужно предварительно настроить некоторые компоненты инфраструктуры.


 См. также: [Раздел с предварительной настройкой окружения для модулей TIONIX](#) (см. стр. 112).

## Установка модуля

### Процесс установки

1. Установите пакет модуля:

```
dnf -y install python3-tionix_approvie
```

 Стандартные пути до файлов конфигурации:

- `/etc/tionix` — основной каталог конфигурации.
- `/etc/tionix/tionix.yaml` — общий файл конфигурации TIONIX.
- `/etc/tionix/approvie.yaml` - основной файл конфигурации сервиса Approvie.

2. Используйте эту конфигурацию по пути `/etc/tionix/approvie.yaml`:

 Необходимо использовать синтаксис вложенных коллекций [YAML](#)<sup>195</sup>.

```
identity:
  url: "http://controller:5000"
  user: "admin"
  password: "password"
  domain: "Default"
  project: "admin"

filter:
  blocked_roles: ["vdi-user"]
```

190 <https://conf.tionix.ru/pages/viewpage.action?pagelId=174030874#id-Глоссарий-httpcheck-term>

191 <https://conf.tionix.ru/pages/viewpage.action?pagelId=174030874#id-Глоссарий-httpcheck-term>

192 <https://conf.tionix.ru/pages/viewpage.action?pagelId=174030874#id-Глоссарий-httpcheck-term>

193 <https://conf.tionix.ru/pages/viewpage.action?pagelId=174030874#id-Глоссарий-httpcheck-term>


194 <https://conf.tionix.ru/pages/viewpage.action?pagelId=174030874#id-Глоссарий-httpcheck-term>

195 [https://symfony.ru/doc/current/components/yaml/yaml\\_format.html#yaml-format-collections-ru](https://symfony.ru/doc/current/components/yaml/yaml_format.html#yaml-format-collections-ru)



```
policy:
supported_methods: ["GET", "HEAD", "POST", "PUT", "PATCH", "DELETE"]

common:
debug: false
```

 Подробное описание конфигурации для этого сервиса предоставлено в этом [подразделе](#) (см. стр. 171).

## Настройка сервиса Cinder

После установки пакета `python3-tionix_approvie`, необходимо настроить сервис [Cinder](#)<sup>196</sup> для использования правил [HttpCheck](#)<sup>197</sup>. Для этого необходимо:

1. Создайте файл политик `/etc/cinder/policy.yaml` с параметрами, указанными в `/etc/cinder/policy.yaml.example`.
2. В конфигурационном файле `/etc/cinder/cinder.conf` добавьте путь до файла политик:

```
[oslo_policy]
policy_file = policy.yaml
```

3. Перезапустите сервис [Cinder](#)<sup>198</sup>:

```
systemctl restart openstack-cinder-api.service
```

### Важно

Использовать файл политик с операциями, в которых указаны правила типа [HttpCheck](#)<sup>199</sup>. Пример доступен по пути `/etc/cinder/policy.yaml.example`.

## Настройка сервиса Glance

После установки пакета `python3-tionix_approvie`, необходимо настроить сервис [Glance](#)<sup>200</sup> для использования правил [HttpCheck](#)<sup>201</sup>. Для этого необходимо:

1. Создайте файл политик `/etc/glance/policy.yaml` с параметрами, указанными в `/etc/glance/policy.yaml.example`.
2. В конфигурационном файле `/etc/glance/glance-api.conf` добавьте путь до файла политик:

```
[oslo_policy]
policy_file = policy.yaml
```

3. Перезапустите сервис [Glance](#)<sup>202</sup>:

```
systemctl restart openstack-glance-api.service
```

### Важно

Использовать файл политик с операциями, в которых указаны правила типа [HttpCheck](#)<sup>203</sup>. Пример доступен по пути `/etc/cinder/policy.yaml.example`.

## Настройка сервиса Nova

После установки пакета `python3-tionix_approvie`, необходимо настроить сервис [Nova](#)<sup>204</sup> для использования правил [HttpCheck](#)<sup>205</sup>. Для этого необходимо:

<sup>196</sup> <https://conf.tionix.ru/pages/viewpage.action?pagelId=174030874#id-Глоссарий-cinder-term>

<sup>197</sup> <https://conf.tionix.ru/pages/viewpage.action?pagelId=174030874#id-Глоссарий-httpcheck-term>

<sup>198</sup> <https://conf.tionix.ru/pages/viewpage.action?pagelId=174030874#id-Глоссарий-cinder-term>

<sup>199</sup> <https://conf.tionix.ru/pages/viewpage.action?pagelId=174030874#id-Глоссарий-httpcheck-term>

<sup>200</sup> <https://conf.tionix.ru/pages/viewpage.action?pagelId=174030874#id-Глоссарий-glance-term>

<sup>201</sup> <https://conf.tionix.ru/pages/viewpage.action?pagelId=174030874#id-Глоссарий-httpcheck-term>

<sup>202</sup> <https://conf.tionix.ru/pages/viewpage.action?pagelId=174030874#id-Глоссарий-glance-term>

<sup>203</sup> <https://conf.tionix.ru/pages/viewpage.action?pagelId=174030874#id-Глоссарий-httpcheck-term>

<sup>204</sup> <https://conf.tionix.ru/pages/viewpage.action?pagelId=174030874#id-Глоссарий-nova-term>

<sup>205</sup> <https://conf.tionix.ru/pages/viewpage.action?pagelId=174030874#id-Глоссарий-httpcheck-term>

1. Создайте файл политик `/etc/nova/policy.yaml` с параметрами, указанными в `/etc/nova/policy.yaml.example`.
2. В конфигурационном файле `/etc/nova/nova.conf` добавьте путь до файла политик:

```
[oslo_policy]
policy_file = policy.yaml
```

3. Перезапустите сервис [Nova](#)<sup>206</sup>:

```
systemctl restart openstack-nova-api.service
systemctl restart openstack-nova-metadata-api.service
systemctl restart openstack-nova-os-compute-api.service
```

#### Важно

Использовать файл политик с операциями, в которых указаны правила типа [HttpCheck](#)<sup>207</sup>. Пример доступен по пути `/etc/cinder/policy.yaml.example`.

## Настройка сервиса Neutron

После установки пакета `python3-tionix_approvie`, необходимо настроить сервис [Neutron](#)<sup>208</sup> для использования правил [HttpCheck](#)<sup>209</sup>. Для этого необходимо:

1. Создайте файл политик `/etc/neutron/policy.yaml` с параметрами, указанными в `/etc/neutron/policy.yaml.example`.
2. В конфигурационном файле `/etc/neutron/neutron.conf` добавьте путь до файла политик:

```
[oslo_policy]
policy_file = policy.yaml
```

3. Перезапустите сервис [Neutron](#)<sup>210</sup>:

```
systemctl restart neutron-server.service
```

#### Важно

Использовать файл политик с операциями, в которых указаны правила типа [HttpCheck](#)<sup>211</sup>. Пример доступен по пути `/etc/cinder/policy.yaml.example`.

## Настройка сервиса Keystone

После установки пакета `python3-tionix_approvie`, необходимо настроить сервис [Keystone](#)<sup>212</sup> для использования правил [HttpCheck](#)<sup>213</sup>. Для этого необходимо:

1. Создайте файл политик `/etc/keystone/policy.yaml` с параметрами, указанными в `/etc/keystone/policy.yaml.example`.
2. В конфигурационном файле `/etc/keystone/keystone.conf` добавьте путь до файла политик:

```
[oslo_policy]
policy_file = policy.yaml
```

3. В файле настроек `/etc/keystone/keystone.conf` добавьте атрибут проекта администратора:

```
[resource]
admin_project_name = admin
```

Где `admin` - имя проекта администратора.

4. Перезапустите сервис [Keystone](#)<sup>214</sup>:

206 <https://conf.tionix.ru/pages/viewpage.action?pagelId=174030874#id-Глоссарий-nova-term>

207 <https://conf.tionix.ru/pages/viewpage.action?pagelId=174030874#id-Глоссарий-httpcheck-term>

208 <https://conf.tionix.ru/pages/viewpage.action?pagelId=174030874#id-Глоссарий-neutron-term>

209 <https://conf.tionix.ru/pages/viewpage.action?pagelId=174030874#id-Глоссарий-httpcheck-term>

210 <https://conf.tionix.ru/pages/viewpage.action?pagelId=174030874#id-Глоссарий-neutron-term>

211 <https://conf.tionix.ru/pages/viewpage.action?pagelId=174030874#id-Глоссарий-httpcheck-term>

212 <https://conf.tionix.ru/pages/viewpage.action?pagelId=174030874#id-Глоссарий-keystone-term>

213 <https://conf.tionix.ru/pages/viewpage.action?pagelId=174030874#id-Глоссарий-httpcheck-term>

214 <https://conf.tionix.ru/pages/viewpage.action?pagelId=174030874#id-Глоссарий-keystone-term>

```
apachectl stop
apachectl start
```

**⚠ Важно**

Использовать файл политик с операциями, в которых указаны правила типа [HttpCheck](#)<sup>215</sup>.  
Пример доступен по пути `/etc/cinder/policy.yaml.example`.

**Настройка RBAC-сервиса**

В составе пакета `python3-tionix_approvie` идут `systemd`-файлы для основных сервисов платформ OpenStack.

- Создайте файлы политик:
  - `/etc/tionix/cinder_policy.yaml`;
  - `/etc/tionix/glance_policy.yaml`;
  - `/etc/tionix/keystone_policy.yaml`;
  - `/etc/tionix/neutron_policy.yaml`;
  - `/etc/tionix/nova_policy.yaml`.

Данные файлы должны содержать параметры, которые указаны в соответствующих файлах по умолчанию с окончанием `example`.

- Активируйте необходимые службы:

```
systemctl enable tionix-{keystone,cinder,glance,nova,neutron}-rbac.service
```

- Запустите необходимые службы:

```
systemctl start tionix-{keystone,cinder,glance,nova,neutron}-rbac.service
```

- Проверьте наличие `systemd`-файлов сервисов командой:

```
systemctl list-unit-files tionix-*-rbac.service
```

- Создайте роль `vdi-user`:

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

**✓ Примечание**

Для того, чтобы включить функционал обработки политик по внешнему источнику, необходимо установить файлы политик с указанием [HttpCheck](#)<sup>216</sup> правил в сервисе платформы OpenStack.

**Пример файла политик с правилом HttpCheck**

Пример файла политик с правилом [HttpCheck](#)<sup>217</sup> для сервиса [Cinder](#)<sup>218</sup>:

```
"volume:attachment_create": "http://127.0.0.1:8081/handle_rule"
"volume:attachment_update": "http://127.0.0.1:8081/handle_rule"
"volume:attachment_delete": "http://127.0.0.1:8081/handle_rule"
"volume:attachment_complete": "http://127.0.0.1:8081/handle_rule"
"volume:multiattach_bootable_volume": "http://127.0.0.1:8081/handle_rule"
```

**Описание файла конфигурации Approvie**

В описании процесса по установке сервиса Approvie предложена стандартная конфигурация для настройки. Это страница содержит подробное описание этой конфигурации.

При изменении конфигурации необходимо перезапустить все компоненты сервиса Approvie:

<sup>215</sup> <https://conf.tionix.ru/pages/viewpage.action?pagelId=174030874#id-Глоссарий-httpcheck-term>

<sup>216</sup> <https://conf.tionix.ru/pages/viewpage.action?pagelId=174030874#id-Глоссарий-httpcheck-term>


<sup>217</sup> <https://conf.tionix.ru/pages/viewpage.action?pagelId=174030874#id-Глоссарий-httpcheck-term>

<sup>218</sup> <https://conf.tionix.ru/pages/viewpage.action?pagelId=174030874#id-Глоссарий-cinder-term>

```
systemctl restart tionix-{keystone,cinder,glance,nova,neutron}-rbac.service
```

## Таблица конфигурации

 Легенда таблицы доступна [на этой странице](#)<sup>219</sup>.

 Файл формата [yaml](#)<sup>220</sup> [критичен к отступам текста](#)<sup>221</sup> 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.

Конфигурационный файл состоит из секций:

- `identity`
- `filter`
- `policy`
- `common`

Параметры секции `identity`:

Параметр	Описание	Значение по умолчанию
<code>url</code>	Указатель на ресурс типа HTTP сервиса <a href="#">Identity</a> <sup>222</sup> .	<code>http://controller:5000</code>
<code>user</code>	Привилегированный пользователь сервиса <a href="#">Identity</a> <sup>223</sup> .	<code>admin</code>
<code>password</code>	Пароль привилегированного пользователя для подключения к сервису <a href="#">Identity</a> <sup>224</sup> .	<code>123456</code>
<code>domain</code>	Домен привилегированного пользователя сервиса <a href="#">Identity</a> <sup>225</sup> .	<code>Default</code>
<code>project</code>	Проект привилегированного пользователя сервиса <a href="#">Identity</a> <sup>226</sup> .	<code>admin</code>

Параметры секции `filter`:

Параметр	Описание	Значение по умолчанию
<code>blocked_roles</code>	Список ролей предназначенные для блокирования.	<code>['vdi-user']</code>

Параметры секции `policy`:

Параметр	Описание	Значение по умолчанию
<code>operations_table</code>	Путь до CSV-таблицы, в которой находятся информация по API-методам платформы <a href="#">OpenStack</a> <sup>227</sup> .	<code>flat_table_operations.csv</code>

219 <https://conf.tionix.ru/pages/viewpage.action?pagelId=205881354#id-Условныеобозначения-table-format-desc>

220 <https://yaml.org/>

221 <https://yaml.org/spec/1.2.2/#61-indentation-spaces>

222 <https://conf.tionix.ru/pages/viewpage.action?pagelId=174030874#id-Глоссарий-keystone-term>

223 <https://conf.tionix.ru/pages/viewpage.action?pagelId=174030874#id-Глоссарий-keystone-term>

224 <https://conf.tionix.ru/pages/viewpage.action?pagelId=174030874#id-Глоссарий-keystone-term>

225 <https://conf.tionix.ru/pages/viewpage.action?pagelId=174030874#id-Глоссарий-keystone-term>

226 <https://conf.tionix.ru/pages/viewpage.action?pagelId=174030874#id-Глоссарий-keystone-term>

227 <https://conf.tionix.ru/pages/viewpage.action?pagelId=174030874#id-Глоссарий-openstack-term>

Параметр	Описание	Значение по умолчанию
supported_methods	Список методов, которые используются для создания политик.	['GET', 'HEAD', 'POST', 'PUT', 'PATCH', 'DELETE']

Параметры секции common:

Параметр	Описание	Значение по умолчанию
debug	Включить режим отладки.	false

#### ✓ Примечание

По умолчанию для сервиса обработки политик RBAC отладочная информация журналируется с помощью journal.

## 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](#)<sup>228</sup>. 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:

```
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'
```

228 <http://yaml.org>

```

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](#)<sup>229</sup> критичен к отступам текста<sup>230</sup> 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](#)<sup>231</sup>.

Имя параметра	Описание
<b>CINDER_VERSION</b>	Версия CinderClient для подключения к службе Cinder.
<b>JOURNAL_API_LISTEN</b>	IP-адрес, на котором будет запущена служба Journal API.
<b>JOURNAL_API_LISTEN_PORT</b>	Порт, на котором будет запущена служба Journal API.

229 <https://yaml.org/>

230 <https://yaml.org/spec/1.2.2/#61-indentation-spaces>

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

Имя параметра	Описание
<b>JOURNAL_API_LOGFILE</b>	Путь до лог файла службы Journal API.
<b>JOURNAL_KEYSTONE_LISTENER_LOGFILE</b>	Путь до лог файла службы Keystone listener.
<b>JOURNAL_LISTENER_LOGFILE</b>	Путь до лог файла службы Journal listener.
<b>JOURNAL_NOVA_LISTENER_LOGFILE</b>	Путь до лог файла службы Nova listener.
<b>NEUTRON_VERSION</b>	Поддерживаемая версия Neutron API.
<b>SENTRY</b>	Настройки системы Sentry.
<b>RABBIT_QUEUES</b>	Настройки сервиса выполнения асинхронных задач.
<b>LOG_LEVEL</b>	<p>Уровень журналирования модулей. Доступные значения:</p> <ul style="list-style-type: none"> <li>• DEBUG;</li> <li>• INFO (по умолчанию);</li> <li>• WARNING;</li> <li>• ERROR;</li> <li>• CRITICAL.</li> </ul> <p>Значения являются регистронезависимыми. По умолчанию INFO.</p>
<b>NOVA_RABBIT_VHOST</b>	Виртуальный хост RabbitMQ, используемый сервисом Nova.
<b>KEYSTONE_RABBIT_VHOST</b>	Виртуальный хост RabbitMQ используемый сервисом Keystone.
<b>TRACEBACK_ENABLED</b>	<p>Параметр для вывода трассировки ошибки интерпретатором в журнал сервисов. Возможные значения:</p> <ul style="list-style-type: none"> <li>• True;</li> <li>• False (по умолчанию).</li> </ul> <p>Параметры являются регистронезависимыми.</p>
<b>KEYSTONE → auth_url</b>	Адрес внутренней точки доступа к сервису Keystone.
<b>KEYSTONE → auth_version</b>	Версия протокола сервиса Keystone.
<b>KEYSTONE → auth_user</b>	Имя пользователя в Keystone для сервисов Tionix.
<b>KEYSTONE → auth_password</b>	Пароль пользователя в Keystone для сервисов Tionix.
<b>KEYSTONE → auth_tenant</b>	Название проекта Keystone, используемый сервисами Tionix.
<b>KEYSTONE → compute_service_name</b>	Имя сервиса в Keystone, используемый Nova (по умолчанию compute).
<b>KEYSTONE → volume_service_name</b>	Тип сервиса в Keystone, используемый Cinder (по умолчанию volume).
<b>KEYSTONE ceilometer_service_name</b> →	Тип сервиса Ceilometer, используемый Ceilometer (по умолчанию metering).

Имя параметра	Описание
<b>KEYSTONE</b> → <b>user_domain_name</b>	Название домена пользователя.
<b>KEYSTONE</b> → <b>project_domain_name</b>	Название домена проекта.
<b>KEYSTONE</b> → <b>service_user</b>	Имя пользователя, добавленный в проект service.
<b>KEYSTONE</b> → <b>service_password</b>	Пароль пользователя, добавленный в проект service.
<b>KEYSTONE</b> → <b>service_project</b>	Название проекта, добавленный в проект service.
<b>KEYSTONE</b> <b>service_user_domain_name</b> →	Название домена Keystone пользователя, добавленный в проект service.
<b>KEYSTONE</b> <b>service_project_domain_name</b> →	Название домена Keystone проекта, добавленный в проект service.
<b>DB</b> → <b>ENGINE</b>	Тип базы данных.
<b>DB</b> → <b>USER</b>	Пользователь базы данных.
<b>DB</b> → <b>PASSWORD</b>	Пароль к базе данных.
<b>DB</b> → <b>HOST</b>	Узел, на котором запущена СУБД.
<b>DB</b> → <b>PORT</b>	Порт, на котором запущена СУБД.
<b>DB</b> → <b>MAX_POOL_SIZE</b>	Максимальное размер пула соединений к СУБД.
<b>DB</b> → <b>MAX_OVERFLOW</b>	Максимальный размер переполнения пула.
<b>DB</b> → <b>POOL_RECYCLE</b>	Время жизни соединения в пуле. При достижении времени соединение в пуле будет закрыто.
<b>DB</b> → <b>POOL_TIMEOUT</b>	Время таймаута соединения в пуле.
<b>RABBIT_QUEUES</b> → <b>broker_type</b>	Тип брокера сообщений для передачи сообщений.
<b>RABBIT_QUEUES</b> → <b>host</b>	Узел, на котором расположен брокер сообщений.
<b>RABBIT_QUEUES</b> → <b>port</b>	Порт, на котором расположен брокер сообщений.
<b>RABBIT_QUEUES</b> → <b>username</b>	Имя пользователя.
<b>RABBIT_QUEUES</b> → <b>password</b>	Пароль пользователя.
<b>RABBIT_QUEUES</b> → <b>durable</b>	Режим сохранения состояния очередей при перезапуске RabbitMQ. Возможные значения: <ul style="list-style-type: none"> <li>• True;</li> <li>• False.</li> </ul>
<b>SENTRY</b> → <b>DSN</b>	Адрес, на который отправляются сообщения о событиях системой мониторинга ошибок Sentry.



Имя параметра	Описание
<b>SENTRY → ENABLED</b>	Параметр, включающий систему мониторинга ошибок Sentry. Допустимые значения: <ul style="list-style-type: none"><li>• True;</li><li>• False.</li></ul> По умолчанию False.
<b>SENTRY → LOG_LEVEL</b>	Уровень логирования системы мониторинга ошибок Sentry.
<b>BLOCKED_ROLES</b>	Список ролей платформы OpenStack, которые необходимо заблокировать на этапе авторизации в модулях ТИОНИКС.