

# Ce-NetConnection

## Руководство администратора

ver.2.0

Для WINDOWS/LINUX

## Содержание

1. Введение Se-NetConnection.....	3
1.1. Назначение документа.....	3
1.2. Термины .....	3
1.3. Назначение продукта .....	5
1.4. Технические требования .....	5
2. Подсистема seNes.....	6
2.1. Основные функции.....	7
2.2. Установка.....	8
2.2.1. Установка Docker.....	9
2.2.2. Установка Docker Compose.....	10
2.2.3. Развёртывание и запуск проекта.....	11
2.2.4. Обновление.....	15
2.3. Конфигурирование.....	17
2.3.1. Настройка сертификата.....	17
2.3.2. Настройка сервиса сбора.....	19
2.3.3. Настройка сервиса DeviceMaintenance.....	20
2.3.4. Настройка сервиса Identity.....	21
2.3.5. Настройка сервиса CENC .....	21
2.3.6. Настройка сервиса Logger.....	24
2.3.7. Настройка сервиса DataStore.....	25
2.3.8. Настройка системы логирования.....	25
3. Подсистема seCloud.....	27
3.1. Основные функции.....	27
3.2. Установка.....	28
3.2.1. Установка Docker.....	28
3.2.2. Установка Docker Compose.....	29
3.2.3. Развёртывание и запуск проекта.....	30
3.2.4. Обновление.....	32
3.3. Конфигурирование.....	35
3.3.1. SNMP .....	35
3.3.2. Modbus.....	38

## 1. Введение Ce-NetConnection

### 1.1. Назначение документа

Этот документ является Руководством администратора Ce-NetConnection.

Для комфортной работы с Ce-NetConnection пользователям необходимо:

- Знать основы работы с браузером.
- Владеть инструментами конфигурирования и администрирования OS Windows, OS Linux, системами виртуализации, контейнеризации Docker и СУБД PostgreSQL.
- Уметь пользоваться командной строкой и технической документацией к применяемым компонентам системы.

Руководство администратора предназначено для следующих целей:

- Помочь пользователю корректно установить и развернуть продукт.
- Ознакомить пользователя с процессом установки обновлений.

### 1.2. Термины

– **ESB** (Enterprise Service Bus) – связующее программное обеспечение, обеспечивающее централизованный и унифицированный событийно-ориентированный обмен сообщениями между различными информационными системами на принципах сервис-ориентированной архитектуры.

– **MDM** (Meter data management) – класс прикладных программ, применяемых предприятиями энергетического сектора для управления данными, полученными с приборов учёта энергии.

– **HES** (Head End System) – система, обеспечивающая коммуникацию с приборами интеллектуального учёта, для сбора, измерения, контроля параметров и предоставления доступа пользователям и внешним системам.

– **DLMS** (Device Language Message Specification) – открытый протокол для обмена данными с приборами учета.

– **СПОДЭС** – спецификация протокола обмена данными электронных счетчиков построенный на базе DLMS.

– **IEC 61968** – представляет собой серию стандартов, определяющих обмен информацией между системами распределения электроэнергии.

– **IEC 61968-100(2022)** – интеграция приложений в электроэнергетику общего пользования. Системные интерфейсы для управления распределением. Часть 100. Профили реализации.

– **CENC** – сервер канала связи, основным назначением которого является обеспечение канала связи между устройствами, имеющих не постоянный (динамический) IP-адрес и ПО верхнего уровня.

– **eEnergO** - программное обеспечение предназначенное для измерения и многотарифного коммерческого учета электрической энергии и мощности, автоматизированного сбора, хранения, обработки и отображения данных по энергопотреблению.

– **HesDLMS** - протокол, регламентирующий обмен данными между приборами учета и системами сбора данных, в основе которого лежит клиент-серверная архитектура.

– **СПОДЭС** - спецификация протокола обмена данными электронных счетчиков.

– **МЭК-104** - протокол информационного обмена, реализованный в соответствии с ГОСТ Р МЭК 60870-5-104-2004.

– **MKSP-1EE** - модуль управления и мониторинга для электропитающих установок постоянного тока типа.

– **IMEI** - международный идентификатор мобильного оборудования.

– **ModBus** - открытый коммуникационный протокол. Применяется в промышленности для организации связи между электронными устройствами.

– **CEA** - протокол обмена УСПД 164-01М, CE805 и CE805М.

### **1.3. Назначение продукта**

Ce-NetConnection представляет собой систему для обеспечения возможности взаимодействия с приборами учета, просмотра и работы с данными приборов учета, возможности конфигурирования и обновления приборов учета.

Состоит из коммуникационной системы CENC в подсистеме обеспечения связи и сбора ceNES и прикладной системы MDM в подсистеме ceCloud. Более подробная информация о каждой из подсистем содержится в соответствующих разделах данной документации.

Ce-NetConnection предоставляет возможность запуска на ОС Windows и ОС Linux (Ubuntu, Debian), в том числе отечественных Ред ОС и Альт СП Сервер.

### **1.4. Технические требования**

Для корректной работы Ce-NetConnection компьютер должен соответствовать следующим минимальным требованиям:

- Минимальное разрешение экрана 1280x1024.
- Оперативная память от 8ГБ.
- Подключение к интернету.
- Браузер.

Рекомендованные браузеры:

- Google Chrome v.123.

– Firefox v.124.

– Opera v.109.

Операционные системы:

– Требования к ОС для серверной части должны соответствовать актуальным требованиям для установки Docker 4.28.x (с ядром Engine 25.x). Смотрите раздел 2.1 по установке Docker.

– Требования к ОС для клиентской части должны соответствовать требованиям браузеров, характеристикам монитора и оперативной памяти из пункта выше.

При развертывании приложения на ПК, выступающем сервером и клиентом, требования выше должны быть совмещены.



### **ВНИМАНИЕ!**

При использовании VPN и Proxu возможны сетевые проблемы или сложности у служб обеспечивающих работу Docker, WSL, Hyper-V. Ознакомьтесь с официальным руководством Docker и при необходимости обратитесь к системному администратору для консультации и решения совместного использования VPN, Proxu и Docker,

## **2. Подсистема ceHes**

«ceHes» – коммуникационная система для организации и обеспечения взаимодействия с приборами учёта. Областью применения в рамках данной версии является серверная (облачная) платформа в виде микросервисной архитектуры в Docker-контейнерах.

Обеспечивает интеграцию с внешними MDM системами потребителя через предоставление REST-API на основе стандарта IEC 61968-100 (2022).

Система позволяет организовывать связь и обеспечивает доступ к основным функциям приборов.

Поддерживаемые приборы учёта:

- CE207 SPODES (поддержка версий 10.x, 12.x).
- CE307 SPODES (поддержка версий 10.x, 12.x).
- CE208 SPODES (поддержка версий 10.x, 12.x).
- CE308 SPODES (поддержка версий 10.x, 12.x).

Поддерживаемые функции:

- Чтение данных измерений (в том числе профилей и параметров сети).
- Чтение журналов событий.
- Чтение состояния реле.
- Изменений (управление) состоянием реле.
- Чтение и запись (синхронизация) времени.

Основными областями применения ceNes являются:

- Интеллектуальные системы учета электроэнергии (ИСУЭ).
- Розничный рынок электроэнергии для электросетевых компаний.
- Управляющие компании: СНТ, ДНТ, ТСЖ, УК и другие.
- Объекты АСКУЭ «нетребовательных потребителей» с поддержкой приборов учёта по протоколу СПОДЭС.

## 2.1. Основные функции

В ceNes существует четыре роли пользователей по умолчанию: пользователь, оператор, администратор и m2m. У каждой роли свой набор разрешений по умолчанию:

- **Пользователь.** Имеет доступ к просмотру основных форм системы и чтения архивных данных показаний, состояний и событий счетчиков.

– **Оператор**. Имеет доступ уровня пользователь и дополнительные возможности:

– Управление устройствами: добавлять, редактировать, удалять, настраивать параметры каналов связи и протоколов.

– Управление реле устройств.

– Управление расписаниями задач.

– **Администратор**. Имеет доступ уровня оператор, а также доступ к управлению системой: настройка сервера, управление пользователями, просмотр логов.

– **m2m**. Имеет доступ к чтению списка устройств, данных и возможность обращаться к REST-API интеграции на основе стандарта IEC 61968-100(2022).

## 2.2. Установка

Для начала нужно подготовить систему, установив в ней Docker и Docker-Compose. Необходимо установить актуальную версию с [официального сайта](#).

На текущий момент это Docker Desktop 4.28.0 (Engine 25.0.3, Compose 2.24.6)

Ниже на рисунках (Рисунок 1) и (Рисунок 2) приведена демонстрация версий для Windows и Linux.





Рисунок 1 – Docker Desktop, просмотр версии

```
admin@cehes-dev: ~  
admin@cehes-dev:~$ docker --version  
Docker version 26.0.0, build 2ae903e  
admin@cehes-dev:~$ docker-compose --version  
Docker Compose version v2.26.0  
admin@cehes-dev:~$
```

Рисунок 2 – Просмотр версии Docker Desktop через консоль



### ВНИМАНИЕ!

Для установки на Windows необходимо иметь права локального администратора, а для Linux права уровня sudo/root.

#### 2.2.1. Установка Docker

Актуальные системные требования, описание процесса установки и ссылки на загрузку Docker текущей версии приведены в официальной документации по ссылкам:

Для Windows: <https://docs.docker.com/desktop/windows/install>.

Для Linux: <https://docs.docker.com/desktop/linux/install>.



### ВНИМАНИЕ!

Необходимо устанавливать Docker Engine не ниже версии 24.x.

## 2.2.2. Установка Docker Compose

Для Windows он будет установлен в составе Docker Desktop.

Для Linux, если установка производилась не с пакетом Docker Desktop, необходима ручная установка.

Ниже приведен пример установки для Ubuntu 22.04.

Начнем с определения последнего выпуска Docker Compose на странице выпусков (<https://github.com/docker/compose/releases>).



### ВНИМАНИЕ!

В примерах команд ниже замените версию v2.26.0 на актуальную.

Запустите следующую команду для загрузки Docker Compose и предоставьте глобальный доступ к этому ПО в своей системе:

```
$ sudo curl -k -L
"https://github.com/docker/compose/releases/download/v2.26.0/
docker-compose-$(uname -s)-$(uname -m)" -o
/usr/local/bin/docker-compose
```

Вариант запуска команды с указанием прокси-сервера:

```
$ sudo curl -x 'http://10.5.0.9:3128' -L
"https://github.com/docker/compose/releases/download/v2.
26.0/docker-compose-$(uname -s)-$(uname -m)" -o
/usr/local/bin/docker-compose
```

Вариант запуска команды с указанием прокси-сервера и игнорирование SSL сертификата (параметр `-k` или `--insecure`):

```
$ sudo curl -k -x 'http://10.5.0.9:3128' -L
"https://github.com/docker/compose/releases/download/v2.
26.0/docker-compose-$(uname -s)-$(uname -m)" -o
/usr/local/bin/docker-compose
```

Затем необходимо задать правильные разрешения, чтобы сделать команду `docker-compose` исполняемой:

```
$ sudo chmod +x /usr/local/bin/docker-compose
```

Чтобы проверить успешность установки, запустите следующую команду:

```
$ sudo docker-compose --version
```

Вывод будет выглядеть следующим образом:

```
$ Docker Compose version v2.26.0.
```

### 2.2.3. Развёртывание и запуск проекта

После того как `Docker` и `Docker-Compose` установлены, достаточно запустить команду развёртывания проекта из репозитория Nexus (сервис Энергомера Софт).

Для этого необходимо скопировать файлы `docker-compose.yml` и `.env` в любую папку (не рекомендуется использовать длинные пути в папках или кириллические символы).

После чего, перейдя к папке с файлами в консоли, выполнить команду (см. шаг 3 в разделе 2.4 Обновление):

Ubuntu:

```
$ sudo docker-compose -f docker-compose.hes.yaml -p hes  
up -d
```

Windows:

```
docker-compose -f docker-compose.hes.yaml -p hes up -d
```



### ВНИМАНИЕ!

Для обеспечения безопасности рекомендуется в `docker-compose.yml` перед развертыванием, прописать пароль системного администратора СУБД PostgreSQL (заменить `root` на требуемый в следующих параметрах - `POSTGRES_USER=root - POSTGRES_PASSWORD=root`).



### ВНИМАНИЕ!

Для Windows консоль управления необходимо открыть от имени администратора.



### ВНИМАНИЕ!

Для успешного выполнения всех действий необходимо наличие интернета. В случае, если интернет доступен через прокси-сервер, то необходимо настроить систему и Docker на работу через него (рекомендуется использовать интернет без прокси-сервера).



### ВНИМАНИЕ!

Для исключения бесконтрольного расширения дискового пространства при внутренней процедуре логирования Docker консольного

вывода контейнеров необходимо настроить ограничения на файлы логов Docker (официальная документация доступна по [ссылке](#)).

Например, добавив ограничения:

```
"log-opts": {  
  "max-file": "5",  
  "max-size": "10m"  
}
```

Это не более 5 файлов логов архива с размером не более 10МБ.

Для Docker Desktop под Windows можно выполнить настройки (Рисунок 3).

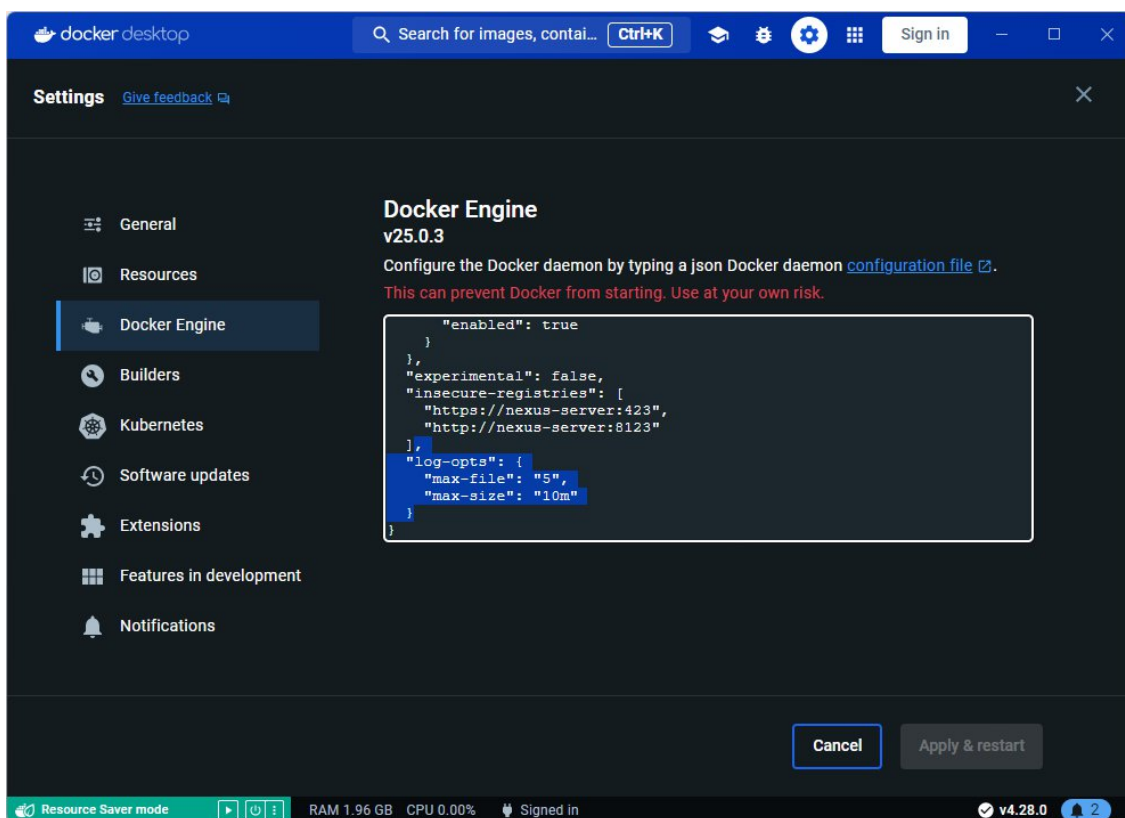


Рисунок 3 – Настройка ограничений

После изменения настроек необходимо нажать кнопку «Apply & restart».



### ВНИМАНИЕ!

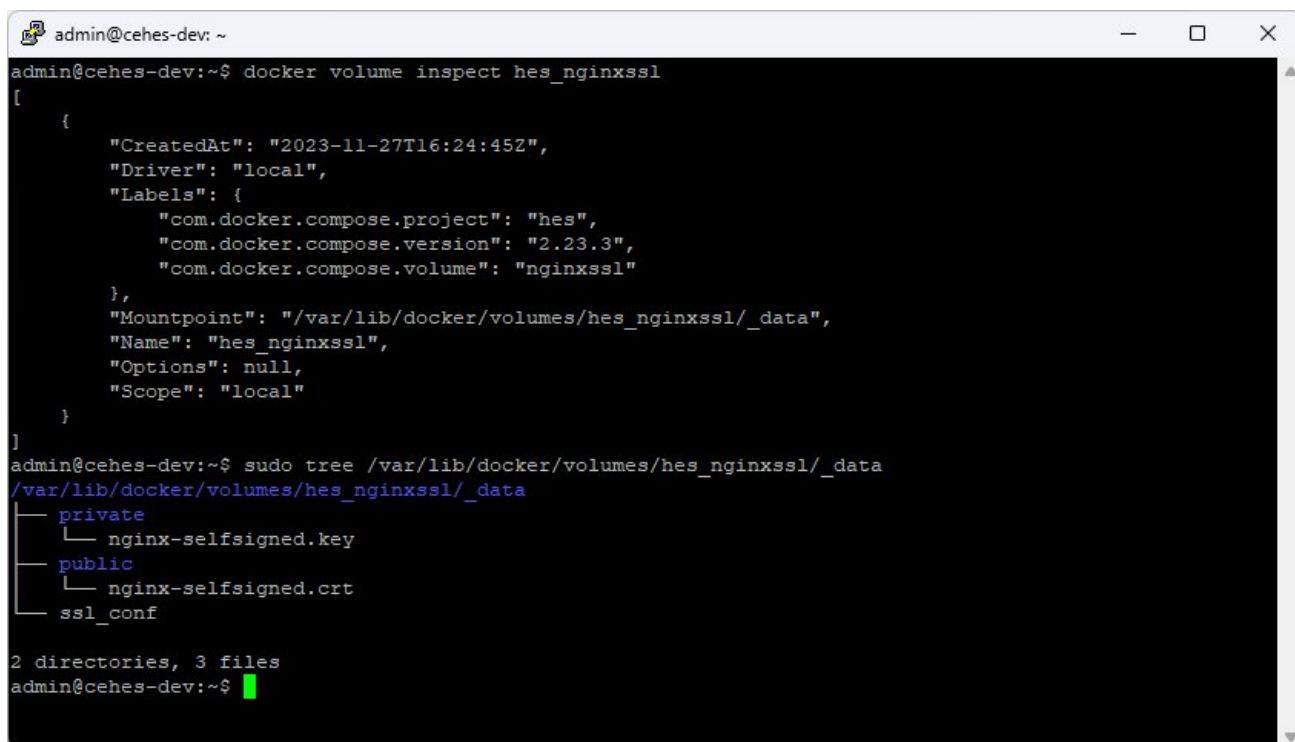
Возможна установка и настройка сертификата безопасности для подключения по HTTPS.

При необходимости его можно установить по пути хранения тома cloud\_nginxssl. Путь можно узнать в параметре Mountpoint, выполнив команду:

```
docker volume inspect hes_nginxssl
```

Пример вывода (Рисунок 4):

```
[
  {
    "CreatedAt": "2023-11-27T16:24:45Z",
    "Driver": "local",
    "Labels": {
      "com.docker.compose.project": "hes",
      "com.docker.compose.version": "2.23.3",
      "com.docker.compose.volume": "nginxssl"
    },
    "Mountpoint":
"/var/lib/docker/volumes/hes_nginxssl/_data",
    "Name": "hes_nginxssl",
    "Options": null,
    "Scope": "local"
  }
]
```



```
admin@cehes-dev: ~
admin@cehes-dev:~$ docker volume inspect hes_nginxssl
[
  {
    "CreatedAt": "2023-11-27T16:24:45Z",
    "Driver": "local",
    "Labels": {
      "com.docker.compose.project": "hes",
      "com.docker.compose.version": "2.23.3",
      "com.docker.compose.volume": "nginxssl"
    },
    "Mountpoint": "/var/lib/docker/volumes/hes_nginxssl/_data",
    "Name": "hes_nginxssl",
    "Options": null,
    "Scope": "local"
  }
]
admin@cehes-dev:~$ sudo tree /var/lib/docker/volumes/hes_nginxssl/_data
/var/lib/docker/volumes/hes_nginxssl/_data
├── private
│   └── nginx-selfsigned.key
├── public
│   └── nginx-selfsigned.crt
└── ssl_conf

2 directories, 3 files
admin@cehes-dev:~$
```

Рисунок 4 – Пример вывода в консоли

По пути `/var/lib/docker/volumes/hes_nginxssl/_data` в подпапках `private` и `public` необходимо расположить файлы ключа (`nginx-selfsigned.key`) и сертификата (`nginx-selfsigned.crt`).

Пример подготовки самоподписанного сертификата см. далее [3.1. Настройка сертификата](#). Нужно обратить внимание на обязательное наличие в сертификате всех вариантов альтернативных имён DNS и IP в `[alt_names]`.

После успешного развертывания контейнеров веб-интерфейс приложения будет доступен по ссылке <https://localhost> либо по адресу IP сервера, на котором оно было развёрнуто.

#### 2.2.4. Обновление

Рекомендуется следующая последовательность шагов:

1) Остановить сервисы (Рисунок 5). Этот шаг является необязательным.

Ubuntu:

```
sudo docker-compose -f docker-compose.hes.yaml -p hes  
down
```

Windows:

```
docker-compose -f docker-compose.hes.yaml -p hes down
```

```
admin@cehes-dev: ~/hes
admin@cehes-dev:~/hes$ sudo docker-compose -p hes down
[*] Running 20/20
  Container hes-nginx-gateway   Removed      3.4s
  Container hes-logger          Removed      4.6s
  Container hes-email           Removed      3.4s
  Container hes-exchanger       Removed      4.3s
  Container hes-maintenance-api  Removed     10.9s
  Container hes-esb-iec61968     Removed      4.6s
  Container hes-localizator     Removed      3.3s
  Container hes-synchronizer    Removed      4.6s
  Container hes-data-gateway     Removed      4.3s
  Container hes-dev-docs        Removed      3.4s
  Container hes-commandgateway   Removed      3.7s
  Container hes-web              Removed      3.4s
  Container hes-scheduler       Removed      3.0s
  Container hes-devices         Removed      0.8s
  Container hes-datastore       Removed      2.8s
  Container hes-identity        Removed      1.6s
  Container api-gateway         Removed      0.6s
  Container hes-psqlserver      Removed      0.8s
  Container hes-rabbitmq-1     Removed      7.2s
  Network hes_hes-net          Removed      0.3s
admin@cehes-dev:~/hes$
```

Рисунок 5 – Остановка сервисов

2) Обновить сервисы (Рисунок 6). Этот шаг можно выполнить повторно, чтобы убедиться, что все обновления прошли успешно и более система не обнаруживает новые версии образов.

Ubuntu:

```
sudo docker-compose -f docker-compose.hes.yaml -p hes pull
```

Windows:

```
docker-compose -f docker-compose.hes.yaml -p hes pull
```

```
admin@cehes-dev: ~/hes
admin@cehes-dev:~/hes$ sudo docker-compose -p hes pull
[*] Pulling 19/19
  logger Pulled          0.9s
  data-gateway Pulled    0.9s
  api-gateway Pulled     0.8s
  datastore Pulled       0.8s
  scheduler Pulled       0.8s
  hes-commandgateway Pulled 1.0s
  devices Pulled         1.1s
  hes-web Pulled         0.9s
  localizator Pulled     0.9s
  hes-synchronizer Pulled 0.8s
  hes-maintenance-api Pulled 1.0s
  hes-esb-iec61968 Pulled 0.9s
  psqlserver Pulled      0.8s
  rabbitmq Pulled        1.1s
  exchanger Pulled       1.0s
  hes-dev-docs Pulled    1.1s
  identity Pulled        1.1s
  hes-nginx-gateway Pulled 1.0s
  email Pulled           1.0s
admin@cehes-dev:~/hes$
```

Рисунок 6 – Обновление сервисов



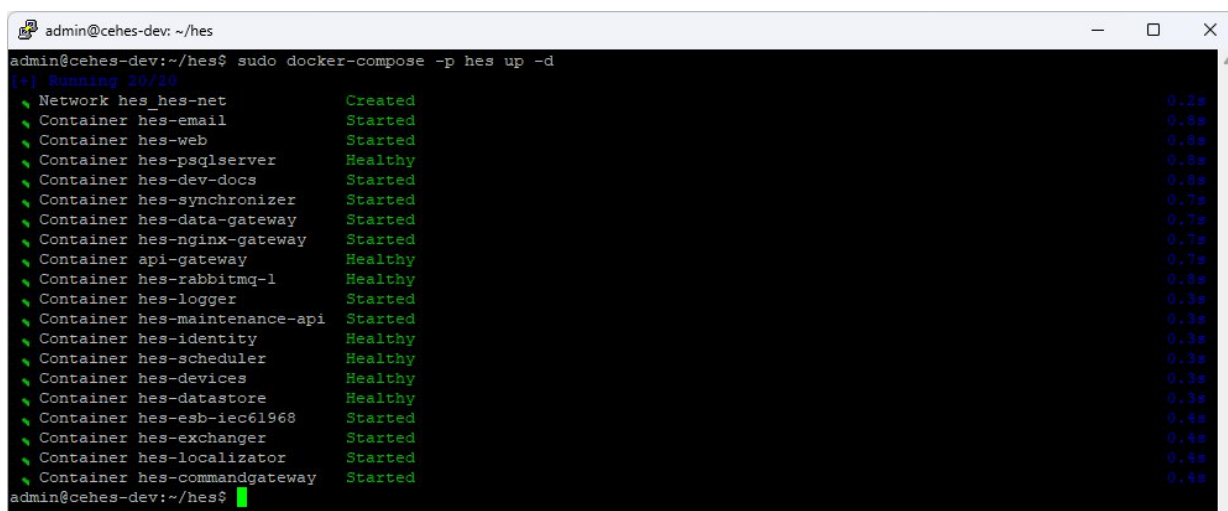
3) Запустить сервисы (Рисунок 7). Этот шаг может потребовать некоторое время, которое необходимо сервисам для инициализации настроек при их старте. Необходимо дождаться, пока у каждого сервиса будет выведено состояние Started или Healthy.

Ubuntu:

```
sudo docker-compose -f docker-compose.hes.yaml -p hes up -d
```

Windows:

```
docker-compose -f docker-compose.hes.yaml -p hes up -d
```



```
admin@cehes-dev: ~/hes
admin@cehes-dev:~/hes$ sudo docker-compose -p hes up -d
[*] Running 30/30
✔ Network hes_hes-net          Created          0.3s
✔ Container hes-email         Started          0.8s
✔ Container hes-web           Started          0.8s
✔ Container hes-psqlserver    Healthy         0.8s
✔ Container hes-dev-docs     Started          0.8s
✔ Container hes-synchronizer  Started          0.7s
✔ Container hes-data-gateway  Started          0.7s
✔ Container hes-nginx-gateway Started          0.7s
✔ Container api-gateway       Healthy         0.7s
✔ Container hes-rabbitmq-1    Healthy         0.8s
✔ Container hes-logger        Started          0.3s
✔ Container hes-maintenance-api Started          0.3s
✔ Container hes-identity      Healthy         0.3s
✔ Container hes-scheduler     Healthy         0.3s
✔ Container hes-devices       Healthy         0.3s
✔ Container hes-datastore     Healthy         0.3s
✔ Container hes-esb-iec61968  Started          0.4s
✔ Container hes-exchanger     Started          0.4s
✔ Container hes-localizator   Started          0.4s
✔ Container hes-commandgateway Started          0.4s
admin@cehes-dev:~/hes$
```

Рисунок 7 – Запуск сервисов

## 2.3. Конфигурирование

### 2.3.1. Настройка сертификата

Файл сертификата должен быть расположен по пути хранения тома hes\_nginxssl. (см. подраздел [2.2.3. Развёртывание и запуск проекта](#), в разделе [2. Установка](#)): /var/lib/docker/volumes/hes\_nginxssl/\_data. Имя файла должно быть nginx-selfsigned.crt

Для генерации самоподписанного сертификата, можно использовать openssl.

Пример создания самоподписанного сертификата для Linux:

1. Создайте новую папку для работы и перейдите в неё:

```
mkdir /new-certs
cd /new-certs
```

2. Сгенерируйте приватный ключ:

```
openssl genrsa -out nginx-selfsigned.key 2048
```

3. Создайте текстовый файл `ssl_conf` следующего содержания:

```
[req]
default_bits      = 4096
prompt           = no
default_md       = sha256
x509_extensions  = v3_req
distinguished_name = dn

[dn]
C                = RU
ST               = {край/область/штат}
L               = {город/населенный пункт}
O               = {организация}
CN              = {доменное имя сервера}
emailAddress    = {электронная почта}

[v3_req]
subjectAltName = @alt_names

[alt_names]
DNS.1          = {доменное имя сервера}
IP.1           = {ip-адрес сервера}
```



### ВНИМАНИЕ!

Замените значения в фигурных скобках на валидные.

4. Сгенерируйте сертификат командой:

```
openssl req -new -x509 -key nginx-selfsigned.key -days
730 -out nginx-selfsigned.crt -config <(cat ssl_conf)
```

Альтернативный способ генерирования ключа без файла конфигурации:

```
sudo openssl req -x509 -nodes -days 730 -newkey rsa:2048
-out selfsigned.crt -keyout nginx-selfsigned.key -addext
"subjectAltName = DNS.1:{короткое доменное имя сервера},
DNS.2:{доменное имя сервера}, IP.1:{ip-адрес сервера}"
```

В данном случае все параметры, описанные в файле конфигурации, будут запрашиваться для ввода в процессе генерации ключа.

5. Переместите ключ `nginx-selfsigned.key` в папку `private`, которая была примонтирована к сервисам `nginxserver`.

6. Переместите сертификат `nginx-selfsigned.crt` в папку `public`, которая была примонтирована к сервисам `nginxserver`.

7. Перезапустите сервисы.

Подробности можно изучить в [официальной документации OpenSSL](#).

### 2.3.2. Настройка сервиса сбора

Для настройки системы сбора возможно использовать переменные окружения.

Для этого в файле `docker-compose.yml` в списке сервисов необходимо найти сервис `exchanger` и установить требуемые значения переменных окружения. Например, для установки работы сбора в 1000 потоков:

```
exchanger:
  environment:
    - exchange__concurrency=1000
```

Список переменных окружения для сбора (Таблица 1):

Таблица 1 – Список переменных окружения для сбора

Переменная окружения	Значение по умолчанию	Описание
<code>exchange__concurrency</code>	10	Максимальное количество параллельных потоков сбора/отправок команд
<code>exchange_jobLifetime</code>	00:05:00	Максимальное время

		жизни задачи сбора/отправки команды, после которого она будет отменена, в случае если отсутствует активность канала связи
--	--	---

Переменные окружения также можно настроить, введя требуемые значения в соответствующий файл **.env**, расположенный рядом с **docker-compose.yaml**.

Пример:

<pre>EXCHANGER_CONCURRENCY=1000 EXCHANGER_JOB_LIFETIME=00:05:00 HES_NAME=DEMO</pre>
---

### 2.3.3. Настройка сервиса DeviceMaintenance

Сервис DeviceMaintenance – содержит ряд переменных окружения, которые могут использоваться для конфигурирования поведения сервиса

Список переменных окружения (Таблица 2):

Таблица 2 – Список переменных окружения сервиса DeviceMaintenance

Переменная окружения	Значение по умолчанию	Описание
HostInfo__Name	UNAVAILABLE	Имя хоста, которое будет отображаться на дашборде и странице с информацией о хосте в поле «Серийный номер».
HostInfo__IsCloud	true	Флаг того, что ПО запущено в облаке

Переменные окружения также можно настроить, введя требуемые значения в соответствующий файл **.env**, расположенный рядом с **docker-compose.yaml**.

### 2.3.4. Настройка сервиса Identity

Сервис Identity – содержит ряд переменных окружения, которые могут использоваться для конфигурирования поведения сервиса (Таблица 3).

Таблица 3 – Список переменных окружения сервиса Identity

Переменная окружения	Значение по умолчанию	Описание
HostURI		URI/IP-адрес хоста Параметр необходимо указать в соответствии с настроенной инфраструктурой, для возможности корректного восстановления пароля пользователя
AuthSettings__Signing		Ключ для подписи JWE – токена
AuthSettings__Encryption		Ключ для шифрования JWE-токена

### 2.3.5. Настройка сервиса CENC

Сервис CENC – содержит следующий ряд переменных окружения, которые могут использоваться для конфигурирования его поведения.

Список переменных окружения указан в Таблица 4.

Таблица 4 – Список переменных окружения сервиса CENC

Переменная окружения	Значение по умолчанию	Описание
CENC__Device__Listen__0	11001	Порт для подключения устройств по TCP/UDP
CENC__Device__Listen__1	11002	Порт для подключения устройств по TCP/UDP
CENC AMR Listen 0	22002	Порт для подключения ПО



**ВНИМАНИЕ!**

Числа в конце 0, 1 – индексы в массиве, нумерация в котором начинается с 0; таким образом, в случае если необходимо настроить 100 портов – последнее число - 99.

Настройка отправки событий в ceHES.

Необходимо указать массив событий, которые должны отправляться в ceHES.

Список событий, которые может отправлять CENC, приведен в REF [\\_Ref12 \h \\\* MERGEFORMAT](#) Таблице 5.

Таблица 5 – События, отправляемые CENC

Событие	Описание
SET_CENC_DEVICE_CONNECTED	Устройство подключено
SET_CENC_DEVICE_DISCONNECTED	Устройство отключено
SET_CENC_DEVICE_ACCESS_START	Начало использования CENC в качестве канала связи
SET_CENC_DEVICE_ACCESS_FINISH	Окончание использования CENC в качестве канала связи
SET_CENC_ERROR	Системная ошибка CENC
SET_CENC_AMR_USER_ADDED	AMR-агент добавлен
SET_CENC_AMR_USER_UPDATE_NAME	Обновлено имя AMR-агента
SET_CENC_AMR_USER_UPDATE_PASSWORD	Обновлен пароль AMR-агента

Для настройки событий, можно воспользоваться переменными среды и передать необходимые события из docker-compose файла.

Пример:

```
Plugins__Energomera.Hes.CENC.Plugins.CEHesIntegrationPlugin__notifications__0=SET_CENC_DEVICE_CONNECTED
Plugins__Energomera.Hes.CENC.Plugins.CEHesIntegrationPlugin__notifications__1=SET_CENC_DEVICE_DISCONNECTED
```

В данном примере – настроили передачу событий SET\_CENC\_DEVICE\_CONNECTED, SET\_CENC\_DEVICE\_DISCONNECTED в ceHES.

По умолчанию CENC отправляет события:

- SET\_CENC\_DEVICE\_CONNECTED
- SET\_CENC\_DEVICE\_DISCONNECTED
- SET\_CENC\_ERROR
- SET\_CENC\_AMR\_USER\_ADDED
- SET\_CENC\_AMR\_USER\_UPDATE\_NAME
- SET\_CENC\_AMR\_USER\_UPDATE\_PASSWORD

Пример пользовательской настройки сервиса CENC в docker-compos'e:

```
hes-cenc:
  restart: always
  image: hub.energomera.ru/hes/hes.cenc
  container_name: hes-cenc
  hostname: hes.cenc
  networks:
    - hes-net
  ports:
    - "65000:65000/tcp"
    - "65000:65000/udp"
    - "65001:65001/tcp"
    - "65001:65001/udp"
  # Пример на случай, если на хост-машине проброс должен отличаться
    - "5000:65002/tcp"
    - "5000:65002/udp"
    - "64000:64000/tcp"
  # Пример на случай, если на хост-машине проброс должен отличаться
    - "5001:64001/tcp"
  # Для проброса портов прямого доступа
  # - "7500-8000:7500-8000"
  environment:
    # кастомные порты прослушивания подключения устройств
    # внутри контейнера
```

```

# их необходимо смаппить с портами хоста в секции `ports`
- CENC__Device__Listen__0=65000
- CENC__Device__Listen__1=65001
- CENC__Device__Listen__2=65002
# кастомные порты прослушивания подключения ПО верхнего
уровня
# внутри контейнера
# их необходимо смаппить с портами хоста в секции `ports`
- CENC__AMR__Listen__0=64000
- CENC__AMR__Listen__1=64001
depends_on:
  rabbitmq:
    condition: service_healthy
  devices:
    condition: service_healthy
  identity:
    condition: service_healthy
  psqlserver:
    condition: service_healthy

```

### 2.3.6. Настройка сервиса Logger

Сервис Logger – служит для хранения и предоставления логов и событий системы. Так же сервис выполняет операции очистки логов, в соответствии с переданными настройками.

Базовая настройка очистки логов может быть осуществлена с помощью переменных окружения (Таблица 6).

Таблица 6 – Список переменных окружения сервиса Logger

Переменная окружения	Значение по умолчанию	Описание
DebugLogsLifetime__Fatal	60.00:00:00	Время хранения отладочных логов уровня Fatal
DebugLogsLifetime__Error	60.00:00:00	Время хранения отладочных логов уровня Error
DebugLogsLifetime__Warning	60.00:00:00	Время хранения отладочных логов уровня Warning



DebugLogsLifetime__Information	10.00:00:00	Время хранения отладочных логов уровня Information
DebugLogsLifetime__Debug	2.00:00:00	Время хранения отладочных логов уровня Debug
DebugLogsLifetime__Trace	1.00:00:00	Время хранения отладочных логов уровня Trace

### 2.3.7. Настройка сервиса DataStore

Сервис DataStore служит для хранения и предоставления измерений, журналов событий, состояний, истории выполненных команд. Так же сервис выполняет операции очистки собранных данных, в соответствии с переданными настройками.

Некоторые настройки, которые при необходимости можно отредактировать, вынесены в переменные окружения (Таблица 7).

Таблица 7 – Список переменных окружения сервиса DataStore

Переменная окружения	Значение по умолчанию	Описание
archiveClear/launchFrequency	1.00:00:00	Периодичность запуска задачи очистки данных
archiveClear/functionDepth	365.00:00:00	Глубина хранения журнала истории выполненных команд

### 2.3.8. Настройка системы логирования

Все сервисы, по умолчанию, имеют возможность логгировать в файл, консоль, либо в СУБД.

В случае необходимости – каждому сервису можно индивидуально настроить логгирование. Для этого – необходимо подключиться к контейнеру и произвести редактирование файла `nlog.config`, который расположен в папке с приложением(/app).

Формат и примеры настройки файла `nlog.config` может быть найден по ссылке <https://nlog-project.org/config/>



### 3. Подсистема seCloud

«seCloud» - это облачный сервис, предназначенный для удаленной работы с приборами учёта. Сервис позволяет отслеживать данные о приборах с помощью графиков, таблиц и журналов событий с помощью стандартного web-браузера.

Клиент может использовать seCloud на своих серверах, либо воспользоваться услугой по размещению сервиса на серверах правообладателя.

Основными областями применения seCloud являются:

- Интеллектуальные системы учета электроэнергии (ИСУЭ).
- Розничный рынок электроэнергии для электросетевых компаний.
- Управляющие компании: СHT, ДHT, ТСЖ, УК и другие.
- На объектах АСКУЭ «нетребовательных потребителей», с поддержкой приборов учёта по протоколу СПОДЭС.

К дополнительным областям применения относятся:

- Системы мониторинга электрохимзащиты (ЭХЗ) по протоколу MODBUS.
- Системы мониторинга электропитающих устройств (ЭПУ) по протоколу SNMP.

#### 3.1. Основные функции

В seCloud существует три типа прав пользователей: абонент, менеджер, администратор. Каждому типу доступны свои программные функции:

- **Абонент.** Имеет доступ к учетной записи, где отображается информация об абоненте, заключенных договорах, и показание счетчиков.

– **Менеджер.** Имеет доступ к списку проектов, к информации о системе, к данным по энергопотреблению, к данным о параметрах сети, к данным о телеметрии, к данным о журналах событий. Также менеджеру доступны следующие функции:

– Управление устройствами: добавлять, редактировать, удалять, заменять, демонтировать, отключать.

– Импортирование данных.

– Управление реле устройств.

– Просмотр и редактирование информации на геокарте.

– Управление расписаниями сервисов.

– Работа с устройствами по протоколу Modbus.

– Работа с устройствами по протоколу SNMP.

– Работа с устройствами по протоколу IEC104.

– **Администратор.** Имеет те же права, что и менеджер, а также доступ к управлению системой: настройка сервера, управление пользователями, настройка импорта из систем сEnergo и HesDLMS, просмотр логов.

## 3.2. Установка

Для начала нужно подготовить систему установив в ней Docker и Docker-Compose.

Для работы с приборами учёта (связь с ними, сбор показаний, передача команд управления реле) в текущей версии 1.1 необходима установка сEnergo 4.8 (*только Windows*). Интеграция реализуется через сервис IntegratorСenergo путем взаимодействия с БД сEnergo напрямую (см. руководство пользователя).

**Внимание!** Для установки необходимо иметь права локального администратора.

### 3.2.1. Установка Docker

Данный шаг можно пропустить, если Docker уже установлен.

Актуальные системные требования, описание процесса установки и ссылки на загрузку Docker текущей версии приведены в официальной документации по ссылкам:

Для Windows: <https://docs.docker.com/desktop/windows/install>.

Для Linux: <https://docs.docker.com/desktop/linux/install>.

### 3.2.2. Установка Docker Compose

Данный шаг можно пропустить, если Docker Compose уже установлен.

Для Windows он будет установлен в составе Docker Desktop.

Для Linux если установка производилась не пакетов Docker Desktop либо необходима отдельная установка возможна установка вручную.

Ниже приведен пример установки для Ubuntu 22.04.

Начнем с определения последнего выпуска Docker Compose на странице выпусков (<https://github.com/docker/compose/releases>). На момент написания настоящего документа наиболее актуальной стабильной версией является версия 2.14.0.

Запустите следующую команду для загрузки Docker Compose и предоставьте глобальный доступ к этому ПО в своей системе как docker-compose:

```
$ sudo curl -k -L
"https://github.com/docker/compose/releases/download/v2.14.0/docker-compose-
$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

Вариант запуска команды с указанием прокси сервера:

```
$ sudo curl -x 'http://10.5.0.9:3128' -L
"https://github.com/docker/compose/releases/download/v2.14.0/docker-
compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

Вариант запуска команды с указанием прокси сервера и игнорирование SSL сертификата (параметр -k или --insecure):

```
$ sudo curl -k -x 'http://10.5.0.9:3128' -L
"https://github.com/docker/compose/releases/download/v2.14.0/docker-
compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

Затем необходимо задать правильные разрешения, чтобы сделать команду docker-compose исполняемой:

```
$ sudo chmod +x /usr/local/bin/docker-compose
```

Чтобы проверить успешность установки, запустите следующую команду:

```
$ sudo docker-compose --version
```

Вывод будет выглядеть следующим образом:

```
Docker Compose version v2.14.0
```

### 3.2.3. Развёртывание и запуск проекта

После того как Docker и Docker-Compose установлены достаточно запустить команду развёртывания проекта из репозитория DockerHub.

Для этого необходимо скопировать файл docker-compose.yml в любую папку и перейдя к ней в командной консоли выполнить команду:

Ubuntu:

```
$ sudo docker-compose -f docker-compose.cloud.yaml -p cloud up -d
```

Windows:

```
$ docker-compose -f docker-compose.cloud.yaml -p cloud up -d
```

**Внимание!** Для обеспечения безопасности рекомендуется в docker-compose.yml перед развёртыванием прописать пароль системного администратора СУБД PostgreSQL (заменить root на требуемый в следующих параметрах - POSTGRES\_USER=root - POSTGRES\_PASSWORD=root).

**Внимание!** Для Windows консоль управления необходимо открыть от имени администратора.

**Внимание!** Для успешного выполнения всех действие необходимо наличие интернета. В случае если интернет доступен через прокси-сервер, то необходимо настроить систему и Docker на работу через него (рекомендуется использовать интернет без прокси-сервера).

**Внимание!** Для исключения бесконтрольного расширения дискового пространства при внутренней процедуре логирования Docker консольного вывода контейнеров необходимо настроить ограничения на файлы логов Docker (официальная документация доступна по [ссылке](#)).

Например, добавив ограничения:

```
"log-opts": {  
  "max-file": "5",  
  "max-size": "10m"  
}
```

Это не более 5 файлов логов архива, с размером не более 10МБ.

Для Docker Desktop под Windows можно выполнить настройки, как на изображении ниже (Рисунок 8).

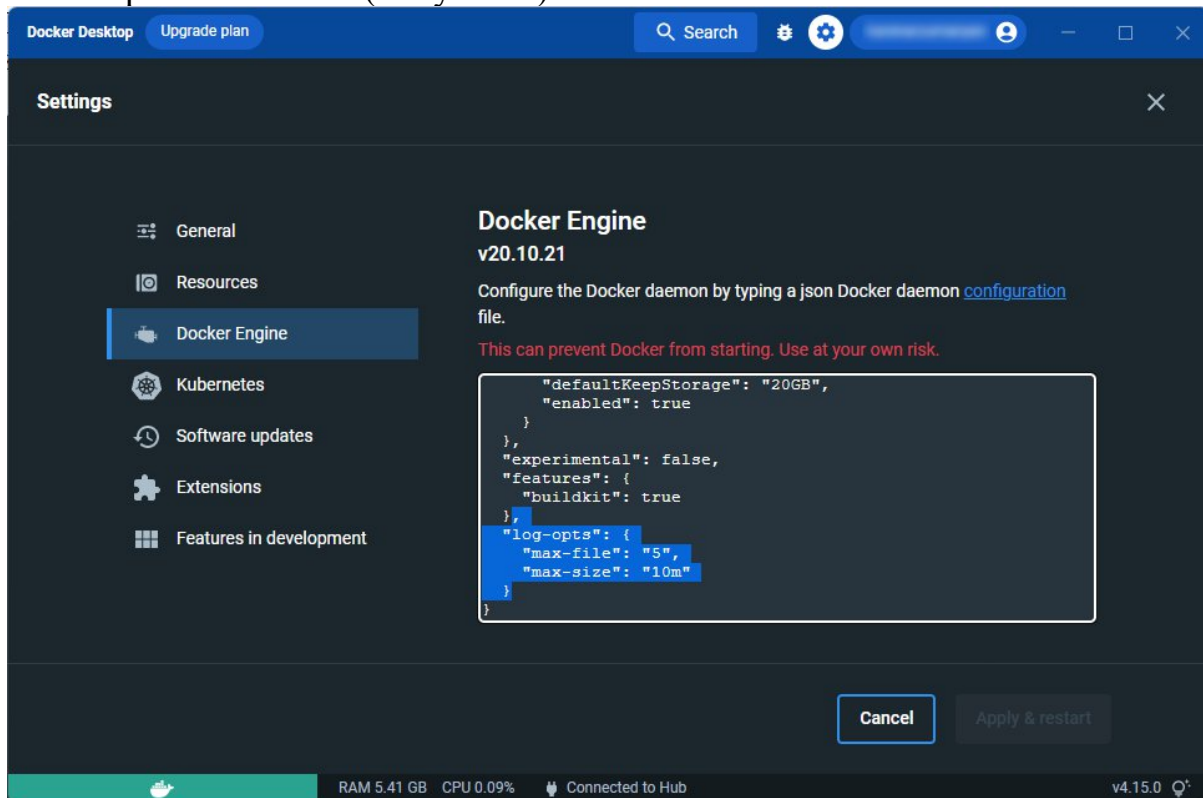


Рисунок 8

После изменения настроек необходимо нажать кнопку «Apply&restart».

**Внимание!** Для настройки сертификата безопасности подключения по HTTPS возможна его установка.

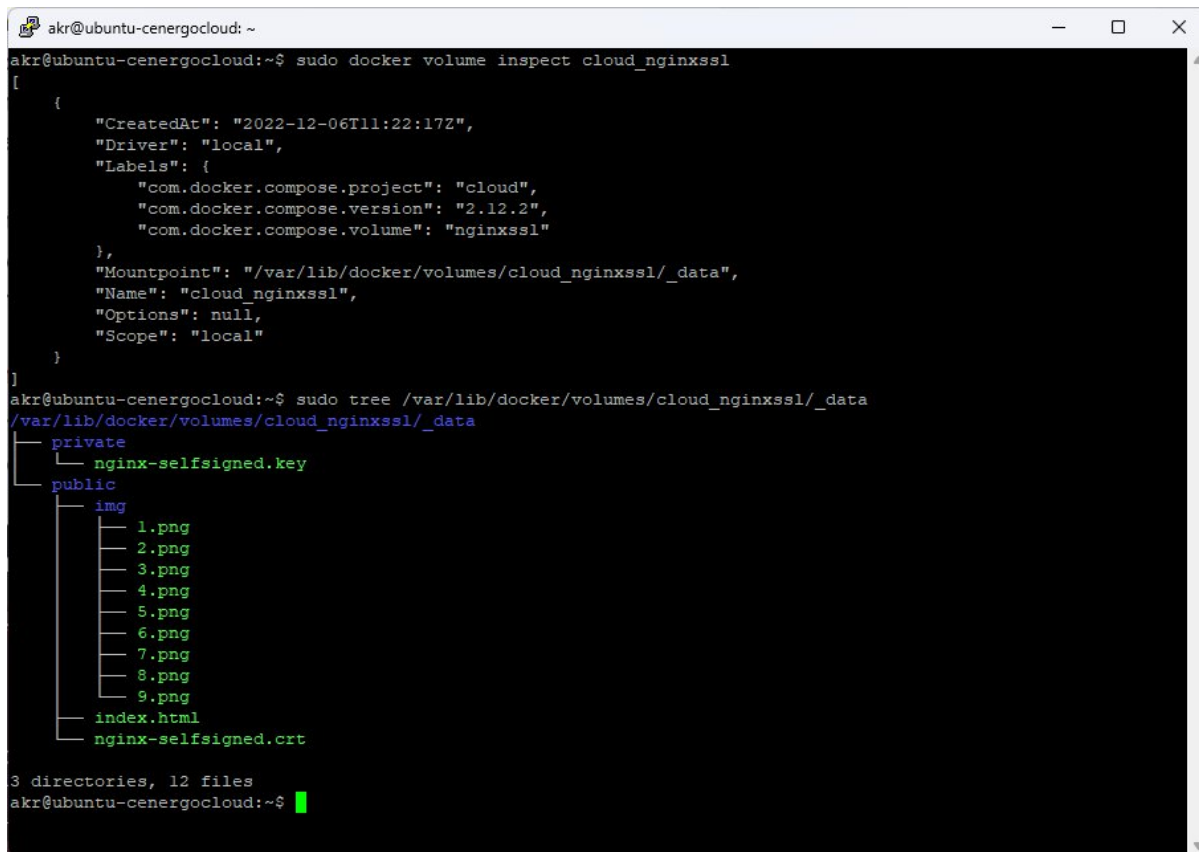
При необходимости его можно установить по пути хранения тома cloud\_nginxssl. Путь можно узнать в параметре Mountpoint, выполнив команду:

```
$ docker volume inspect cloud_nginxssl
```

Пример вывода (Рисунок 9):

```
[  {    "CreatedAt": "2022-11-29T13:24:59Z",    "Driver": "local",    "Labels": {      "com.docker.compose.project": "cloud",      "com.docker.compose.version": "2.12.2",      "com.docker.compose.volume": "nginxssl"    },    "Mountpoint": "/var/lib/docker/volumes/cloud_nginxssl/_data",    "Name": "cloud_nginxssl",
```

```
"Options": null,  
"Scope": "local"  
}  
]
```



```
akr@ubuntu-cenergocloud: ~  
akr@ubuntu-cenergocloud:~$ sudo docker volume inspect cloud_nginxssl  
[  
  {  
    "CreatedAt": "2022-12-06T11:22:17Z",  
    "Driver": "local",  
    "Labels": {  
      "com.docker.compose.project": "cloud",  
      "com.docker.compose.version": "2.12.2",  
      "com.docker.compose.volume": "nginxssl"  
    },  
    "Mountpoint": "/var/lib/docker/volumes/cloud_nginxssl/_data",  
    "Name": "cloud_nginxssl",  
    "Options": null,  
    "Scope": "local"  
  }  
]  
akr@ubuntu-cenergocloud:~$ sudo tree /var/lib/docker/volumes/cloud_nginxssl/_data  
/var/lib/docker/volumes/cloud_nginxssl/_data  
├── private  
│   └── nginx-selfsigned.key  
├── public  
│   └── img  
│       ├── 1.png  
│       ├── 2.png  
│       ├── 3.png  
│       ├── 4.png  
│       ├── 5.png  
│       ├── 6.png  
│       ├── 7.png  
│       ├── 8.png  
│       └── 9.png  
└── index.html  
    nginx-selfsigned.crt  
  
3 directories, 12 files  
akr@ubuntu-cenergocloud:~$
```

Рисунок 9

По пути `/var/lib/docker/volumes/cloud_nginxssl/_data` в подпапках `private` `public` необходимо расположить файлы ключа (`nginx-selfsigned.key`) и сертификата (`nginx-selfsigned.crt`).

Пример подготовки самоподписанного сертификата см. далее 3.1.1 Регистрация в один клик, в разделе SNMP. Нужно обратить внимания на обязательное наличие в сертификате всех вариантов альтернативных имён DNS и IP в `[alt_names]`.

После успешного развертывания контейнеров веб интерфейс проекта будет доступен по адресу `https://localhost`

### 3.2.4. Обновление

Рекомендуется следующая последовательность шагов:

1) Остановить сервисы (Рисунок 10). Этот шаг является необязательным.

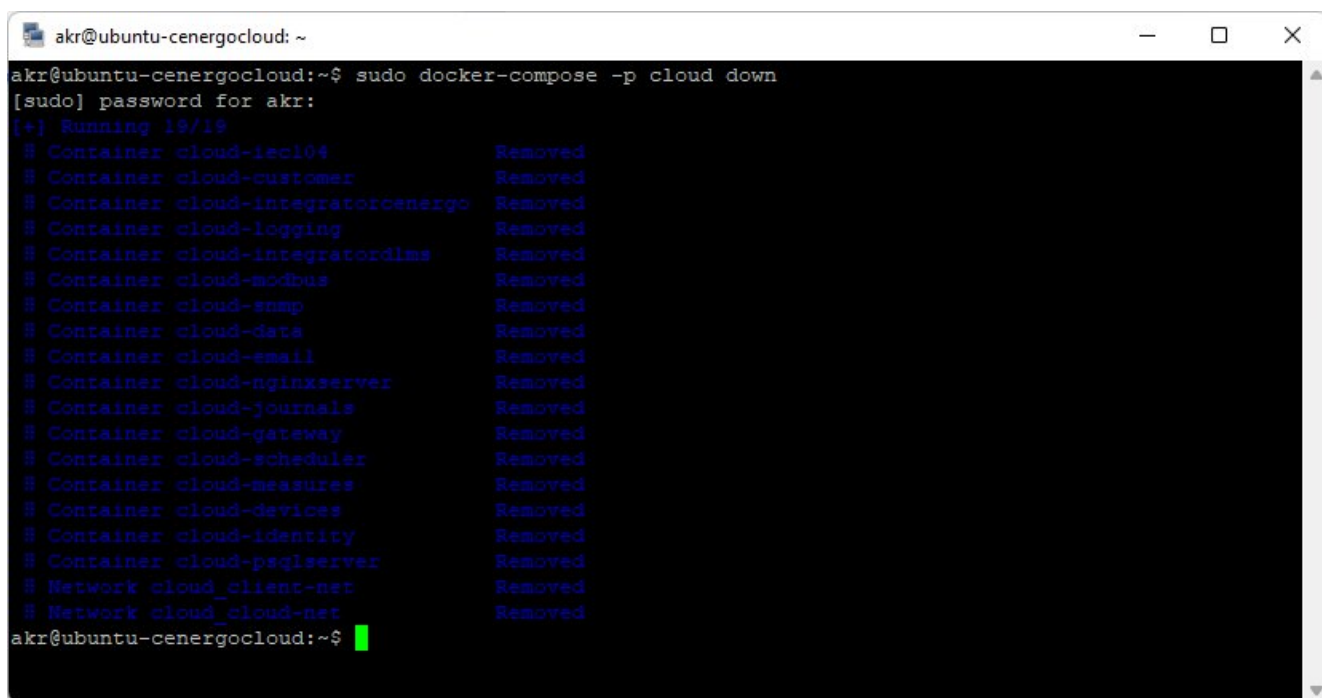


Ubuntu:

```
$ sudo docker-compose -f docker-compose.cloud.yaml -p cloud down
```

Windows:

```
$ docker-compose -f docker-compose.cloud.yaml -p cloud down
```



```
akr@ubuntu-cenergocloud: ~  
akr@ubuntu-cenergocloud:~$ sudo docker-compose -p cloud down  
[sudo] password for akr:  
(+) Running 19/19  
# Container cloud-iacl04 Removed  
# Container cloud-customer Removed  
# Container cloud-integratorcenergo Removed  
# Container cloud-logging Removed  
# Container cloud-integratordims Removed  
# Container cloud-mcdbus Removed  
# Container cloud-snap Removed  
# Container cloud-data Removed  
# Container cloud-email Removed  
# Container cloud-nginxserver Removed  
# Container cloud-journals Removed  
# Container cloud-gateway Removed  
# Container cloud-scheduler Removed  
# Container cloud-measures Removed  
# Container cloud-devices Removed  
# Container cloud-identity Removed  
# Container cloud-psqlserver Removed  
# Network cloud_client-net Removed  
# Network cloud_cloud-net Removed  
akr@ubuntu-cenergocloud:~$
```

Рисунок 10

2) Обновить сервисы (Рисунок 11). Этот шаг можно выполнить повторно чтобы убедиться, что все обновления прошли успешно и более системе не обнаруживает новые версии образов.

Ubuntu:

```
$ sudo docker-compose -f docker-compose.cloud.yaml -p cloud pull
```

Windows:

```
$ docker-compose -f docker-compose.cloud.yaml -p cloud pull
```

```
kr@ubuntu-cenergocloud: ~  
kr@ubuntu-cenergocloud:~$ sudo docker-compose -p cloud pull  
[+] Running 17/27  
# integratordlms Pulled 3.7s  
# data Pulled 3.6s  
# identity Pulled 3.5s  
# snmp Pulled 3.8s  
# modbus Pulled 3.7s  
# iec104 Pulled 3.6s  
# devices Pulled 3.9s  
# integratorcenergo Pulled 3.6s  
# gateway Pulled 3.7s  
# customer Pulled 3.8s  
# measures Pulled 3.8s  
# email Pulled 3.9s  
# scheduler Pulled 3.9s  
# journals Pulled 3.9s  
* nginxserver Pulling 4.2s  
# 8ee3204ce13b Already exists 0.0s  
.: a1484661dfe6 Downloading 556.3kB/7.236MB 1.6s  
.: 2f78a3560d10 Download complete 1.6s  
.: a517401f7a94 Download complete 1.6s  
.: 294d17c34d13 Waiting 1.6s  
.: 7051f5a2f4b1 Waiting 1.6s  
.: 977b508a19e0 Waiting 1.6s  
.: f12dac2be4d4 Waiting 1.6s  
.: c811c0ce884b Waiting 1.6s  
.: 0f56b475a58f Waiting 1.6s  
# logging Pulled 3.0s  
# postgres Pulled 3.8s
```

Рисунок 11

## 2) Запустить сервисы:

Ubuntu:

```
$ sudo docker-compose -f docker-compose.cloud.yaml -p cloud up -d
```

Windows:

```
$ docker-compose -f docker-compose.cloud.yaml -p cloud up -d
```

```
kr@ubuntu-cenergocloud: ~  
kr@ubuntu-cenergocloud:~$ sudo docker-compose -p cloud up -d  
[+] Running 19/19  
# Network cloud_cloud-net Created 0.2s  
# Network cloud_client-net Created 0.2s  
# Container cloud-postgresqlserver Started 4.5s  
# Container cloud-identity Started 3.1s  
# Container cloud-gateway Started 14.0s  
# Container cloud-journals Started 12.6s  
# Container cloud-devices Started 14.3s  
# Container cloud-email Started 12.2s  
# Container cloud-measures Started 13.9s  
# Container cloud-logging Started 13.8s  
# Container cloud-customer Started 14.0s  
# Container cloud-integratordlms Started 18.8s  
# Container cloud-data Started 18.4s  
# Container cloud-integratorcenergo Started 18.6s  
# Container cloud-scheduler Started 18.9s  
# Container cloud-nginxserver Started 18.5s  
# Container cloud-modbus Started 22.4s  
# Container cloud-iec104 Started 22.1s  
# Container cloud-snmp Started 22.1s  
kr@ubuntu-cenergocloud:~$
```

Рисунок 12

## 3.3. Конфигурирование

### 3.3.1. SNMP

Сервис SNMP - использует ряд настроек, которые могут быть изменены в случае необходимости.

Для изменения настроек - необходимо переопределить ключи с помощью переменных среды перед запуском контейнера. Как это может быть сделано описано в документации docker-compose: <https://docs.docker.com/compose/environment-variables/>.

**Внимание!** Использование «:» для разделения иерархических ключей - не поддерживается рядом операционных систем, потому - лучшим вариантом будет использование «\_\_» ([подробнее](#)).

Для настройки сервиса SNMP используются следующие ключи:

Автоматическая регистрация устройства:

Auth\_\_EntryPoint - веб-сервер через который устройство МКSP-1EE сможет обратиться к ceCloud для проведения процедуры автоматической регистрации в системе.

**Пример:**

Auth\_\_EntryPoint=<https://www.cecloud.ru>.

**Внимание!** Значение пути должно включать схему (**[Ошибка! Недопустимый объект гиперссылки.](#)** или **[Ошибка! Недопустимый объект гиперссылки.](#)** в зависимости от настроек сервера). По умолчанию используется https.

**Внимание!** В случае использования реверс-прокси, он должен быть настроен на редирект контейнеру **cloud.nginxserver** либо **cloud.gateway**.

#### 3.1.1 Регистрация в один клик

Функция контроллера «Регистрация в один клик» – требует сертификат сервера, содержащего информацию о домене и IP-адресе сервера.

Файл сертификата должен быть расположен **по пути хранения тома cloud\_nginxssl**. (см. Развёртывание и запуск проекта, в разделе 2.2. Установка): /var/lib/docker/volumes/cloud\_nginxssl/\_data/public. Имя файла должно быть nginx-selfsigned.crt

Для генерации самоподписанного сертификата, можно использовать openSSL.

## Пример создания самоподписанного сертификата для Linux.

### 1. Создайте новую папку для работы и перейдите в неё:

```
$ mkdir /new-certs  
$ cd /new-certs
```

### 2. Сгенерируйте приватный ключ:

```
$ openssl genrsa -out nginx-selfsigned.key 2048
```

### 3. Создайте текстовый файл `ssl_conf` следующего содержания:

```
[req]  
default_bits      = 4096  
prompt           = no  
default_md        = sha256  
x509_extensions  = v3_req  
distinguished_name = dn  
  
[dn]  
C                = RU  
ST               = {край/область/штат}  
L                = {город/населенный пункт}  
O                = {организация}  
CN               = {доменное имя сервера}  
emailAddress     = {электронная почта}  
  
[v3_req]  
subjectAltName = @alt_names  
  
[alt_names]  
DNS.1           = {доменное имя сервера}  
IP.1            = {ip-адрес сервера}
```

**Внимание!** Замените значения в фигурных скобках валидными в вашем случае.

### 4. Сгенерируйте сертификат командой:

```
$ openssl req -new -x509 -key nginx-selfsigned.key -days 730 -out nginx-selfsigned.crt -config <(cat ssl_conf)
```

Альтернативный способ генерирования ключа без файла конфигурации:

```
$ sudo openssl req -x509 -nodes -days 730 -newkey rsa:2048 -out selfsigned.crt -keyout nginx-selfsigned.key -addext "subjectAltName = DNS.1:{короткое доменное имя сервера}, DNS.2:{доменное имя сервера}, IP.1:{ip-адрес сервера}"
```

В данном случае все параметры описанные в файле конфигурации будут запрашиваться для ввода в процессе генерации ключа.

5. Переместите ключ `nginx-selfsigned.key` в папку `private`, которая была примонтирована к сервисам `nginxserver` и `snmp`.
6. Переместите сертификат `nginx-selfsigned.crt` в папку `public`, которая была примонтирована к сервисам `nginxserver` и `snmp`.
7. Перезапустите сервисы.

Подробности можно изучить в официальной документации OpenSSL <https://www.openssl.org/docs/manmaster/man1/openssl-req.html>

### 3.3.2. Modbus

В сервисе Modbus используются настройки, которые могут быть изменены в случае необходимости.

Для соединения устройств с сервисом Modbus используется внешний порт 21050. В случае если данный порт занят, его можно заменить на необходимый в файле `docker-compose.yml`.

Для этого нужно открыть в любом тестовом редакторе файл `docker-compose.yml`. Затем в файле найти сервис «`modbus`», внутри сервиса найти раздел «`ports`».

Порты имеют следующий формат:

«внешний\_порт\_docker:внутренний\_порт\_контейнера».

Чтобы изменить порта для работы с устройствами нужно найти прописанные порты `-21050:5050` и значение внешнего порта заменить на необходимый для вашей сети.

В случае если `seCloud` запущен чтобы применить изменённые настройки нужно его перезапустить для этого нужно остановить сервисы путем выполнения команды:

Ubuntu:

```
$ sudo docker-compose -f docker-compose.cloud.yaml -p cloud down
```

Windows:

```
$ docker-compose -f docker-compose.cloud.yaml -p cloud down
```

И после запустить сервисы:

Ubuntu:

```
$ sudo docker-compose -f docker-compose.cloud.yaml -p cloud up -d
```

Windows:

```
$ docker-compose -f docker-compose.cloud.yaml -p cloud up -d
```