ООО “ТРИДИВИ”

OMNI Platform 1.9.4

Руководство администратора

Миасс - 2023

Содержание

1. Требования

1.1. Минимальные системные требования

Для развертывания сервера OMNI Platform необходимо подготовить машину со следующими характеристиками:

ЦП:

- 4 ядра (потока)
- Частота - 3 ГГц
- Расширения набора команд: AVX/AVX2

Пример подходящего ЦП - Intel® Xeon® E3-1220 v5

ОЗУ - 16 ГБ

Накопитель SSD:

- Свободное пространство от 100 ГБ
- Не менее 20% свободного места в файловой системе

В случае развертывания отдельного сервера БД (по умолчанию сервер базы данных запускается на машине, где развернута Платформа) потребуется машина с характеристиками ниже:

- ЦП: 2 ядра
- 8 ГБ ОЗУ
- `PostgreSQL` >= 11.2
- SSD/HDD 100 ГБ

Убедитесь, что на машине установлено следующее ПО:

- ОС Ubuntu 20.04.4
- Docker версии - 20.10.17
- Kubernetes версии - 1.23.8
- Helm версии - 3.9.1

Примечание: Для установки Docker, Kubernetes и Helm можно воспользоваться скриптом `on_premise/setup/install-packages.sh` поставляемым вместе с дистрибутивом (требуется подключение к интернету).

1.2. Требования для GPU (опционально)

Nvidia Container Toolkit версии - 1.11.0

Примечание: требуется видеокарта с графическим процессором от компании Nvidia не ниже GTX 1080 Ti с поддержкой версии CUDA 10.2.

1.3. Настройка Docker для использования GPU (опционально)

1.3.1 Конфигурация Docker

Для установки `nvidia-container-runtime` в качестве низкоуровневой среды выполнения по умолчанию добавьте следующие строки в файл конфигурации, который находится по адресу `/etc/docker/daemon.json`:

```
"default-runtime": "nvidia",
"runtimes": {
    "nvidia": {
        "path": "/usr/bin/nvidia-container-runtime",
        "runtimeArgs": []
    }
}
```

1.3.2 Применение конфигурации

Перезапустите docker-service, выполнив следующую команду:

```
$ sudo systemctl restart docker
```

2. Инструкция по развертыванию

2.1. Обновление Платформы

Если вы получили новую версию Платформы и при этом уже используете развернутую версию, последовательно выполните описанные ниже команды, в противном случае пропустите пункт 2.1.

1. Перейдите в папку `on_premise` развернутой версии Платформы и удалите Платформу, выполнив команду:

```
$ ./setup/uninstall-platform.sh
```

2. Убедитесь, что все контейнеры сервисов остановлены. Для этого используйте команду:

```
$ watch 'kubectl get pods'
```

Статус сервисов из состояния “Running” должен перейти в состояние “Terminating”. В результате, все сервисы должны пропасть из отображаемой таблицы.

3. Удалите развернутый Kubernetes кластер, выполнив команду:

```
$ sudo kubeadm reset
```

4. Удалите вспомогательные файлы Kubernetes кластера, выполнив команду:

```
$ sudo rm -rf ~/.kube/
```

5. Выполните сброс таблиц IPVS вашей системы, используя команду:

```
$ sudo ipvsadm --clear
```

Примечание: При удалении Платформы вся база данных сохраняется в директории `/kv/pgdata`. Для дальнейшего использования базы в процессе установки новой версии Платформы необходимо указать те же авторизационные данные и имя базы данных. В противном случае, удалите папку `/kv/pgdata (sudo rm -rf /kv/pgdata)` для создания новой базы данных в процессе развертывания Платформы.

2.2. Подготовка: загрузка образов и создание Kubernetes кластера

2.2.1 Подготовка

Скачайте и распакуйте дистрибутив OMNI Platform на машину, где планируется выполнить установку. Ссылка на дистрибутив должна быть отправлена вам в письме.
Откройте системную консоль, перейдите в директорию `on_premise` внутри дистрибутива и проверьте содержимое папки, используя команду:

```
$ find -maxdepth 1
```

Файлы и папки, содержащиеся в дистрибутиве, будут выведены в консоль. Пример вывода:

```
./deploy
./ingress-nginx-4.2.0.tgz
./OMNIAgent_Linux_x64.run
./OMNIAgent_Windows_x64.exe
./integration_tests
./kube-flannel.yml
./kubeadm.yaml
./license_server
./nvidia-device-plugin-0.12.2.tgz
./pdf_docs
./platform_images.tar.gz
./setup
./upload_script
```

Основные элементы дистрибутива:

- `./pdf_docs/administrator_guide.pdf` - руководство администратора содержит информацию необходимую для развертывания OMNI Platform;
- `./pdf_docs/user_guide.pdf` - руководство пользователя содержит информацию необходимую для использования OMNI Platform;
- `./pdf_docs/integration_api.pdf` - справочник по API;
- `./pdf_docs/release_notes.pdf` - описание изменений, включая информацию о новом функционале, исправленным ошибкам и улучшениям;
- `./pdf_docs/agent_user_guide.pdf` – руководство пользователя OMNI-агента;
- `./OMNIAgent_Linux_x64.run` и `./OMNIAgent_Windows_x64.exe` - установочные файлы OMNI-агента для Linux и Windows соответственно;
- `./license_server` - файлы необходимые для запуска лицензионного сервера с помощью которого осуществляется лицензирование OMNI Platform;

- `./integration_tests` - скрипты для автоматического тестирования Платформы после развертывания;
- `./setup/settings.env` - файл конфигурации экземпляра OMNI Platform.
- `./upload_script` - папка, в которой находится скрипт загрузки изображений из датасета для создания профилей на Платформе.

Дальнейшие команды выполняются в системной консоли из директории `on_premise`.

2.2.2 Загрузка образов

Загрузите в локальный `registry` образы из архива:

```
$ sudo docker load -i platform_images.tar.gz
```

Загрузка образов может длиться около пяти минут.

2.2.3 Ввод переменных окружения

Откройте файл конфигурации `./setup/settings.env` используя текстовый редактор и установите значения следующих переменных:

- `MASTER_NODE_IP_ADDRESS` - IP-адрес машины, на которой выполняется развертывание. Вы можете узнать его у Вашего системного администратора.
- `DOMAIN` - корневое доменное имя. После развертывания доступ к API и веб-интерфейсу OMNI Platform будет осуществляться по адресу `http://platform.<DOMAIN>`. IP-адрес для доменного имени `platform.<DOMAIN>` должен быть сконфигурирован на DNS-сервере (подробнее в пункте 2.3.2 данного руководства).
- `RABBIT_USER` и `RABBIT_PASSWORD` - имя пользователя и пароль для доступа к брокеру сообщений, используется для внутреннего взаимодействия сервисов OMNI Platform. Задайте произвольное имя, состоящее из латинских букв, без пробелов и пароль, состоящий из латинских букв и цифр, без пробелов.
- `POSTGRES_USER`, `POSTGRES_PASSWORD` и `POSTGRES_DB` - параметры подключения к базе данных. При первом развертывании OMNI Platform задайте произвольное имя пользователя и название базы данных, состоящих из латинских букв, без пробелов и сгенерируйте пароль, состоящий из латинских букв и цифр. База данных будет создана автоматически.
- `SERVICE_KEY` – секретный ключ, необходимый для внутреннего взаимодействия сервисов OMNI Platform. Сгенерируйте произвольную строку, состоящую из латинских букв и цифр без пробелов.

- `LIC_KEY` – лицензионный ключ. Обычно ключ отправляется в сопроводительном письме с дистрибутивом. При отсутствии ключа обратитесь к вашему менеджеру по продажам.
- `PLATFORM_ADMIN_EMAIL` и `PLATFORM_ADMIN_PASSWORD` - учетные данные, которые будут использоваться для доступа в панель администратора OMNI Platform. При первом развертывании OMNI Platform пользователь с правами администратора будет создан автоматически. Укажите корректный email и сгенерируйте пароль состоящий из латинских букв и цифр длиной не менее 8-ми символов.
- `PLATFORM_DEFAULT_EMAIL`, `PLATFORM_DEFAULT_PASSWORD` - учетные данные пользователя для доступа в веб-интерфейс OMNI Platform. При первом развертывании пользователь будет создан автоматически. Укажите корректный email и сгенерируйте пароль состоящий из латинских букв и цифр длиной не менее 8-ми символов.
- `EMAIL_HOST`, `EMAIL_PORT`, `EMAIL_HOST_USER`, `EMAIL_HOST_PASSWORD`, `EMAIL_USE_SSL` – параметры доступа к SMTP-серверу. SMTP-сервер используется для отправки писем, например при сбросе пароля или для оповещений. Чтобы отключить отправку писем, оставьте данные поля пустыми. Для получения параметров доступа к SMTP-серверу обратитесь к администратору вашей сети. Для включения/отключения SSL-протокола, укажите для параметра `EMAIL_USE_SSL` значения `true/false`, соответственно. Изначально значение для параметра не указано, что по умолчанию определяется как `false`.
- `EMAIL_FROM` – значение, которое будет отправлятся в заголовке FROM и отображаться в качестве отправителя письма. Требования к формату данного поля могут зависеть от SMTP-сервера. Пример значения поля FROM – `"Bob Example" <bob@example.org>`.
- `QUERY_LIMIT` – ограничение количества возвращаемых элементов в API запросах для получения сущностей системы. Увеличение данного лимита не рекомендуется, т.к. время выполнения API запроса может увеличиться в несколько раз. Также, обратите внимание, что увеличение лимитов приведет к ухудшению работы системы.
- `ENABLE_PROFILE_AUTOGENERATION` – авто-создание профилей для приходящих активностей с агента. Необходимо учитывать, что при включении данной опции будет увеличенный расход ресурсов лицензии (размер базы данных). Если функция не требуется, то нужно оставить поле пустым, в противном случае установить значение 1.
- `USE_CUDA` - отвечает за использование CUDA ядер в сервисах обработки изображений. 0 - отключить GPU, 1 - включить GPU для `processing` сервиса.

Сохраните изменения в файле.

2.2.4 Установка и настройка кластера

Запустите команду для создания и настройки кластера.

```
$ ./setup/init-cluster.sh
```

Эта команда выполняет следующие действия:

1. Инициализация узла для развертывания кластера
2. Создание секретов
3. Создание необходимых папок
4. Установка ingress-controller
5. Установка nvidia-device-plugin, если включено использование видеокарты

2.2.5 Проверка работоспособности кластера

После инициализации главного узла убедитесь, что все узлы готовы к работе и имеют статус Ready. Для проверки выполните следующую команду:

```
$ kubectl get nodes
```

В результате в терминале будет отображен следующий вывод:

```
NAME          STATUS      ROLES                   AGE     VERSION
master-node   Ready       control-plane,master    11d     v1.23.8
```

Для проверки всех элементов кластера запустите следующую команду:

```
$ kubectl get all --all-namespaces
```

2.3. Настройка лицензирования

Примечание: Если вы обновили Платформу, используя команды из пункта 2.1, сразу переходите к пункту 2.4.

2.3.1 Установка сервера лицензий

Запустите команду для установки и запуска сервера лицензий:

```
$ ./setup/install-lic-server.sh
```

Проверьте что сервер лицензий находится в статусе Running, выполнив следующую команду:

```
$ sudo service floatingserver status
```

Откройте файл `./setup/settings.env` и для переменной `LIC_SERVER_URL` укажите значение по шаблону `http://<LIC_SERVER_IP_ADDRESS>:8090`, где `LIC_SERVER_IP_ADDRESS` - IP-адрес машины, на которой установлен сервер лицензий. Если сервис запускается на той же машине, где развернут кластер, то `LIC_SERVER_IP_ADDRESS` будет совпадать с `MASTER_NODE_IP_ADDRESS`.

Убедитесь, что сервер лицензий доступен, перейдите в браузере по адресу сервера лицензий, на странице должна отобразится форма входа.

2.3.2 Оффлайн активация лицензии

Запустите команду для генерации оффлайн запроса на лицензию:

```
$ ./setup/activate-lic-server.sh --generate-offline
```

В результате выполнения команды в директории `on_premise` должен появиться файл `request-offline.license`.

Отправьте сгенерированный файл запроса `request-offline.license` на почту технической поддержки. В ответном письме будет отправлен файл лицензии.

Поместите полученный лицензионный файл в папку `on_premise`.
Откройте файл конфигурации `./setup/settings.env` , используя текстовый редактор, и заполните значение переменной `OFFLINE_LICENSE_FILE` именем файла лицензии и его расширением, если присутствует, через точку.

Запустите команду для активации полученной лицензии:

```
$ ./setup/activate-lic-server.sh --activate-offline
```

Пример вывода в консоль при успешной активации лицензии:

```
[2022-09-08 01:30:36+05:00] INF Offline activating license key...
[2022-09-08 01:30:36+05:00] INF License activated successfully!
```

2.4. Развертывание Платформы

2.4.1 Запуск развертывания

Запустите скрипт для развертывания Платформы в кластере:

```
$ ./setup/deploy.sh
```

Для отслеживания процесса развертывания откройте ещё одну вкладку терминала и введите следующую команду:

```
$ watch 'kubectl get pods'
```

Наличие у всех pods статуса `Running` означает, что Платформа запущена.

2.4.2 Настройка DNS

Для обеспечения доступа к Платформе DNS сервер вашей сети должен содержать запись о том, что домен `platform.<DOMAIN>` доступен по адресу `<MASTER_NODE_IP_ADDRESS>`. Значения переменных можно получить из файла `./setup/settings.env`, заполненного в пункте 2.1.3. Обратитесь к администратору вашей сети, чтобы выполнить данную конфигурацию.

Для целей тестирования можно указать IP-адрес и домен в файле `/etc/hosts` на Linux или `C:\Windows\System32\drivers\etc\hosts` на Windows. Для этого добавьте в конец данного файла новую строку вида `<MASTER_NODE_IP_ADDRESS> platform.<DOMAIN>`, подставив значения соответствующих переменных и сохраните файл. Обратите внимание, для редактирования файла `hosts` необходимо обладать правами администратора.

Для использования Платформы с той же машины, где выполнено развертывание, можно воспользоваться скриптом. Он автоматически добавит необходимую запись в файл `/etc/hosts`.

```
$ ./setup/add-dns.sh
```

2.4.3 Описание элементов развернутой системы

Для получения статуса сервисов Платформы выполните команду:

```
$ kubectl get pods
```

В консоль будет выведен список сервисов, их статус, количество перезапусков и время с момента создания сервиса. Пример вывода:

```
NAME                                READY   STATUS    RESTARTS     AGE
backend-dep-5478dd8d88-lfjp8        1/1     Running   0            18h
broker-dep-697f74bbbd-qqx5v         1/1     Running   0            18h
cache-dep-589c74b79f-kfblz          1/1     Running   0            18h
db-dep-8669dd8c47-mmltv             1/1     Running   0            18h
gateway-dep-67dcf75646-6vwlw        1/1     Running   0            18h
image-api-dep-79df7ff5cc-fmmdp      1/1     Running   0            18h
matcher-dep-7cd8fc9849-j2qnw        1/1     Running   0            18h
processing-dep-775bdf5875-kxrnq     1/1     Running   0            18h
quality-dep-7c47d85787-2qn58        1/1     Running   2 (18h ago)  18h
redis-dep-5d8cd4d657-88rzf          1/1     Running   0            18h
```

Ниже приведено краткое описание сервисов:

- `backend-dep` - основной контейнер Платформы, отвечает за работу большей части API;
- `db-dep` - экземпляр базы данных PostgreSQL, хранит всю информацию Платформы;
- `gateway-dep` - сервис nginx, отвечает за доступ к Платформе и за работу веб-интерфейса Платформы;
- `matcher-dep` - отвечает за поиск людей по базе;
- `processing-dep` - отвечает за детекцию людей на изображении с последующим созданием шаблона для поиска, используется при выполнении запроса на детекцию лиц;
- `quality-dep` - отвечает за расчет качества изображения;
- `image-api-dep` - отвечает за работу сервиса ImageAPI, доступного по URL /image-api/.

2.4.4 Масштабирование

В случае, когда нагрузка возрастает, для стабилизации работы Платформы предусмотрено масштабирование следующих сервисов в ручном режиме:

1. `processing-dep`
2. `quality-dep`
3. `backend-dep`
4. `gateway-dep`
5. `ingress-nginx-controller`

Описание сервисов указано в пункте 2.4.3.

Для масштабирования сервиса необходимо выполнить следующую команду:

```
$ kubectl scale deployment <SERVICE_NAME> --replicas <COUNT>
```

где `<SERVICE_NAME>` - наименование сервиса (например, `gateway-dep`), а `<COUNT>` - количество экземпляров сервиса.

Примечание: Для масштабирования сервиса `ingress-nginx-controller` необходимо в конец команды добавить аргумент “`-n ingress-nginx`”.

При масштабировании необходимо руководствоваться следующей информацией:

1. Для обработки большего числа одновременно поступающих запросов, следует масштабировать `backend-dep`, `ingress-nginx-controller`, `gateway-dep`. Количество экземпляров сервисов необходимо указывать согласно формуле: `<REQUESTS>/<CPU_COUNT>`, где `<REQUESTS>` - это желаемое количество запросов, одновременно находящихся в обработке, а `<CPU_COUNT>` - это количество логических ядер ЦП.
2. Если большая часть запросов связана с обработкой изображений, необходимо масштабировать `processing-dep` и `quality-dep`. Количество экземпляров сервисов не должно превышать количество физических ядер ЦП.

Пример:

Для поддержания нагрузки в A запросов/сек, направленной на обработку изображений, на сервере с физическим количеством ядер ЦП равным B и логическим количеством ядер равным C, следует масштабировать сервисы следующим образом:

- `processing-dep` - $min(A,\ B)$ экземпляров
- `quality-dep` - $min(A,\ B)$ экземпляров
- `backend-dep` - A/C экземпляров
- `gateway-dep` - A/C экземпляров
- `ingress-nginx-controller` - A/C экземпляров

Для сохранения параметров масштабирования, откройте файл `./deploy/values.yaml`, найдите блок `service_replicas` и укажите для сервисов новые подобранные значения. При следующих установках Платформы сервисы будут автоматически масштабироваться до указанных значений.

2.4.5 Резервное копирование и восстановление базы данных

Для создания резервной копии базы данных выполните следующую команду:

```
$ ./setup/db-backup.sh <dump_path>
```

Для восстановления базы данных выполните следующую команду:

```
$ ./setup/db-restore.sh <dump_path>
```

`dump_path` - путь до дампа базы данных

3. Проверка работоспособности и отладка

3.1 Проверка работоспособности

Дистрибутив содержит скрипт для автоматической проверки работоспособности Платформы. Укажите URL Платформы (`http://platform.<DOMAIN>`), email пользователя (`<PLATFORM_DEFAULT_EMAIL>`) и выполните команду:

```
$ python3 integration_tests/main.py <platform url> <user email> --show-trace
```

После запуска скрипта, в консоли будет запрошен пароль, введите пароль пользователя (`<PLATFORM_DEFAULT_PASSWORD>`) и нажмите клавишу Enter.
В случае успешного выполнения тестов, в консоль будет выведен следующий текст:

```
Test data prepared successfully
------------------------------------------------------------------------
------------------------------
create_search_profile test started
create_search_profile test succeeded
------------------------------------------------------------------------
------------------------------
Test data deleted successfully
```

3.1.1 Возможные ошибки при тестировании и пути их решения

При возникновении ошибок тестирования система возвращает следующий результат:

```
Error: <error type>
Error message: <error message>
```

`Error type` обозначает тип возникшей ошибки, а `error message` сообщает уточняющую информацию об ошибке.

Команды для отладки сервисов описаны в пункте 3.2, список сервисов и их зона ответственности - в пункте 2.4.3.

Далее перечислены комбинации ошибок и сообщений с возможными шагами по их устранению.

ConnectionError:

- **<urlopen error Wrong url format: asdasd>**
  Введен неверный формат url, необходимо ввести корректный адрес.
- **<urlopen error [Errno -2] Name or service not known> \ <urlopen error [Errno 111] Connection refused>**
  Введен url недоступного сервиса. Проверьте корректность введённого адреса, а также убедитесь, что Платформа развернута корректно и доступна извне. Если вы обращаетесь по домену, проверьте, что файл /etc/hosts имеет именно тот домен, который указывает на ip адрес развернутой Платформы.

- **HTTP Error 405: Not Allowed**
  Убедитесь, что введенный url ведёт именно на Платформу, а не на сторонний сервис.

- **HTTP Error 502: Bad Gateway / HTTP Error 503: Service Temporarily Unavailable**
  Убедитесь, что сервис `backend-dep` развернут.

PlatformError:

- **connection to server at "localhost" (::1), port 5432 failed: Connection refused Is the server running on that host and accepting TCP/IP connections?**
  Убедитесь, что база данных доступна и работает правильно.

- **Authorization error**
  Убедитесь, что вы ввели правильные пароль и почту пользователя.

- **Wrong answer from server. JSON can not decoded**
  Убедитесь, что введенный url ведёт именно на Платформу, а не на сторонний сервис.

- **License has not been leased yet**
  Убедитесь, что сервер лицензий работает, и у Платформы есть к нему доступ. Дополнительно проверьте, что лицензия активирована корректно.

- **Low quality photo**
  Проверьте, что сервис, отвечающий за вычисление качества фотографий, доступен и работает корректно.

- **Profile not searched**
  Убедитесь, что сервис, отвечающий за поиск по базе персон, доступен и работает корректно.

При появлении любых других типов ошибок или сообщений, которые не получается отладить и устранить на месте, а также невозможности устранения вышеописанных ошибок, обратитесь в нашу службу поддержки.

3.2 Отладка

В случае некорректной работы сервиса логи можно получить с помощью следующей команды, подставив имя интересующей поды:

```
$ kubectl logs $POD_NAME
```

Или через команду получения логов предыдущей попытки запуска:

```
$ kubectl logs $POD_NAME --previous
```

Или через раздел `Events` вывода следующей команды:

```
$ kubectl describe pod $POD_NAME
```

4. Получение данных для доступа к системе

Передайте пользователю учетные данные для входа в web-интерфейс, URL Платформы, токен доступа к API. Как получить данную информацию описано ниже.

4.1. Получение пароля и почты пользователя

Пароль и почту пользователя можно найти в файле конфигурации Платформы `settings.env` в переменных `PLATFORM_DEFAULT_PASSWORD` и `PLATFORM_DEFAULT_EMAIL` соответственно.

4.2. URL развернутого сервера

Домен для развертывания Платформы указывается в файле `settings.env` в переменной `DOMAIN`. В результате доступ к Платформе можно получить по url: `http://platform.<DOMAIN>`

Например: `http://platform.your-company.com`

4.3. Токен доступа

Чтобы получить токен доступа к API для зарегистрированного пользователя под `PLATFORM_DEFAULT_EMAIL`, используйте команду:

```
$ ./setup/get-token.sh
```

Либо откройте в браузере web-интерфейс Платформы используя URL из предыдущего пункта. Войдите в Платформу используя учетные данные пользователя из пункта 4.1, перейдите по ссылке *Platform API* в блоке *Ресурсы* на главной странице веб-интерфейса. Отправьте следующий запрос в консоли Graphql:

```
query{
  me {
    workspaces {
      accesses {
        id
      }
    }
  }
}
```

Пример ответа сервера при успешном выполнении запроса:

```
{
  "data": {
    "me": {
      "workspaces": [
        {
          "accesses ": [
            {
              "id": "3460a1d7-214c-48c3-b2be-aa6d2a6bca09"
            }
          ]
        }
      ]
    }
  }
}
```

Значение поля `id` - токен доступа к API.

5. Полезные команды для работы с Платформой

Запуск Платформы:

```
$ ./setup/deploy.sh
```

Остановка Платформы:

```
$ ./setup/uninstall-platform.sh
```

Запуск сервера лицензий:

```
$ sudo service floatingserver start
```

Остановка сервера лицензий:

```
$ sudo service floatingserver stop
```

6. Устранение неисправностей

Ошибка генерации оффлайн запроса на предоставление лицензии:

Проблема: при выполнении команды `./setup/activate-lic-server.sh --generate-offline` появляется ошибка:

```
ERR Missing file path for offline activation request file! Specify path using '--offline-request' option.
```

Решение: убедитесь, что в файле `./setup/settings.env` указаны ключ лицензии в переменной `LIC_KEY` и адрес сервера лицензий в переменной `LIC_SERVER_URL`.

Ошибка установки пакетов Docker, Kubernetes и Helm

Проблема: при выполнении команды скрипта `on_premise/setup/install-packages.sh` появляется ошибка:

```
E: Sub-process /usr/bin/dpkg returned an error code (1)
```

Решение 1: ошибка может быть вызвана поврежденной базой данных dpkg. В этом случае выполните перенастройку пакетного менеджера dpkg с помощью команды:

```
$ sudo dpkg --configure -a
```

Решение 2: если ошибки появляются во время установки пакетов ПО, можно принудительно установить пакет, используя аргумент `-f`:

```
$ sudo apt install -f
OR
$ sudo apt install --fix-broken
```

Аргументы `-f` и `--fix-broken` равноценно используются для исправления зависимостей, нарушенных в результате прерванной загрузки пакета.

Решение 3: Если предыдущие два решения не помогли устранить проблему, попробуйте удалить или стереть проблемный пакет ПО, выполнив команду:

```
$ sudo apt remove --purge package_name
```

Решение 4: Вы также можете вручную удалить все файлы, связанные с проблемным пакетом, выполнив команду, указанную ниже. Файлы находятся в директории **`/var/lib/dpkg/info`**.

```
$ sudo ls -l /var/lib/dpkg/info | grep -i package_name
```

После просмотра списка файлов перенесите их в папку **`/tmp`** :

```
$ sudo mv /var/lib/dpkg/info/package-name.* /tmp
```

Также удалить файлы вручную можно с помощью команды:

```
$ sudo rm -r /var/lib/dpkg/info/package-name.*
```

Ошибка с `nvidia-device-plugin` при проверке элементов кластера

Проблема: при выполнении команды `kubectl get all --all-namespaces` появляется ошибка:

```
Error: failed to start container "nvidia-device-plugin-ctr": Error response from daemon: failed to create shim task: OCI runtime create failed: runc create failed: unable to start container process: error during container init: error running hook #0: error running hook: exit status 1, stdout: , stderr: Auto-detected mode as 'legacy'
```

```
nvidia-container-cli: initialization error: nvml error: driver/library version mismatch: unknown
```

```
stranger@stranger:~/Platform/Platform_op-v1-9-1-rc3-fcb/on_premise$ kubectl get all --all-namespaces
NAMESPACE              NAME                                           READY   STATUS             RESTARTS      AGE
ingress-nginx          pod/ingress-nginx-controller-58c78bb84c-9vz2l  1/1     Running            0             46s
kube-flannel           pod/kube-flannel-ds-rgmj6                      1/1     Running            0             51s
kube-system            pod/coredns-64897985d-4vqxk                    1/1     Running            0             51s
kube-system            pod/coredns-64897985d-j4tjb                    1/1     Running            0             51s
kube-system            pod/etcd-master-node                           1/1     Running            21            66s
kube-system            pod/kube-apiserver-master-node                 1/1     Running            23            65s
kube-system            pod/kube-controller-manager-master-node        1/1     Running            26            65s
kube-system            pod/kube-proxy-drhl8                           1/1     Running            0             51s
kube-system            pod/kube-scheduler-master-node                 1/1     Running            23            65s
nvidia-device-plugin   pod/nvdp-nvidia-device-plugin-5gj8l            0/1     RunContainerError  3 (5s ago)    41s
```

Решение:

1. Для получения информации о вашей видеокарте и доступных драйверах выполните следующую команду:

```
ubuntu-drivers devices
```

2. В выводе консоли указано, что в системе установлена видеокарта «GeForce GTX 1050 Ti», а рекомендуемый драйвер — «nvidia-driver-515».

```
 == /sys/devices/pci0000:00/0000:00:10.0 ==
modalias : pci:v000010DEd00001C82sv00001458sd00003764bc03sc00i00
vendor   : NVIDIA Corporation
model    : GP107 [GeForce GTX 1050 Ti]
manual_install: True
driver   : nvidia-driver-510-server - distro non-free
driver   : nvidia-driver-450-server - distro non-free
driver   : nvidia-driver-390 - distro non-free
driver   : nvidia-driver-520 - distro non-free
driver   : nvidia-driver-418-server - distro non-free
driver   : nvidia-driver-515-server - distro non-free
driver   : nvidia-driver-515 - distro non-free recommended
driver   : nvidia-driver-510 - distro non-free
driver   : nvidia-driver-470-server - distro non-free
driver   : nvidia-driver-470 - distro non-free
driver   : xserver-xorg-video-nouveau - distro free builtin
```

3. Для установки рекомендуемого (recommended) драйвера выполните команду:

```
sudo apt install nvidia-driver-515
```

4. После установки драйвера вы можете просмотреть состояние видеокарты с помощью инструмента мониторинга `nvidia-smi`:

```
stranger@stranger:~/Platform/Platform_op-v1-9-1-rc3-fcb/on_premise$ nvidia-smi
Mon Nov  7 09:33:05 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 515.65.01    Driver Version: 515.65.01    CUDA Version: 11.7     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA GeForce ...  Off  | 00000000:00:10.0 Off |                  N/A |
|  0%   46C    P0    N/A /  72W |      0MiB /  4096MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

5. Посмотреть версию драйвера можно с помощью команды:

```
cat /proc/driver/nvidia/version
```

```
stranger@stranger:~/Platform/Platform_op-v1-9-1-rc3-fcb/on_premise$ cat /proc/driver/nvidia/version
NVRM version: NVIDIA UNIX x86_64 Kernel Module  515.65.01  Wed Jul 20 14:00:58 UTC 2022
GCC version:  gcc version 11.2.0 (Ubuntu 11.2.0-19ubuntu1)
```

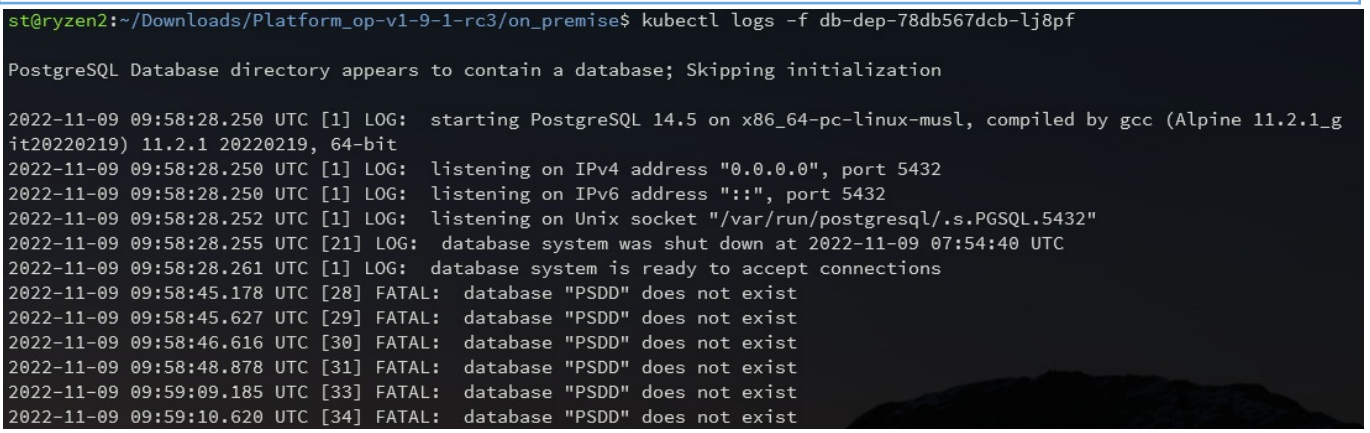
Ошибка при развертывании Платформы в кластере:

Проблема: при выполнении команды `./setup/deploy.sh` появляется ошибка:

```
NAME                                READY   STATUS             RESTARTS      AGE
agent-sync-dep-5594b487b9-bt9rb     0/1     CrashLoopBackOff   5 (65s ago)   4m7s
backend-dep-7b49b78d64-r8s9n        0/1     CrashLoopBackOff   5 (60s ago)   4m7s
broker-dep-f6dfdf55b-7bhhq          1/1     Running            0             4m7s
cache-dep-7dbc644bcf-h6w5d          1/1     Running            0             4m7s
db-dep-78db567dcb-lj8pf             1/1     Running            0             4m7s
gateway-dep-544c8b67fd-7wh7z        1/1     Running            0             4m6s
image-api-dep-9447f8b64-5ffcx       1/1     Running            0             4m7s
matcher-dep-97cb47c4c-m9cfl         0/1     CrashLoopBackOff   5 (67s ago)   4m7s
processing-dep-679496d79b-j7rtf     1/1     Running            0             4m7s
quality-dep-6b98d6645d-c9ww9        0/1     CrashLoopBackOff   5 (60s ago)   4m7s
redis-dep-5d8cd4d657-qwgx8          1/1     Running            0             4m7s
```

Решение: запросите лог `db-dep` с помощью команды:

```
kubectl logs -f <полное имя поды>
```



Если отображается ошибка о неверных имени базы данных или авторизационных данных, повторно разверните кластер (См. п. 2.1).