Инструкция по установке POLYHUB

Содержание

Требования к системе

Серверная часть

Минимальные требования к серверному оборудованию следующие:

- 8 vCPU (2.8 ГГц+)
- 12GB RAM
- 90GB HDD

Из которых:

Сервис POLYHUB:

- 2 vCPU
- 2GB RAM
- 10GB HDD

Сервис каталога:

- 2 vCPU
- 4GB RAM
- 20GB HDD

Сервис поиска в каталоге:

- 2 vCPU
- 2GB RAM
- 30GB HDD

Внутренняя база данных системы:

- 2 vCPU
- 4GB RAM
- 30GB HDD

Ориентировочная формула для подсчета конфигурации в зависимости от количества пользователей и используемых модулей: дополнительно к минимальным системным требованиям необходимо RAM 256-512МБ CPU 0.1 vCPU в среднем на каждого пользователя. Конечная конфигурация уточняется в каждом случае отдельно.

Операционная система: Astra Linux Special Edition 1.6 (Смоленск) или аналог.

Права пользователя, разворачивающего приложение: user - non-root with sudo privileges.

Дополнительные требования к установленным приложениям: Docker версии 20.10.0 и до 25, Docker -compose версия 1.29 и выше.

Локальная сеть

Все компоненты платформы должны находиться в одной подсети или должна обеспечиваться прозрачная маршрутизация. Не рекомендуется использовать NAT. В рамках ознакомления рекомендуется отключить брандмауэры. Внутри локальной сети между всеми компонентами не должно быть ограничений по передаче данных. Для доступа из внешней сети достаточно открыть порт, используемый POLYHUB (порт задается при установке). При использовании системы с установленными антивирусами или комплексными системами защиты необходимо обеспечить свободную работу, сетевую активность и взаимодействие компонентов.

Установка версии

Установка и настройка Docker

1. Установить Docker в соответствии с инструкцией: https://docs.docker.com/install/linux/docker-ce/ubuntu/.
   Примечание - Рекомендуемая версия 25.0.5.
2. Установить Docker Compose в соответствии с инструкцией: https://docs.docker.com/compose/install/.
   Примечание - Рекомендуемая версия 1.29.2.

Установка компонентов на один сервер

- Убедиться, что сервис запущен
- Создать директорию для файлов образов

```
cd ~ && mkdir install
```

- Скопировать из полученного дистрибутива в созданную директорию архивы базовых образов (где <version> - версия приложения):
  - r5_dwash_back:<version>.tar
  - r5_dwash_front:<version>.tar
  - r5_dwash_skipper:<version>.tar
  - r5_dwash_origin:<version>.tar
  - r5_dwash_dagster:<version>.tar
  - r5_redis:7-alpine.tar
  - r5_postgres13:1.0.tar
- Скопировать из полученного дистрибутива в созданную директорию архивы образов опциональных компонент:
  - r5_dwash_alert:<version>.tar
  - r5_dwash_report:<version>.tar
  - r5_dwash_catalog:<version>.tar
  - opensearch:2.tar
- Выполнить команды:

```
docker load -i ~/install/r5_dwash_back:<version>.tar
docker load -i ~/install/r5_dwash_front:<version>.tar
docker load -i ~/install/r5_dwash_skipper:<version>.tar
docker load -i ~/install/r5_dwash_origin:<version>.tar
docker load -i ~/install/r5_dwash_dagster:<version>.tar
docker load -i ~/install/r5_redis:6-alpine.tar
docker load -i ~/install/r5_postgres13:1.0.tar
```

- Выполнить команды загрузки опциональных компонент, если требуется их установка:

```
docker load -i ~/install/r5_dwash_alert:<version>.tar
docker load -i ~/install/r5_dwash_report:<version>.tar
docker load -i ~/install/r5_dwash_catalog:<version>.tar
docker load -i ~/install/opensearch:2.tar
```

- Создать директории сервиса

```
cd ~ && mkdir polyhub && cd polyhub
```

- Скопировать из полученного дистрибутива в созданную директорию инициализирующие компоненты
    - manage.py
    - __init__.py
    - docker-compose_prod.yml.tmpl
- Разместить в созданной директории файлы ключей (key.pem) и сертификатов (cert.pem) (путь до файлов может быть задан при установке).
- Развернуть сервис используя одну из команд:
    - для обычной установки

```
python3 manage.py --deploy --release
```

    - для установки с каталогом

```
python3 manage.py --deploy --release --with-catalog
```

Описание утилиты управления сервисом POLYHUB manage.py

Автоматизирует процесс установки, обновления и управления POLYHUB.

Принимает следующие аргументы:

`-h`, `--help` вывести текст справки

`--deploy` развернуть сервисы с нуля

`--rebuild` пересобрать docker образы

`--release` собрать версионные образы

`--update` обновить сервисы

`--restart` перезапустить сервисы

`--version` вывести версию приложения

`--backup [BACKUP]` создать резервную копию базы

`--restore [RESTORE]` восстановить базу из резервной копии

`--docker-repo [DOCKER_REPO]` задать репозиторий docker-образов, по умолчанию: test.polymedia.ru/r5/

`--add-skipper-volume ADD_SKIPPER_VOLUME ADD_SKIPPER_VOLUME ADD_SKIPPER_VOLUME` `<vol_name> <src_path> <dst_relative_path>`, добавить том в `resources/skipper`

`--no-upgrade` не обновлять базу данных

`--with-catalog` сервис поставляется с модулем метаданных

`--ci` автоматический деплой.

Примеры запуска:

- `python3 manage.py --deploy --release` выполнить установку приложения;

- `python3 manage.py --deploy --release --with-catalog` выполнить установку приложения с каталогом данных;
- `python3 manage.py --update --release --restart` выполнить обновление приложения;
- `python3 manage.py --no-upgrade --backup /path/to/backup` выполнить резервное копирование БД;
- `python3 manage.py --no-upgrade --restore /path/to/backup` выполнить восстановление БД из резервной копии.

Порядок обновления системы

При обновлении необходимо обеспечить наличие базовых образов в системе, а также инициализирующих компонентов (manage.py, __init__.py), аналогично как при установке.

Ниже указан общий порядок обновления:

1. убедиться, что сервис запущен
2. обновить файлы __init__.py и manage.py
3. запустить обновление
    a. для обычного обновления

```
python3 manage.py --update --release -restart
```

    b. для обновления с каталогом

```
python3 manage.py --update --release --restart --with-catalog
```

Перед обновлением рекомендуется создать резервную копию БД.

Резервное копирование изменяемых при обновлении файлов конфигурации выполняется автоматически в папку ./backups.

Резервное копирование

Для резервного копирования БД используется утилита управления manage.py:

- создание: `python3 manage.py --no-upgrade --backup /path/to/backup`
- восстановление: `python3 manage.py --no-upgrade --restore /path/to/backup`

Для резервной копии требуется наличие свободного дискового пространства на:

- целевом устройстве под хранение резервной копии
- контейнере dwash-db (директория /tmp)

Функционал резервного копирования и восстановления БД предназначен для использования на одном и том же экземпляре сервиса.

Восстановление секретов

Для восстановления БД на другом экземпляре, например, если было выполнено новое развертывание, необходимо также восстановить секреты:

- FERNET_KEY
    - используется для обратимого шифрования ключей и паролей, требующих возможности их расшифровки,

    - в случае компрометации должен быть заменен на новый,
    - после активации нового секрета пароли провайдеров и секретные опции, при наличии таковых, должны быть заданы заново;
- JWT_SECRET_KEY
    - используется для подписи токенов,
    - в случае компрометации должен быть заменен на новый,
    - после активации нового секрета все токены публичного API должны быть перегенерированы;
- SECRET_KEY
    - используется для подписи сессионных куки,
    - в случае компрометации должен быть заменен на новый;
- DATABASE_URL
    - используется для подключения к БД сервиса,
    - в случае компрометации секрета пароль БД в нем должен быть заменен на новый, в том числе и в самой БД.

Секреты задаются в app.secrets и должны быть известны только использующему их экземпляру сервиса.

В случае замены новые секреты активируются перезапуском сервиса.

Резервное копирование файлов конфигурации и томов сервиса необходимо выполнять средствами хоста, на котором развернут сервис.

Управление ресурсами

- Все ресурсы должны располагаться на томе resources
- поддерживается следующие типа ресурсов: отчеты, файлы облака, метаданные хранилища, уведомления, где:
    - отчеты в resources/reporting (вложенный том reports)
    - корзины облака в resources/cloud
    - метаданные в resources/dwh
    - уведомления в resources/alert (вложенный том alerts)
- относительный путь из атрибута db провайдера типа LOCAL добавляется к базовой директории ресурса
- для отчетов в качестве db провайдера по умолчанию используется reports.