DBA
DIGITAL BUSINESS ADAPTATION

ПО «СИСТЕМА ИНТЕЛЛЕКТУАЛЬНОЙ ДИАГНОСТИКИ ЭНЕРГООБЪЕКТОВ»

Инструкция по развертыванию

Барнаул
2022

Содержание

Начало работы 3
Подготовка сервера: 3
Развертывание 3
Устанавливаем Portainer**:** 3
Настраиваем Portainer: 4
Создаем stack для приложения: 6
Установка RabbitMQ: 7
Установка PostgreSQL: 8
Установка pgAdmin: 9
Установка Seq: 10
Поднимаем сервис авторизации: 11
Поднимаем nginx: 12
Поднимем клиент авторизации: 13
Поднимаем сервис интерпретации: 14
Поднимаем медиасервис: 16
Поднимаем интеграционную шину: 17
Поднимаем сервис загрузки: 18
Поднимаем ui-side: 19
Настраиваем DNS: 20

Начало работы

Важно. Образ виртуальной машины в формате Hyper-V. Необходима версия hyper-v сервера не ниже 2019.

При развертывании СИДЭ на Ubuntu необходимо выполнить каждый шаг. Если разворачивается образ виртуальной машины, достаточно добавить DNS записи (последняя страница). В целях безопасности все пароли от регистров были удалены с образа. Не забудьте добавить их, если разворачиваете СИДЭ “с нуля”.

Данные для авторизации:

ui-side:
- Логин – admin
- Пароль – admin

portainer:
- Логин – admin
- Пароль – 123456789

auth-client:
- Логин – admin
- Пароль – admin

pgadmin:
- Пароль – 123456789

postgres:
- user: postgres
- password: baguvix

Подготовка сервера:

1. Подключаемся по ssh к виртуальной машине с Ubuntu;
2. Получаем root-права командой: **sudo -s** (далее вводим логин и пароль);
3. Обновляем информацию по пакетам: **apt upgrade;**
4. Обновляем пакеты: **apt upgrade;**
5. Устанавливаем Docker командой **apt install docker.io .**

Развертывание

Устанавливаем Portainer**:**

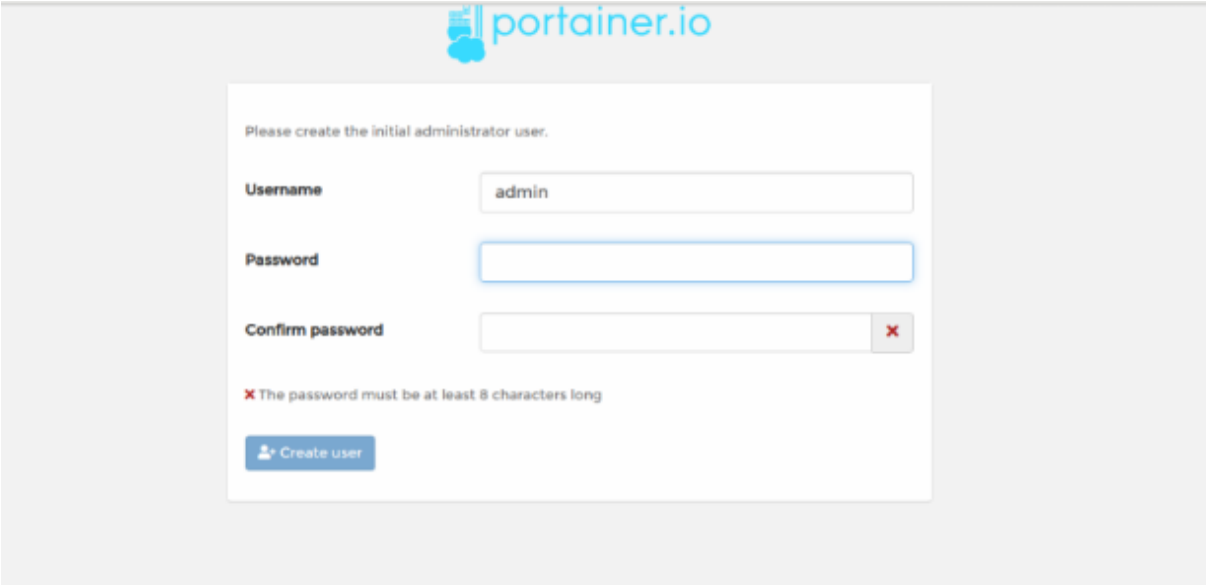
1. Создаем хранилище данных для Portainer: **docker volume create portainer_data ;**
2. Устанавливаем и запускаем контейнер Portainer: **docker run -d -p 9000:9000 --name=portainer --restart=always -v /var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer-ce.** Эта команда настраивает автоматический запуск

Portainer после перезагрузки, а также постоянное хранилище, чтобы Ваши настройки не потерялись при удалении и повторном разворачивании контейнера;

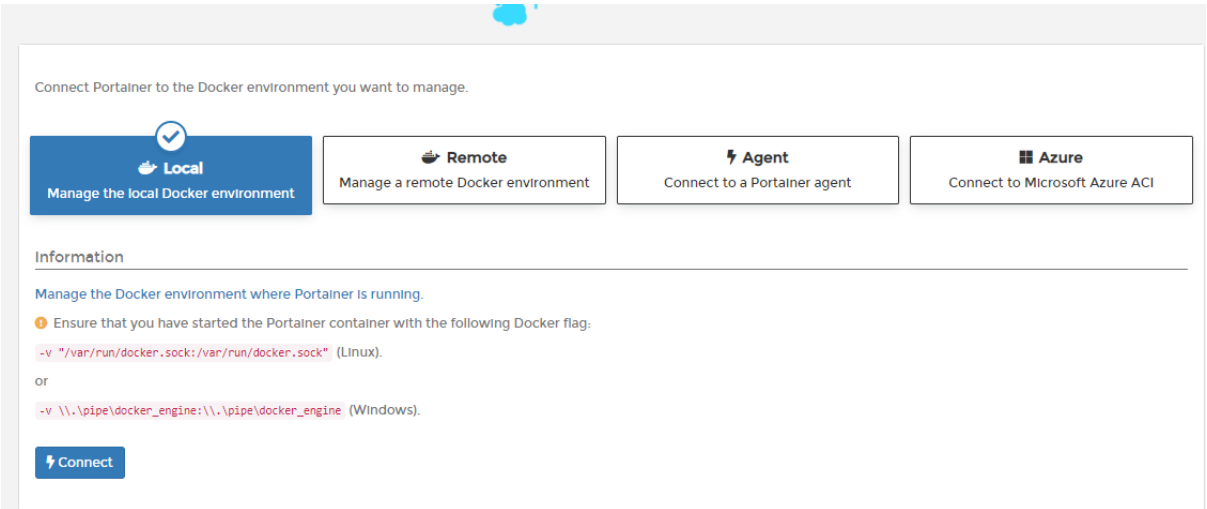
3. Убеждаемся, что portainer запущен: **docker ps .**

Настраиваем Portainer:

1. Получить доступ к программе Вы можете через веб-интерфейс на порту 9000. Откройте его в браузере. Введите имя пользователя и пароль, под которым вы будете входить в систему.



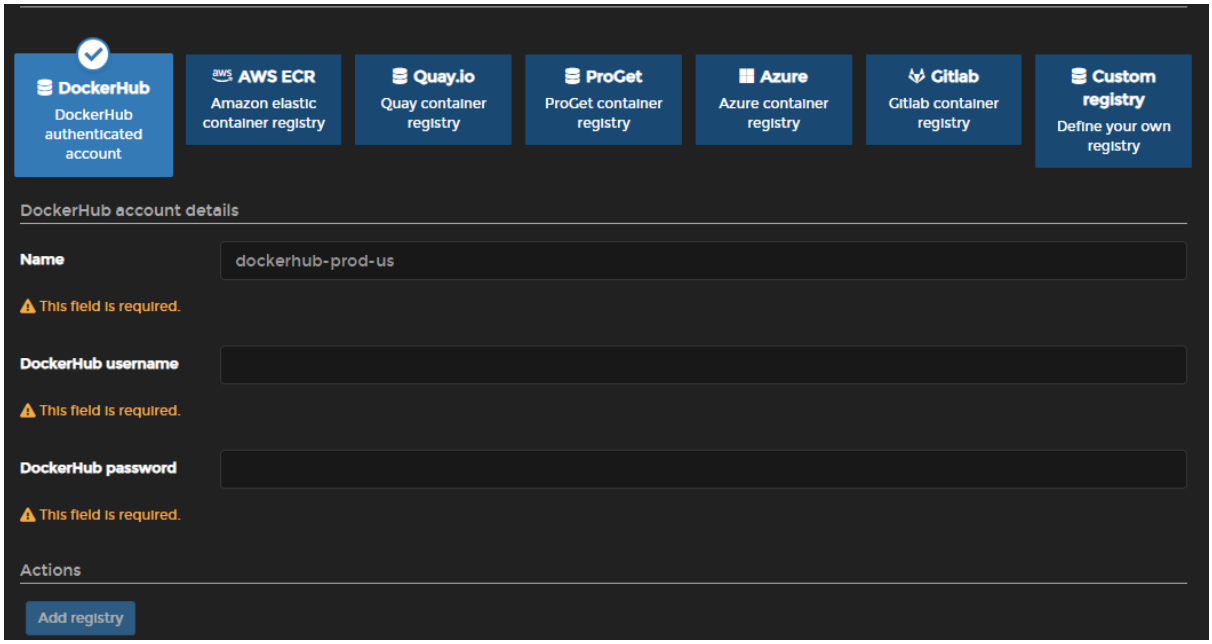
2. Выберем метод подключения к Docker. Для начала можно подключиться к локальному сервису Docker. Для этого выберите Loca ;



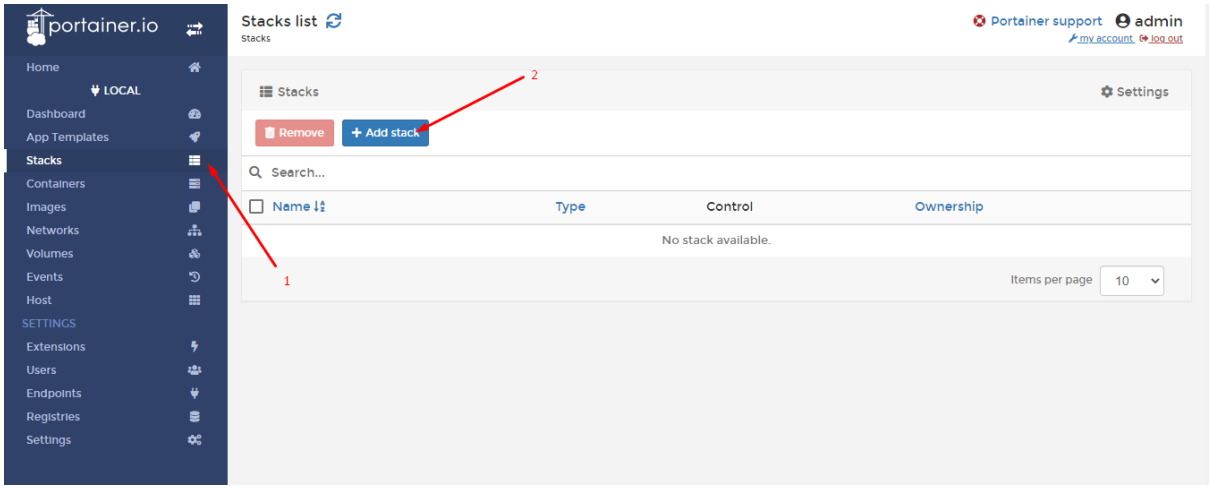
3. Переходим в консоль и устанавливаем swarm режим для докера:

docker swarm ini ;

4. Используя любой ftp manager, закидываем папку с конфигами на сервер (далее будем копировать их в нужные директории);
5. Переходим в пункт Swarm- Registries и добавляем регистр.



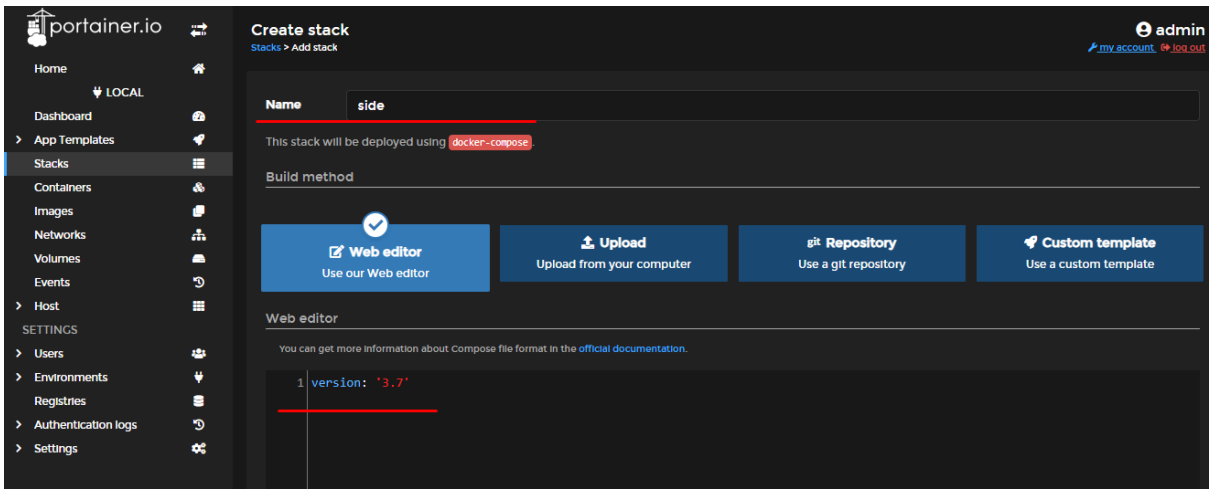
Создаем stack для приложения:



1.Переходим на панель Stacks в Portainer;

2.Нажимаем на Add stack;

3.Вводим название stack и пишем версию в compose файл;



4.Спускаемся вниз страницы и нажимаем ‘Deploy the stack’ .

Установка RabbitMQ:

1. Добавляем в compose файл конфигурацию для контейнера;

```
services:

  #RABBITMQ
  rabbitmq:
    image: rabbitmq:3-management-alpine
    hostname: rabbitmq-dev
    container_name: rabbitmq-dev
    ports:
      - "5672:5672"
      - "15672:15672"
    volumes:
       - /var/configs/rabbitmq.conf:/etc/rabbitmq/rabbitmq.conf
       - /var/configs/rabbitmq_definitions.json:/definitions.json
       - rabbit-volume:/var/lib/rabbitmq/mnesia
    restart: always
```

2. Закидываем по указанным в блоке volumes путям соответствующие конфиги;
3. Добавляем блок volumes в самый низ compose файла;

```
volumes:
  rabbit-volume:
```

4. Нажимаем ‘Update the stack’ в portainer .

Установка PostgreSQL:

1. Добавляем в compose файл (блок services) конфигурацию для контейнера;

```
#POSTGRESQL
  postgres:
    hostname: postgres
    container_name: postgres
    image: postgres
    environment:
      POSTGRES_USER: ${POSTGRES_USER:-postgres}
      POSTGRES_PASSWORD: ${POSTGRES_PASSWORD:-baguvix}
      PGDATA: /data/postgres
    volumes:
      - postgres:/data/postgres
    ports:
      - "5432:5432"
      - "6556:5432"
    restart: always
```

2. Обновляем блок volumes:

```
volumes:
  rabbit-volume:
  postgres:
```

3. Нажимаем ‘Update the stack’ в portainer;

4. Далее нам понадобится 3 базы для сервисов (сервис интеграции, медиа сервис и сервис авторизации) сервис интеграции использует EF core с подходом code first, поэтому базу для него создавать не нужно. Для создания оставшихся баз используются контейнеры.
   - **sudo docker run -it --rm hub4dba/mediaservicecreatedatabase "User Id=postgres;Password=baguvix;Host=postgres;Database=mediaservice;Port=5432;Integrated Security=False"**
   - **sudo docker run -it --rm hub4dba/authenticationcreatedatabase "User Id=postgres;Password=baguvix;Host=postgres;Database=authservice;Port=5432;Integrated Security=False**

Установка pgAdmin:

1. Добавляем в compose файл (блок services) конфигурацию для контейнера (необходимо подставить значения логина и пароля);

```
  #PGADMIN
  pgadmin:
    container_name: pgadmin
    image: dpage/pgadmin4
    environment:
      PGADMIN_DEFAULT_EMAIL: ${PGADMIN_DEFAULT_EMAIL:-
sv_kulaga@dba.ooo}
      PGADMIN_DEFAULT_PASSWORD: ${PGADMIN_DEFAULT_PASSWORD:-
baguvix}
      PGADMIN_CONFIG_SERVER_MODE: 'False'
    volumes:
       - /var/configs/pgadmin:/var/lib/pgadmin
    ports:
      - "${PGADMIN_PORT:-5050}:80"
    restart: always
```

2. Создаем директорию pgadmin по указанному в volumes пути;
3. Меняем право доступа к созданной папке

   sudo chmod a+wrx {папка указанная в volumes};
4. Нажимаем ‘Update the stack’ в portainer;
5. Заходим в pgAdmin и подключаемся к бд.

Установка Seq:

1. Добавляем в compose файл (блок services) конфигурацию для контейнера (необходимо подставить значения логина и пароля);

```
#SEQ
seq:
  hostname: seq
  container_name: seq
  image: datalust/seq:latest
  environment:
    - ACCEPT_EULA=Y
  volumes:
    - /var/seqdata:/data
  ports:
    - "5433:80"
    - "5434:5341"
  restart: always
```

2. Создаем папку по указанному в volumes пути;
3. Меняем право доступа к созданной папке

   sudo chmod a+wrx {папка указанная в volumes} ;
4. Нажимаем ‘Update the stack’ в portainer.

Поднимаем сервис авторизации:

1. Добавляем в compose файл (блок services) конфигурацию для контейнера (необходимо подставить значения логина и пароля);

```
#сервис авторизации
auth-service:
  hostname: auth-service
  image: hub4dba/authenticationservice:latest
  container_name: auth-service
  volumes:
    - /var/configs/appsettings_auth.json:/app/appsettings.json
  ports:
    - "9600:80"
  restart: always
```

2. Копируем соответствующий конфигурационный файл по пути, указанному в блоке volumes;
3. Нажимаем ‘Update the stack’ в portainer.

Поднимаем nginx:

1. Добавляем в compose файл (блок services) конфигурацию для контейнера (необходимо подставить значения логина и пароля);

```
nginx:
  hostname: nginx
  image: nginx:stable-alpine
  container_name: nginx
  environment:
    - "TZ=Asia/Barnaul"
  ports:
    - "80:80"
    - "443:443"
    - "8080:8080"
  volumes:
    - /var/wwwroot/:/usr/share/nginx/html/
    - /var/configs/nginx.conf:/etc/nginx/nginx.conf
  restart: always
```

2. Копируем соответствующий конфигурационный файл по пути, указанному в блоке volumes, и создаем wwwrot директорию;
3. Нажимаем ‘Update the stack’ в portainer.

   Внимание! Он (nginx) может вставать и падать до тех пор, пока оставшиеся сервисы не будут подняты.

Поднимем клиент авторизации:

1. Добавляем в compose файл (блок services) конфигурацию для контейнера (необходимо подставить значения логина и пароля)

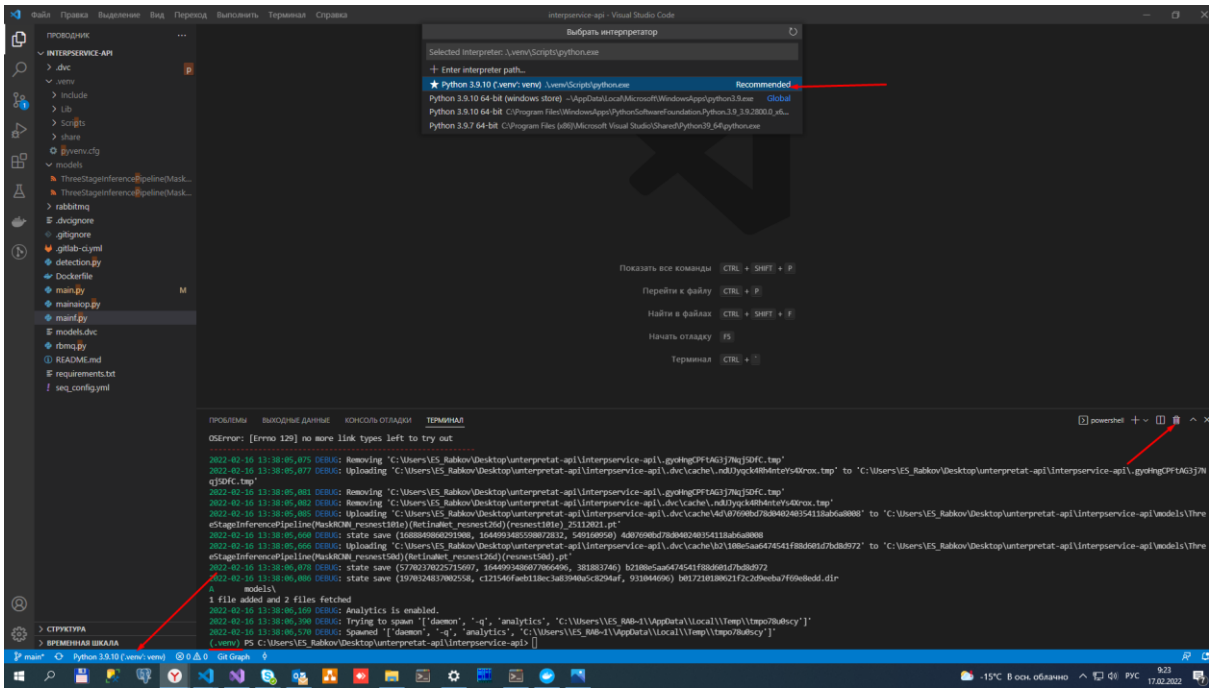
```
#Клиент сервиса авторизации
  auth-client:
    hostname: auth-client
    image: hub4dba/authenticationclient:latest
    container_name: auth-client
    ports:
      - "9500:80"
    volumes:
      - /var/configs/auth_client_config.js:/var/www/web/settings.js
    restart: always
```

2. Копируем соответствующий конфигурационный файл по пути, указанному в блоке volumes;
3. Нажимаем ‘Update the stack’ в portainer.

Поднимаем сервис интерпретации:

К сожалению, на данный момент Dockerfile, предоставленный разработчиками сервиса, выдает ошибку: «Не может отработать dvc». Для того чтобы это исправить, необходимо вручную выгрузить модель, подсунуть его в папку проекту и подкорректировать Dockerfile, убрав из него блок, отвечающий за выгрузку модели. Более того, конфигурационные файлы в данном сервисе не предусмотрены, поэтому для нормальной работы сервиса в другом окружении придется править его исходники (например, для работы с gpu).

1. Загрузка модели (при помощи vscode):
   - Открываем проект в vscode ;
   - ctrl + ~ откроет консоль ;
   - Создаем виртуальное окружение: **python -m venv .venv ;**
   - vscode предложит перейти на новое виртуальное окружение - соглашаемся (Если не предложит, можно выбрать вручную в левом нижнем углу, 3-я плашка статус-панели);



   - перезапускаем текущий сеанс терминала (именно перезапускаем, а не скрываем и открываем), у строчки терминала должен появиться (.venv) префикс, как на скрине;
   - Устанавливаем dvc: **pip install dvc[ssh] ;**
   - Загружаем модель: **dvc pull --remote storage-ml –v ;**
   - Модель будет лежать в папке models.
2. По ftp сбрасываем модель на сервер и собираем образ:

   docker build -t interpretation-service ;

3. Добавляем в docker compose файл определение сервиса;

```
#Сервис интерпретации
interpretation-service:
  image: interpretation-service:latest
  container_name: interpretation-service
  volumes:
    - /var/TempRepo:/app/TempRepo
  restart: always
```

4. Нажимаем ‘Update the stack’ в portain.

Поднимаем медиасервис:

1. Добавляем в compose файл (блок services) конфигурацию для контейнера (необходимо подставить значения логина и пароля)

```
#Медиа сервис
media-service:
  hostname: media-service
  image: hub4dba/mediaserviceservice:secure
  container_name: media-service
  ports:
    - "9626:80"
  volumes:
    - /var/configs/appsettings_mediaservice.json:/app/appsettings.json
    - /var/media:/storage
  restart: always
```

2. Добавим файл конфигураций по указанному пути;
3. Добавим директорию media по пути, указанному в volumes;
4. Поменяем права на директорию

   sudo chmod a+wrx {папка указанная в volumes} .

Поднимаем интеграционную шину:

1. Добавляем в compose файл (блок services) конфигурацию для контейнера (необходимо подставить значения логина и пароля)

```
# Шина интеграции
integration-service:
    hostname: integration-service
    image:
registry.dba.ooo/shiryaev/sideintegrationbus/integrationservice:main
    container_name: integration-service
    ports:
      - "9625:80"
    volumes:
      -
/var/configs/appsettings_integrationservice.json:/app/appsettings.json
      - /var/TempRepo:/app/TempRepo
    restart: always
```

2. Копируем файл конфигурации по указанному пути;
3. Создаем директорию и меняем ее права

   sudo chmod a+wrx {папка указанная в volumes} .

Поднимаем сервис загрузки:

1. Добавляем в compose файл (блок services) конфигурацию для контейнера (необходимо подставить значения логина и пароля)

```
#Сервис загрузки фото
loading-service:
  hostname: loading-service
  image: registry.dba.ooo/shiryaev/sideintegrationbus/loadingservice:main
  container_name: loading-service
  ports:
    - "9624:80"
  volumes:
    - /var/configs/appsettings_loadingservice.json:/app/appsettings.json
    - /var/TempRepo:/app/TempRepo
  restart: always
```

2. Копируем файл конфигурации по указанному пути;
3. Создаем директорию и меняем ее права

   sudo chmod a+wrx {папка указанная в volumes} .

Поднимаем ui-side:

1. Добавляем в compose файл (блок services) конфигурацию для контейнера (необходимо подставить значения логина и пароля)

```
  #Веб-интерфейс
  ui-side:
    hostname: ui-side
    image: registry.dba.ooo/ershovdmitry/ui-side/ui-side:side-
rosreestr
    container_name: ui-side
    ports:
      - "9623:80"
    restart: always
```

2. Копируем файл конфигурации по указанному +пути.

Для того чтобы изменить базовые адреса, по которым ui-side обращается к интеграционной шине и сервису авторизации, нужно поменять их в файле исходного кода src/config.ts и пересобрать контейнер (для PocPeестра на гитлабе была создана отдельная ветка, в которой указаны базовые адреса, предполагающие, что записи будут добавлены в DNS).

Настраиваем DNS:

Большинство сервисов общаются по внутренней сети docker, но веб-интерфейс вынужден обращаться к хосту по имени. Чтобы избежать лишнего конфигурирования, было использовано решение с DNS.

В своем dns необходимо добавить 4 имени для ip сервера, на котором развернут СИДЭ:

- integration-service.dba.ooo
- ui-side-rosreestr.dba.ooo
- auth-service.dba.ooo
- media-service.dba.ooo

Если нет возможности внести записи во внутренний DNS, то можно внести записи в hosts файл (C:\Windows\System32\drivers\etc\hosts). Тогда система будет работать только на том рабочем месте, где настроен hosts.

```
# The IP address and the host name should be separated by at least one
# space.
        IP address integration-service.dba.ooo
        IP address ui-side-rosreestr.dba.ooo
        IP address auth-service.dba.ooo
        IP address media-service.dba.ooo
```