Инструкция по установке и эксплуатации экземпляра Платформы Abanking OEM

Инструкции по разворачиванию docker-compose

- Требования к инфраструктуре
  - Операционная система
  - Стороннее ПО
- Распаковка контейнеров и загрузка образов
- Настройки Nginx
  - Настройки хостового Nginx
  - Настройка сим-линк
  - Применение новой конфигурации Nginx
  - Nginx в docker-compose
- Настройка сервисов
  - Настройка почты в микросервисе уведомлений
- Запуск
  - Создание сети
  - Запуск инфраструктурных микросервисов
  - Увеличение максимального количества подключений PostgreSQL
  - Запуск приложения
- Проверка
  - Проверяем запущенные контейнеры
  - Доступность Swagger
- Примененние бэкапа данных
  - Разархивировать бекап
  - Остановить работу продукта
  - Копирование файлов в примонтированную директорию
  - Применение бекапа postgresql
  - Запустить продукт

Требования к инфраструктуре

Операционная система

ОС	Версия

Debian	`10 и старше`
Ubuntu	`20 и старше`
ALT Linux	`10 и старше`
Astra Linux	`1.7 и старше`

Стороннее ПО

Компонент	Версия
Docker	`19.03.0 и старше`
docker-compose	`2.24.7 и старше`
Nginx	`1.18.0 и старше`
PostgreSQL[1]	`16-alpine`
RabbitMQ[2]	`3.12.4-management`
Redis[3]	`7.2.0`

Примечание: Версии ПО указаны для развертывания новых стендов, на существующих обновления не требуются.

Обязательным требованием для запуска является наличие на хостовой машине следующего ПО:

- Docker
- docker-compose

Рабочая конфигурация предполагает наличие двух веб-серверов Nginx:

- nginx-1 - веб-сервер установленный на хостовой ОС;
- nginx-2 - контейнер с веб-сервером nginx. используется для маршрутизации между микросервисами.

Веб-сервер nginx-1 может располагаться как на одной машине с сервисами, так и на отдельной. Основным требованием является наличие у веб-сервера доступа к порту хостовой машины, на которой запущен nginx-2 - маршрутизатор запроса на сервисы (по умолчанию используется 10000 порт).

[1] PostgreSQL - компонент не входит в состав поставки платформы.
[2] RabbitMQ - компонент не входит в состав поставки платформы.
[3] Redis (open-source edition)- компонент не входит в состав поставки платформы.

Важно: Для каждого публичного кабинета, созданного на платформе, требуется наличие домена, спроксированного на стенд, а также SSL-сертификат.

Распаковка контейнеров и загрузка образов

На сервере, создать в корне каталог */docker-containers*, куда распаковать содержимое архива abankingeom*.tar.gz*.

Распаковка конфигурации

```
sudo mkdir /docker-containers
sudo tar -xvf abankingeom.tar.gz -C /docker-containers
```

Если с сервера нет доступа в интернет, для загрузки образов с Harbor, то поставка происходит через SFTP, для этого необходимо:

1. Загрузить архивы с SFTP на сервер, где будет размещен комплект поставки
2. Перейти в каталог с образами (каждый образ в архиве имеет вид <SERTVICE_NAME>.tar.gz)
3. Выполнить загрузку следующей командой:

Распаковка и загрузка образов

```
sudo docker load -i <SERVICE_NAME>.tar.gz                                    #
Для загрузки одного образа
sudo ls -1 *.tar.gz | xargs --no-run-if-empty -L 1 docker load -i   # Для
загрузки всех образов в каталоге
```

Проверить список загруженных образов можно командой:

Вывод списка образов

```
docker image ls
```

Перейти в каталог */docker-containers.* Структура каталога, из комплекта поставки, соответствует примеру ниже.

Все микросервисы приложений должны находиться в директории *./services*, все инфраструктурные микросервисы должны находиться в директории *./infra.*

Структура директорий

```
/docker-containers   # основной каталог, в который размещаются все каталоги с
файлами docker-compose и настройками сервисов
├── ./services    # каталог с сервисами
|   ├── ./docker-compose.yml  #  файл содержащий описание архитектуры docker-
контейнеров
|   ├── .env  # общий файл переменных среды
|   ├── ./action-api  #  каталог сервиса
|   |   ├── DefaultSetting
```

```
│   │   │   └── appsettings.DefaultSetting.json  #  файл настроек сервиса
│   │   └── .env  # файл переменных среды данного сервиса
│   ├── ./application-api
│   │   ├── DefaultSetting
│   │   │   └── appsettings.DefaultSetting.json
│   │   └── .env
│   └── ./...
│       ├── DefaultSetting
│       │   └── appsettings.DefaultSetting.json
│       └── .env
└── ./infra   #  каталог с инфраструктурными сервисами
    ├── docker-compose.yml
    ├── .env
    ├── ./postgres   #  Каталог появится, после старта комоуза.
    └── ./rabbitmq
```

Настройки Nginx

Настройки хостового Nginx

Добавить проксирование с веб-сервера реверс-прокси, установленный непосредственно на хост с сервисами (с того, что должно проксировать на nginx в контейнер, который на 10000 порту).

В примере ниже - если на хосте установлен Nginx, то конфигурация расположенная в */etc/nginx/sites-available* проксировала запросы на Nginx в контейнере).

Директиву *server_name* заменить на свой домен.

Пример конфигурации хостового Nginx

```
server {
  # публичные домены
  server_name
    client-lk-browser-dev.abankingeom.ru # домен
    ...
    <some_name> # другие домены

  listen 80;

  location / {

     client_max_body_size 30m;

     proxy_set_header  Host $host;
     proxy_set_header  X-Forwarded-Proto $scheme;
     proxy_set_header  X-Real-IP $remote_addr;
     proxy_set_header  X-Forwarded-For $proxy_add_x_forwarded_for;

     # блок настроек необходимый для проксирования websocket запросов
     proxy_set_header Upgrade $http_upgrade;
     proxy_set_header Connection $http_connection;
```

```
    proxy_cache_bypass $http_upgrade;
    proxy_http_version 1.1;

    proxy_pass        http://127.0.0.1:10000/;

  }
#для блокировки swagger'a, интернал рестов, дебаг рестов

  if ($request_uri ~* (swagger|internal|debug)) {
    set $block_me_now A;
  }

  if ($remote_addr != xx.xx.xx.xx) {  # добавить разрешенный адрес
    set $block_me_now "${block_me_now}B";
  }

  if ($block_me_now = AB) {
    return 403;
    break;
  }

  location ~ .*(/internal/.*) {
     deny all;
  }

  location ~ .*(/debug/.*) {
    deny all;
  }

  location ~ .*(/swagger/.*) {
    deny all;
  }

  location ~ .*(/document-api/public/.*) {
    deny all;
  }
}
```

Настройка сим-линк

После создания необходимо конфигурации приведенной выше, необходимо перейти в директорию */etc/nginx/sites-enabled* и настроить сим-лик.

Для этого потребуется выполнить следующую команду:

Создание сим-линк

```
sudo ln -s ../sites-available/<имя файла с конфигурацией> <имя файла с конфигурацией>
```

Применение новой конфигурации Nginx

После добавления файла конфигурации в */etc/nginx/sites-available* и настройки сим-линк в */etc/nginx/sites-enabled* требуется проверить ее на синтаксические ошибки, если они отступают, выполнить перегрузку хвостового Nginx для их активации:

Применение конфигурации

```
sudo nginx -t                          # Проверка синтаксиса файла конфигурации
sudo nginx -s reload           # Перезапуск Nginx
```

Nginx в docker-compose

Отредактировать директиву *server_name* для каждого домена (типовой конфиг будет предоставлен) в *docker-compose/services/nginx/conf.d/default.conf* - этот файл смонтирован в контейнер с хоста через volume.

Пример конфигурации Nginx в контейнере

```
server {
    listen 80;

    client_max_body_size 1000m;

    server_name
    ~^[a-z]+?-[a-z]+?\.domain\.ru$
    .main-dev.corp.abankingeom.ru
    client-lk-browser-dev.abankingeom.ru;

    location /sso/ {
        proxy_pass http://idsrv-api/;
    }

    location /workflow-api/ {
        proxy_pass http://workflow-api/;
    }

    location /profile-api/ {
        proxy_pass http://profile-api/;
    }

    location /client-document-api/ {
        proxy_pass http://client-document-api/;
    }

    location /action-api/ {
        proxy_pass http://action-api/;
    }

    location /application-api/ {
        proxy_pass http://application-api/;
    }

    location /orchestrator-api/ {
        proxy_pass http://orchestrator-api/;
    }
```

```
    location /catalog-api/ {
        proxy_pass http://catalog-api/;
    }

    ...
```

Настройка сервисов

Общие настройки окружения микросервисов находятся в файле */docker-containers/services/.env*.

Пример .env

```
# Реджестри
REGISTRY=artsofte

# Версии включаемых образов
ABANKING_OEM=abankingeom-1.0.0

# Настройки окружения сервисов
IdentityServer__Authority=http://idsrv-api/
IdentityServer__UseRevocationJwtAccessToken=true

ApplicationServices__MonitoringService__Mock__Enable=false
ApplicationServices__ConfirmationService__Host=http://cs-api
ApplicationServices__NotificationService__Host=http://ns-api

HttpServiceHosts__ClientDocumentHost=http://client-document-api
HttpServiceHosts__FileServerHost=http://file-api
HttpServiceHosts__ExternalHost=http://external-api
HttpServiceHosts__GenerateDocumentHost=http://generate-document-api
ProfileApiHttpConnectionServiceSettings__ProfileApiHostAddress=http://profile
-api
HttpServiceHosts__TemplateDocumentRenderHost=http://template-render-api

# Настройки подключения к реляционной БД
Dal__AutoMigration=true
Dal__ServerType=postgresql
Dal__DapperCoreConnectionSetting__Host=postgres
Dal__DapperCoreConnectionSetting__Port=5432
Dal__DapperCoreConnectionSetting__UserID=postgres
Dal__DapperCoreConnectionSetting__Password=rw_dev
Dal__DapperCoreConnectionSetting__Pooling=true
Dal__DapperCoreConnectionSetting__MaxPoolSize=30

RabbitSetting__RabbitAmpqHost=rabbitmq:5672
RabbitSetting__RabbitHttpUri=http://rabbitmq:15672
RabbitSetting__User=mquser
RabbitSetting__Password=Pulse1data
RabbitSetting__RabbitHttpApiEnabled=true
RabbitSetting__VHost=abankingeom

Redis__SentinelConnectionOptions=redis:26379
Redis__ServiceName="serviceName=mymaster"
```

```
Logger__OurLogLevel=Information
Logger__LogLevelFile=Information
Logger__LogLevelConsole=Information
Logger__SystemLogLevel=Warning
Logger__LoggingMiddlewareEnabled=true
Logger__UnwantedForLogPart[0]=swagger

TenantEnvironment__IsSearchOrchestrator=false

ConfirmOperation__Default__IsUseDbSettings=true
Debug__IsEnable=false
```

В каждом каталоге сервиса, присутствует файл настроек */docker-containers/services/<SERVICE_NAME>/.env*. В нём указаны переменные среды, уникальные для данного микросервиса. Например:

Пример файла настроек микросервиса

```
# Имя реляционной базы данных микросервиса
Dal__DapperCoreConnectionSetting__Database=document-search-api
```

В поставке, уже указаны реквизиты, нужные для подключения к БД в контейнере, на одном хосте с сервисами.

В случае, когда БД размещается на отдельном хосте, необходимо изменить эти реквизиты (хост, порт, логи, пароль, имя пользователя). Сервисы сами создают БД при первом старте.

Все микросервисы бекенда подключаются к RabbitMQ, PostgreSQL и Redis.

Примечание: База данных PostgreSQL, размещается в Docker-контейнере, только в варианте тестового развертывания. В продакшене, БД размещается и устанавливается на отдельном сервере.

В файле */docker-containers/services/docker-compose.yml* указан список контейнеров с конфигурациями запуска. В нём можно использовать переменные для тегов.

Пример указания переменной тега

```
  provider-lk-browser:
    image: abankingeom-provider-lk-browser:${ABANKING_OEM_VERSION}
    networks:
      - network
    volumes:
      - "./provider-lk-browser/settings/:/usr/share/nginx/html/settings"
    restart: unless-stopped
```

В файле */docker-containers/services/.env* можно задать переменную тегов.

Пример указания тегов версий

```
ABANKING_OEM_VERSION=1.0.0
```

Настройка почты в микросервисе уведомлений

Важно: До старта docker-compose необходимо установить настройки SMTP/SMPP сервиса и реквезиты почты/телефона администратора продукта (провайдера).

В */docker-containers/services/ns-api/DefaultSetting/appsettings.DefaultSetting.json* прописываем настройки для подключения к почтовому серверу (Host, Port, User, Password, From).

Настройки ns-api

```
{
  "AvailableSendingTimeSettings": null,
  "Smtp": {
    "Enable": true,
    "Anonymous": false,
    "Host": "smtp.mail.ru",
    "Port": 587,
    "User": "abankingeom@mail.ru",
    "Password": "",
    "From": "abankingeom@mail.ru",
    "CheckCertificateRevocation": false,
    "SslMode": 1,
    "ContextualKey": ""
  },
  "Smpp": {
    "IsMocked": false,
    "Enable": false,
    "Host": "",
    "Password": "",
    "UserName": "",
    "FromName": "",
    "ContextualKey": ""
  },
  "ContextualSmtp": [],
  "ContextualSmpp": [],
  "WebMessengerConfig": {
    "Enable": true,
    "MockAllSms": true,
    "Webhook": "",
    "FromName": ""
  }
}
```

В */docker-containers/services/profile-api/.env* прописываем почту для отправки пароля учетной записи провайдера:

Настройки profile-api

```
ProviderCreator:Email=confirm@gmail.com
```

Запуск

Создание сети

Создать виртуальную сеть в докере для работы сервисов (этот пункт опционально - сеть можно оставить дефолтную и убрать этот параметр из docker-compose.yml, либо указать свое название сети для компоуза и всех сервисов).

Создание сети

```
docker network create network
```

Запуск инфраструктурных микросервисов

Перейти в каталог */docker-compose/infra* и запустить инфраструктурные сервисы, необходимые для работы приложений.

Запуск инфраструктуры

```
cd /docker-containers/infra
docker-compose up -d
```

Увеличение максимального количества подключений PostgreSQL

В случае необходимости в настройках PostgreSQL возможно увеличить максимальное кол-во подключений.

Для этого:

Если PostgreSQL развернут процессом:

1. Открыть файл */var/lib/postgres/postgresql.conf*
2. Вместо *max_connections = 100*, указать *max_connections = 1000*
3. Перезапустить БД командой

   Перезапуск процесса PostgreSQL

   ```
   sudo systemctl restart postgresql
   ```

Если в контейнере:

1. Открыть файл */docker-containers/abankingeom-infra/postgres/postgresql.conf*
2. Вместо *max_connections = 100*, указать *max_connections = 1000*
3. Перейти в директорию инфраструктуры и перезапустить БД

   Перезапуск контейнера PostgreSQL

   ```
   cd /docker-containers/infra
   ```

```
docker-compose up -d --force-recreate postgres
```

Запуск приложения

Запустить образы приложений из директории */docker-compose/services*

Запуск мискросервисов

```
cd /docker-compose/services
docker-compose up -d
```

Проверка

Проверяем запущенные контейнеры

Проверить, что контейнеры не в рестарте, имеют статус *Up* больше 20 секунд. Команда выполняется без привязки к каталогу:

Вывод информации о запущенных контейнерах

```
docker ps
```

Вывод при исправной работе

```
CONTAINER ID   IMAGE
COMMAND                  CREATED             STATUS              PORTS
NAMES
cd5b71d2377f   abankingeom-orchestrator-api:abankingeom-dev
"dotnet Api.dll"         3 minutes ago       Up 3 minutes        80/tcp,
443/tcp
abankingeom-dev_orchestrator-api_1
29a6d66fc09c   abankingeom-profile-api:abankingeom-dev
"dotnet Api.dll"         5 minutes ago       Up 5 minutes        80/tcp,
443/tcp
abankingeom-dev_profile-api_1
d7665dce2d5f   abankingeom-workflow-api:abankingeom-dev
"dotnet Api.dll"         5 minutes ago       Up 5 minutes        80/tcp,
443/tcp
abankingeom-dev_workflow-api_1
ccdc2e2a56cd   abankingeom-reserve-copy-api:abankingeom-dev
"dotnet Api.dll"         10 minutes ago      Up 10 minutes       80/tcp,
443/tcp
abankingeom-dev_reserve-copy-api_1
6a4598c052cc   abankingeom-client-document-api:abankingeom-dev
"dotnet Api.dll"         11 minutes ago      Up 11 minutes       80/tcp,
443/tcp
abankingeom-dev_client-document-api_1
e942a3f11b43   abankingeom-async-operation-api:abankingeom-dev
"dotnet Api.dll"         15 minutes ago      Up 15 minutes       80/tcp,
443/tcp
abankingeom-dev_async-operation-api_1
3f01a2b748fb   abankingeom-action-api:abankingeom-dev
"dotnet Api.dll"         15 minutes ago      Up 15 minutes       80/tcp,
443/tcp
```

Вывод при наличии ошибок

```
CONTAINER ID   IMAGE
COMMAND                  CREATED             STATUS
PORTS
NAMES
cd5b71d2377f   abankingeom-orchestrator-api:abankingeom-dev
"dotnet Api.dll"         3 minutes ago       Restarting (139) 18 seconds ago
80/tcp, 443/tcp
abankingeom-dev_orchestrator-api_1
29a6d66fc09c   abankingeom-profile-api:abankingeom-dev
"dotnet Api.dll"         5 minutes ago       Up 5 minutes
80/tcp, 443/tcp
abankingeom-dev_profile-api_1
d7665dce2d5f   abankingeom-workflow-api:abankingeom-dev
"dotnet Api.dll"         5 minutes ago       Up 5 minutes
80/tcp, 443/tcp
abankingeom-dev_workflow-api_1
ccdc2e2a56cd   abankingeom-reserve-copy-api:abankingeom-dev
"dotnet Api.dll"         10 minutes ago      Up 10 minutes
80/tcp, 443/tcp
abankingeom-dev_reserve-copy-api_1
6a4598c052cc   abankingeom-client-document-api:abankingeom-dev
"dotnet Api.dll"         11 minutes ago      Up 11 minutes
80/tcp, 443/tcp
abankingeom-dev_client-document-api_1
e942a3f11b43   abankingeom-async-operation-api:abankingeom-dev
"dotnet Api.dll"         15 minutes ago      Restarting (139) 57 seconds ago
80/tcp, 443/tcp
abankingeom-dev_async-operation-api_1
3f01a2b748fb   abankingeom-action-api:abankingeom-dev
"dotnet Api.dll"         15 minutes ago      Up 15 minutes
80/tcp, 443/tcp
```

Далее войти в кабинет провайдера по дольмену указанному в *server_name* хостового Nginx, например, provider-lk-browser-dev.abankingeom.ru.

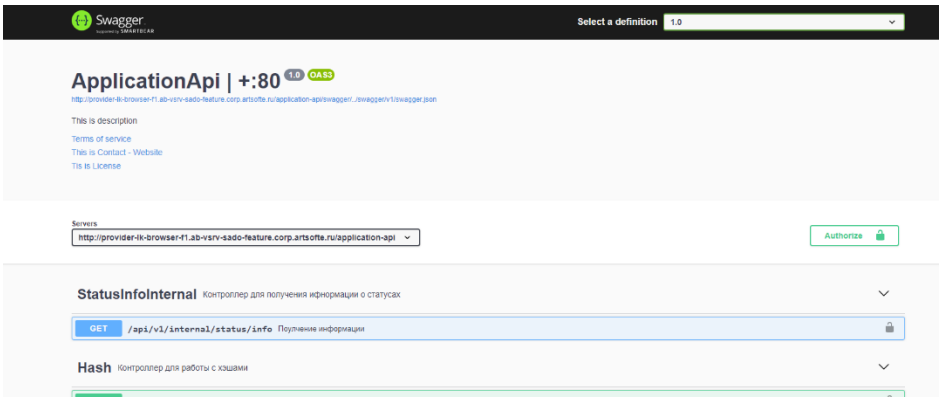
Стартовая страница должна успешно загрузиться, без ошибок.

Доступность Swagger

Если Swagger включен на хостовом Nginx и в */docker-containers/services/.env* (SwaggerDocs:Enable=true), можно проверить доступность каждого из сервисов бекенда, выполнив http://provider-lk-browser-dev.abankingeom.ru/action-api/swagger (вместо action-api, может быть любой другой сервис).

Должна загрузится страница следующего вида:

Скриншот



Примененние бэкапа данных

Бекап - резервное копирование базы данных для последующего восстановления.

Бекапы БД нужны, чтобы сохранить состояние БД с возможностью вернуть это состояние системе.

Поставка NOCODE использует БД postgresql. Каждая из этих БД предоставляет функционал снятия и применения бекапов.

Помимо данных в БД, продукт хранит файлы пользователей и файлы, относящиеся к процессам и кабинетам (тематизация, шаблоны документов и т.д.).

Разархивировать бекап

Необходимо подключиться к целевой ВМ, перейти в директорию с архивом и разархивировать его.

Для разархивирование можно воспользоваться командой:

```
tar -xvf <ИМЯ АРХИВА>.tar.gz
```

Например:

```
tar -xvf bckp_2024_01_01.tar.gz
```

Остановить работу продукта

Чтобы данные бекапа были консистентными, необходимо остановить микросервисы. Для этого в директории с docker-compose.yml, в котором описаны микросервисы NOCODE, выполняем команду

```
docker-compose down
```

Копирование файлов в примонтированную директорию

Чтобы перенести файлы в примонтированную директорию:

cp -r <исходная директория> <целевая директория, в которую переносим>

При этом переносимые файлы перезапишут существующие в примонтированной директории.

Пример

```
> docker cp db.out 9ea66a1cde79:/var/lib/postgresql
> docker exec -it 9ea66a1cde79 bash
> cd /var/lib/postgresql
> psql -U postgres -f db.out postgres
*ВОССТАНАВЛИВАЕТСЯ БЕКАП, ОЖИДАЕМ*
*НАЖАТЬ CTRL+D*
```

Применение бекапа postgresql

Для применения бекапа воспользуемся pg_restore внутри контейнера postgresql.

1. находясь в директории с файлом бекапа скопируем файл бекапа в контейнер: *docker cp db.out <postgres_container_name>:/var/lib/postgresql/*
2. `docker exec -it <postgres_container> bash`
3. `cd /var/lib/postgresql`
4. `psql -U postgres -f db.out postgres`

Запустить продукт

После снятия бекапов, можно запустить сервисы продукта. Для этого в директории с docker-compose.yml, в котором описаны микросервисы NOCODE, выполняем команду

docker-compose up -d