Описание установки C-View-in-docker

Для автоматизации установки нашей системы C-VIEW существует контейнеризированная версия - c-view-in-docker. Содержит в себе portal-admin, c-view-frontend и все вспомогательные сервисы, необходимые для разворачивания портала за несколько минут.

Все необходимые для развертывания файлы находятся в одноименном репозитории c-view-in-docker.

Как запускать

Для полноценной работы инсталляции на основе c-view-in-docker нужно будет получить от заказчика 6 доменных имен и указать их в .env файле в корневом каталоге репозитория:

```
PORTAL_DOMAIN=portal.infra-test1.openstack
PORTAL_ADMIN_DOMAIN=portal-admin.infra-test1.openstack
PORTAL_SD_DOMAIN=sd.portal-admin.infra-test1.openstack
CENTRIFUGO_DOMAIN=websock.infra-test1.openstack
MINIO_DOMAIN=s3.infra-test1.openstack
MINIO_CONSOLE_DOMAIN=console.s3.infra-test1.openstack
```

Плюс не забыть сменить пароли и секреты от MYSQL, CENTRIFUGO, RABBIT и т.д.

Затем перейти в каталог cview-portal-admin и скопировать туда репозиторий portal-admin в одноименную директорию:

```
cd cview-portal-admin
git clone ssh://git@gitlab.coms.ru:2202/portal/portal-admin.git
```

Таким же образом необходимо поступить с фронтендом портала. Переходим в каталог cview-client-nginx и клонируем репозиторий c-view-frontend:

```
cd ../cview-client-nginx
git clone ssh://git@gitlab.coms.ru:2202/frontend/c-view-frontend.git
```

Остаемся в этом каталоге и редактируем файл .env.development. Вставляем туда следующие переменные:

```
VUE_APP_BACKEND_HOST=http://portal-admin.infra-test1.openstack
VUE_APP_BACKEND_SOCKET_URL=ws://websock.infra-test1.openstack/connection/
websocket
VUE_APP_CLIENT_ID=UWaet1Aiqu9quainieCoo7thongo3vahke7doo7thietheeZ
VUE_APP_BACKEND_BASE_URL=http://portal-admin.infra-test1.openstack/json-rpc/
VUE_APP_BACKEND_HOST=http://portal-admin.infra-test1.openstack
VUE_APP_ADMIN_URL=http://portal-admin.infra-test1.openstack
```

Обращаем внимание на то, что VUE_APP_CLIENT_ID должен совпадать с PORTAL_API_CLIENTS из корневого .env-файла

Возвращаемся в корень c-view-in-docker и запускаем docker-compose:

```
cd ../../
docker-compose up -d --build
```

Сборка небыстрая и занимает около 20 минут.

После сборки необходимо создать суперпользователя в системе:

```
docker-compose exec cview-portal-admin php ./yii install/create-admin
adminuser 1234567 admin@example.org
```

На этом с запуском все. Переходим в админку по доменному имени, которое задавали в переменной PORTAL_ADMIN_DOMAIN и добавляем сервер мониторинга и создаем компанию.

Подгружаем несколько хостов и только после этого проверяем фронтенд.

Cron-задачи

Для добавления новых cron-задач, в случае необходимости в этом, нужно отредактировать файл cview-portal-admin/etc/crontab/portal-crontab и добавить в него новую строку по шаблону:

```
40 0 * * * yii-cron-wrapper.sh ${PORTAL_ADMIN_DOMAIN} scheduled-work/device-
filters
```

В конце файла обязательно нужно оставить пустую строку, иначе crond не заработает.

Описание основных Dockerfile и сервисов

Основу сборки составляют portal-admin и c-view-frontend.

cview-portal-admin

Это основной контейнер системы который отвечает за работу с FASTCGI-сервером php-fpm, supervisor-задач и cron-задач.

Построен на базовом образе **`debian:stable-slim.`**

Структура каталога для сборки:

```
cview-portal-admin
├── Dockerfile
├── build_project_env.sh
├── etc
│   ├── crontab
│   │   ├── portal-crontab
│   │   ├── yii-cron-wrapper.sh
│   │   └── yii-cron-wrapper.sh_wo_flock
│   ├── fpm
│   │   ├── php-fpm.conf
│   │   ├── php.ini
│   │   └── pool.d
│   │       └── www.conf
│   └── supervisor
│       ├── conf.d
│       │   ├── yii-consumer.ini
│       │   ├── yii-handler-processes.ini
│       │   └── yii-rsm-incidents.ini
│       └── supervisord.conf
└── portal.customs
    ├── clickhouse.cview_monitoring_tasks.php
    ├── clickhouse.devices_statuses_history.php
    ├── clickhouse.items_statistics.php
    ├── clickhouse.sla_dates.php
    ├── db.console.php
    ├── db.custom.php
    ├── params.custom.php
    ├── redis.connection.custom.php
    └── redis.custom.php
```

- Директория etc - содержит основные конфиги php-fpm, supervisord и cron
- Директория portal.customs - содержит все кастомные конфигурационные файлы портала. Редактировать их не надо, т.к. все значения подставляются из .env файла.
- build_project_env.sh - файл инициализации. Импортирует глобальные переменные, чтобы они были доступны для crond. Проверяет что RabbitMQ, Clickhouse и MySQL доступны и только после этого запускает supervisor и php-fpm.

cview-client-nginx

Контейнер с фронтом C-VIEW.

Выполняет мультистейджинг сборку кода из репозитория c-view-frontend, затем копирует собранные артифакты в имедж с веб-сервером nginx.

Построен на базовом образе **`nginx:stable-alpine.`**

Структура каталога для сборки:

```
cview-client-nginx
├── Dockerfile
└── etc
    ├── nginx.conf
    └── templates
        └── portal-client.conf.template
```

- Директория etc - содержит в себе конфигурационные файлы для nginx.

Сами конфигурационные файлы собираются при помощи переменных из основного .env-файла c-view-in-docker при помощи `envsubst, который включен в базовый образ по-умолчанию.`

cview-admin-nginx

Контейнер с сервером nginx для обслуживания статики и проксирования запросов к php-скриптам portal-admin.

Построен на базовом образе **nginx:stable-alpine.**

Структура каталога для сборки:

```
cview-admin-nginx
├── Dockerfile
├── etc
│   └── nginx.conf
└── templates
    ├── centrifugo.conf.template
    ├── monitoring.conf.template
    ├── portal-admin.conf.template
    └── portal-sd.conf.template
```

- Директория etc - содержит в себе конфигурационные файлы для nginx. Шаблоны так же собираются при помощи `envsubst.`

proxy-nginx

Этот контейнер отвечает за выход всех сервисов наружу, т.е. проброс портов на хост-машину и проксирование запросов по нужным эндпоинтам.

Построен на базовом образе **nginx:stable-alpine.**

Структура каталога для сборки:

```
proxy-nginx
├── Dockerfile
└── etc
    └── nginx
        ├── proxy_params
        └── templates
            ├── centrifugo.conf.template
            ├── default.conf.template
            ├── minio.conf.template
            ├── portal-admin.conf.template
            ├── portal-client.conf.template
            └── portal-sd.conf.template
```

percona

Этот контейнер отвечает за БД percona-mysql.

Построен на базовом образе **percona:ps-5.7.35.**

Структура каталога:

```
percona
├── conf
│   └── my.cnf
└── init.d
    ├── 10-init-users.sh
    └── 20-init-db.sh
```

- Директория conf - содержит конфигурационные файлы для mysql.
- Директория init.d содержит инициализационные скрипты, которые по данным из .env файла создают пользователей и базу данных для портала.

clickhouse-server

Контейнер отвечает за БД clickhouse.

Построен на базовом образе **yandex/clickhouse-server.**

```
clickhouse-server
├── etc
│   └── config.xml
```

```
└── init.d
    └── init-db.sh
```

- Директория etc - содержит основные файлы конфигурации сервера clickhouse.
- init.d/init-db.sh - инициализационный файл, который при старте контейнера создает базы данных из env-файла.

rabbitmq

Контейнер сервера очередей rabbitmq.

Построен на базовом образе **rabbitmq:3.8.11-management**.

Структура каталога:

```
rabbitmq
├── Dockerfile
└── init.sh
```

- Файл init.sh - отвечает за создание пользователей указанных в .env файле.

redis

Контейнер отвечает за по БД Redis.

Построен на базовом образе **redis:6.2-alpine**

Структура каталога:

```
redis
└── redis.conf
```

- redis.conf - конфигурационный файл сервера redis.

centrifugo

Контейнер отвечает за работу веб-сокетов.

Конфиг-файлов не требует, настраивается через env переменные.

Построен на базовом образе **centrifugo/centrifugo:v2.8.4**

sphinxsearch

Контейнер отвечает за работу поисковой системы sphinx для фронта портала.

Построен на базовом образе **alpine:latest**

Структура каталога:

```
sphinxsearch
├── Dockerfile
├── etc
│   └── sphinx.conf
└── init.sh
```

- Директория etc - содержит конфиг. файлы сервиса sphinxsearch.
- init.sh - выполняет запуск searchd при старте контейнера.

minio

Контейнер поднимает объектное S3 хранилище Minio.

Построен на базовом образе **quay.io/minio/minio:RELEASE.2022-05-26T05-48-41Z.**

Настраивается полностью в docker-compose.yml

Как обновлять

Для обновления версии приложения в контейнерах, просто обновляем код в директориях cview-portal-admin/portal-admin, cview-client-nginx/c-view-frontend.

При обновлениис-view-frontend обязательно сохраняем содержимое .env.development, и после перезалива кода из репозитория - снова помещаем содержимое в cview-client-nginx/c-view-frontend/.env.development, иначе соберется нерабочий дистрибутив фронтенда.

И выполняем:

```
docker-compose up -d --build
```

Контейнеры, в случае изменения кода, пересоберутся, и будут запущены необходимые миграции.