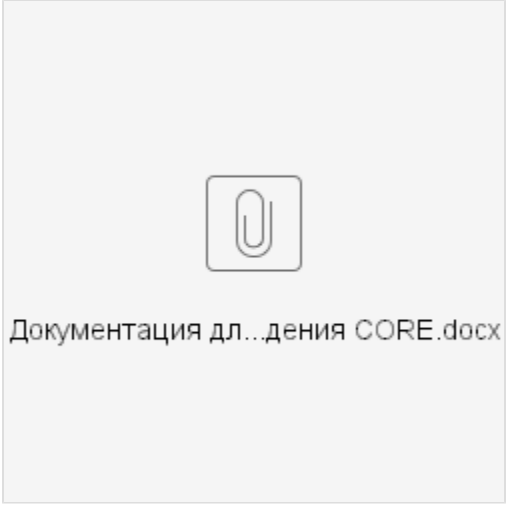
Развёртывание платформы

Инструкция по развёртыванию платформы CORE:



Развертывание с помощью докера

Выложен в https://github.com/essence-community/core

Как запустить:

0. Создать виртуалку со следующими параметрами:

- Linux/Windows
- оперативки минимум 6 гб
- жесткого минимум 40 гб
- доступ в интернет.
  Если интернет по прокси, тогда надо добавить в docker-compose.yml в блоки web-build и gate-build после environment:
  http_proxy: http://HOST:PORT
  https_proxy: http://HOST:PORT

1. Установить docker. Подробнее о работе с docker тут
2. Установить docker-compose
3. Склонировать репозиторий

```
git clone https://github.com/essence-community/core
```

Лирическое отступление для Windows

Если установка производится на Windows, для успешного выполнения следующего шага необходимо в файле docker-compose.yml (расположен в '\core\docker') удалить строку:

```
- './gate/config:/opt/work_gate/configs'
```

В случае необходимости повторной сборки необходимо выполнить команду:

```
git clean -fdx
```

4. Зайти в папку склонированного репозитория и выполнить

```
git submodule init
git submodule update -f --remote
```

5. Зайти в папку docker

6. Запустить сборку образов:

```
docker-compose -f docker-compose.yml build
```

7. Запуск

```
docker-compose -f docker-compose.yml up

docker-compose -f docker-compose.yml up -d
```

Патчинг

1. Скачиваем нужный релиз бд https://github.com/essence-community/core-backend/tree/2.0.0/dbms
2. Скачиваем patcher
3. Прописываем в PATH ссылку на патчер
4. Настраиваем liquibase.properties на БД, с которой хотим снять дамп
5. Заходим в папку dbms

```
patcher -c query -qp " " -p liquibase.
properties                                    --    meta
patcher -c syssetting -p liquibase.
properties
--
patcher -c message -p liquibase.
properties
--    meta   t_message   1000+
patcher -c page -o "id //" -p liquibase.properties         --    meta
  //
patcher -c class -p liquibase.
properties
--    meta
patcher -c lang -p liquibase.
properties
--    meta
patcher -c lang -o ru_RU -p liquibase.
properties                                                          --
  meta     (ru_Ru)
```

6. Полученные дампы прописываем в include meta/meta.xml. Дампы page должны в списке идти самыми последними
7. Если были добавлены новые сервисы, то необходимо включить их в файл dbms\t_query\query.xml

Если в репозитории были изменения после скачивания релиза необходимо:

1. зайти в docker и выполнить:

```
git submodule update -f --remote
docker-compose start liquibase
docker-compose ps
```

2. `liquibase   exit 0,`
3.

Инструкция по развёртыванию

1. Введение

1.1 Цель

Данный документ содержит последовательность действий по развёртыванию и обновлению «Технологической платформы CORE».

1.2 Определения и сокращения

Термин	Описание
CORE	Технологическая платформа CORE
ПО	Программное обеспечение
ИР	Информационные ресурсы
АПК	Аппаратно-программный комплекс

2. Требования к квалификации системного администратора

Для обслуживания Системы администратор должен обладать следующими навыками:

- Навыки работы в серверных Linux либо опыт администрирования *nix подобных систем или Windows;
- Опыт администрирования БД PostgreSql 10+;
- Опыт администрирования веб-сервера nginx 1.14 и выше.

Для поддержки Системы системному администратору необходимо:

- Ознакомиться с документацией производителей используемого аппаратного и системного программного обеспечения.

3. Назначение и условия применения

3.1 Назначение

«Технологическая платформа CORE» представляет собой комплексное решение для построения ERP системы.

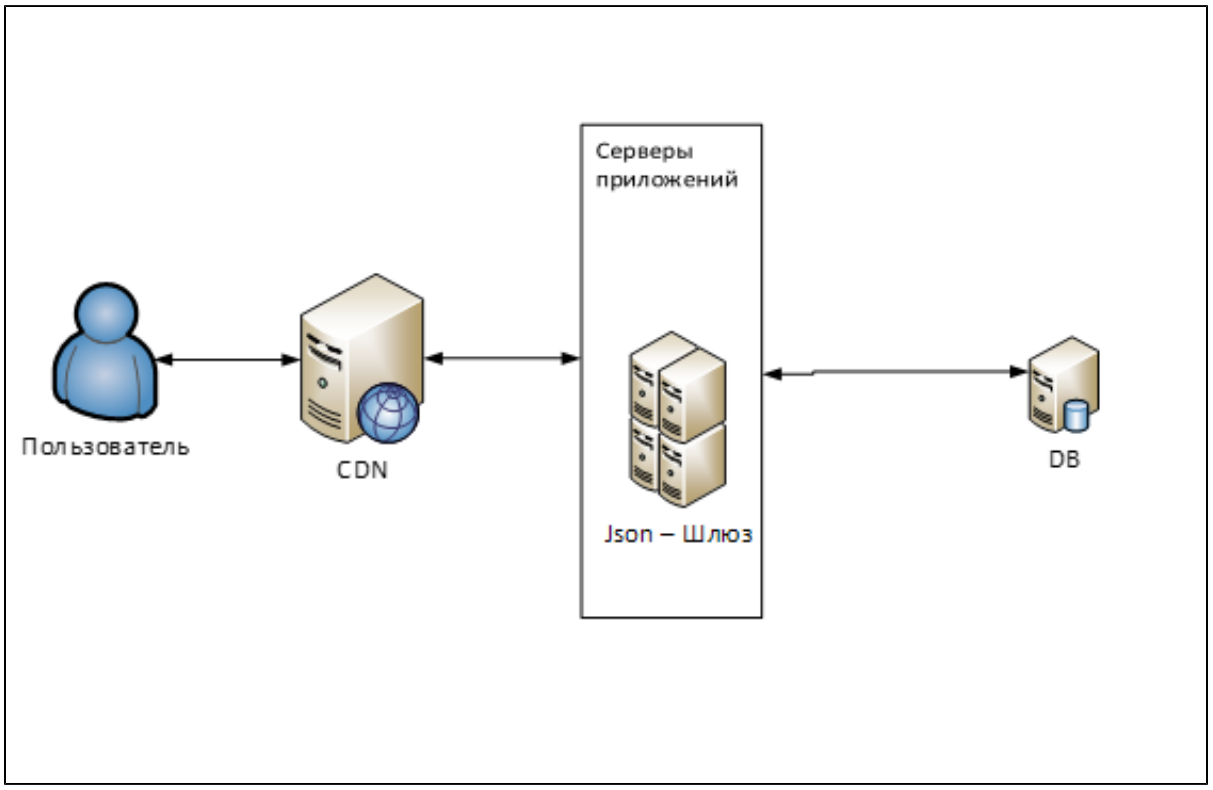
3.2 Условия применения

Для работы Системы на сервере предварительно следует установить:

Сервер	ПО
Сервер БД CORE	СУБД PostgreSQL 10+
	ОС Linux или Windows
Сервер CDN	ОС Linux или Windows
	Nginx 1.14 и выше
Сервер приложений	ОС Linux или Windows
	node.js 10 и выше

4. Логическая схема системы

Логическая схема серверов представлена на рисунке ниже:



5. Описание операций установки и настройки

5.1 Установка NodeJS

5.1.1 Установка NodeJS

Установка производится согласно инструкции: https://nodejs.org/en/download/package-manager/

5.2 Развёртывание базы данных

5.2.1 Развёртывание базы данных

1. Для развёртывания базы данных Системы на сервере БД CORE необходимо установить СУБД PostgreSQL.
2. Создать пользователей:
    - Суперпользователь, от которого идет заливка бд:

```
CREATE ROLE s_su WITH
  LOGIN
  SUPERUSER
  INHERIT
  CREATEDB
  CREATEROLE
  NOREPLICATION;
```

    - Пользователь для входа шлюза:

```
CREATE ROLE s_mc WITH LOGIN NOSUPERUSER INHERIT NOCREATEDB
NOCREATEROLE NOREPLICATION;


ALTER ROLE s_mc SET search_path TO s_mt, public, pg_catalog;
```

    - Пользователь без возможности авторизации, которому будут принадлежать функции и процедуры:

```
CREATE ROLE s_mp WITH
  NOLOGIN
  NOSUPERUSER
  INHERIT
  NOCREATEDB
  NOCREATEROLE
  NOREPLICATION;
```

3. Создать контейнер БД:

```
CREATE DATABASE core
  WITH
  OWNER = s_su
  TEMPLATE = template0
  ENCODING = 'UTF8'
  LC_COLLATE = 'ru_RU.UTF-8'
  LC_CTYPE = 'ru_RU.UTF-8'
  TABLESPACE = pg_default
  CONNECTION LIMIT = -1;

CREATE EXTENSION "uuid-ossp"
    SCHEMA public;
```

4. Распаковать архив релиза и настроить liquibase.properties, указав адрес и порт БД

```
driver: org.postgresql.Driver
url: jdbc:postgresql://127.0.0.1:5432/core
username: s_su
password: s_su
```

5. Запустить файл update

5.3 Развёртывание шлюза

Json – шлюз представляет собой отдельные приложения (файлы-архивы с именами ungate_*.zip).

Создаем файлы конфигурации (например, в папке opt/work_gate/configs для linux или в папке c:\work_gate\configs для Windows):

- t_providers.toml;
- t_context.toml;
- t_plugins.toml;
- t_events.toml;
- t_servers.toml;
- logger.json.

Ниже приводятся настройки этих файлов.

5.3.1 Установка шлюза

Распаковываем архив ungate_*.zip в папку (например, /opt/work_gate/ungate для Linux)

Заходим в папку и выполняем команду:

```
sudo npm install -g yarn
yarn install
```

5.3.2 Настройка списка серверов t_servers.toml

```
[[data]]
ck_id = "core.example.com"
cv_ip = "192.168.1.1"
```

где

ck_id — наименование ноды (hostname машины либо переменная окружения GATE_NODE_NAME, которая задается при старте сервиса)

cv_ip — ip/dns ноды

5.3.3 Настройка подключения контекста данных t_context.toml к БД

```
[[data]]
ck_id = "core"
cv_path = "/api"
cv_description = "  "
ck_d_plugin = "CorePGContext"

  [data.cct_params]
  debug = true
  disableCache = true
  poolMax = 100
  connectString = "postgres://:@127.0.0.1:5432/ "
```

где

ck_id — уникальное наименование

cv_path — путь доступа

cv_description — Описание

ck_d_plugin — наименование плагина

[data.cct_params] — настройки плагина

debug — включаем отладочную информацию в ответе

disableCache — признак отключения кэширования страниц

poolMax — максимальный пулл

connectString — строка подключения «postgres://логин:пароль@127.0.0.1:5432/наименование базы»

5.3.4 Настройка подключения провайдеров данных t_providers.toml к БД

Настройки провайдеров

```
[[data]]
ck_id = "admingate"
cv_description = " "
ck_d_plugin = "admingate"
cct_params = { }

[[data]]
ck_id = "auth"
cv_description = " "
cl_autoload = true
ck_d_plugin = "AuthMock"

  [data.cct_params]
  adminUser = "admin_core"
  adminPassword = "123456"
  viewUser = "view_core"

[[data]]
ck_id = "meta"
cv_description = "   "
cl_autoload = true
ck_d_plugin = "PostgreSQLDb"

  [data.cct_params]
  core = true
  poolMax = 100
  connectString = "postgres://:@127.0.0.1:5432/ "
```

где

ck_id — уникальное наименование

cl_autoload — загрузка при старте шлюза

cv_description — Описание

ck_d_plugin — наименование плагина

adminUser - логин администратора

adminPassword — пароль администратора

5.3.5 Настройка подключения плагинов t_plugins.toml

```
[[data]]
cv_name = "preparequery"
cv_description = "   filter,sort"
ck_d_provider = "meta"
ck_d_plugin = "PrepareQuery"
cl_required = 1
cl_default = 0
cct_params = { }
ck_id = "PrepareQuerymeta"
cn_order = 1

[[data]]
cv_name = "extractrow"
cv_description = "   json"
ck_d_provider = "meta"
ck_d_plugin = "JsonRowColumnExtractor"
cl_required = 1
cl_default = 0
ck_id = "extractRowmeta"
cn_order = 2

  [data.cct_params]
  columns = "json,result"
  extractSingleColumn = false
```

5.3.6 Настройка системы оповещений t_events.toml

```
[[data]]
cv_description = " meta"
ck_d_plugin = "CorePgNotification"
ck_id = "meta"

  [data.cct_params]
  authProvider = "auth"
  connectString = "postgres://:@127.0.0.1:5432/ "

[[data]]
cv_description = " meta"
ck_d_plugin = "CorePgSemaphore"
ck_id = "semaphore"

  [data.cct_params]
  connectString = "postgres://:@127.0.0.1:5432/ "
```

5.3.7 Настройка логирования logger.json

Создаем папку /opt/work_gate/logs

```
{
    "handlers": {
        "errors": {
            "class": "rufus/handlers/rotating",
            "file": "/opt/work_gate/logs/error.log",
            "level": "ERROR",
            "maxSize": "30mb",
            "maxFile": "30"
        },
        "main": {
            "class": "rufus/handlers/rotating",
            "file": "/opt/work_gate/logs/main.log",
            "maxSize": "30mb",
            "maxFile": "30"
        },
        "console": {
            "class": "rufus/handlers/console"
        }
    },
    "loggers": {
        "root": {
            "level": "TRACE",
            "handlers": ["main","errors"]
        }
    }
}
```

5.3.8 Запуск сервера windows

1. Зайдем в папку с распакованным сервером и выполним команды:

```
yarn global add node-windows
yarn link node-windows
echo LOGGER_CONF=c:\work_gate\configs\logger.json>>.env.svc
echo PROPERTY_DIR=c:\work_gate\configs>>.env.svc
echo GATE_UPLOAD_DIR=c:\work_gate\tmp>>.env.svc
echo NEDB_TEMP_DB=c:\work_gate\tmp\db>>.env.svc
yarn install-svc
```

   где

   LOGGER_CONF — ссылка на настройки логера

   PROPERTY_DIR — указываем папку, где лежат настройки (t_providers.toml, t_context.toml, t_plugins.toml)

2. Запуск:

```
sc start gate-core
```

5.3.9 Запуск сервера linux systemctl

1. Создаем скрипт запуска /opt/work_gate/start.sh

```
#!/bin/bash
cd /opt/work_gate/ungate
/opt/work_gate/ungate/node_modules/.bin/nodemon 1>/dev/null 2>/opt
/work_gate/logs/daemon_error.log
```

   где

   /opt/work_gate/ungate — место, где распакован шлюз

2. выполняем команду:

```
sudo chmod +x /opt/work_gate/start.sh
```

3. Создаем сервис /etc/systemd/system/gate-core.service

```
[Unit]
Description=Core Nodemon
After=network.target

[Service]
#User=web-nodejs
LimitNOFILE=infinity
LimitNPROC=infinity
LimitCORE=infinity
Environment=LOGGER_CONF=/opt/work_gate/configs/logger.json
Environment=GATE_CLUSTER_NUM=4
Environment=PROPERTY_DIR=/opt/work_gate/configs
Environment=GATE_UPLOAD_DIR=/opt/work_gate/tmp
Environment=NEDB_TEMP_DB=/opt/work_gate/tmp/db
WorkingDirectory=/opt/work_gate/ungate
ExecStart=/opt/work_gate/start.sh

[Install]
WantedBy=multi-user.target
```

   где

   PROPERTY_DIR — указываем папку, где лежат настройки (t_providers.toml, t_context.toml, t_plugins.toml)

   LOGGER_CONF — ссылка на настройки логера

   WorkingDirectory — папка распакованного шлюза

4. Запуск:

```
systemctl start gate-core
```

5.4 Настройка nginx на CDN сервере Системы

На **CDN** сервере должен быть предустановлен веб-сервер **nginx** и создана директория для хранения фалов приложения.

В созданную директорию необходимо распаковать архив **core_*.zip** (например, /opt/www_core или c:\www_core).

В конфигурационном файле веб-сервера conf.d/core.conf необходимо прописать путь подключения к json-шлюзу и виртуальному хосту приложения.

5.4.1 Подключение к json-шлюзам

Имя параметра: server

Значение параметра: проксирование запросов к json-шлюзам и ссылка на директорию с файлами сборки

Пример:

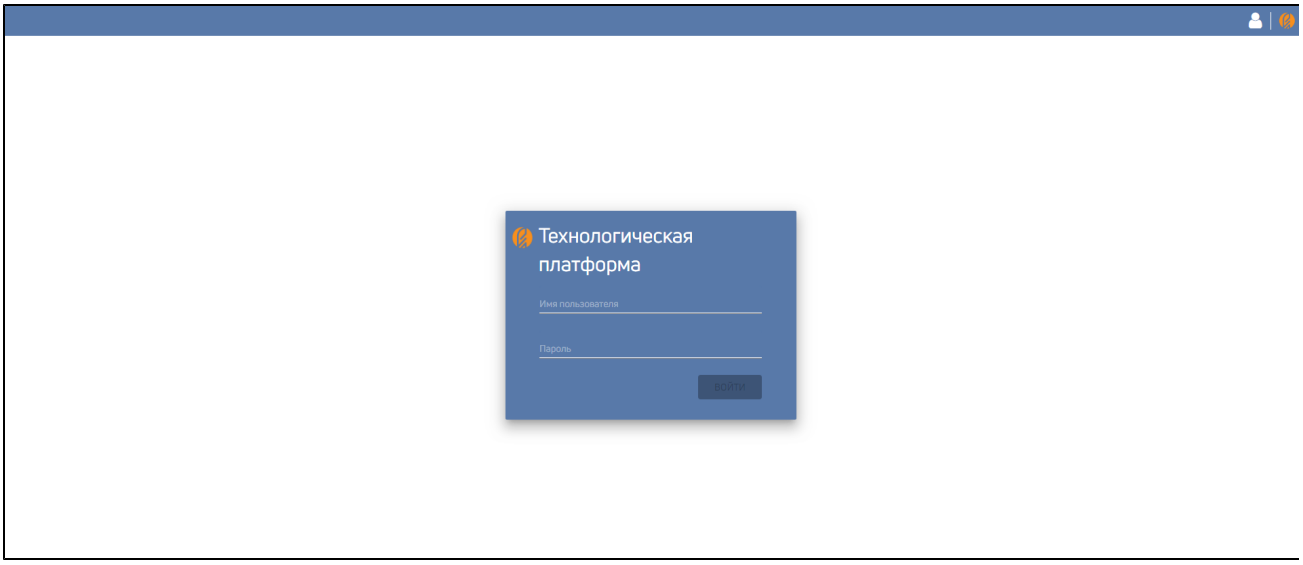
```
map $http_upgrade $connection_upgrade {
    default upgrade;
    ''      close;
}

server {
    listen       80;
    server_name  localhost;
    access_log  /var/log/nginx/access_core.log  main;
    root   /opt/www_core; #

    location / {
        try_files $uri $uri/ /index.html;
        index  index.html index.htm;
    }
    location /gate-core {
        proxy_pass http://127.0.0.1:8080/api;
    }

    location /core_notification {
        proxy_pass http://127.0.0.1:8080/notification;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "Upgrade";
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }
}
```

где

http://127.0.0.1:8080 - адрес машины с запущенным шлюзом

5.5 Проверка функционирования Системы

1. Перейдите по ссылке http://$CDN_HOST/
2. Ответ сервера должен быть следующего вида:



3. В окне авторизации введите логин и пароль (см. пункт 5.3.3). Если Система работоспособна, то отобразится главная страница, с возможностью выбора модуля:

5.6 Работа в режиме кластера

5.6.1 Настройка шлюза

1. Добавляем дополнительные сервера в t_servers.toml

```
[[data]]
ck_id = "core1.example.com"
cv_ip = "192.168.1.1"
[[data]]
ck_id = "core2.example.com"
cv_ip = "192.168.1.2"
```

2. Перезагружаем шлюзы

5.6.2 Настройка conf.d/core.conf

```
upstream nodejscluster {
        least_conn;
        ip_hash;
        server 192.168.1.1:8080;
        server 192.168.1.2:8080;
    }
map $http_upgrade $connection_upgrade {
    default upgrade;
    ''      close;
}

server {
    listen       80;
    server_name  localhost;
    access_log  /var/log/nginx/access_core.log  main;
    root   /opt/www_core; #

    location / {
        try_files $uri $uri/ /index.html;
        index  index.html index.htm;
    }
    location /gate-core {
        proxy_pass http://nodejscluster/api;
    }

    location /core_notification {
        proxy_pass http://nodejscluster/notification;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "Upgrade";
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }
}
```

6 Обновление

6.1 Обновление БД

1. Снимаем дамп с БД
2. Распаковываем архив и настраиваем liquibase.properties

```
driver: org.postgresql.Driver
url: jdbc:postgresql://127.0.0.1:5432/core
username: s_su
password: s_su
```

3. Запускаем update

6.2 Обновление шлюза

1. Распаковываем архив ungate_*.zip в папку ранее установленного шлюза
2. Заходим в папку и выполняем команду

```
yarn install
```

3. Перезапускаем шлюз

6.3 Обновление фронт

Распаковываем архив с фронтом в папку nginx (/opt/www_core)