|  |  |  |  |
| --- | --- | --- | --- |
|  | | | |
|  |  |  | |
| Документация, необходимая для установки и эксплуатации программного продукта CORE    **На 13 страницах** | | | |
|  | | | |
|  |  | |  |

Оглавление

[1 Введение 2](#__RefHeading___Toc2267_66293550)

[2 Требования к квалификации системного администратора 3](#__RefHeading___Toc2269_66293550)

[3 Назначение и условия применения 3](#__RefHeading___Toc2271_66293550)

[4 Логическая схема системы 3](#__RefHeading___Toc2273_66293550)

[5 Описание операций 4](#__RefHeading___Toc2275_66293550)

[5.1.1 Установка NodeJS 4](#__RefHeading___Toc2277_66293550)

[5.2.1 Развёртывание базы данных 4](#__RefHeading___Toc2279_66293550)

[5.3 Развёртывание шлюза 5](#__RefHeading___Toc2281_66293550)

[5.3.1 Установка шлюза 5](#__RefHeading___Toc2283_66293550)

[5.3.2 Настройка списка серверов t\_servers.toml 6](#__RefHeading___Toc2051_3606195140)

[5.3.3 Настройка подключения контекста данных t\_context.toml к БД 6](#__RefHeading___Toc2285_66293550)

[5.3.4 Настройка подключения провайдеров данных t\_providers.toml к БД 7](#__RefHeading___Toc2287_66293550)

[5.3.5 Настройка подключения плагинов t\_plugins.toml 8](#__RefHeading___Toc2289_66293550)

[5.3.6 Настройка системы оповещени t\_events.toml 8](#__RefHeading___Toc2053_3606195140)

[5.3.7 Настройка логирования logger.json 9](#__RefHeading___Toc2291_66293550)

[5.3.8 Запуск сервера windows 9](#__RefHeading___Toc2293_66293550)

[5.3.9 Запуск сервера linux systemctl 9](#__RefHeading___Toc2295_66293550)

[5.4 Настройка nginx на CDN сервере Системы 10](#__RefHeading___Toc2297_66293550)

[5.4.1 Подключение к json-шлюзам 10](#__RefHeading___Toc2299_66293550)

[5.4.2 Подключение к файлам сборки conf.d/core.conf 11](#__RefHeading___Toc2301_66293550)

[5.5 Проверка функционирования Системы 11](#__RefHeading___Toc2303_66293550)

[5.6 Работа в режиме кластера 12](#__RefHeading___Toc2055_3606195140)

[5.6.1 Настройка шлюза 12](#__RefHeading___Toc2057_3606195140)

[5.6.2 Настройка conf.d/core.conf 13](#__RefHeading___Toc2059_3606195140)

# Введение

* 1. Цель

Данный документ содержит последовательность действий по разворачиванию «Технологической платформы».

* 1. Определения и сокращения

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

# 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;

# Назначение и условия применения

* 1. Назначение

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

* 1. 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 |

# Логическая схема системы

Логическая схема серверов представлена на рисунке ниже:



Рис. 1 Логическая схема системы

# Описание операций

* 1. Установка NodeJS

### Установка NodeJS

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

* 1. Развёртывание базы данных

### Развёртывание базы данных

Для развёртывания базы данных Системы на сервере БД Core необходимо установить СУБД PostgreSQL.

Создать пользователей:

Суперпользователь от которого идет заливка бд:

Пользователь для входа шлюза:

CREATE ROLE s\_su WITH

LOGIN

SUPERUSER

INHERIT

CREATEDB

CREATEROLE

NOREPLICATION;

Пользователь без возможности авторизации которому будут принадлежать функции и процедуры:

CREATE ROLE s\_mc WITH

LOGIN

NOSUPERUSER

INHERIT

NOCREATEDB

NOCREATEROLE

NOREPLICATION;

ALTER ROLE s\_mc SET search\_path TO s\_mt, public, pg\_catalog;

Создать контейнер БД:

CREATE ROLE s\_mp WITH

NOLOGIN

NOSUPERUSER

INHERIT

NOCREATEDB

NOCREATEROLE

NOREPLICATION;

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;

Установить дамп из архива.

## Развёртывание шлюза

Json – шлюз представляют собой отдельные приложение (файлы-архивы с именами ungate\_\*.zip). Создаем файлы конфигурации (напиример opt/work\_gate/configs в linux или c:\work\_gate\configs в Windows) t\_providers.toml, t\_context.toml, t\_plugins.toml, t\_events.toml, t\_servers.toml, logger.json. Ниже приводятся настройки, которые указываются в этом файле.

### Установка шлюза

Распаковываем архив ungate\_\*.zip например в /opt/work\_gate/ungate

Заходим в папку и выполняем команду

sudo npm install -g yarn

yarn install

### Настройка списка серверов t\_servers.toml

ck\_id — наименование ноды

[[data]]

ck\_id = "core.example.com"

cv\_ip = "192.168.1.1"

cv\_ip — ip/dns ноды

### Настройка подключения контекста данных t\_context.toml к БД

*ck\_id — уникальное наименование*

[[data]]

ck\_id = "core"

cv\_path = "/api"

cv\_description = "Основные настройки метаинформации"

ck\_d\_plugin = "CorePGContext"

[data.cct\_params]

debug = true

disableCache = true

poolMax = 100

connectString = "postgres://логин:пароль@127.0.0.1:5432/наименование базы"

*cv\_path — путь доступа*

*cv\_description — Описание*

*ck\_d\_plugin — наименование плагина*

*[data.cct\_params] — настройки плагина*

*debug — включаем отладачную информацию в ответ*

*disableCache — признак отключения кэширования страниц*

*poolMax — максимальный пулл*

*connectString — строка подключения «postgres://логин:пароль*[*@127.0.0.1*](mailto:s_mc@127.0.0.1)*:5432/наименование базы»*

### Настройка подключения **провайдеров данных** t\_providers.toml к БД

Настройки провайдеров

[[data]]

ck\_id = "admingate"

cv\_description = "Провайдер администрирования"

ck\_d\_plugin = "admingate"

cct\_params = { }

[[data]]

ck\_id = "auth"

cv\_description = "Провайдер авторизации"

cl\_autoload = true

ck\_d\_plugin = "AuthMock"

[data.cct\_params]

adminUser = "admin\_core"

adminPassword = "123456"

viewUser = "view\_core"

[[data]]

ck\_id = "meta"

cv\_description = "Провайдер работы с метаинформацией"

cl\_autoload = true

ck\_d\_plugin = "PostgreSQLDb"

[data.cct\_params]

core = true

poolMax = 100

connectString = "postgres://логин:пароль@127.0.0.1:5432/наименование базы"

*ck\_id — уникальное наименование*

*cl\_autoload — загрузка при старте шлюза*

*cv\_description — Описание*

*ck\_d\_plugin — наименование плагина*

***adminUser - логин учетки администратора***

***adminPassword — пароль администратора***

### Настройка подключения плагинов t\_plugins.toml

[[data]]

cv\_name = "preparequery"

cv\_description = "Плагин преобразования запросов 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 = "Плагин распаковывания вложеного 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

### Настройка системы оповещени t\_events.toml

[[data]]

cv\_description = "Оповещение meta"

ck\_d\_plugin = "CorePgNotification"

ck\_id = "meta"

[data.cct\_params]

authProvider = "auth"

connectString = "postgres://логин:пароль@127.0.0.1:5432/наименование базы"

[[data]]

cv\_description = "Семафор meta"

ck\_d\_plugin = "CorePgSemaphore"

ck\_id = "semaphore"

[data.cct\_params]

connectString = "postgres://логин:пароль@127.0.0.1:5432/наименование базы"

### 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"]

}

}

}

### Запуск сервера windows

Зайдем в папку с распакованным сервером и выполним команды:

LOGGER\_CONF — ссылка на настройки логера

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

PROPERTY\_DIR — указываем папку где лежат настрой t\_providers.toml, t\_context.toml, t\_plugins.toml

Запуск

sc start gate-core

### Запуск сервера linux systemctl

Создаем скрипт запуска /opt/work\_gate/start.sh

где /opt/work\_gate/ungate — место где распакован шлюз

#!/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

sudo chmod +x /opt/work\_gate/start.sh

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

PROPERTY\_DIR — указываем папку где лежат настрой t\_providers.toml, t\_context.toml, t\_plugins.toml

[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

LOGGER\_CONF — ссылка на настройки логера

WorkingDirectory — папка распакованого шлюза

Запуск

systemctl start gate-core

## 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.

### Подключение к json-шлюзам

Значение параметра: проксирование запросов к json-шлюзам

Пример:

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*](http://127.0.0.1:8080/) *адрес машины с запущеным шлюзом*

### Подключение к файлам сборки conf.d/core.conf

Имя параметра: server

Значение параметра: 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; # путь к папке с распакованым приложением

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;

}

}

## Проверка функционирования Системы

1. Перейдите по ссылке http://$CDN\_HOST/

2. Ответ сервера должен быть следующего вида:

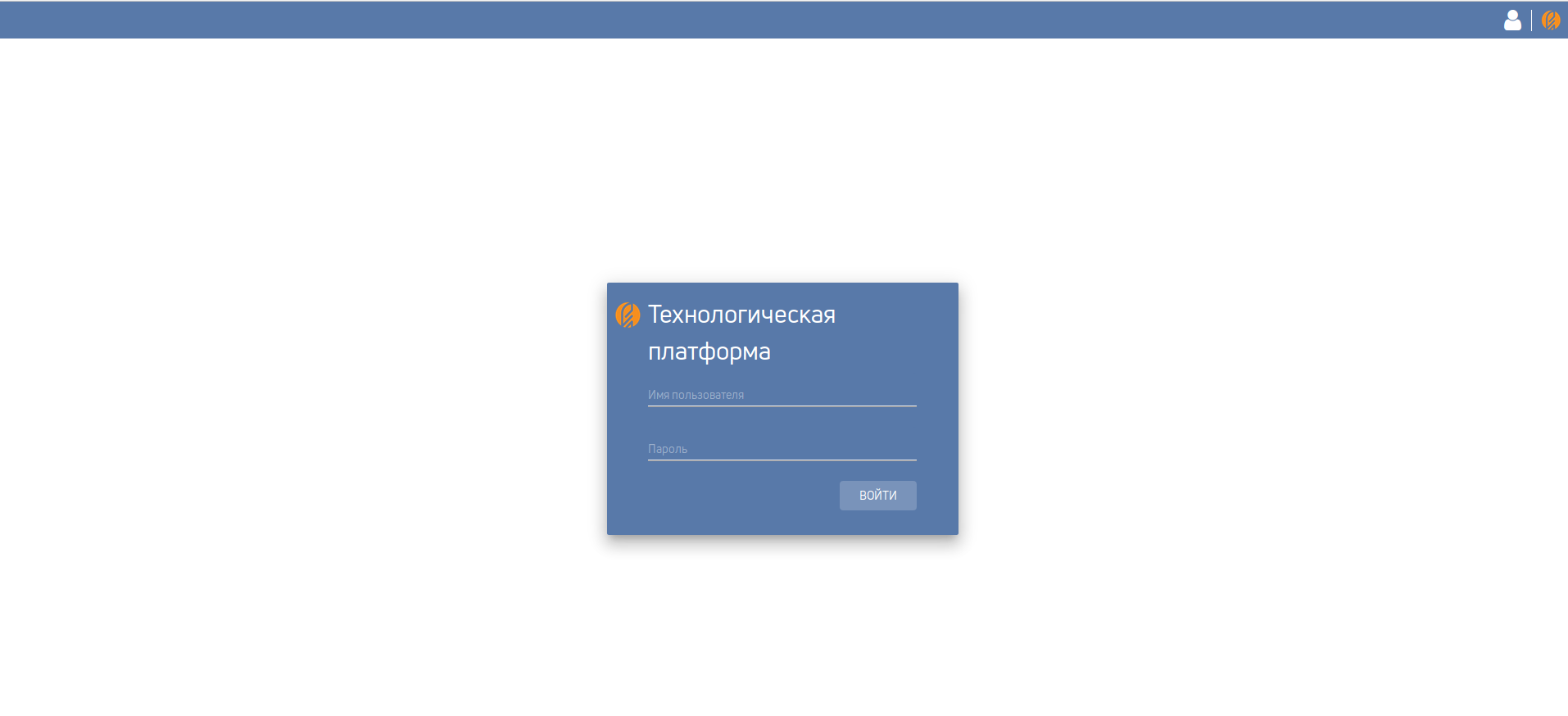


Рис. 1 Авторизация в системе CORE

3. В окне авторизации введите логин: Администратор и пароль (Логин и пароль мы указывали ранее в пункте 5.3.3). Если Система работоспособна – то отобразится окно с сообщением об успешной **авторизации**, после чего появится главная страница, с возможностью выбора модуля:



**Рис.** 1 **Главная страница CORE**

## Работа в режиме кластера

### Настройка шлюза

Добавляем дополнительные сервера в t\_servers.toml

Перезагружаем шлюзы

[[data]]

ck\_id = "core1.example.com"

cv\_ip = "192.168.1.1"

[[data]]

ck\_id = "core2.example.com"

cv\_ip = "192.168.1.2"

### 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; # путь к папке с распакованым приложением

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;

}

}