**ЦИФРОВАЯ ПЛАТФОРМА РАСПРЕДЕЛЕННОГО РЕЕСТРА ФНС РОССИИ**

**(ЦПРР ФНС России)**

**Подсистема администрирования**

**Техническое описание**

Руководство администратора

# ИСТОРИЯ ИЗМЕНЕНИЯ

|  |  |  |
| --- | --- | --- |
| **Дата** | **Описание** | **Автор** |
| 28.03.2022 | Cформирована первая версия руководства | Waves |
| 23.08.2022 | Произведены изменения:   * удалено описание развертывания посредством Ansible; * добавлено описание развертывания на docker swarm | Waves |
| 13.12.2022 | Множественные терминологические правки | Феликс Делюда |

# СОДЕРЖАНИЕ

[ИСТОРИЯ ИЗМЕНЕНИЯ 2](#_Toc122009405)

[СОДЕРЖАНИЕ 3](#_Toc122009406)

[О РУКОВОДСТВЕ АДМИНИСТРАТОРА 5](#_Toc122009407)

[1 ОБЩИЕ СВЕДЕНИЯ 6](#_Toc122009408)

[2 ОПИСАНИЕ СЕТЕВОЙ СТРУКТУРЫ ПЛАТФОРМЫ 7](#_Toc122009409)

[3 ТРЕБОВАНИЯ К ВИРТУАЛЬНЫМ МАШИНАМ 12](#_Toc122009410)

[3.1 Общие требования и допущения, принятые при формировании требований к виртуальным машинам 12](#_Toc122009411)

[3.2 Требования к виртуальным машинам 12](#_Toc122009412)

[3.3 Требования к лицензионному обеспечению 15](#_Toc122009413)

[4 НАСТРОЙКА СЕРВЕРОВ POSTGRYSQL ДЛЯ РАЗВЕРТЫВАНИЯ БЛОКЧЕЙН-НОД И СЕРВИСОВ ПЛАТФОРМЫ 16](#_Toc122009414)

[4.1 Установка СУБД «PostgrySQL» 16](#_Toc122009415)

[4.2 Установка настроек для PostgreSQL-серверов 16](#_Toc122009416)

[4.2.1 Настройка мастер-сервера для Сервисов 16](#_Toc122009417)

[4.2.2 Настройка slave-сервера для Сервисов 20](#_Toc122009418)

[4.2.3 Настройка серверов блокчейн-нод 21](#_Toc122009419)

[4.3 Создание баз данных и пользователей для серверов блокчейн-нод 22](#_Toc122009420)

[5 УСТАНОВКА И НАСТРОЙКА ПЛАТФОРМЫ 24](#_Toc122009421)

[5.1 Развертывание узла на docker swarm 24](#_Toc122009422)

[5.2 Установка посредством пакетного менеджера «Helm» 27](#_Toc122009423)

[5.2.1 Подготовка к установке 27](#_Toc122009424)

[5.2.2 Выполнение установки 29](#_Toc122009425)

[6 ОБНОВЛЕНИЕ ВЕРСИИ ОБРАЗОВ ПЛАТФОРМЫ 30](#_Toc122009426)

[6.1 Общие сведения о процедуре обновления 30](#_Toc122009427)

[6.2 Алгоритм обновления системы, развернутой с использованием пакета «install\_pack» 30](#_Toc122009428)

[6.2.1 Обновление версии образов Сервиса 30](#_Toc122009429)

[6.2.2 Обновление образов UI 31](#_Toc122009430)

[6.3 Алгоритм обновления системы, развернутой с использованием Kubernetes 31](#_Toc122009431)

[7 ОТВЕТЫ НА ЧАСТО ЗАДАВАЕМЫЕ ВОПРОСЫ (F.A.Q.) 34](#_Toc122009432)

# О РУКОВОДСТВЕ АДМИНИСТРАТОРА

В настоящем руководстве приводится техническое описание Цифровой платформы распределенного реестра ФНС России (далее – Платформа) в следующих аспектах:

* сетевая структура;
* системные требования;
* установка и настройка Платформы и информационных сервисов Платформы (далее – Сервис), реализованных на ее базе;
* обновление Платформы и Сервисов, реализованных на ее базе;

Руководство адресовано специалистам, которые осуществляют развертывание и администрирование Платформы в рамках исполнения своих рабочих функций.

В настоящем руководстве применяются следующие способы обозначения сведений обязательного или рекомендательного характера (таблица 1):

Таблица 1 – Способы обозначения сведений

|  |  |
| --- | --- |
| **Указатель** | **Пояснение** |
| **ВНИМАНИЕ!** | Важные сведения, которые необходимо в обязательном порядке учитывать при работе с Платформой |
| **ПРИМЕЧАНИЕ** | Дополнительные сведения о работе Платформы, имеющие рекомендательный или поясняющий характер |

# ОБЩИЕ СВЕДЕНИЯ

Цифровая платформа распределенного реестра ФНС России представляет собой единую платформу, реализованную с применением технологии распределенного реестра (блокчейн-сети) и предназначенную для обеспечения и поддержания работы бизнес-приложений, функционирующих на базе этой платформы (далее – Сервисы).

Платформа позволяет реализовывать следующие основные функции:

* отправка и получение данных из блокчейн-сети;
* управление учетными записями пользователей Сервисов, реализованных на базе Платформы;
* мониторинг работы блокчейн-сети Платформы и Сервисов, реализованных на базе Платформы.

# ОПИСАНИЕ СЕТЕВОЙ СТРУКТУРЫ ПЛАТФОРМЫ

Предусмотрены следующие схемы развертывания сетевой структуры Платформы:

* схема стандартной сетевой структуры, изображенная на рисунке 1;
* схема минимальной сетевой структуры, изображенная на рисунке 2.

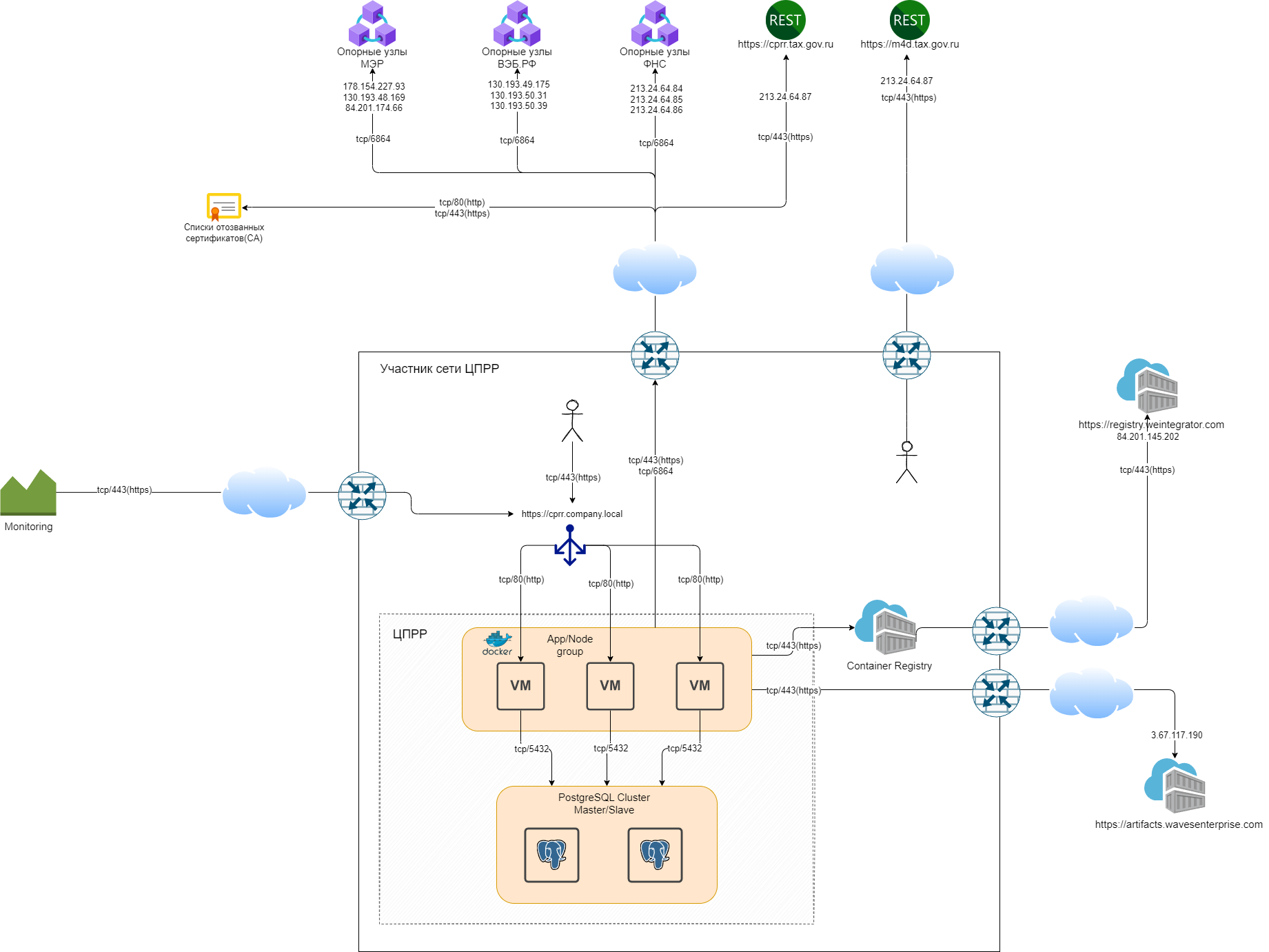
Аннотация к схемам сетевой структуры приведена в таблицах 3 и 4.

При формировании схемы сетевой структуры (рисунки 1 и 2) принимаются следующие допущения:

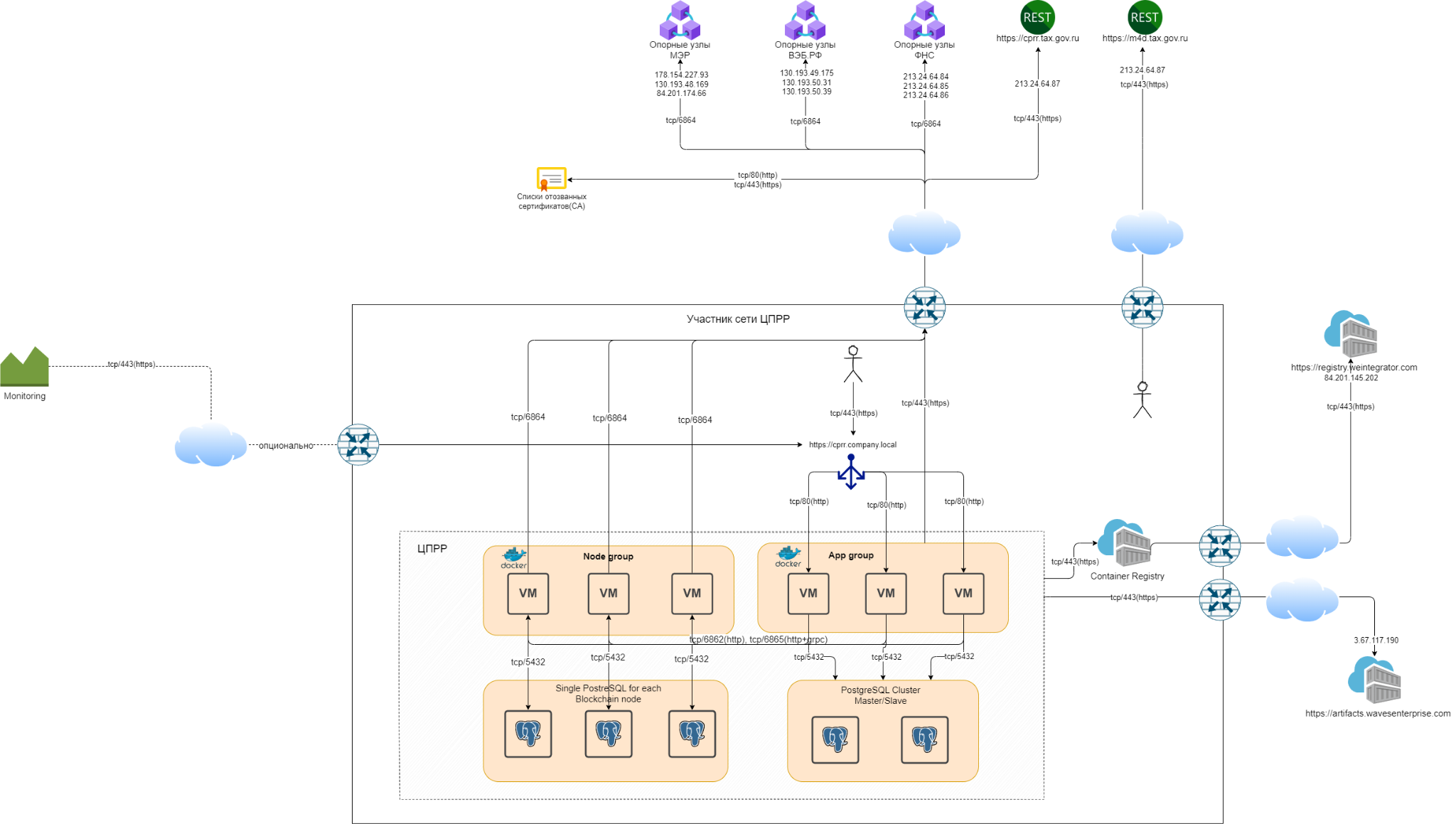
1. Внешние доменные имена, выделенные под Платформу, находятся в домене четвертого уровня:

|  |
| --- |
| *\*.credit-registry.local.<orgnization>.ru* |

1. Сервера, выделенные на схеме пунктиром (рисунки 1 и 2), конфигурируются лицами, ответственными за эксплуатацию сервиса в организациях, предоставляющих соответствующий сервис.
2. Развертывание прокси-сервера *nginx-tcp-proxy-0* выполняется опционально.



**Рисунок** **1 – Схема минимальной сетевой структуры Платформы**



**Рисунок** **2 – Схема стандартной сетевой структуры Платформы**

**Описание docker-контейнеров, представленных на схеме:**

*app-group:*

backend, frontend приложений и nginx. По одному контейнеру для каждого Сервиса:  
zpreg-support - backend приложения “Программы поддержки бизнеса”

zpreg-subsidy - backend приложения “279”

zpreg-subsidy-second - backend приложения “279-2”

zpreg-priv - backend приложения 696

zpreg-support-ui - frontend приложения “Программы поддержки бизнеса”

zpreg-subsidy-ui - frontend приложения “279”

zpreg-subsidy-second-ui - frontend приложения “279-2”

zpreg-priv-ui - frontend приложения 696

nginx - nginx

Набор сервисов, устанавливаемых на каждом узле:  
vst-files - сервис передачи файлов

vst-oauth2 - сервис авторизации

vst-identity - сервис данных о компании и пользователях узла

*node-group:*

Контейнеры блокчейн нод требуются для корректного функционирования Сервисов и обмена данными с опорными узлами сети.

**Базы данных, представленные на схеме:**  
  
БД Сервиса 422

app\_db\_pg\_name: app\_db

БД Сервиса 696 (если не планируется разворачивать данный Сервис, создавать БД не требуется)

priv\_app\_db\_pg\_name: priv\_app\_db

БД Сервиса 279 (если не планируется разворачивать данный Сервис, создавать БД не требуется)

subsidy\_app\_db\_pg\_name: subsidy\_app\_db

БД Сервиса 279-2 (если не планируется разворачивать данный Сервис, создавать БД не требуется)

subsidy\_second\_app\_db\_pg\_name: subsidy\_second\_app\_db

Обязательная БД

# ------------------- vst files ------------------- #

vst\_files\_pg\_name: vst\_files

# ------------------- vst identity ------------------- #

vst\_identity\_pg\_name: vst\_identity

# ТРЕБОВАНИЯ К ВИРТУАЛЬНЫМ МАШИНАМ

## Общие требования и допущения, принятые при формировании требований к виртуальным машинам

Все виртуальные машины (ВМ) на этапе установки системы (10 рабочих дней с начала работ по установке системы) должны иметь доступ в сеть интернет по портам 80, 443 для доступа к публичным репозиториям ПО с открытым исходным кодом:

* github.com;
* репозитории Ubuntu LTS соответствующей версии;
* download.docker.com;
* artifacts.wavesenterprise.com.

## Требования к виртуальным машинам

Требования, предъявляемые к ВМ, изложены в таблице 4. В таблице приведены наименования ВМ в соответствии со схемой сетевой структуры (рисунки 1 и 2).

Таблица 4 – Требования к виртуальным машинам

| **Наименование на сетевой схеме** | **Доменное имя** | **Сертификат SSL** | **ОС** | **SSD(bronze+)** | **vCPU** | **RAM** | **Примечание** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| node-0 | node-0.credit-registry.<bank>.ru | Да | Ubuntu LTS 20.04, RHEL 8.4+/CentOS 7 | 50Gb ОС +  100Gb Data Volume | 4 | 8Gb | Использует Docker.  Docker имеет доступ к диску (монтирует volume) |
| node-1 | [node-1.credit-registry.<bank>.ru](http://node-0.credit-registry.local.psbank.ru/) |
| node-2 | [node-2.credit-registry.<bank>.ru](http://node-0.credit-registry.local.psbank.ru/) |
| app-0  app-0  app-0 | – | Нет | Ubuntu LTS 20.04, RHEL 8.4+/ CentOS 7 | 50Gb ОС | 4 | 8Gb | Использует Docker |
| db-node-0 (single)  db-node-1 (single)  db-node-2 (single) | – | – | Ubuntu LTS 20.04, RHEL 8.4+/ CentOS 7 | 50G OC +  100Gb Data Volume | 2 | 8Gb | – |
| db-app-0  db-app-1  (master/slave) | – | – | Ubuntu LTS 20.04, RHEL 8.4+/CentOS 7 | 50G OC +  100Gb Data Volume | 4 | 16Gb | – |
| docker-registry | registry.credit-registry.<bank>.ru | Да | Ubuntu LTS 20.04, RHEL 8.4+/CentOS 7 | 50G OC | 2 | 8Gb | Использует Docker |

## Требования к лицензионному обеспечению

Для обеспечения работоспособности Платформы требуются следующие лицензии:

* Блокчейн-платформа – 3 шт. (мастер-токен предоставляется ФНС России);
* КРИПТО-ПРО CSP 5.0 (server) – 3 шт.;
* КРИПТО-ПРО Java CSP (server) – 3 шт.

В случае использования варианта с минимальной конфигурацией с двумя блокчейн нодами, требуется две лицензии вместо трёх.

# НАСТРОЙКА СЕРВЕРОВ POSTGRYSQL ДЛЯ РАЗВЕРТЫВАНИЯ БЛОКЧЕЙН-НОД И СЕРВИСОВ ПЛАТФОРМЫ

Процесс настройки PostgreSQL-серверов для развертывания блокчейн-нод и Сервисов Платформы предполагает последовательное выполнение следующих процедур:

* + - 1. Установка СУБД «PostgreSQL» на виртуальные машины.
      2. Установка настроек для PostgreSQL-серверов.
      3. Создание баз данных и пользователей для PostgreSQL-серверов блокчейн-нод.

## Установка СУБД «PostgrySQL»

Установите СУБД «PostgreSQL» на виртуальные машины:

* мастер-сервер PostgreSQL;
* слейв-сервер PostgreSQL;
* cервера PostgreSQL для баз данных блокчейн-нод.

Ниже приведен перечень пакетов для установки:

|  |
| --- |
| **Пакеты для установки** |
| "postgresql{{ postgresql\_version\_dotless }}-server"  "postgresql{{ postgresql\_version\_dotless }}-contrib"  "postgresql{{ postgresql\_version\_dotless }}-libs"  "postgresql{{ postgresql\_version\_dotless }}"  "pg\_repack{{ postgresql\_version\_dotless }}" |

## Установка настроек для PostgreSQL-серверов

### Настройка мастер-сервера для Сервисов

Для настройки мастер-сервера для Сервисов последовательно выполните следующие действия:

1. Установите настройки в файле *postgresql.conf* мастер-сервера для Сервисов согласно следующему примеру:

|  |
| --- |
| **Пример** |
| listen\_addresses = '\*'  max\_connections = 100  shared\_buffers = 4GB  work\_mem = 10485kB  maintenance\_work\_mem = 1GB  effective\_io\_concurrency = 200  max\_worker\_processes = 4  max\_parallel\_maintenance\_workers = 2  max\_parallel\_workers\_per\_gather = 2  max\_parallel\_workers = 4  wal\_buffers = 16MB  max\_wal\_size = 4GB  min\_wal\_size = 1GB  checkpoint\_completion\_target = 0.9  random\_page\_cost = 1.1  effective\_cache\_size = 12GB  default\_statistics\_target = 100  max\_wal\_senders = 3  wal\_keep\_size = 64  wal\_level = replica |

1. Создайте пользователей и базы данных к ним на мастер-сервере БД для Сервисов согласно следующему примеру:

|  |
| --- |
| **Пример** |
| Для баз данных достаточно создать одного пользователя. Имя пользователя указывается в конфигурационных файлах при развертывании Сервиса. Имя пользователя может быть любым.  БД Сервиса 422  # ------------------- zpreg app db ------------------- #  app\_db\_pg\_name: app\_db  app\_db\_pg\_user: app\_db  app\_db\_pg\_pass: ohsiel1yae1Yopha  БД Сервиса 696 (если не планируется разворачивать данный Сервис, создавать БД не требуется)  # ------------------- priv app db ------------------- #  priv\_app\_db\_pg\_name: priv\_app\_db  priv\_app\_db\_pg\_user: priv\_app\_db  priv\_app\_db\_pg\_pass: eyahpitee0Ahh3ar  БД Сервиса 279 (если не планируется разворачивать данный Сервис, создавать БД не требуется)  # ------------------- subsidy app db ------------------- #  subsidy\_app\_db\_pg\_name: subsidy\_app\_db  subsidy\_app\_db\_pg\_user: subsidy\_app\_db  subsidy\_app\_db\_pg\_pass: ophuTohTuphoo5ah  БД Сервиса 279-2 (если не планируется разворачивать данный Сервис, создавать БД не требуется)  # ------------------- subsidy second app db ------------------- #  subsidy\_second\_app\_db\_pg\_name: subsidy\_second\_app\_db  subsidy\_second\_app\_db\_pg\_user: subsidy\_second\_app\_db  subsidy\_second\_app\_db\_pg\_pass: eyahpitee0Ahh3ar  Обязательная БД  # ------------------- vst files ------------------- #  vst\_files\_pg\_name: vst\_files  vst\_files\_pg\_user: vst\_files  vst\_files\_pg\_pass: shon6thoh5eiCai7  # ------------------- vst identity ------------------- #  vst\_identity\_pg\_name: vst\_identity  vst\_identity\_pg\_user: vst\_identity  vst\_identity\_pg\_pass: eiwieloeLaeL0Mah  # ------------------- vst identity ------------------- #  vst\_oauth2\_pg\_name: vst\_oauth2  vst\_oauth2\_pg\_user: vst\_oauth2  vst\_oauth2\_pg\_pass: eipooka3rei8Aivu |

1. Предоставьте пользователям привилегии ALL для соответствующих БД в соответствии со следующим примером:

|  |
| --- |
| **Пример** |
| app\_db  priv\_app\_db  subsidy\_app\_db  subsidy\_second\_app\_db  tpf\_app\_db  we\_dictionary\_app\_db  poar\_app\_db  vst\_files  vst\_identity  vst\_oauth2 |

1. Выполните команду перезапуска сервера БД:

|  |
| --- |
| **Command** |
| systemctl restart {{ имя\_процесса\_postgresql }} |

### Настройка slave-сервера для Сервисов

Для настройки slave-сервера для Сервисов последовательно выполните следующие действия:

1. Создайте replication-пользователя.
2. Установите настройки ролей для replication-пользователя: REPLICATION,LOGIN.
3. Внесите изменения в файл *pg\_hba.conf* в соответствии со следующим примером:

|  |
| --- |
| **Пример** |
| local all postgres peer  local all all peer  host replication {{ postgres\_replication user }} {{ slave\_IP }}/32 md5  host all all '127.0.0.1/32' md5"  host all all '::1/128' md5 |

1. Если политика безопасности требует открыть доступ к БД в рамках внутренней сети, внесите соответствующие изменения.
2. Добавьте данные доступа в файл */root/.pgpass*:

|  |
| --- |
| **Пример** |
| {{ адрес\_master\_сервера }}:5432:\*:{{ postgres\_replication\_user }}:{{ postgres\_replication\_pass }} |

Права на файл *.pgpass: "0600"*.

1. Запустите копирование данных с мастер-сервера:

|  |
| --- |
| **Comand** |
| command: pg\_basebackup -h {{ адрес\_master\_сервера }} -D /var/lib/postgresql/13/main/ -P -U {{ postgres\_replication\_user }} --wal-method=stream |

1. Измените настройки */etc/postgresql/13/main/postgresql.conf: hot\_standby = on*.
2. В файле */etc/postgresql/13/main/recovery.conf* добавьте блок:

|  |
| --- |
| **Пример** |
| standby\_mode = 'on'  primary\_conninfo = 'host={{ адрес\_master\_сервера }} port=5432 user={{ postgres\_replication\_user }} password={{ postgres\_replication\_pass }} sslmode=disable'  trigger\_file = '/etc/postgresql/13/main/trigger\_file' |

1. После внесения изменений выполните команду:

|  |
| --- |
| **Comand** |
| systemctl restart {{ имя\_процесса\_postgresql }} |

### Настройка серверов блокчейн-нод

Для настройки серверов блокчейн-ноды последовательно выполните следующие действия:

1. Установите настройки в файле *postgresql.conf* для серверов блокчейн-нод в соответствии со следующим примером:

|  |
| --- |
| **Пример** |
| listen\_addresses = '\*'  max\_connections = 100  shared\_buffers = 2GB  work\_mem = 10485kB  maintenance\_work\_mem = 512MB  effective\_io\_concurrency = 200  max\_worker\_processes = 2  max\_parallel\_maintenance\_workers = 1  max\_parallel\_workers\_per\_gather = 1  max\_parallel\_workers = 2  wal\_buffers = 16MB  max\_wal\_size = 4GB  min\_wal\_size = 1GB  checkpoint\_completion\_target = 0.9  random\_page\_cost = 1.1  effective\_cache\_size = 6GB |

1. Перезапустите сервера баз данных для блокчейн-нод.

## Создание баз данных и пользователей для серверов блокчейн-нод

Для создания баз данных и пользователей для PostgrySQL-серверов блокчейн-ноды последовательно выполните следующие действия:

1. Создайте пользователей и базы данных для серверов блокчейн-нод в в соответствии со следующим примером:

|  |
| --- |
| **Пример** |
| # ------------------- privacy node ------------------- #  privacy\_node\_0\_pg\_name: privacy\_node\_0  privacy\_node\_0\_pg\_user: privacy\_node\_0  privacy\_node\_0\_pg\_pass: Ao9iw7aeth6ohnoB  # ------------------- privacy node ------------------- #  privacy\_node\_1\_pg\_name: privacy\_node\_1  privacy\_node\_1\_pg\_user: privacy\_node\_1  privacy\_node\_1\_pg\_pass: uoghaeh0theiT2Wu  # ------------------- privacy node ------------------- #  privacy\_node\_2\_pg\_name: privacy\_node\_2  privacy\_node\_2\_pg\_user: privacy\_node\_2  privacy\_node\_2\_pg\_pass: Ia3xai7Ohgahroh0 |

1. Выдайте пользователям привилегии ALL для соответствующих баз данных:

|  |
| --- |
| **Пример** |
| privacy\_node\_0  privacy\_node\_1  privacy\_node\_2 |

# УСТАНОВКА И НАСТРОЙКА ПЛАТФОРМЫ

## Развертывание узла на docker swarm

- системные требования

3 vm os linux, 4 cpu/8 RAM, hdd/ssd 100гб

- для корректной работы docker swarm нужно открыть порты:

TCP port 2377 for cluster management communications

TCP and UDP port 7946 for communication among nodes

UDP port 4789 for overlay network traffic

- установить

docker/docker-compose, wget

- На каждую ноду перенесите содержимое переданного архива в

/opt/fns/app/

в результате получим:

*$ ls /opt/fns/app/*

*configs docker-compose.yml env*

в каталоге app будут каталоги configs и env, а также файл docker-compose.yml

- создайте каталоги на новых нодах (по три экземпляра на каждой ноде)

/opt/node-0-data/data

/opt/node-1-data/data

/opt/node-2-data/data

- загрузите стейт на этих нодах

wget https://artifacts.wavesenterprise.com/repository/fileshare/node/data-20220125.tar.gz

tar xf data-20220125.tar.gz

mv data/\* /opt/node-0-data/data/

mv data/\* /opt/node-1-data/data/

mv data/\* /opt/node-2-data/data/

- создайте каталог /opt/licenses на нодах и переместите в каждый из каталогов соответствующий файл лицензии из имеющихся (передаются в составе конфигурационного архива по пути .../configs/node)

/opt/licenses/node-0{1,2}.license

- создайте базы pgsql и пропишите доступы в соответствующих env файлах

privacy\_node\_0 - base\_pack/configs/node/node-0.conf

privacy\_node\_1 - base\_pack/configs/node/node-1.conf

privacy\_node\_2 - base\_pack/configs/node/node-2.conf

vst\_files - vst-files.env

vst\_identity - vst-identity.env

vst\_oauth2 - vst-oauth2.env

Ниже приведен список баз основных сервисов, необходимых для работы Платформы. Необходимо отдельно развернуть бд для Сервисов Платформы, которые вы будете использовать. Имена баз можно уточнить у технической поддержки.

- выполните команду на ноде 0 и получите id ноды

docker swarm init

- на остальных нодах выполните

docker swarm join --token SWMTKN-1-3mdg96j2m264y1whjij2bir4pnyucm2wglk927g7xzlaqs9yrp-atim4squpyvcc7wz2pybu1c0x 10.130.0.31:2377

- получить id нод

docker node ls

- выполните на первой ноде заменив id

docker node update <yzwkq4avguhlh0yghr5aflskb> --label-add name=node-0

docker node update <yzwkq4avguhlh0yghr5aflskb> --label-add name=node-1

docker node update <yzwkq4avguhlh0yghr5aflskb> --label-add name=node-2

- войдите в docker-registry на всех нодах

docker login registry.weintegrator.com

(логин и пароль будут переданы вместе с конфигурационным архивом)

- запустите

docker stack deploy --compose-file=docker-compose.yml --with-registry-auth cprr

- полезные команды

docker service ls - просмотр состояния сервисов

docker stack ps --no-trunc cprr - подробное состояние кластера

docker stack rm cprr - остановка кластера

docker service scale cprr\_nginx=0 - остановить сервис

docker service scale cprr\_nginx=1 - запустить сервис

docker service logs cprr\_nginx - просмотр логов

## Установка посредством пакетного менеджера «Helm»

### Подготовка к установке

Для подготовки к установке системы необходимо последовательно выполнить следующие действия:

1. Подготовка СУБД. Для подготовки СУБД необходимо выполнить следующие действия:

* подготовить СУБД, пользователей и базы;
* в файле *environment/kaluga-astral/node/node-config.yaml* скорректировать *connection\_string* к БД;
* в файле *values-kaluga-astral.yaml* скорректировать *connection\_string* к БД.

|  |
| --- |
| **ПРИМЕЧАНИЕ**  Здесь и далее в качестве примера приведены наименования файлов *values-kaluga-astral.yaml* и *environment/kaluga-astral/node/node-config.yaml*, используемых АО «Калуга Астрал». |

1. Установка Dockerhost. Существуют два варианта установки:

* по инструкции на отдельный хост;
* в куб чартом.

Для установки в куб чартом требуется *\*privileged: true\**. Для минимизации рисков безопасности под Dockerhost создается отдельная нод-группа, к которой через *taints* прибивается *pod*.

Пример установки Dockerhost чартом:

|  |
| --- |
| **Command** |
| ```bash  helm repo add --username=username --password=password integrator-helm https://artifacts.wavesenterprise.com/repository/integrator-helm/  helm upgrade -i dockerhost -n dockerhost -f values-dockerhost.yaml integrator-helm/dockerhost --create-namespace  ``` |

В отдельном контейнере находится *fluentbit*. При необходимости в качестве эндпоинтов можно указать *loki*, либо *elasticsearch* для возможности анализа логов смарт-контрактов.

В случае установки по инструкции на отдельный хост:

* поменять тип сервиса блокчейн-нод на *LoadBalancer* в файле *values-kaluga-astral.yaml* для возможности подключения извне к grpc-эндпоинтам нод блокчена;
* заменить значение для *grpc-server.host* в файле *environment/kaluga-astral/node/node-config.yaml* на соответствующие адреса, полученные для сервисов.

1. Установка Docker-registry. Существует три варианта установки:

* использовать существующие с проксированием;
* развернуть по переданной инструкции на отдельном хосте;
* развернуть в кубе <https://github.com/goharbor/harbor-helm>.

В файле *environment/kaluga-astral/node/node-config.yaml* заменить *default-registry-domain* на домен вашего docker-registry.

В файле *values-kaluga-astral.yaml* заменить все *\*repository:\** на соответствующие пути в вашем *registry*.

1. Настройка Ingress. В качестве ingress-контроллера используется nginx-ingress *<https://github.com/kubernetes/ingress-nginx/tree/main/ charts/ingress-nginx>*.

В файле *values-kaluga-astral.yaml* необходимо заменить *cprr.kaluga-astral.ru* на домен, который будете использовать под этот проект.

TLS-сертификат генерируется вручную и подкладывается в виде секрета, либо генерируется автоматически через *acme* при условии установленного в куб cert-manager *<https://cert-manager.io/docs/installation>*

### Выполнение установки

Для установки выполните:

|  |
| --- |
| **Command** |
| ```bash  cd <каталог с чартом>  helm upgrade -i cprr -n cprr -f values-kaluga-astral.yaml \  --set-file node.node.wallet=environment/kaluga-astral/node/wallet.tar.gz \  --set-file node.node.config=environment/kaluga-astral/node/node-config.yaml \  --set-file node.node.license=environment/kaluga-astral/node/node-license.yaml \  --set-file vst-oauth2.userCompanies=environment/kaluga-astral/user-companies.json \  --set-file vst-oauth2.oauthClients=environment/kaluga-astral/oauth-clients.yml \  --create-namespace ./  ``` |

Имя релиза *cprr* менять не рекомендуется.

# ОБНОВЛЕНИЕ ВЕРСИИ ОБРАЗОВ ПЛАТФОРМЫ

## Общие сведения о процедуре обновления

Обновление версии приложения выполняется путем замены действующей версии образа на публикуемую обновленную.

Алгоритм обновления образов зависит от способа развертывания Платформы:

* с использованием пакета установки *install\_pack* (посредствам системы управления конфигурациями «Ansible»);
* с использованием системы оркестрации контейнеров «Kubernetes» (посредством пакетного менеджера «Helm»).

Далее приводится описание алгоритмов обновления для каждого способа развертывания Платформы.

## Алгоритм обновления системы, развернутой с использованием пакета «install\_pack»

Если развертывание выполнено с использованием инсталляционного пакета *install\_pack*, применяются описанные ниже алгоритмы обновления версии образов приложений и UI.

### Обновление версии образов Сервиса

Для обновления версии образов Сервиса последовательно выполните следующие действия:

1. Перейдите в каталог *install\_pack/roles/app/vars* и найдите файл *main.yml*.
2. Замените версии образов у соответствующих Сервисов:

|  |
| --- |
| **Пример** |
| zpreg\_subsidy\_webapp\_app\_image: fns/zpreg/zpreg-subsidy-webapp-app:**1.8.0**  *------------------------------------------------------------------------------------------------*  *В приведенном примере* ***1.8.0*** *– версия приложения, которую требуется заменить* |

1. После этого вернитесь в каталог *install\_pack* и выполните команду развертывания Сервисов.

### Обновление образов UI

Для обновления версий образов UI последовательно выполните следующие действия:

1. Перейдите в каталог *install\_pack/roles/nginx/vars* и найдите файл *main.yml*.
2. Откройте файл с помощью доступного редактора (vi, nano) и замените версию у соответствующего образа:

|  |
| --- |
| **Пример** |
| zpreg\_ui\_image: fns/zpreg/zpreg-ui:**3ad6ae42**  *---------------------------------------------------------------------------------------------*  ***3ad6ae42*** *- та самая версия Сервиса, которую нам нужно заменить* |

1. Вернитесь в каталог *install\_pack* и запустите команду развертывания веб-приложения:

|  |
| --- |
| **Пример** |
| ansible-playbook playbooks/nginx.yml |

## Алгоритм обновления системы, развернутой с использованием Kubernetes

Если развертывание выполнено с использованием системы оркестрации контейнеров «Kubernetes», применяется следующий алгоритм обновления:

1. Перейдите на машину, у которой есть доступ к инструменту командной строки Kubernetes – *kubectl*.
2. Откройте список деплойментов в неймспейсе, который использовался для развертывания приложений и ноды:

|  |
| --- |
| **Пример** |
| kubectl get deployments -n **zpreg**  *--------------------------------------------------------------------------------------------------*  *В приведенном примере* ***zpreg*** *– имя неймспейса, который использовался для развёртывания* |

1. Далее с помощью команды *edit* откройте соответствующий деплоймент и отредактируйте версию образа:

|  |
| --- |
| **Пример** |
| kubectl edit deployment **vst-files** -n zpreg  *----------------------------------------------------------------------------------------*  *В приведенном примере* ***vst-files*** *– имя редактируемого деплоймента* |

1. Образ расположен в пути [*spec:/template:/spec:/containers:/image*](http://spec/template:/spec:/containers:/image):

|  |
| --- |
| **Пример** |
| image: docker-registry-1.local.ru/we/vst-files:**1.12.0**  *-----------------------------------------------------------------------------------------*  ***1.12.0*** *– самая версия приложения, которую необходимо заменить* |

1. После внесения изменений сохраните деплоймент и выйдите из него. Выполните проверку статуса подов с помощью команды *get pods*:

|  |
| --- |
| **Пример** |
| kubectl get pods -n zpreg |

1. Для обновления смарт-контрактов отредактируйте *value* в *zpreg-app.contract-hash* и *zpreg-app.contract-image* по пути [*spec:/template:/spec:/containers:/env*](http://spec/template:/spec:/containers:/env):

|  |
| --- |
| **Пример** |
| - name: zpreg-app.contract-hash    value: 0339dcc80adab4913bda510ae6f61e01f515d6188d87eda11ee0b6a4dc19ffb3  - name: zpreg-app.contract-image    value: sc/zpreg-contract-app:2.20.0-v2.20.0 |

# ОТВЕТЫ НА ЧАСТО ЗАДАВАЕМЫЕ ВОПРОСЫ (F.A.Q.)

|  |
| --- |
| 1. **Какие данные подлежат резервированию?** |
| Обязательно: резервные копии БД группы app.  Желательно: резервные копии БД группы node. |
| 1. **Каким образом можно обеспечить резервирование и отказоустойчивость для узла исполнения смарт-контрактов docker-host?** |
| Резервировать docker-host нет необходимости, поскольку аналогичные хосты есть у других участников сети. Если ваш узел не сможет осуществлять майнинг, это могут делать узлы других участников. |
| 1. **Какова оценка возможных объемов баз данных и их роста для типовых случаев, стандартных нагрузок?** |
| Объем БД - 50Гб/год. |
| 1. **Какие требования предъявляются к сети передачи данных?** |
| 100Mb канал вовне и внутри. |
| 1. **Имеется ли возможность совместить TCP и HTTP прокси-сервер на одном виртуальном сервере?** |
| Технических ограничений нет, настройка производится собственными специалистами. |
| 1. **Какова конфигурация двух серверов СУБД app node?** |
| Конфигурация Master/Replica, ручной феловер. можно самостоятельно сделать автоматический феловер в соответствии с принятыми внутри компании практиками. |
| 1. **Возможно ли использование собственного внутреннего репозитория образов?** |
| Совместимость возможна, если репозиторий построен на основе реестра «Harbor». |
| 1. **Какую версию лицензий КРИПТО-ПРО CSP следует использовать?** |
| КРИПТО-ПРО CSP 5.0 (server). Версия должна быть 5.0. |