АО «Концерн ГРАНИТ»
АКЦИОНЕРНОЕ ОБЩЕСТВО

Система распределенного хранения данных

«КВАНТ-РЕЕСТР»

Инструкция по установке программного обеспечения

2021

АННОТАЦИЯ

Настоящий документ содержит сведения о системе распределенного хранения данных «КВАНТ-РЕЕСТР» (далее –комплексной информационной системе «Квант-реестр», КИС «Квант-реестр», Системе), включающие в себя данные о назначении, функциях, структуре и условиях выполнения, а также сведения по установке, настройке, запуску, завершению и резервному восстановлению Системы.

СОДЕРЖАНИЕ

1. Общие сведения

1.1.Назначение

Система предназначена для децентрализованного и защищенного файлового хранилища, что обеспечивает оперативный контроль и управление средствами защиты информации от несанкционированного доступа.

1.2.Функции

Система обеспечивает выполнение следующих функций:

- формирование приватного элемента для пользователя с гарантированными вероятностными свойствами, т.е. пользователь должен иметь приватный идентификатор или ключ, никому не известный кроме него, выработанный при помощи датчика случайных чисел с гарантированными статистическими свойствами, или взаимодействие с функцией подписи аппаратного средства электронной подписи, соответствующего требованиям 63-ФЗ «Об электронной подписи»;
- формирование сетевого имени (идентификатора, которым пользователь представляется в системе) на основе указанного выше приватного элемента, исключающего возможность выявления связей между сетевым именем и множеством открытых данных о физическом лице или организации, или взаимодействие с функцией предоставления открытого ключа аппаратного средства электронной подписи, соответствующего требованиям 63-ФЗ «Об электронной подписи»;
- безопасное хранение приватного элемента у пользователя для обеспечения защищенности от несанкционированного доступа к нему или хранение приватного элемента на стороне аппаратного средства электронной подписи, соответствующего требованиям 63-ФЗ «Об электронной подписи».;
- наличие «точки входа» для пользователя (оператора) распределенного реестра;

- авторизация пользователя для оператора при помощи электронной подписи, использующей приватный элемент пользователя;
- безопасный транспорт (как минимум с сохранение неизменности информации, получаемой от пользователя) для передачи информации от пользователя к оператору распределенного реестра;
- контроль целостности и авторства каждой информационной единицы, помещаемой в распределенный реестр, с помощью электронной подписи пользователя;
- формирование подтверждений у оператора распределенного реестра факте помещения информации в распределенный реестр (например, путем выдачи заверенных оператором квитанций пользователям);
- наличие механизма формирования и обработки запросов по выдаче информации из распределенного реестра по запросам его участников (клиентов), обеспечивающего защищенность данного запроса (также авторизацию и контроль неизменности запроса)
- хранение файлов в экземплярах СУБД;
- хранение метаинформации о файлах в логе транзакций блокчейна;
- поиск файлов по уникальному идентификатору;
- отправка метаинформации о файлах по запросу;
- контроль доступа пользователей к хранимым данным;
- организация необходимой структуры базы данных;
- получение информации об установленной версии QHB;
- проверка состояния соединения с сервером базы данных;
- возможность выполнения метакоманд, а таже различных функций для автоматизации широкого спектра задач;
- создание и удаление экземпляра базы данных;
- создание и удаление новой учетной записи;
- запись данных в базу, обеспечение записи данных, вводимых пользователем в БД через интерактивный терминал qsql;
- управление хранением данных;

- чтение данных (выполнение запросов пользователя на получение интересующих данных);
- редактирование существующих записей;
- удаление записей;
- реализация поддержки языка описания данных и языка запросов;
- обеспечение восстановления БД после сбоя;
- создание резервной копии кластера QHB;
- непрерывное архивирование и восстановление на момент времени журнала упреждающей записи (WAL);
- кластеризация БД;
- переиндексация БД;
- организация синхронного и асинхронного взаимодействия клиентских приложений с БД;
- индексирование, позволяющее оптимизировать производительность базы данных;
- параллелизм;
- функция больших объектов, обеспечивающая потоковый доступ к пользовательским данным;
- ввод запросов в интерактивном режиме, из файла, конвейера ранее запущенной программы, из аргументов командной строки на пользовательской консоли;
- очистка БД и генерация внутренней статистики.

1.3. Условия выполнения

Техническое обеспечение должно учитывать имеющиеся стандартные технические решения и оборудования Заказчика. Ниже, в таблице (Таблица 1) представлены требования к Системе и программному обеспечению.

Таблица 1 - Требования к Системе и программному обеспечению

№ п/п	Техническое средство/программное обеспечение	Характеристики	Примечание
1	Процессор	Процессоры архитектур: x86-64 ARM Эльбрус	
2	Операционная система	Семейство Linux на всех вышеуказанных архитектурах	
3	Оперативная память	Не менее 4 ГБ оперативной памяти	
4	Жесткий диск	Не менее 200 МБ (не учитывая размер базы данных)	При выборе дискового пространства для базы данных необходимо ориентироваться на конкретную задачу

2. Установка, настройка и запуск на Centos 7

2.1.Установка

Для запуска на Ubuntu 18.04 см. [USAGE-ubuntu.md](./USAGE-ubuntu.md).

1) Необходимо распаковать предоставленный архив в произвольную директорию на диске:

```
tar -xf qlad-centos-1.0.0-rc.1.tar.gz
```

Все перечисленные ниже шаги выполняются от пользователя с root-правами в директории `qlad-centos-1.0.0-rc.1`.

2) Установить репозиторий EPEL:

```
yum -y install epel-release
```

3) Установить libsodium:

```
yum install -y libsodium
```

4) Установить Python3 и pip:

```
yum install -y python3 python3-pip
```

5) Установить Protobuf:

```
yum install -y wget emacs-filesystem zlib-devel
wget
https://cbs.centos.org/kojifiles/packages/protobuf/3.6.1/4.el7/x86_64/protobuf-3.6.1-4.el7.x86_64.rpm
```

wget https://cbs.centos.org/kojifiles/packages/protobuf/3.6.1/4.el7/x86_64/protobuf-compiler-3.6.1-4.el7.x86_64.rpm

wget https://cbs.centos.org/kojifiles/packages/protobuf/3.6.1/4.el7/x86_64/protobuf-devel-3.6.1-4.el7.x86_64.rpm

rpm -i protobuf-3.6.1-4.el7.x86_64.rpm

rpm -i protobuf-compiler-3.6.1-4.el7.x86_64.rpm

rpm -i protobuf-devel-3.6.1-4.el7.x86_64.rpm

```

6) Установить Exonum Launcher:

```
pip3 install exonum-launcher --upgrade --no-binary=protobuf
```

7) Установить Qlad Launcher plugin:

```
pip3 install -e launcher_instance_plugin
```

8) Установить Qlad:

```
rpm -i qlad*.rpm
```

9) Установить QHB, QCP и Metricsd:

```
rpm --import \
    https://repo.granit-concern.ru/qhb/keys/RPM-GPG-KEY-qhb && \
yum-config-manager --add-repo \
    https://repo.granit-concern.ru/qhb/std-1/centos/7/x86_64/qhb.repo
yum install -y qcp metricsd qhb-core qhb-contrib
```
```

```
    mkdir -p /opt/qhb-data \
      && chown qhb:qhb /opt/qhb-data \
      && su - qhb -c "/usr/local/qhb/bin/initdb -D /opt/qhb-data --encoding=UTF8" \
      && echo "metrics_collector_id = 100500" >> /opt/qhb-data/qhb.conf \
      && echo "shared_preload_libraries = 'librbytea'" >> /opt/qhb-data/qhb.conf
```

10) Установить NodeJS:

```
    curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.35.3/install.sh | bash
    export NVM_DIR="$HOME/.nvm"
    [ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"
    nvm install node
```

2.2.Запуск

1) Запустить QHB, QCP и metricsd

```
    cp ./metricsd-config.yaml /etc/metricsd/config.yaml
    cp ./qcp-config.yaml /etc/qcp/config.yaml
    su - qhb -c "/usr/local/qhb/bin/qhb_ctl -D /opt/qhb-data -l /opt/qhb-data/logfile start"
    systemctl start metricsd
    systemctl start qcp
```

2) Инициализировать базу

```
    cp *.sql /opt/qhb-data
    chown qhb:qhb /opt/qhb-data/init-files.sql /opt/qhb-data/init-logs.sql
    su - qhb -c "/usr/local/qhb/bin/psql -h localhost -p 8080 -f /opt/qhb-data/init-files.sql"
```

su - qhb -c "/usr/local/qhb/bin/psql -h localhost -p 8080 -f /opt/qhb-data/init-logs.sql"

```

3) Сгенерировать конфигурацию блокчейн-сети из 1 узла

```

qlad generate-template config/template.toml --validators-count 1

sed -i 's/propose_timeout_threshold = 500/propose_timeout_threshold = 1/' config/template.toml

sed -i 's/max_propose_timeout = 200/max_propose_timeout = 600000/' config/template.toml

sed -i 's/min_propose_timeout = 10/min_propose_timeout = 200/' config/template.toml

sed -i 's/first_round_timeout = 3000/first_round_timeout = 1200000/' config/template.toml

qlad generate-config config/template.toml config/1 --peer-address 127.0.0.1:7001 -n

qlad finalize config/1/sec.toml config/1/node.toml --public-api-address 127.0.0.1:8201 --private-api-address 127.0.0.1:8301 --public-configs config/1/pub.toml

```

**Замечание**: если указанные адреса `--public-api-address` и/или `--private-api-address`

отличаются от приведенных в примере выше, следует также обновить содержимое файла `qlad.yaml`

в корне распакованного архива - оно должно содержать корретные адреса узла в секции `network`.

4) Запустить узел блокчейна

```

qlad run --node-config config/1/node.toml --db-path db/1 --master-key-pass pass --host "127.0.0.1" --port 8080 &

```

5) Инициализировать блокчейн, используя Launcher:

```
python3 -m exonum_launcher -i ./qlad.yaml
```

**Замечание**. В конфигурационном файле, предоставленном по умолчанию, публичный и приватный адреса узла указаны как `http://localhost:8201` и `http://localhost:8301` соответственно.

Обновить содержимое `qlad.yaml`, если хотите использовать другие адреса.

По умолчанию, блокчейн инициализируется с одним админ-пользователем с приватным ключом

`89f28ef7604f738d86067a1f01314fd2bc23be2383b509c6d8ca861da0b6de0ce3e2ba391445701dcbf1a96bf63bec18efc118b4b0c8f1fa3e38516e6c20e200`. Ключ можно изменить, отредактировав файл `qlad.yaml`.

6) Запустить фронтенд

Фронтенду передаются два параметра:

- `port`: внешний порт для пользователей фронтенда. После запуска, фронтенд будет доступен
  по адресу `http://localhost:<port>` в браузере.
- `api-root`: публичный адрес узла блокчейна. Должен соответствовать адресу
  `--public-api-address`, переданному команде `qlad run` ранее.

Оба этих адреса должны быть доступны для внешних пользователей, обновите настройки фаервола, если необходимо.

```
cd frontend
node server.js --port 2867 --api-root=http://localhost:8201
```

После этого фронтенд доступен по адресу `http://localhost:2867`..
```

3. Установка, настройка и запуск под Ubuntu 18.04

3.1. Установка

Данный раздел описывает отличия от [процесса установки на Centos 7](./USAGE.md), которые проявляются при установке на Ubuntu 18.04.

1) Установка QHB

QHB можно установить из официальной поставки DEB-пакетов:

```
apt install gnupg2 apt-transport-https wget
wget -qO - https://repo.granit-concern.ru/qhb/keys/RPM-GPG-KEY-qhb | sudo apt-key add -
echo 'deb https://repo.granit-concern.ru/qhb/std-1/debian stretch main' >> /etc/apt/sources.list
apt update
apt install qhb-core qhb-contrib qcp metricsd
```

Для установки qhb-core также требуется пакет `libicu-dev`, предоставляющий libicu версии 57. Данный пакет отсутствует в стандартных репозиториях Ubuntu и должен быть установлен отдельно.

3.2.Запуск

1) Запуск QCP и metricsd

Systemd-юниты QCP и metricsd требуют редактирования: необходимо заменить использование `/bin/sh` на `/usr/bin/sh`. Также требуется вручную создать пользователей и добавить их в группу qhb.

2) Исполняемый файл узла

В отличие от Centos 7, для Ubuntu 18.04 предоставляется не установочный rpm-пакет, а просто исполняемый файл `qlad`. Его можно разместить в директории, доступной из PATH, или запускать из локальной директории.

Для работы исполняемый файл требует установленной библиотеки `libsodium-dev`.

3) Запуск сервиса

Если при попытке запуска сервиса через `exonum-launcher` происходит ошибка

```
Invalid proto descriptor for file "service.proto":
  service.proto: A file with this name is already in the pool.
```

Нужно выполнить следующую команду и повторить попытку запуска.

```
export PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION='python'
```

Дополнительная информация:
https://github.com/protocolbuffers/protobuf/issues/3002.

4. Резервное восстановление

В Систему интегрирован механизм автоматического восстановления в случае сбоя, поэтому участие администратора безопасности или инженеров службы эксплуатации не требуется. Данный механизм восстанавливает работу Системы в случае его непредвиденного закрытия или в случае сбоя.

Перечень принятых сокращений

Сокращение, обозначение	**Расшифровка**
CD/DVD	Compact Disc (компакт диск) / Digital Versatile Disc (цифровой многоцелевой диск)
CPU	Центральный процессор (с англ. «Central processing unit»)
Fiber Channel	Семейство протоколов для высокоскоростной передачи
HTTP	Протокол прикладного уровня передачи данных (с англ. «HyperText Transfer Protocol»)
IP	Маршрутизируемый протокол сетевого уровня стека TCP/IP (с англ. «Internet protocol» – межсетевой протокол)
REST	Метод взаимодействия компонентов распределённого приложения в сети Интернет, при котором вызов удаленной процедуры представляет собой обычный HTTP-запрос, а необходимые данные передаются в качестве параметров запроса (с англ. «Representational state transfer»)
АБИ	Администратор безопасности информации
АПМДЗ	Аппаратно-программный модуль доверенной загрузки
АРМ	Автоматизированное рабочее место
БД	База данных
ГОСТ	Государственный стандарт
ЕПП	Единое пространство пользователей
КСПД	Корпоративная сеть передачи данных
ЛВС	Локальная вычислительная сеть
НСД	Несанкционированный доступ

Сокращение, обозначение	**Расшифровка**
ОС	Операционная система
ОЭ	Опытная эксплуатация
ПО	Программное обеспечение
ПС	Программные средства
ПЭВМ	Персональная электронная вычислительная машина
РД	Руководящий документ
СЗИ	Средство защиты информации
СПО	Специальное программное обеспечение
СУБД	Система управления базами данных
ТЗ	Техническое задание
ТП	Технологический проект
УЗНИ	Учёт защищаемых носителей информации
УСЗИ	Модуль управления средствами защиты информации
ЭВМ	Электронная вычислительная машина

Термины и определения

Термин	Определение
Блокчейн	Выстроенная по определённым правилам непрерывная последовательная цепочка блоков, содержащих информацию. Связь между блоками обеспечивается не только нумерацией, но и тем, что каждый блок содержит свою собственную хеш-сумму и хеш-сумму предыдущего блока. Для изменения информации в блоке придётся редактировать и все последующие блоки. Чаще всего копии цепочек блоков хранятся на множестве разных компьютеров независимо друг от друга. Это делает крайне затруднительным внесение изменений в информацию, уже включённую в блоки. Блокчейн децентрализовано хранится на узлах распределенной компьютерной сети.
Консенсус	Механизм, используемый в распределенных системах и блокчейнах, предназначенный для достижения согласованного состояния между несколькими независимыми агентами или процессами.
Конфигурация сети	Набор параметров, определяющих поведение сети. Он включает параметры алгоритма консенсуса, например, время принятия блоков, список узлов сети, список пользователей и их прав.
Узел	Устройство, хранящее полную копию истории транзакций блокчейна и соединенное с другими узлами сети. Узлы доступны для пользователей через их приватные и публичные API.
Доверенная временная метка	Процесс надёжного отслеживания времени создания и изменения документа. Надёжность подразумевает, что никто (включая владельца) не сможет внести изменение во

Термин	Определение
	временную метку после её создания при условии, что целостность метки не будет нарушена.