АО «ИЦР»

**СКАН.**

**Инструкция по установке экземпляра ПО**

2024 год

Стандартным вариантом использования программного продукта СКАН является его развертывание в ЦОД в качестве «облачного» SaaS-сервиса. Пользователи получают доступ к Системе СКАН через Интернет с помощью Интернет-браузера.

Данная поставка предназначена исключительно в демонстрационно-ознакомительных целях. Поставка включает:

1. WebApi.tar - Веб-приложение, упакованное в Docker контейнер
2. WebApp.tar - Приложение API, упакованное в Docker контейнер

Общая последовательность действий для развертывания приложения:

* Подготовить исполнительные сервера согласно п.1
* Настроить сетевое взаимодействие для получения внешних данных по защищенным транспортным каналам HTTPS
* Разместить исполняемые файлы приложения и конфигурационные файлы в рабочем каталоге (архив с дистрибутивами прилагается)
* Развернуть базу данных согласно п. 2
* Обеспечить запуск и согласованное выполнение компонентов приложения согласно п.3.

Ссылка: https://share.interfax.ru/s/Ek7mKdaw7AZSnpg

Название файла: Interfax\_SCAN\_Disoto.7z

1. **Технические требования**

Требования к серверам приложений:

* 8-ми ядерный 64-х разрядный процессор;
* от 64 ГБ оперативной памяти;
* жесткий диск SSD от 500 ГБ;
* операционная система Ubuntu;
* фреймворк .NET 6.0+;
* Docker 24.0.6

Требования к серверу баз данных:

* 32 ядра 64-х разрядного процессора;
* от 512 ГБ оперативной памяти;
* жёсткий диск SSD + NVMe 20 ТБ;
* операционная система РЕД ОС 8;
* СУБД Postgres Pro Standard 16, архитектура x86-64

1. **Развертывание базы данных**

2.1. Настройка прав и доступов

На хостах БД необходима возможность подключения по 22 порту (ssh) и sudo права для выполнения настроек.

Для работы PostgreSQL требуется чтобы на хостах были открыты доступы по портам 5432 (postgresql) и 6432 (пулер соединений pgbouncer, используется для подключения приложений к СУБД).

2.2. Настройка системы, установка пакетов, настройка PostgresPro Standard

Настройки выполняем из под пользователя root (с правами sudo), если не указано иное.

Меняем таймзону на Москву:

*$ timedatectl set-timezone Europe/Moscow*

Отключаем firewalld (если установлен):

*$ systemctl stop firewalld*

*$ systemctl disable firewalld*

Отключаем Selinux (если установлен и включен):

*$ vim /etc/selinux/config*

Меняем значение параметра SELINUX на: SELINUX=disabled

Перезагружаем систему:

*$ reboot*

Ставим некоторые утилиты, обновляем пакеты ОС:

*$ dnf install -y htop net-tools telnet wget*

*$ dnf upgrade*

*$ reboot*

Настраиваем репозиторий PostgresPro:

*$ wget http://repo.postgrespro.ru/std/std-16/keys/pgpro-repo-add.sh*

*$ sh pgpro-repo-add.sh*

*$ rm pgpro-repo-add.sh*

Устанавливаем PostgresPro Standard 16:

*$ dnf install -y postgrespro-std-16 postgrespro-std-16-client postgrespro-std-16-contrib postgrespro-std-16-libs postgrespro-std-16-server postgrespro-std-16-plpython3 postgrespro-std-16-jit*

Ставим дополнительные пакеты:

*$ dnf install -y pg-probackup-std-16 pgbouncer*

Для пользователей root и postgres добавляем в PATH путь к бинарным файлам postgrespro - /opt/pgpro/std-16/bin:

Для root:

*$ vim ~root/.bashrc*

Добавляем в конце файла добавление в переменную PATH :/opt/pgpro/std-16/bin

*----------*

*PATH=$PATH:/opt/pgpro/std-16/bin*

*export PATH*

*----------*

У пользователя postgres выполняем тоже самое, но если нет файла ~postgres/.bash\_profile, то создаем его, если есть то просто редактируем:

*$ vim ~postgres/.bash\_profile*

*----------*

*# .bash\_profile*

*# Get the aliases and functions*

*if [ -f ~/.bashrc ]; then*

*. ~/.bashrc*

*fi*

*# User specific environment and startup programs*

*PATH=$PATH:$HOME/bin:/opt/pgpro/std-16/bin*

*export PATH*

*----------*

*$ chown postgres:postgres ~postgres/.bash\_profile*

*$ chmod 600 ~postgres/.bash\_profile*

После установки пакетов PostgresPro Standart, автоматически была инициализирована директория PGDATA, был создан и запущен сервис:

PGDATA - /var/lib/pgpro/std-16/data/

Сервис - /usr/lib/systemd/system/postgrespro-std-16.service

Останавливаем сервис:

*$ systemctl stop postgrespro-std-16*

Монтируем наш диск для PGDATA /database. Владельцем делаем root:root и выдаем права 755. (Если эти настройки на хосте еще не проведены).

*$ ls -lha /*

*drwx------ 2 root root 4.0K Sep 16 15:50 database*

Создаем в /database необходимые директории и выдаем им следующего владельца и права:

*$ mkdir -p /database/std-16/data*

*$ chown -R postgres:postgres /database/\**

*$ chmod -R 700 /database/\**

Создаем директорию для логов:

*$ mkdir -p /var/log/postgresql*

*$ chown postgres:postgres /var/log/postgresql*

*$ chmod 700 /var/log/postgresql*

Переносим данные из старой PGDATA /var/lib/pgpro/std-16/data/ в новую PGDATA /pgdata/std-16/data/ :

*$ mv /var/lib/pgpro/std-16/data/\* /database/std-16/data/*

Проверяем что из исходной директории все перенеслось в целевую.

Удаляем директории в старом каталоге PGDATA:

*$ rm -rf /var/lib/pgpro/std-16*

Проверяем, у нас должно не быть директории std-16 в /var/lib/pgpro :

*$ ls -lha /var/lib/pgpro*

Создаем (под postgres пользователем) симлинк std-16 на директорию /database/std-16 в директории /var/lib/pgpro:

*$ cd /var/lib/pgpro*

*$ sudo -u postgres ln -s /database/std-16 std-16*

Запускаем сервис postgresql чтобы убедиться что после переносов в новые директории, postgresql работает:

*$ systemctl start postgrespro-std-16*

Задаем пароль для пользователя postgres (суперпользователь СУБД) и пользователя replicator (пользователь для репликации данных) в СУБД:

*$ sudo -iu postgres*

*$ psql*

*$ SET password\_encryption = 'md5';*

*$ alter user postgres with password 'ваш\_пароль\_postgres';*

*$ \q*

*$ exit*

Настраиваем sysctl.conf:

Для начала вычисляем пару значений для страниц в памяти - SBPERCENT это процент от памяти сервера на shared\_buffers, hugepage это расчитанное количество HugePages, minfree это указание ядру сколько свободной памяти в системе стараться держать:

*$ SBPERCENT=40*

*$ ramkb=$(awk '/MemTotal/ {print $2}' /proc/meminfo)*

*$ hugepage=$((ramkb/1024\*(${SBPERCENT}+3)/100/2))*

*$ minfree=$((ramkb\*10/100))*

*$ if [ $minfree -ge 102400 ] ; then*

*$ minfree=102400*

*$ fi*

*$ echo $hugepage*

*$ echo $minfree*

Редактируем файл /etc/sysctl.conf

*$ vim /etc/sysctl.conf*

В конец файла прописываем ($hugepage и $minfree в тексте ниже заменняем на полученные в ходе расчетов выше значения):

*----------*

*#### postgres tune ####*

*# минимальное использование SWAP*

*vm.swappiness=1*

*# число больших страниц (2Мб) зависит от объема ОЗУ, ставим примерно 40%. если размер кэша данных 32 Гб, это требует 16384 страницы, но включить нужно с запасом*

*vm.nr\_hugepages=$hugepage*

*# расчитываем по методу mamonsu-tune*

*# максимальный объем системной памяти, которую можно заполнить dirty pages - грязными страницами;*

*# 64 Мб*

*vm.dirty\_bytes = 67108864*

*# объём системной памяти, который можно заполнить dirty pages до того, как фоновые процессы pdflush/flush/kdmflush запишут их на диск;*

*# 16 Мб*

*vm.dirty\_background\_bytes = 16777216*

*# Данный параметр говорит ядру стараться держать часть памяти свободной (до 10%), а чтобы удовлетворить это требование, ему приходится запускать дефрагментацию раньше.*

*vm.min\_free\_kbytes = $minfree*

*# время, в течение которого шедуллер считает процесс достаточно горячим, чтобы не мигрировать его на другой CPU*

*kernel.sched\_migration\_cost\_ns = 5000000*

*# ядро не пытается выделить память путем ее высвобождения, а выделяет там где есть в том числе и на другой NUMA-ноде.*

*vm.zone\_reclaim\_mode = 0*

*# отключить работу NUMA для CPU*

*kernel.numa\_balancing=0*

*# кол-во открытых файлов (1М)*

*fs.file-max = 1000000*

*# сколько файлов может отслеживать один пользователь*

*fs.inotify.max\_user\_watches = 100000*

*# размер памяти, который будет гарантированно не занят в системе, чтобы у нас могли работать SSH и пр.*

*vm.admin\_reserve\_kbytes=100000*

*----------*

Сразу пробуем выделить HugePages в системе и проверяем результат:

*$ sysctl -w vm.nr\_hugepages=$hugepage*

*$ grep Huge /proc/meminfo*

Отключаем прозрачные большие страницы:

*$ vim /lib/systemd/system/disable-transparent-huge-pages.service*

*----------*

*# It's recommended to disable transparent huge pages for postgres service*

*# https://www.stephenrlang.com/2018/01/disabling-transparent-huge-pages-in-linux/*

*# cat /sys/kernel/mm/transparent\_hugepage/enabled*

*# always madvise [never]*

*# cat /sys/kernel/mm/transparent\_hugepage/defrag*

*# always madvise [never]*

*#*

*# However once the system reboots, it will go back to its default value again.*

*# To make the setting persistent on CentOS 7 and Ubuntu 16.04, you can disable THP on system startup by making a systemd unit file:*

*[Unit]*

*Description=Disable Transparent Huge Pages*

*DefaultDependencies=no*

*After=sysinit.target local-fs.target*

*Before=basic.target*

*[Service]*

*Type=oneshot*

*ExecStart=/bin/bash -c 'echo never > /sys/kernel/mm/transparent\_hugepage/enabled'*

*ExecStart=/bin/bash -c 'echo never > /sys/kernel/mm/transparent\_hugepage/defrag'*

*[Install]*

*WantedBy=basic.target*

*----------*

$ chmod 644 /lib/systemd/system/disable-transparent-huge-pages.service

$ systemctl daemon-reload

$ systemctl enable disable-transparent-huge-pages.service

$ systemctl start disable-transparent-huge-pages.service

Перезагруажем систему:

$ reboot

Конфигурационные файлы postgresql.conf pg\_hba.conf, ident.conf:

Переносим эти файлы на хост, например в директорию /tmp/pg\_conf и копируем с заменой подготовленные конфигурационные файлы для PostgreSQL - postgresql.conf и pg\_hba.conf в PGDATA и настраиваем права:

*$ mkdir /tmp/pg\_conf*

*$ chmod 777 /tmp/pg\_conf*

Далее заменяем ими текущие файлы конфигурации PostgreSQL:

*$ cp /tmp/pg\_conf/postgresql.conf /database/std-16/data/postgresql.conf*

*$ chown postgres:postgres /database/std-16/data/postgresql.conf*

*$ chmod 600 /database/std-16/data/postgresql.conf*

*$ cp /tmp/pg\_conf/pg\_hba.conf /database/std-16/data/pg\_hba.conf*

*$ chown postgres:postgres /database/std-16/data/pg\_hba.conf*

*$ chmod 600 /database/std-16/data/pg\_hba.conf*

*$ cp /tmp/pg\_conf/pg\_ident.conf /database/std-16/data/pg\_ident.conf*

*$ chown postgres:postgres /database/std-16/data/pg\_ident.conf*

*$ chmod 600 /database/std-16/data/pg\_ident.conf*

Создаем файл триггер для архивации wal:

*$ touch /databse/std-16/data/archive\_active.trigger*

*$ chown postgres:postgres /database/std-16/data/archive\_active.trigger*

*$ chmod 644 /database/std-16/data/archive\_active.trigger*

Перезапускаем PostgreSQL:

*$ systemctl restart postgrespro-std-16*

Добавляем сервис postgrespro-std-16 в автозагрузку:

*$ systemctl enable postgrespro-std-16*

Создаем директорию под скрипты обслуживания PostgreSQL:

*$ mkdir /var/lib/pgsql/scripts*

*$ chown postgres:postgres /var/lib/pgsql/scripts*

*$ chmod 700 /var/lib/pgsql/scripts*

Переносим на хост скрипты для системного cron, приложенные к инструкции. Допустим положим их в директорию /tmp/pg\_scripts:

*$ mkdir /tmp/pg\_scripts*

*$ chmod 777 /tmp/pg\_scripts*

Переносим эти скрипты в /var/lib/pgsql/scripts:

*$ mv /tmp/pg\_scripts/vacuum.sh /var/lib/pgsql/scripts*

*$ mv /tmp/pg\_scripts/vacuum\_freeze.sh /var/lib/pgsql/scripts*

*$ chown postgres:postgres /var/lib/pgsql/scripts/\*.sh*

*$ chmod 750 /var/lib/pgsql/scripts/\*.sh*

Добавляем вызов этих скриптов в cron пользователя postgres:

*$ sudo -u postgres crontab -e*

*----------*

*# Vacuum*

*0 0 8-31 \* \*/7 /var/lib/pgsql/scripts/vacuum.sh*

*0 0 1-7 1,6 \*/7 /var/lib/pgsql/scripts/vacuum\_freeze.sh*

*----------*

1. **Настройка ролей и глобальных объектов, загрузка дампа БД**

*Настройки выполняем из под пользователя root (с правами sudo), если не указано иное.*

*Создаем директорию /tmp/db\_scripts и помещаем в нее скрипты для БД, приложенные к инструкции:*

*$ mkdir /tmp/db\_scripts*

*$ chmod 777 /tmp/db\_scripts*

*Настраиваем права на эти скрипты для пользователя postgres:*

*$ chown -R postgres:postgres /tmp/db\_scripts*

*$ chmod 640 /tmp/db\_scripts/\*.sql*

*Все команд, приведенные ниже в этом пункте выполняем в ос под пользователем postgres:*

*$ sudo -iu postgres*

*В скрипте scan4site\_global\_superuser.sql нужно предварительно задать пароль в строке:*

*$ vim /tmp/db\_scripts/scan4site\_global\_superuser.sql*

*----------*

*ALTER ROLE scan4site WITH NOSUPERUSER INHERIT NOCREATEROLE NOCREATEDB LOGIN NOREPLICATION NOBYPASSRLS PASSWORD '<пароль>';*

*----------*

*Выполняем скрипт scan4site\_global\_superuser.sql:*

*$ psql -U postgres -d postgres -f /tmp/db\_scripts/scan4site\_global\_superuser.sql*

*Создаем базу данных scan4site, выполняя скрипт scan4site\_schema\_superuser.sql:*

*$ psql -U postgres -d postgres -f /tmp/db\_scripts/scan4site\_schema\_superuser.sql*

*Создаем в бд scan4site необходимые объекты из под пользователя scan4site, подключившись к бд scan4site, выполняя скрипт scan4site\_schema.sql (может потребоваться ввести пароль пользователя scan4site, заданный в первом скрипте):*

*$ psql -U scan4site -h localhost -d scan4site -f /tmp/db\_scripts/scan4site\_schema.sql*

1. **Развертывание приложения**

Для запуска на стенде должен быть установлен докер и докер-композ

Необходимо выполнить следующие действия:

* Скопировать файлы docker-compose.yml, nginx.conf, webapi.tar, webapp.tar в рабочую директорию и перейти в нее
* Загрузить образ webapi командой docker load ./webapi.tar
* Загрузить образ webapp командой docker load ./webapp.tar
* Запустить приложения командой docker-compose up -d
* Убедиться что приложение доступно перейдя в браузере по адресу http://localhost

1. **Требования к рабочему месту пользователя**

Для корректной работы программного продукта СКАН компьютер пользователя должен отвечать следующим техническим характеристикам:

* Доступ в интернет по защищенным транспортным каналам HTTPS/SSL
* Совместимая версия браузера:
  + Microsoft Edge версии 44 и выше;
  + Google Chrome 69 и выше;
  + Яндекс.Браузер 18.9 и выше;
  + Mozilla Firefox 62 и выше;
  + Safari 12 и выше;
  + Opera 56 и выше

Также для работы с программным продуктом СКАН в настройках браузера должно быть разрешено:

* выполнять JavaScript;
* принимать cookies