# Руководство по настройке кластера PostgreSQL на основе repmgr на CentOS 8

Для организации отказоустойчивого кластера PostgreSQL используется синхронная репликация master-standby, настроенная при помощи утилиты repmgr, и HAProxy. HAProxy используется для скрытия от клиента IP mastera, который изменится после автоматического failover. Для примера рассмотрим настройку кластера из 3-х нод (рисунки 1 – 4):



Рисунок 1 – Штатная работа кластера



Рисунок 2 – Выход из строя (не доступность) master



Рисунок 3 – Автоматический failover и реструктуризация кластера



Рисунок 4 – Штатная работа кластера с новым master

Чтение и запись производится клиентом через HAProxy только в master[[1]](#footnote-1). Для определения, является ли нода master, HAProxy по порту 5445 вызывает http-сервисы, развернутые на всех нодах кластера. Если нода является master, сервис вернет http-статус 200, иначе 503.

## 1. Репликация на основе repmgr12

<https://www.youtube.com/watch?v=LvDHdgQqaQw>

Установим PostgreSQL (локально из пакетов или из репозитория в интернет, см. «Руководство по установке TimescaleDB под CentOS 8») и repmgr12 на всех нодах:

# sudo dnf install -y https://download.postgresql.org/pub/repos/yum/reporpms/EL-8-x86\_64/pgdg-redhat-repo-latest.noarch.rpm

# sudo dnf -qy module disable postgresql

# sudo dnf install -y postgresql12-server repmgr12 postgresql12 postgresql12-contrib postgresql12-libs

# sudo /usr/pgsql-12/bin/postgresql-12-setup initdb

# sudo systemctl enable postgresql-12

# sudo systemctl start postgresql-12

На всех нодах настроим переменную окружения PATH для удобства работы с pgsql:

# sudo vim /var/lib/pgsql/.bash\_profile

PATH=$PATH:$HOME/bin:/usr/pgsql-12/bin

export PATH

На всех нодах добавляем repmgr12 в автозапуск:

# sudo systemctl enable repmgr12

Добавляем настройки repmgr в postgresql.conf (достаточно для ноды 1):

# sudo vim /var/lib/pgsql/12/data/repmgr.conf

shared\_preload\_libraries = 'repmgr'

max\_wal\_senders = 10

max\_replication\_slots = 15

wal\_level = 'replica'

hot\_standby = on

archive\_mode = on

archive\_command = '/bin/true'

# sudo echo -e "include\_if\_exists 'repmgr.conf'" >> /var/lib/pgsql/12/data/postgresql.conf

# sudo tail /var/lib/pgsql/12/data/postgresql.conf

Добавляем пользователей для репликации в pg\_hba.conf (достаточно для ноды 1):

# sudo vim /var/lib/pgsql/12/data/pg\_hba.conf

# TYPE DATABASE USER ADDRESS METHOD

# "local" is for Unix domain socket connections only

local all all trust

# IPv4 local connections:

host all all 0.0.0.0/0 trust

# IPv6 local connections:

host all all ::1/128 ident

# Allow replication connections from localhost, by a user with the

# replication privilege.

local replication repmgr trust

host replication repmgr 0.0.0.0/0 trust

host replication repmgr ::1/128 trust

local repmgr repmgr trust

host repmgr repmgr 0.0.0.0/0 trust

Назначаем владельца и права на конфиги (достаточно для ноды 1):

# sudo chown postgres:postgres /var/lib/pgsql/12/data/{pg\_hba.conf,repmgr.conf}

# sudo chmod 600 /var/lib/pgsql/12/data/{pg\_hba.conf,repmgr.conf}

Перезапускаем postgresql12 на ноде 1

# sudo systemctl restart postgresql-12

# sudo systemctl status postgresql-12

Теперь настраиваем repmgr:

Выполняем настройку схемы по умолчанию (достаточно для ноды 1):

# sudo su - postgres

# createuser -s repmgr

# createdb repmgr -O repmgr

# psql

# postgres=# \c repmgr

# postgres=# ALTER USER repmgr SET search\_path TO repmgr, "$user", public;

# postgres=# \q

Дальнейшая настройка repmgr происходит для каждой ноды. Полезно сначала сохранить оригинальный конфигурационный файл:

# sudo mv /etc/repmgr/12/repmgr.conf{,.b}

# sudo vim /etc/repmgr/12/repmgr.conf

Для ноды 1:

# vim /etc/repmgr/12/repmgr.conf

node\_id=1

node\_name='node1'

conninfo='host=192.168.0.29 dbname=repmgr user=repmgr connect\_timeout=4'

data\_directory='/var/lib/pgsql/12/data/'

use\_replication\_slots=yes

log\_file='/var/log/repmgr/repmgr.log'

pg\_bindir='/usr/pgsql-12/bin/'

ssh\_options='-q -o ConnectTimeout=10 -o "StrictHostKeyChecking no"'

failover=automatic

priority=100

reconnect\_attempts=6

reconnect\_interval=10

promote\_command='/usr/pgsql-12/bin/repmgr standby promote -f /etc/repmgr/12/repmgr.conf'

follow\_command='/usr/pgsql-12/bin/repmgr standby follow -f /etc/repmgr/12/repmgr.conf -W --upstream-node-id=%n'

service\_start\_command='sudo systemctl start postgresql-12'

service\_stop\_command='sudo systemctl stop postgresql-12'

service\_restart\_command='sudo systemctl restart postgresql-12'

service\_reload\_command='sudo systemctl reload postgresql-12'

Регистрируем ноду 1 как master:

# sudo su - postgres

# repmgr primary register

# repmgr cluster show

Для ноды 2:

# vim /etc/repmgr/12/repmgr.conf

node\_id=2

node\_name='node2'

conninfo='host=192.168.0.9 dbname=repmgr user=repmgr connect\_timeout=4'

data\_directory='/var/lib/pgsql/12/data/'

use\_replication\_slots=yes

log\_file='/var/log/repmgr/repmgr.log'

pg\_bindir='/usr/pgsql-12/bin/'

ssh\_options='-q -o ConnectTimeout=10 -o "StrictHostKeyChecking no"'

failover=automatic

priority=90

reconnect\_attempts=6

reconnect\_interval=10

promote\_command='/usr/pgsql-12/bin/repmgr standby promote -f /etc/repmgr/12/repmgr.conf'

follow\_command='/usr/pgsql-12/bin/repmgr standby follow -f /etc/repmgr/12/repmgr.conf -W --upstream-node-id=%n'

service\_start\_command='sudo systemctl start postgresql-12'

service\_stop\_command='sudo systemctl stop postgresql-12'

service\_restart\_command='sudo systemctl restart postgresql-12'

service\_reload\_command='sudo systemctl reload postgresql-12'

Регистрируем ноду 2 как standby:

# sudo su - postgres

# repmgr standby register

# repmgr cluster show

Для ноды 3:

# vim /etc/repmgr/12/repmgr.conf

node\_id=3

node\_name='node3'

conninfo='host=192.168.0.7 dbname=repmgr user=repmgr connect\_timeout=4'

data\_directory='/var/lib/pgsql/12/data/'

use\_replication\_slots=yes

log\_file='/var/log/repmgr/repmgr.log'

pg\_bindir='/usr/pgsql-12/bin/'

ssh\_options='-q -o ConnectTimeout=10 -o "StrictHostKeyChecking no"'

failover=automatic

priority=80

reconnect\_attempts=6

reconnect\_interval=10

promote\_command='/usr/pgsql-12/bin/repmgr standby promote -f /etc/repmgr/12/repmgr.conf'

follow\_command='/usr/pgsql-12/bin/repmgr standby follow -f /etc/repmgr/12/repmgr.conf -W --upstream-node-id=%n'

service\_start\_command='sudo systemctl start postgresql-12'

service\_stop\_command='sudo systemctl stop postgresql-12'

service\_restart\_command='sudo systemctl restart postgresql-12'

service\_reload\_command='sudo systemctl reload postgresql-12'

Регистрируем ноду 3 как standby:

# sudo su - postgres

# repmgr standby register

# repmgr cluster show

Дальше на всех узлах создаем скрипты и службы для определения, является ли нода master.

Редактируем файл sudoers:

# sudo visudo

Defaults:postgres !requiretty

postgres ALL = NOPASSWD: /bin/systemctl stop postgresql-12, \

/bin/systemctl start postgresql-12, \

/bin/systemctl restart postgresql-12, \

/bin/systemctl reload postgresql-12, \

/bin/systemctl status postgresql-12

Устанавливаем socat (чтобы запускать скрипты в службах):

# sudo dnf install socat

Копируем скрипт pg\_ms.sh[[2]](#footnote-2) в /usr/local/bin и делаем его исполняемым:

# sudo vim /usr/local/bin/pg\_ms.sh

# sudo chmod +x /usr/local/bin/pg\_ms.sh

Создаем службу, которая будет по http-запросу выполнять скрипт pg\_ms.sh:

# sudo vim /etc/systemd/system/pg\_ms.service

[Unit]

Description=PostgreSQL master check

[Service]

Type=simple

User=postgres

Group=postgres

ExecStart=/usr/bin/socat -t20 TCP-LISTEN:5445,reuseaddr,fork EXEC:/usr/local/bin/pg\_ms.sh

Restart=on-failure

[Install]

WantedBy=multy-user.target

Перезагружаем юниты, добавляем созданную службу в автозагрузку, запускаем:

# sudo systemctl daemon-reload

# sudo systemctl enable pg\_ms

# sudo systemctl start pg\_ms

# sudo systemctl status pg\_ms

Проверяем отдачу службы для всех нод

# sudo curl -i 192.168.0.29:5445

# sudo curl -i 192.168.0.7:5445

# sudo curl -i 192.168.0.9:5445

Добавляем правила в firewal

# sudo firewall-cmd --zone=public --permanent --add-port 5445/tcp

# sudo firewall-cmd --zone=public --permanent --add-port 5432/tcp

# sudo firewall-cmd --reload

## 2. HAProxy

Устанавливаем HAProxy:

<http://www.haproxy.org/download/2.5/src/haproxy-2.3.5.tar.gz>

# sudo wget http://www.haproxy.org/download/2.5/src/haproxy-2.5.5.tar.gz -O ~/haproxy.tar.gz

# sudo tar xzvf ~/haproxy.tar.gz -C ~/

# cd ~/haproxy-2.5.5

# sudo yum groupinstall "Development tools"

# sudo make TARGET=linux-glibc USE\_PCRE=1 USE\_OPENSSL=1 USE\_ZLIB=1

# sudo make install

# sudo mkdir -p /etc/haproxy

# sudo mkdir -p /var/lib/haproxy

# sudo touch /var/lib/haproxy/stats

# sudo ln -s /usr/local/sbin/haproxy /usr/sbin/haproxy

# sudo cp ~/haproxy-2.5.5/examples/haproxy.init /etc/init.d/haproxy

# sudo chmod 755 /etc/init.d/haproxy

# sudo systemctl daemon-reload

# sudo chkconfig haproxy on

# sudo useradd -r haproxy

Настраиваем конфигурацию haproxy:

# sudo vim /etc/haproxy/haproxy.cfg

defaults

mode tcp

frontend pgmaster\_front

bind \*:5432

default\_backend pgmaster\_back

backend pgmaster\_back

option httpchk GET

http-check expect status 200

server pgmaster1 192.168.0.29:5432 check port 5445

server pgmaster2 192.168.0.7:5432 check port 5445

server pgmaster3 192.168.0.9:5432 check port 5445

Перезагружаем haproxy:

# sudo systemctl restart haproxy

# sudo systemctl status haproxy

1. Со standby возможно только чтение, но этот сценарий здесь не рассматривается [↑](#footnote-ref-1)
2. См. приложение [↑](#footnote-ref-2)