

RuBackup

Система резервного копирования и восстановления данных

Настройка отказоустойчивого кластера PostgreSQL



RuBackup

Версия 1.5

2021 г.

Содержание

Введение.....	3
Используемое ПО.....	3
Перед установкой.....	4
Установка ПО.....	4
Настройка кластера.....	5
Настройка PostgreSQL.....	6
Настройка ресурсов кластера.....	8
Создание ресурсов кластера.....	8
Создание ресурсов фенсинга.....	10
Настройка доступа по ssh.....	10
Приложения.....	12
Приложение 1.....	12
Приложение 2.....	13
Приложение 3.....	14

Введение

Система резервного копирования (СРК) RuBackup использует СУБД PostgreSQL для хранения параметров системы и метаданных резервных копий. Поэтому для бесперебойной работы СРК RuBackup в отказоустойчивой конфигурации необходимо также обеспечить постоянную доступность базы данных.

Данное руководство содержит шаги настройки отказоустойчивого кластера PostgreSQL с синхронной репликацией из трёх узлов в минимальной конфигурации на основе программного обеспечения (ПО) Pacemaker и Corosync.

Примеры конфигурации в данном руководстве, используются для демонстрации работы кластера в закрытой сети. Для более тонкой настройки используйте официальную документацию PostgreSQL и Pacemaker, ориентируясь на используемую рабочую среду и принятую политику безопасности в вашей организации.

Используемое ПО

Для развёртывания отказоустойчивого кластера PostgreSQL используется следующее ПО:

- Операционная система Ubuntu 20.04 LTS
- Гипервизор KVM
- СУБД PostgreSQL 12.6
- Pacemaker 2.0.3
- Corosync 3.0.3
- Fence-agents 4.5.2
- Resource-agents 4.7.0

Допускается использование ПО версий выше указанных.

Перед установкой

Для развёртывания кластера необходимо подготовить инфраструктуру, состоящую из минимум трёх узлов. Возможно использование как физических, так и виртуальных машин (как в примере ниже).

Узлы могут иметь произвольные названия. Здесь и ниже три узла называются *node1*, *node2* и *node3*.

Выполните следующие действия:

1. В качестве узлов подготовьте три виртуальные машины одинаковой конфигурации и установите на них операционную систему Ubuntu 20.04 LTS.
2. Настройте синхронизацию времени между всеми узлами.
3. На всех узлах добавьте IP-адреса всех узлов в файл `/etc/hosts`.

Установка ПО

На все подготовленные узлы кластера установите необходимое ПО при помощи команды:

```
$ sudo apt install corosync pcs pacemaker fence-agents postgresql
```

Поскольку пакет `pacemaker` содержит версию пакета `resource-agents`, которая не поддерживает работу с PostgreSQL версии 12, следует заменить его пакетом версии 4.7.0. Для этого:

1. Удалите `resource-agents` со всеми зависимостями:

```
$ sudo dpkg --remove --force-depends resource-agents
```

2. Скачайте и установите пакет `resource-agents` версии 4.7.0:

```
$ wget http://ftp.de.debian.org/debian/pool/main/r/resource-agents/resource-agents_4.7.0-1_amd64.deb
```

```
$ sudo dpkg -i resource-agents_4.7.0-1_amd64.deb
```

Настройка кластера

Для первоначальной настройки кластера выполните следующие действия.

1. Измените пароль пользователя `hacluster`, который был создан автоматически на всех узлах кластера при установке пакетов :

```
$ sudo passwd hacluster
```

2. Произведите аутентификацию на одном из узлов кластера:

```
$ sudo pcs host auth node1 node2 node3
```

3. Создайте кластер под названием `pgcluster`.

```
$ sudo pcs cluster setup --force pgcluster node1 addr=ipaddr node2  
addr=ipaddr node3 addr=ipaddr
```

Параметры команды:

`ipaddr` - IP-адрес добавляемого узла;

`node1`, `node2`, `node3` - сетевые имена узлов кластера.

Внимание! IP-адреса всех узлов кластера должны быть явно указаны в файле `/etc/hosts` на всех узлах кластера.

4. Запустите кластер и настройте автоматический запуск кластера при запуске системы:

```
$ sudo systemctl enable corosync.service
```

```
$ sudo systemctl enable pacemaker.service
```

```
$ sudo pcs cluster enable --all
```

Эти действия завершают настройку кластера.

Чтобы узнать статус кластера используйте команду:

```
$ sudo pcs cluster status
```

Чтобы проверить синхронизацию узлов кластера используйте команду:

```
$ corosync-smartctl | grep members
```

Для отслеживания состояния кластера в реальном времени можно использовать команду:

```
$ sudo crm_mon -Afr
```

Настройка PostgreSQL

После первоначальной настройки кластера следует настроить СУБД PostgreSQL следующим образом.

1. Отключите запуск **postgresql.service** при загрузке системы на каждом узле кластера. Включать и отключать сервис при необходимости теперь будет Pacemaker.

```
$ sudo systemctl disable postgresql.service
```

2. На узле (в нашем случае это *node1*), который первоначально будет являться мастером, инициализируйте новую базу данных:

```
$ sudo -u postgres /usr/lib/postgresql/12/bin/initdb -D  
/var/lib/postgresql/12/main
```

Если база уже запущена, то остановите процесс **postgresql**, очистите директорию `/var/lib/postgresql/12/main` и выполните команду ещё раз.

3. Запустите базу:

```
$ sudo -u postgres /usr/lib/postgresql/12/bin/pg_ctl -D  
/var/lib/postgresql/12/main start
```

4. Создайте пользователя для репликации:

```
$ sudo -u postgres /usr/lib/postgresql/12/bin/createuser --  
replication -P repl
```

5. Измените файл `/var/lib/postgresql/12/main/pg_hba.conf` и добавьте в него необходимые разрешения следующим образом:

```
# TYPE  DATABASE  USER  ADDRESS  METHOD

# "local" is for Unix domain socket connections only
local  all      all            trust
# IPv4 local connections:
host   all      all          127.0.0.1/32  trust
# IPv6 local connections:
host   all      all          ::1/128      trust
#Allow replication connections from localhost, by a user with the
#replication privilege.
local  replication  all            trust
host   replication  all          127.0.0.1/32  trust
host   replication  all          ::1/128      trust
host   replication  all          192.168.110.0/24  trust
host   all        all          192.168.110.0/24  trust
```

Внимание! В двух нижних строках в столбце **ADDRESS** следует указать вашу подсеть.

6. Измените файл `/var/lib/postgresql/12/main/postgresql.conf` и добавьте в него следующие строки:

```
listen_addresses = '*'  
wal_level = replica  
logging_collector = on  
hot_standby = on  
lc_messages = 'C'  
wal_keep_segments = 10
```

7. Перезапустите СУБД PostgreSQL:

```
$ sudo -u postgres /usr/lib/postgresql/12/bin/pg_ctl -D  
/var/lib/postgresql/12/main stop  
$ sudo -u postgres /usr/lib/postgresql/12/bin/pg_ctl -D  
/var/lib/postgresql/12/main start
```

На других двух узлах (*node2* и *node3*) выполните следующие действия:

1. Остановите процесс postgresql, если он запущен.
2. Очистите директорию `/var/lib/postgresql/12/main`.
3. Скопируйте базу данных с мастера (*node1*) при помощи команды:

```
$ sudo -u postgres pg_basebackup -D /var/lib/postgresql/12/main -h  
192.168.110.200 -X stream
```

Параметр `192.168.110.200` в данном случае - это IP-адрес узла *node1*, на котором мы произвели первоначальную настройку PostgreSQL и инициализировали базу данных.

Настройка ресурсов кластера

Создание ресурсов кластера

После настройки СУБД PostgreSQL на всех узлах кластера следует создать ресурсы кластера. Для этого выполните следующие действия:

1. Создайте ресурс с именем *virtual_ip* типа IPAddr2, который будет использоваться для подключения в базе данных PostgreSQL:

```
$ sudo pcs resource create virtual_ip IPAddr2 ip="192.168.110.204"
cidr_netmask="24" meta migration-threshold="0" op monitor
timeout="60s" interval="10s" on-fail="restart" op stop
timeout="60s" interval="0s" on-fail="ignore" op start
timeout="60s" interval="0s" on-fail="stop"
```

В качестве значения параметра *ip* укажите любой адрес из подсети, в которой находятся узлы кластера. Этот IP-адрес будет автоматически добавляться на интерфейс того узла, который будет являться мастером в тот момент времени.

2. Создайте ресурс с именем *my-pgsql* типа *pgsql* для управления конфигурацией PostgreSQL:

```
$ sudo pcs resource create my-pgsql pgsql
pgctl="/usr/lib/postgresql/12/bin/pg_ctl"
psql="/usr/lib/postgresql/12/bin/psql"
pgdata="/var/lib/postgresql/12/main" rep_mode="sync"
node_list="node1 node2 node3" master_ip="192.168.110.204"
restart_on_promote="false" check_wal_receiver="true" pgport="5432"
primary_conninfo_opt="password=12345" repuser="repl"
```

3. Для созданного выше ресурса *my-pgsql* укажите, что он может иметь одно из нескольких состояний и менять их в зависимости от типа узла (master и slave):

```
$ sudo pcs resource promotable my-pgsql promoted-max=1 promoted-
node-max=1 clone-max=3 clone-node-max=1 notify=true
```

Свяжите два созданных выше ресурса, чтобы они запускались вместе на одном узле, и установите очерёдность запуска таким образом, чтобы ресурс *virtual_ip* запускался только после успешного запуска ресурса *my-pgsql*. Для этого создайте группу ресурсов *master-group* и добавьте в неё ресурсы:

```
$ sudo pcs resource group add master-group virtual_ip
$ sudo pcs constraint colocation add master-group with Master my-
pgsql-clone
```



```
$ sudo pcs constraint order promote my-pgsql-clone then start  
master-group symmetrical=false score=INFINITY  
$ sudo pcs constraint order demote my-pgsql-clone then stop  
master-group symmetrical=false score=0
```

Создание ресурсов фенсинга

Для защиты разделяемых ресурсов и изоляции узла кластера при его неисправности существует механизм фенсинга (изоляция).

Чтобы вывести список доступных fence-agents используйте команду:

```
$ sudo pcs stonith list
```

При использовании виртуальных машин в качестве узлов кластера можно использовать агент **fence_virsh**.

Чтобы вывести необходимые настройки для выбранного агента используйте команду:

```
$ sudo pcs stonith describe fence_virsh
```

Настройка доступа по ssh

Чтобы настроить доступ по ssh к серверу с гипервизором под пользователем root по ключу выполните следующие действия.

1. На сервере в файле `/etc/ssh/sshd_config` установите значение параметра `PermitRootLogin` равное `yes`.

2. Перезагрузите службу `sshd`:

```
$ sudo systemctl restart sshd.service
```

3. На всех узлах сгенерируйте ключи при помощи команды:

```
$ ssh-keygen
```

4. На всех узлах отправьте публичный ключ на сервер с гипервизором:

```
$ ssh-copy-id root@192.168.110.1
```

5. На сервере в файле `/etc/ssh/sshd_config` прокомментируйте параметр `PermitRootLogin` (чтобы он не применялся в конфигурации).

6. Перезагрузите службу `sshd` для применения настроек:

```
$ sudo systemctl restart sshd.service
```

Для проверки работы `fence_virsh` перед настройкой можно использовать команду:

```
$ sudo fence_virsh -a 192.168.110.1 -l root -n node1 -x -k  
/home/user/.ssh/id_rsa -o list
```

Параметры команды:

`-a 192.168.110.1` - IP-адрес сервера, на котором запущен гипервизор KVM;

-l root - логин пользователя для подключения к серверу с гипервизором по ssh;

-n *node1* — название виртуальной машины в гипервизоре;

-k /home/user/.ssh/id_rsa - путь к ключу, созданному при помощи команды ssh-keygen.

В результате выполнения команда выведет список всех виртуальных машин в гипервизоре.

Теперь следует создать и настроить ресурсы фенсинга для всех узлов кластера. Выполните следующие действия.

1. Создайте ресурс фенсинга *fence_node1* для первого узла (*node1*) при помощи команды:

```
$ sudo pcs stonith create fence_node1 fence_virsh  
pcmk_host_list="node1" ipaddr="192.168.110.1" login="root"  
identity_file="/home/u/.ssh/id_rsa" pcmk_reboot_action="reboot"  
pcmk_monitor_timeout=60s plug=node1
```

Параметры команды:

pcmk_host_list - какими узлами кластера может управлять данный ресурс;

plug - название виртуальной машины в гипервизоре.

2. Аналогично создайте ресурсы *fence_node2* и *fence_node3* для узлов *node2* и *node3*.

После создания ресурсов фенсинга для каждого узла, необходимо настроить их таким образом, чтобы они не запускались на тех узлах, для перезагрузки которых они созданы.

1. Для ресурса *fence_node1* выполните команду:

```
$ sudo pcs constraint location fence_node1 avoids node1=INFINITY
```

2. Выполните аналогичную команду для других узлов:

```
$ sudo pcs constraint location fence_node2 avoids node2=INFINITY
```

```
$ sudo pcs constraint location fence_node3 avoids node3=INFINITY
```

3. Перезагрузите все виртуальные машины в кластере.

На этом базовая настройка отказоустойчивого кластера PostgreSQL окончена.

Приложения

Приложение 1

Содержимое файла конфигурации PostgreSQL 12

```
/var/lib/postgresql/12/main/postgresql.conf  
max_connections = 100  
shared_buffers = 128MB  
dynamic_shared_memory_type = posix  
max_wal_size = 1GB  
min_wal_size = 80MB  
log_timezone = 'Europe/Moscow'  
datestyle = 'iso, mdy'  
timezone = 'Europe/Moscow'  
lc_messages = 'C.UTF-8'  
lc_monetary = 'C.UTF-8'  
lc_numeric = 'C.UTF-8'  
lc_time = 'C.UTF-8'  
default_text_search_config = 'pg_catalog.english'  
listen_addresses = '*'  
wal_level = replica  
logging_collector = on  
hot_standby = on  
lc_messages = 'C'  
wal_keep_segments = 10
```

Приложение 2

Содержимое файла конфигурации PostgreSQL 12

/var/lib/postgresql/12/main/pg_hba.conf

```
# TYPE  DATABASE        USER            ADDRESS          METHOD

# "local" is for Unix domain socket connections only
local                                all              all              trust
trust

# IPv4 local connections:
host                                all              all              127.0.0.1/32    trust
trust

# IPv6 local connections:
host                                all              all              ::1/128         trust
trust

# Allow replication connections from localhost, by a user with the
# replication privilege.
local                                replication      all              all              trust
trust
host                                replication      all              127.0.0.1/32    trust
trust
host                                replication      all              ::1/128         trust
trust
host                                replication      all              192.168.110.0/24 trust
trust
host                                all              all              all              trust
192.168.110.0/24
```

Приложение 3

Содержимое файла конфигурации Corosync

```
/etc/corosync/corosync.conf
totem {
    version: 2
    cluster_name: pgcluster
    transport: knet
    crypto_cipher: aes256
    crypto_hash: sha256
}

nodelist {
    node {
        ring0_addr: 192.168.110.200
        name: node1
        nodeid: 1
    }

    node {
        ring0_addr: 192.168.110.201
        name: node2
        nodeid: 2
    }

    node {
        ring0_addr: 192.168.110.203
        name: node3
        nodeid: 3
    }
}

quorum {
    provider: corosync_votequorum
}

logging {
    to_logfile: yes
    logfile: /var/log/corosync/corosync.log
    to_syslog: yes
    timestamp: on
}
```

