repmgr+PostgreSQL故障自动转移

1. 创建dockerfile

```
# 使用 CentOS 7 作为基础镜像
FROM centos:7

# 设置环境变量
ENV PG_MODE=primary \
    PG_USER=postgres \
    PG_PASSWORD=postgres \
    NODE_ROLE=master \
    MASTER_NAME=master \
    MASTER_PORT=5432 \
    RE_USER=repmgr \
    NODE_ID=1 \
    #NET_SEGMENT=192.168.0 \
    NODE_NAME=master1 \
    PG_DATADIR=/home/postgres/pgdata \
    PG_REPMGR_CONF=/home/postgres/repmgr.conf \
    PG_BINDIR=/usr/pgsql-12/bin \
    PG_CONFIGDIR=/home/postgres/pgdata/postgresql.conf \
    PRIORITY=1 \
    CONNINFO_HOST=master \
    CONNINFO_PORT=5432 \
    PATH="/usr/pgsql-12/bin:${PATH}"

# 创建postgres用户和组
RUN groupadd -r postgres  \
    && useradd -r -g postgres  postgres

# 安装所需的软件和工具
RUN yum -y install epel-release \
    && yum -y install https://download.postgresql.org/pub/repos/yum/reporpms/EL-7-x86_64/pgdg-redhat-repo-latest.noarch.rpm \
    && yum -y install postgresql12 postgresql12-server postgresql12-contrib repmgr12 \
    && yum -y install which sudo iproute hostname \
```

```
    && yum -y clean all

# 创建脚本存放目录
RUN mkdir -p /home/runtime/ \
    && chown -R postgres:postgres /home/runtime/

# 将 functions 文件和 entrypoint.sh 添加到镜像中
#COPY functions /home/postgres/runtime/functions
COPY functions /home/runtime/functions
#COPY entrypoint.sh /home/postgres/runtime/entrypoint.sh
COPY entrypoint.sh /home/runtime/entrypoint.sh

# 修改文件权限与属性
RUN mkdir -p /home/postgres/pgdata \
    && chown -R postgres:postgres /home/postgres \
    #&& chown -R postgres:postgres /home/postgres/pgdata \
    #&& chown 777 /home/postgres/pgdata \
    #&& chmod +x /home/postgres/runtime/entrypoint.sh \
    && chmod +x /home/runtime/entrypoint.sh \
    #&& chmod +x /home/postgres/runtime/functions
    && chmod +x /home/runtime/functions
    #&& usermod -a -G root postgres \
    #&& chmod 770 /home/postgres

RUN echo 'postgres ALL=(ALL) NOPASSWD: /bin/chown' >> /etc/sudoers

# 切换到 postgres 用户
USER postgres

# 初始化 PostgreSQL 数据目录
#RUN initdb -D ${PG_DATADIR} -U${PG_USER}

# 暴露 PostgreSQL 端口
EXPOSE 5432

# 设置启动脚本
#ENTRYPOINT ["/home/postgres/runtime/entrypoint.sh"]
ENTRYPOINT ["/home/runtime/entrypoint.sh"]
```

2. 打包成镜像

```
docker run build -t centos7-pgsql-repmgr:v1.0
```

3. docker-compose□□ master□□□dockercompose.yaml

```
version: "3"

services:
  pg-master:
    image: centos7-pgsql-repmgr:v1.9.27
    hostname: master1
    container_name: pg-master
    environment:
      PG_MODE: primary
      PG_USER: postgres
      PG_PASSWORD: postgres
      NODE_ROLE: master
      NODE_ID: 1
      NODE_NAME: master1
      #MASTER_NAME: 172.18.41.8
      #MASTER_PORT: 25432
      CONNINFO_HOST: 172.18.41.8
      CONNINFO_PORT: 25432
      PRIORITY: 10
      POSTGRES_DB: repmgr
    ports:
      - "25432:5432"
    volumes:
      - ./pg-data:/home/postgres
```

slave□□□dockercompose.yaml

```
version: "3"

services:
  pg-slave:
    image: centos7-pgsql-repmgr:v1.9.27
    container_name: pg-salve
    hostname: salve1
    #network_mode: host
    environment:
```

```
      PG_MODE: salve
      PG_USER: postgres
      PG_PASSWORD: postgres
      PGPASSWORD: postgres
      RE_USER: repmgr
      NODE_ROLE: slave
      MASTER_NAME: 172.18.41.8
      MASTER_PORT: 25432
      NODE_ID: 2
      NODE_NAME: salve1
      CONNINFO_HOST: 172.18.41.2
      CONNINFO_PORT: 25432
      PRIORITY: 15
      POSTGRES_DB: repmgr
    ports:
      - "25432:5432"
    volumes:
      - ./pg-data:/home/postgres
~
```

master□□□□□salve□□□dockercompose.yaml

```
version: "3"

services:
  pg-slave:
    image: centos7-pgsql-repmgr:v1.9.27
    hostname: salve1
    container_name: pg-salve
    environment:
      MASTER_NAME: 172.18.41.8
      PGPASSWORD: postgres
      MASTER_PORT: 25432
      CONNINFO_HOST: 172.18.41.2
      CONNINFO_PORT: 25432
      NODE_ID: 1
      PRIORITY: 15
      POSTGRES_DB: repmgr
    ports:
      - "25432:5432"
```

```
    volumes:
      - ./pg-data:/home/postgres
```

注：恢复前请做好数据备份，防止数据丢失

附：functions脚本函数

```
#!/bin/bash

# 容器调用入口函数，根据传入命令不同，执行注册或启动主库、注册或启动备库，从新加入集群的操作。
configure_repmgr()
{
  case ${NODE_ROLE} in
    master)
      echo 'master'
      initialize_master
      ;;
    slave)
      echo 'slave'
      initialize_slave
      ;;
  #  rejoin)
  #    echo 'rejoin'
  #    rejoin_node
  #    ;;
    master_slave)
      echo 'master_slave'
      master_slave
      ;;
  esac
}

# 用于设置 PostgreSQL 配置文件 'postgresql.conf' 中的参数。
# 该函数接受两个参数：参数名 (param_name) 和参数值 (param_value),
# 并使用 'sed' 命令在 'postgresql.conf' 文件中查找以参数名开头的行
# （行前可能有一个 # 注释符号），然后用新的参数值替换该行。

set_postgresql_param() {
  param_name="$1"
  param_value="$2"
  sed -i "/^#${param_name} =/c ${param_name} = ${param_value}" ${PG_DATADIR}/postgresql.conf
```

```
}

# 这个函数用来向 PostgreSQL 的 pg_hba.conf 文件中添加一行新的配置
# ${1}: 你想要添加的配置行
# ${PG_DATADIR}: pg_hba.conf 文件的位置
set_hba_param() {
  # 判断是否传入了配置行
  if [ -z "${1}" ]; then
    echo "No configuration line provided."
    return 1
  fi

  # 检查 pg_hba.conf 文件是否存在
  if [ ! -e "${PG_DATADIR}/pg_hba.conf" ]; then
    echo "${PG_DATADIR}/pg_hba.conf does not exist."
    return 1
  fi

  # 检查是否已经拥有写权限，如果没有则尝试获取
  if [ ! -w "${PG_DATADIR}/pg_hba.conf" ]; then
    chmod u+w ${PG_DATADIR}/pg_hba.conf || {
      echo "Could not set write permissions on ${PG_DATADIR}/pg_hba.conf"
      return 1
    }
  fi

  # 添加配置行到 pg_hba.conf 文件
  echo "${1}" >> ${PG_DATADIR}/pg_hba.conf || {
    echo "Could not write to ${PG_DATADIR}/pg_hba.conf"
    return 1
  }

  return 0
}

master_slave() {
  # 检查pgdata-bak是否存在，如果存在则警告并直接启动pgsql
  if [ -d "${PG_DATADIR}-bak" ]; then
      echo "Warning: ${PG_DATADIR}-bak already exists. Starting pgsql directly."
      pg_ctl -D $PG_DATADIR start
```

```
  else
    #export PGPASSFILE=~/.pgpass
    #write_pgpass
    IP=`ping ${MASTER_NAME} -c 1 -w 1 |  sed '1{s/[^(]*(//;s/).*//;q}'`
    # 1.  备份pgsql数据
    mv $PG_DATADIR ${PG_DATADIR}-bak

    # 2.  创建pgsql数据目录
    mkdir $PG_DATADIR

    # 3.  从现在的主库获取备份数据
    #repmgr -f $PG_REPMGR_CONF -h ${MASTER_NAME} -U repmgr -d repmgr -D $PG_DATADIR standby clone
    echo "repmgr -h ${IP} -p ${MASTER_PORT} -U repmgr -d repmgr -f ${PG_REPMGR_CONF} standby clone --dry-run"
    repmgr -h ${IP} -p ${MASTER_PORT} -U repmgr -d repmgr -f ${PG_REPMGR_CONF} standby clone --dry-run
    echo "repmgr -h ${IP} -p ${MASTER_PORT} -U repmgr -d repmgr -f ${PG_REPMGR_CONF} standby clone"
    repmgr -h ${IP} -p ${MASTER_PORT} -U repmgr -d repmgr -f ${PG_REPMGR_CONF} standby clone

    # 4.  启动pgsql
    echo "pg_ctl -D $PG_DATADIR start"
    pg_ctl -D $PG_DATADIR start

    # 5.  重新注册节点
    echo "repmgr -f $PG_REPMGR_CONF standby register --force"
    repmgr -f $PG_REPMGR_CONF standby register --force
  fi
  echo "repmgrd -f ${PG_REPMGR_CONF} --pid-file /tmp/repmgrd.pid --daemonize=false"
  repmgrd -f ${PG_REPMGR_CONF} --pid-file /tmp/repmgrd.pid --daemonize=false
}



# 注册和执行主库操作，包括修改配置文件、创建用户、数据库及插件等，并设置守护进程用于自动故障转移。
initialize_master()
{
  if [[ ! -f ${PG_DATADIR}/PG_VERSION ]]; then
    initdb -D /home/${PG_USER}/pgdata  -U${PG_USER}
```

```
    write_postgresql_config
    write_pg_hba_conf
    write_pgpass
    pg_ctl -D /home/${PG_USER}/pgdata -w start >/dev/null
    psql -U ${PG_USER} -d postgres -h localhost   -c "ALTER USER ${PG_USER} WITH PASSWORD
'${PG_PASSWORD}';" >/dev/null
    psql -U ${PG_USER} -d postgres -h localhost   -c "create database repmgr;" >/dev/null
    psql -U ${PG_USER} -d postgres -h localhost   -c "create extension repmgr;" >/dev/null
    psql -U ${PG_USER} -d repmgr -h localhost  -c "create user repmgr with superuser;"
>/dev/null
    psql -U ${PG_USER} -d repmgr -h localhost  -c "alter user repmgr password
'${PG_PASSWORD}';" >/dev/null
    write_repmgr_conf
    repmgr -f ${PG_REPMGR_CONF} primary register
  else
    pg_ctl -D /home/${PG_USER}/pgdata -w start >/dev/null
  fi
  repmgrd -f ${PG_REPMGR_CONF} --pid-file /tmp/repmgrd.pid --daemonize=false
}

# 注册和执行备库操作，克隆主库，加入repmgr集群，并设置守护进程用于自动故障转移。
initialize_slave()
{
  if [[ ! -f ${PG_DATADIR}/PG_VERSION ]]; then
    write_repmgr_conf
    write_pgpass
    IP=`ping ${MASTER_NAME} -c 1 -w 1 | sed '1{s/[^(]*(//;s/).*//;q}'`
    repmgr -h ${IP} -p ${MASTER_PORT} -U repmgr -d repmgr -f ${PG_REPMGR_CONF} standby clone
--dry-run
    repmgr -h ${IP} -p ${MASTER_PORT} -U repmgr -d repmgr -f ${PG_REPMGR_CONF} standby clone
    pg_ctl -D ${PG_DATADIR} -w start >/dev/null
    repmgr -f ${PG_REPMGR_CONF}  standby register
    repmgrd -f ${PG_REPMGR_CONF} --pid-file /tmp/repmgrd.pid --daemonize=false
  else
    pg_ctl -D ${PG_DATADIR} -w start >/dev/null
    repmgrd -f ${PG_REPMGR_CONF} --pid-file /tmp/repmgrd.pid --daemonize=false
  fi
}

# 将已有节点重新加入到集群，启动守护进行用于自动切换。
```

```
rejoin_node()
{
  if [[ -f ${PG_DATADIR}/PG_VERSION ]]; then
    IP=`ping ${MASTER_NAME} -c 1 -w 1 | sed '1{s/[^(]*(//;s/).*//;q}'`
    if [[ -d /home/${PG_USER}/pgdata-bak ]];then
      rm -fr /home/${PG_USER}/pgdata-bak
    fi
    cp -a /home/${PG_USER}/pgdata /home/${PG_USER}/pgdata-bak
    rm -fr /home/postgres/pgdata/postmaster.pid
   # pg_resetwal -f /home/${PG_USER}/pgdata
    repmgr node rejoin -d "host=${IP} dbname=repmgr user=repmgr" --force-rewind --config-
files="postgresql.conf,postgresql.auto.conf" -f ${PG_REPMGR_CONF} --verbose --dry-run
    repmgr node rejoin -d "host=${IP} dbname=repmgr user=repmgr" --force-rewind --config-
files="postgresql.conf,postgresql.auto.conf" -f ${PG_REPMGR_CONF} --verbose
  fi
}

# □□postgresql.conf□□
write_postgresql_config()
{
  set_postgresql_param "wal_log_hints" "on"
  set_postgresql_param "archive_mode" "on"
  set_postgresql_param "archive_command" "\'test ! -f /home/${PG_USER}/pgarch/%f && cp %p
/home/${PG_USER}/pgarch/%f\'"
  set_postgresql_param "wal_level" "hot_standby"
  set_postgresql_param "listen_addresses" "\'*\'"
  set_postgresql_param "hot_standby" "on"
  set_postgresql_param "max_wal_senders" "10"
  set_postgresql_param "wal_keep_segments" "10"
  set_postgresql_param "port" "${PG_PORT:-5432}"
  set_postgresql_param "max_connections " "100"
  set_postgresql_param "superuser_reserved_connections" "10"
  set_postgresql_param "full_page_writes" "on"
  set_postgresql_param "max_replication_slots" "10"
  set_postgresql_param "synchronous_commit" "on"
  set_postgresql_param "shared_preload_libraries" "repmgr"
  set_postgresql_param "log_destination" "csvlog"
  set_postgresql_param "logging_collector" "on"
  set_postgresql_param "log_directory" "on"
  set_postgresql_param "log_filename" "postgresql-%Y-%m-%d_%H%M%S"
```

```
  set_postgresql_param "log_rotation_age" "1d"
  set_postgresql_param "log_rotation_size" "10MB"
  set_postgresql_param "log_statement" "mod"
  #set_postgresql_param "data_directory" "/home/pgsqlData"
}

# □□repmgr.conf□□
write_repmgr_conf()
{
  echo "node_id=${NODE_ID}" > ${PG_REPMGR_CONF}
  echo "node_name='${NODE_NAME}'" >> ${PG_REPMGR_CONF}
  #echo "conninfo='host=${CONNINFO_HOST} user=repmgr dbname=repmgr connect_timeout=2'" >>
${PG_REPMGR_CONF}
  echo "conninfo='host=${CONNINFO_HOST} port=${CONNINFO_PORT} user=repmgr dbname=repmgr
connect_timeout=2'" >> ${PG_REPMGR_CONF}
  echo "data_directory='${PG_DATADIR}'" >> ${PG_REPMGR_CONF}
  echo "config_directory='${PG_CONFIGDIR}'" >> ${PG_REPMGR_CONF}
  echo "use_replication_slots=true" >> ${PG_REPMGR_CONF}
  echo "reconnect_attempts=4" >> ${PG_REPMGR_CONF}
  echo "reconnect_interval=5" >> ${PG_REPMGR_CONF}
  echo "monitor_interval_secs=2" >> ${PG_REPMGR_CONF}
  echo "retry_promote_interval_secs=300" >> ${PG_REPMGR_CONF}
  echo "pg_bindir='${PG_BINDIR}'" >> ${PG_REPMGR_CONF}
  echo "log_level='INFO'" >> ${PG_REPMGR_CONF}
  echo "log_status_interval=300" >> ${PG_REPMGR_CONF}
  echo "log_facility='STDERR'" >> ${PG_REPMGR_CONF}
  #echo "event_notification_command='${PG_EVENT_NOTIFICATION_SCRIPT}'" >> ${PG_REPMGR_CONF}
  echo "promote_command='repmgr standby promote -f ${PG_REPMGR_CONF}'" >> ${PG_REPMGR_CONF}
  echo "follow_command='repmgr standby follow -f ${PG_REPMGR_CONF} -W --log-to-file'" >>
${PG_REPMGR_CONF}
  echo "failover='automatic'" >> ${PG_REPMGR_CONF}
  echo "priority=${PRIORITY}" >> ${PG_REPMGR_CONF}
  echo "degraded_monitoring_timeout=-1" >> ${PG_REPMGR_CONF}
}

# □□pg_hba.conf□□
write_pg_hba_conf()
{
  set_hba_param " local   replication   ${PG_USER}                                  trust "
  set_hba_param " host    replication   ${PG_USER}      127.0.0.1/32            trust "
```

```
  set_hba_param " local   repmgr        ${PG_USER}                              trust "
  set_hba_param " host    repmgr        ${PG_USER}      127.0.0.1/32            trust "
  #set_hba_param " host    replication   ${PG_USER}      ${NET_SEGMENT}/24          md5 "
  #set_hba_param " host    repmgr        ${PG_USER}      ${NET_SEGMENT}/24          md5 "
  #set_hba_param " host    repmgr        repmgr       ${NET_SEGMENT}/24          md5 "
  set_hba_param " host    replication   ${RE_USER}      0.0.0.0/0          md5 "
  set_hba_param " host    repmgr        ${PG_USER}      0.0.0.0/0          md5 "
  set_hba_param " host    repmgr        repmgr       0.0.0.0/0          md5 "
  set_hba_param " host    all    all    0.0.0.0/0    md5 "
}

# □□.pgpass□□
write_pgpass()
{
    if [ -f ~/.pgpass ]
    then
        rm -f ~/.pgpass
    fi
    echo "*:*:*:${PG_USER}:${PG_PASSWORD}" >> ~/.pgpass
    echo "*:*:repmgr:repmgr:${PG_PASSWORD}" >> ~/.pgpass
    chmod 600 ~/.pgpass
}
```

□2□entrypoint.sh

```
#!/bin/bash
set -e

# shellcheck source=runtime/functions
#source "/home/postgres/runtime/functions"
source "/home/runtime/functions"
sudo chown -R postgres:postgres /home/postgres
if [ ! -d /home/postgres/pgarch/ ];then
  mkdir -p /home/postgres/pgarch/
fi

if [ -f /tmp/repmgrd.pid ];then
  rm -fr /tmp/repmgrd.pid
fi
#□□repmgr
configure_repmgr
```

版本号 #3
创建于 30 十一月 2023 06:35:22
更新于 15 一月 2025 14:56:20