## 主备节点编译安装PostgreSQL V10 (略)

PGHOME --> /usr/local/pg103

PGDATA --> /data/pg103/data

## 初始化数据库服务

(主节点，备节点不需要初始化数据库)

pg\_ctl initdb

或

initdb -D $PGDATA -E UTF8 --locale=C -U postgres

## repmgr安装介质准备

[repmgr v.5.1.0](https://repmgr.org/download/repmgr-5.1.0.tar.gz) （可直接下载，[签名](https://repmgr.org/download/repmgr-5.1.0.tar.gz.asc)， [发行说明](https://repmgr.org/docs/current/release-5.1.0.html)）

SHA15859789e71f93c1315b9520e197b92fe60693418

*请查看旧版本的[下载部分](https://repmgr.org/downloads.html)。*

## 主备节点编译安装repmgr

建议使用postgres用户进行安装

[root@node02 repmgr]# ./configure && make install

checking for a sed that does not truncate output... /bin/sed

checking for pg\_config... /usr/local/pg103/bin/pg\_config

configure: building against PostgreSQL 10.3

checking for gnused... no

checking for gsed... no

checking for sed... yes

configure: creating ./config.status

config.status: creating Makefile

config.status: creating Makefile.global

config.status: creating config.h

Building against PostgreSQL 10

gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Wendif-labels -Wmissing-format-attribute -Wformat-security -fno-strict-aliasing -fwrapv -fexcess-precision=standard -O2 -fPIC -std=gnu89 -I/usr/local/pg103/include/postgresql/internal -I/usr/local/pg103/include -Wall -Wmissing-prototypes -Wmissing-declarations -I. -I./ -I/usr/local/pg103/include/postgresql/server -I/usr/local/pg103/include/postgresql/internal -D\_GNU\_SOURCE -c -o repmgr.o repmgr.c

gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Wendif-labels -Wmissing-format-attribute -Wformat-security -fno-strict-aliasing -fwrapv -fexcess-precision=standard -O2 -fPIC -shared -o repmgr.so repmgr.o -L/usr/local/pg103/lib -Wl,--as-needed -Wl,-rpath,'/usr/local/pg103/lib',--enable-new-dtags -L/usr/local/pg103/lib -lpq

sed -E 's/REPMGR\_VERSION\_DATE.\*""/REPMGR\_VERSION\_DATE "2020-06-11"/' repmgr\_version.h.in >repmgr\_version.h; \

sed -i -E 's/PG\_ACTUAL\_VERSION\_NUM/PG\_ACTUAL\_VERSION\_NUM 100003/' repmgr\_version.h

gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Wendif-labels -Wmissing-format-attribute -Wformat-security -fno-strict-aliasing -fwrapv -fexcess-precision=standard -O2 -fPIC -std=gnu89 -I/usr/local/pg103/include/postgresql/internal -I/usr/local/pg103/include -Wall -Wmissing-prototypes -Wmissing-declarations -I. -I./ -I/usr/local/pg103/include/postgresql/server -I/usr/local/pg103/include/postgresql/internal -D\_GNU\_SOURCE -c -o repmgr-client.o repmgr-client.c

gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Wendif-labels -Wmissing-format-attribute -Wformat-security -fno-strict-aliasing -fwrapv -fexcess-precision=standard -O2 -fPIC -std=gnu89 -I/usr/local/pg103/include/postgresql/internal -I/usr/local/pg103/include -Wall -Wmissing-prototypes -Wmissing-declarations -I. -I./ -I/usr/local/pg103/include/postgresql/server -I/usr/local/pg103/include/postgresql/internal -D\_GNU\_SOURCE -c -o repmgr-action-primary.o repmgr-action-primary.c

gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Wendif-labels -Wmissing-format-attribute -Wformat-security -fno-strict-aliasing -fwrapv -fexcess-precision=standard -O2 -fPIC -std=gnu89 -I/usr/local/pg103/include/postgresql/internal -I/usr/local/pg103/include -Wall -Wmissing-prototypes -Wmissing-declarations -I. -I./ -I/usr/local/pg103/include/postgresql/server -I/usr/local/pg103/include/postgresql/internal -D\_GNU\_SOURCE -c -o repmgr-action-standby.o repmgr-action-standby.c

gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Wendif-labels -Wmissing-format-attribute -Wformat-security -fno-strict-aliasing -fwrapv -fexcess-precision=standard -O2 -fPIC -std=gnu89 -I/usr/local/pg103/include/postgresql/internal -I/usr/local/pg103/include -Wall -Wmissing-prototypes -Wmissing-declarations -I. -I./ -I/usr/local/pg103/include/postgresql/server -I/usr/local/pg103/include/postgresql/internal -D\_GNU\_SOURCE -c -o repmgr-action-witness.o repmgr-action-witness.c

gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Wendif-labels -Wmissing-format-attribute -Wformat-security -fno-strict-aliasing -fwrapv -fexcess-precision=standard -O2 -fPIC -std=gnu89 -I/usr/local/pg103/include/postgresql/internal -I/usr/local/pg103/include -Wall -Wmissing-prototypes -Wmissing-declarations -I. -I./ -I/usr/local/pg103/include/postgresql/server -I/usr/local/pg103/include/postgresql/internal -D\_GNU\_SOURCE -c -o repmgr-action-cluster.o repmgr-action-cluster.c

gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Wendif-labels -Wmissing-format-attribute -Wformat-security -fno-strict-aliasing -fwrapv -fexcess-precision=standard -O2 -fPIC -std=gnu89 -I/usr/local/pg103/include/postgresql/internal -I/usr/local/pg103/include -Wall -Wmissing-prototypes -Wmissing-declarations -I. -I./ -I/usr/local/pg103/include/postgresql/server -I/usr/local/pg103/include/postgresql/internal -D\_GNU\_SOURCE -c -o repmgr-action-node.o repmgr-action-node.c

gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Wendif-labels -Wmissing-format-attribute -Wformat-security -fno-strict-aliasing -fwrapv -fexcess-precision=standard -O2 -fPIC -std=gnu89 -I/usr/local/pg103/include/postgresql/internal -I/usr/local/pg103/include -Wall -Wmissing-prototypes -Wmissing-declarations -I. -I./ -I/usr/local/pg103/include/postgresql/server -I/usr/local/pg103/include/postgresql/internal -D\_GNU\_SOURCE -c -o repmgr-action-service.o repmgr-action-service.c

gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Wendif-labels -Wmissing-format-attribute -Wformat-security -fno-strict-aliasing -fwrapv -fexcess-precision=standard -O2 -fPIC -std=gnu89 -I/usr/local/pg103/include/postgresql/internal -I/usr/local/pg103/include -Wall -Wmissing-prototypes -Wmissing-declarations -I. -I./ -I/usr/local/pg103/include/postgresql/server -I/usr/local/pg103/include/postgresql/internal -D\_GNU\_SOURCE -c -o repmgr-action-daemon.o repmgr-action-daemon.c

gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Wendif-labels -Wmissing-format-attribute -Wformat-security -fno-strict-aliasing -fwrapv -fexcess-precision=standard -O2 -fPIC -std=gnu89 -I/usr/local/pg103/include/postgresql/internal -I/usr/local/pg103/include -Wall -Wmissing-prototypes -Wmissing-declarations -I. -I./ -I/usr/local/pg103/include/postgresql/server -I/usr/local/pg103/include/postgresql/internal -D\_GNU\_SOURCE -c -o configdata.o configdata.c

gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Wendif-labels -Wmissing-format-attribute -Wformat-security -fno-strict-aliasing -fwrapv -fexcess-precision=standard -O2 -fPIC -std=gnu89 -I/usr/local/pg103/include/postgresql/internal -I/usr/local/pg103/include -Wall -Wmissing-prototypes -Wmissing-declarations -I. -I./ -I/usr/local/pg103/include/postgresql/server -I/usr/local/pg103/include/postgresql/internal -D\_GNU\_SOURCE -c -o configfile.o configfile.c

flex -o'configfile-scan.c' configfile-scan.l

gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Wendif-labels -Wmissing-format-attribute -Wformat-security -fno-strict-aliasing -fwrapv -fexcess-precision=standard -O2 -fPIC -std=gnu89 -I/usr/local/pg103/include/postgresql/internal -I/usr/local/pg103/include -Wall -Wmissing-prototypes -Wmissing-declarations -I. -I./ -I/usr/local/pg103/include/postgresql/server -I/usr/local/pg103/include/postgresql/internal -D\_GNU\_SOURCE -c -o configfile-scan.o configfile-scan.c

gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Wendif-labels -Wmissing-format-attribute -Wformat-security -fno-strict-aliasing -fwrapv -fexcess-precision=standard -O2 -fPIC -std=gnu89 -I/usr/local/pg103/include/postgresql/internal -I/usr/local/pg103/include -Wall -Wmissing-prototypes -Wmissing-declarations -I. -I./ -I/usr/local/pg103/include/postgresql/server -I/usr/local/pg103/include/postgresql/internal -D\_GNU\_SOURCE -c -o log.o log.c

gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Wendif-labels -Wmissing-format-attribute -Wformat-security -fno-strict-aliasing -fwrapv -fexcess-precision=standard -O2 -fPIC -std=gnu89 -I/usr/local/pg103/include/postgresql/internal -I/usr/local/pg103/include -Wall -Wmissing-prototypes -Wmissing-declarations -I. -I./ -I/usr/local/pg103/include/postgresql/server -I/usr/local/pg103/include/postgresql/internal -D\_GNU\_SOURCE -c -o strutil.o strutil.c

gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Wendif-labels -Wmissing-format-attribute -Wformat-security -fno-strict-aliasing -fwrapv -fexcess-precision=standard -O2 -fPIC -std=gnu89 -I/usr/local/pg103/include/postgresql/internal -I/usr/local/pg103/include -Wall -Wmissing-prototypes -Wmissing-declarations -I. -I./ -I/usr/local/pg103/include/postgresql/server -I/usr/local/pg103/include/postgresql/internal -D\_GNU\_SOURCE -c -o controldata.o controldata.c

gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Wendif-labels -Wmissing-format-attribute -Wformat-security -fno-strict-aliasing -fwrapv -fexcess-precision=standard -O2 -fPIC -std=gnu89 -I/usr/local/pg103/include/postgresql/internal -I/usr/local/pg103/include -Wall -Wmissing-prototypes -Wmissing-declarations -I. -I./ -I/usr/local/pg103/include/postgresql/server -I/usr/local/pg103/include/postgresql/internal -D\_GNU\_SOURCE -c -o dirutil.o dirutil.c

gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Wendif-labels -Wmissing-format-attribute -Wformat-security -fno-strict-aliasing -fwrapv -fexcess-precision=standard -O2 -fPIC -std=gnu89 -I/usr/local/pg103/include/postgresql/internal -I/usr/local/pg103/include -Wall -Wmissing-prototypes -Wmissing-declarations -I. -I./ -I/usr/local/pg103/include/postgresql/server -I/usr/local/pg103/include/postgresql/internal -D\_GNU\_SOURCE -c -o compat.o compat.c

gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Wendif-labels -Wmissing-format-attribute -Wformat-security -fno-strict-aliasing -fwrapv -fexcess-precision=standard -O2 -fPIC -std=gnu89 -I/usr/local/pg103/include/postgresql/internal -I/usr/local/pg103/include -Wall -Wmissing-prototypes -Wmissing-declarations -I. -I./ -I/usr/local/pg103/include/postgresql/server -I/usr/local/pg103/include/postgresql/internal -D\_GNU\_SOURCE -c -o dbutils.o dbutils.c

gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Wendif-labels -Wmissing-format-attribute -Wformat-security -fno-strict-aliasing -fwrapv -fexcess-precision=standard -O2 -fPIC -std=gnu89 -I/usr/local/pg103/include/postgresql/internal -I/usr/local/pg103/include -Wall -Wmissing-prototypes -Wmissing-declarations -I. -I./ -I/usr/local/pg103/include/postgresql/server -I/usr/local/pg103/include/postgresql/internal -D\_GNU\_SOURCE -c -o sysutils.o sysutils.c

gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Wendif-labels -Wmissing-format-attribute -Wformat-security -fno-strict-aliasing -fwrapv -fexcess-precision=standard -O2 -fPIC repmgr-client.o repmgr-action-primary.o repmgr-action-standby.o repmgr-action-witness.o repmgr-action-cluster.o repmgr-action-node.o repmgr-action-service.o repmgr-action-daemon.o configdata.o configfile.o configfile-scan.o log.o strutil.o controldata.o dirutil.o compat.o dbutils.o sysutils.o -L/usr/local/pg103/lib -lpgcommon -lpgport -L/usr/local/pg103/lib -lpq -L/usr/local/pg103/lib -Wl,--as-needed -Wl,-rpath,'/usr/local/pg103/lib',--enable-new-dtags -lpgcommon -lpgport -lpthread -lz -lreadline -lrt -lcrypt -ldl -lm -o repmgr

gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Wendif-labels -Wmissing-format-attribute -Wformat-security -fno-strict-aliasing -fwrapv -fexcess-precision=standard -O2 -fPIC -std=gnu89 -I/usr/local/pg103/include/postgresql/internal -I/usr/local/pg103/include -Wall -Wmissing-prototypes -Wmissing-declarations -I. -I./ -I/usr/local/pg103/include/postgresql/server -I/usr/local/pg103/include/postgresql/internal -D\_GNU\_SOURCE -c -o repmgrd.o repmgrd.c

gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Wendif-labels -Wmissing-format-attribute -Wformat-security -fno-strict-aliasing -fwrapv -fexcess-precision=standard -O2 -fPIC -std=gnu89 -I/usr/local/pg103/include/postgresql/internal -I/usr/local/pg103/include -Wall -Wmissing-prototypes -Wmissing-declarations -I. -I./ -I/usr/local/pg103/include/postgresql/server -I/usr/local/pg103/include/postgresql/internal -D\_GNU\_SOURCE -c -o repmgrd-physical.o repmgrd-physical.c

gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Wendif-labels -Wmissing-format-attribute -Wformat-security -fno-strict-aliasing -fwrapv -fexcess-precision=standard -O2 -fPIC repmgrd.o repmgrd-physical.o configdata.o configfile.o configfile-scan.o log.o dbutils.o strutil.o controldata.o compat.o sysutils.o -L/usr/local/pg103/lib -lpgcommon -lpgport -L/usr/local/pg103/lib -lpq -L/usr/local/pg103/lib -Wl,--as-needed -Wl,-rpath,'/usr/local/pg103/lib',--enable-new-dtags -lpgcommon -lpgport -lpthread -lz -lreadline -lrt -lcrypt -ldl -lm -o repmgrd

/usr/bin/mkdir -p '/usr/local/pg103/lib/postgresql'

/usr/bin/mkdir -p '/usr/local/pg103/share/postgresql/extension'

/usr/bin/mkdir -p '/usr/local/pg103/share/postgresql/extension'

/usr/bin/mkdir -p '/usr/local/pg103/bin'

/usr/bin/install -c -m 755 repmgr.so '/usr/local/pg103/lib/postgresql/repmgr.so'

/usr/bin/install -c -m 644 .//repmgr.control '/usr/local/pg103/share/postgresql/extension/'

/usr/bin/install -c -m 644 .//repmgr--unpackaged--4.0.sql .//repmgr--unpackaged--5.1.sql .//repmgr--4.0.sql .//repmgr--4.0--4.1.sql .//repmgr--4.1.sql .//repmgr--4.1--4.2.sql .//repmgr--4.2.sql .//repmgr--4.2--4.3.sql .//repmgr--4.3.sql .//repmgr--4.3--4.4.sql .//repmgr--4.4.sql .//repmgr--4.4--5.0.sql .//repmgr--5.0.sql .//repmgr--5.0--5.1.sql .//repmgr--5.1.sql .//repmgr--5.1--5.2.sql .//repmgr--5.2.sql '/usr/local/pg103/share/postgresql/extension/'

/usr/bin/install -c -m 755 repmgr repmgrd '/usr/local/pg103/bin/'

## 配置主节点postgresql.conf

此处只列举与repmgr相关参数

max\_wal\_senders = 10

max\_replication\_slots = 10

wal\_level = 'replica'

hot\_standby = on

wal\_log\_hints= on

archive\_mode = on

archive\_command = 'cp %p /postgresql/archive/%f'

shared\_preload\_libraries ='repmgr'

注意：另外，如果打算使用pg\_rewind，并且集群没有使用数据校验和初始化，则需要启用 wal\_log\_hints，full\_page\_writes。

重启数据库服务生效

## 配置主节点pg\_hba.conf

# TYPE DATABASE USER ADDRESS METHOD

local all all trust

host all all 0.0.0.0/0 trust

host replication repmgr 192.168.137.0/24 trust

pg\_ctl reload生效

连接数据库，创建repmgr用户和数据库

create user repmgr WITH REPLICATION LOGIN PASSWORD '1qaz@WSX';

create database repmgr owner repmgr;

Alter user repmgr superuser;

## **配置主节点repmgr.conf**

在主服务器上 创建repmgr.conf文件。该文件必须至少包含以下参数：

node\_id=1

node\_name='node01'

conninfo='host=node01 user=repmgr dbname=repmgr connect\_timeout=2 port=1921'

data\_directory='/data/pg103/data'

#日志参数

log\_level=INFO

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

*ssh\_options='-p 3582' #表示ssh协议使用的是3582端口，生产环境的ssh协议端口，一般不会是默认的22端口，因此，需要指定ssh协议的端口号,该端口号用于repmgr standby switchover等操作中。*

注意：repmgr.conf不应将其存储在PostgreSQL数据目录中，因为在设置或重新初始化PostgreSQL服务器时，它可能会被覆盖。

有关 的更多详细信息，请参见“ 配置和配置文件”部分repmgr.conf。

## **配置备节点repmgr.conf**

在备服务器上创建repmgr.conf文件。该文件必须至少包含以下参数：

node\_id=2

node\_name='node02'

conninfo='host=node02 user=repmgr dbname=repmgr connect\_timeout=2 port=1921'

data\_directory='/data/pg103/data'

log\_level=INFO

log\_file='/usr/pgsql-12/repmgr.log'

*ssh\_options='-p 3582' #表示ssh协议使用的是3582端口，生产环境的ssh协议端口，一般不会是默认的22端口，因此，需要指定ssh协议的端口号,该端口号用于repmgr standby switchover等操作中。*

## repmgr.conf存放路径

/etc/repmgr.conf

日志目录

[root@mypg02 ~]# mkdir /repmgr/

[root@mypg02 ~]# chown -R postgres:postgres /repmgr

*参考配置参考：*

*node\_id=1 集群中的标识 注意这个一个集群不能有重复，一般用数字来*

*node\_name='192.168.198.21' 需要给节点一个注册的名字，这里可以使用IP 也可以使用机器名等等*

*conninfo='host=192.168.198.21 dbname=repmgr user=repmgr connect\_timeout=2'*

*#本地连接信息，repmgr 操作本地时的连接信息*

*data\_directory='/pgdata/data' #指定当前主机的数据目录*

*replication\_user='repmgr' #进行复制的操作的账户*

*replication\_type='physical' #复制的方式，默认值，*必须为physical（默认值）

*repmgr\_bindir='/usr/local/postgres/bin' #repmgr 的执行文件的目录*

*pg\_bindir='/usr/local/postgres' #配置PG的执行文件*

*ssh\_options='-p 3582' #表示ssh协议使用的是3582端口，生产环境的ssh协议端口，一般不会是默认的22端口，因此，需要指定ssh协议的端口号,该端口号用于repmgr standby switchover等操作中。*

*另一台机器，需要在 node\_id , node\_name , conninfo 等位置做改变，*

## **注册主节点**

**注意点：**

**若是考试机是云主机，scp的端口未必是22，请根据实际情况合理使用scp命令。**

**前提条件：**

**配置好.pgpass文件的内容，主库和备库的连接信息都需要有。**

**修改pgpass文件的权限：chmod 600 .pgpass**

**如下是一个例子：**

**[pg10@localhost ~]$ chmod 600 .pgpass**

**[pg10@localhost ~]$ cat .pgpass**

**192.168.56.101:1921:repmgr:repmgr:1qaz@WSX**

**192.168.56.102:1921:repmgr:repmgr:1qaz@WSX**

**[pg10@localhost ~]$**

**测试.pgpass文件的方法：**

**psql -d repmgr -U repmgr -p 1921 -h IP地址**

**若是psql能免密登录到数据库中，那就表示.pgpass文件是配置正确的。**

要使repmgr支持复制集群，必须向repmgr注册主节点。

这将安装repmgr 扩展和元数据对象，并添加主服务器的元数据记录。

repmgr -f /etc/repmgr.conf primary register

INFO: connecting to primary database...

NOTICE: attempting to install extension "repmgr"

NOTICE: "repmgr" extension successfully installed

NOTICE: primary node record (ID: 1) registered

验证集群状态，如下所示：

repmgr -f /etc/repmgr.conf cluster show

ID | Name | Role | Status | Upstream | Location | Priority | Timeline | Connection string

----+--------+---------+-----------+----------+----------+----------+----------+---------------------------------------------------------

1 | node01 | primary | \* running | | default | 100 | 13 | host=node01 user=repmgr dbname=repmgr connect\_timeout=2

repmgr元数据表中的记录如下所示：

repmgr=# SELECT \* FROM repmgr.nodes;

node\_id | upstream\_node\_id | active | node\_name | type | location | priority | conninfo

| repluser | slot\_name | config\_file

---------+------------------+--------+-----------+---------+----------+----------+---------------------------------------------------

------+----------+-----------+------------------------------

1 | | t | node01 | primary | default | 100 | host=node01 user=repmgr dbname=repmgr connect\_time

out=2 | repmgr | | /etc/repmgr.conf

(1 row)

复制群集中的每个服务器都有其自己的记录。

如果repmgrd 在使用中，字段upstream\_node\_id，active，type将标记节点的地位或作用，并在发生变化时进行更新。

添加模式

repmgr primary register的过程，将安装repmgr扩展，该扩展将创建一个repmgr模式，其中包含repmgr的元数据表以及其他函数和视图。建议设置repmgr用户的搜索路径来包含这个模式名

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

## **克隆备节点**

**前提条件：**

**配置好.pgpass文件的内容，主库和备库的连接信息都需要有。**

repmgr.conf在备用服务器上 创建一个文件，存放在/usr/local/pg103目录：

node\_id=2

node\_name='node02'

conninfo='host=node02 user=repmgr dbname=repmgr connect\_timeout=2 port=1921'

data\_directory='/data/pg103/data'

使用该--dry-run选项检查备用数据库是否可以克隆：

repmgr -h mypg01 -U repmgr -d repmgr -p 1921 -f /etc/repmgr.conf standby clone --dry-run

NOTICE: destination directory "/data/pg103/data" provided

INFO: connecting to source node

DETAIL: connection string is: host=node01 user=repmgr dbname=repmgr

DETAIL: current installation size is 44 MB

INFO: "repmgr" extension is installed in database "repmgr"

INFO: replication slot usage not requested; no replication slot will be set up for this standby

INFO: parameter "max\_wal\_senders" set to 10

NOTICE: checking for available walsenders on the source node (2 required)

INFO: sufficient walsenders available on the source node

DETAIL: 2 required, 10 available

NOTICE: checking replication connections can be made to the source server (2 required)

INFO: required number of replication connections could be made to the source server

DETAIL: 2 replication connections required

NOTICE: standby will attach to upstream node 1

HINT: consider using the -c/--fast-checkpoint option

INFO: all prerequisites for "standby clone" are met

#表示可以克隆了

注意：上面的-h mypg01 表示-h代表源端数据库。

如果未报告任何问题，则可以使用以下方法克隆备用数据库：

repmgr -h mypg01 -U repmgr -d repmgr -p 1921 -f /etc/repmgr.conf standby clone

---

NOTICE: destination directory "/data/pg103/data" provided

INFO: connecting to source node

DETAIL: connection string is: host=node01 user=repmgr dbname=repmgr

DETAIL: current installation size is 44 MB

INFO: replication slot usage not requested; no replication slot will be set up for this standby

NOTICE: checking for available walsenders on the source node (2 required)

NOTICE: checking replication connections can be made to the source server (2 required)

INFO: creating directory "/data/pg103/data"...

NOTICE: starting backup (using pg\_basebackup)...

HINT: this may take some time; consider using the -c/--fast-checkpoint option

INFO: executing:

pg\_basebackup -l "repmgr base backup" -D /data/pg103/data -h node01 -p 1921 -U repmgr -X stream

NOTICE: standby clone (using pg\_basebackup) complete

NOTICE: you can now start your PostgreSQL server

HINT: for example: pg\_ctl -D /data/pg103/data start

HINT: after starting the server, you need to register this standby with "repmgr standby register"

---

这已node01 使用PostgreSQL的pg\_basebackup实用程序从主数据库克隆了PostgreSQL数据目录文件。

recovery.conf 将自动创建一个包含正确参数的文件，以从该主服务器开始流式传输。

注意

默认情况下，主数据库的数据目录中的所有配置文件都将被复制到备用数据库。

通常，这些将是postgresql.conf， postgresql.auto.conf，pg\_hba.conf和 pg\_ident.conf。

这些可能需要在启动备用数据库之前进行修改。

## 启动注册备节点

启动  
pg\_ctl -D $PGDATA start

注册备节点：

repmgr -f /etc/repmgr.conf standby register

INFO: connecting to local node "mypg02" (ID: 2)

INFO: connecting to primary database

WARNING: --upstream-node-id not supplied, assuming upstream node is primary (node ID 1)

INFO: standby registration complete

NOTICE: standby node "mypg02" (ID: 2) successfully registered

通过repmgr cluster show在备用数据库上执行来检查该节点是否已注册：

注销备节点（参考）

repmgr -f /etc/repmgr.conf standby unregister

[postgres@mypg01 data]$ repmgr -f /etc/repmgr.conf standby unregister

INFO: connecting to local standby

INFO: connecting to primary database

NOTICE: unregistering node 1

INFO: standby unregistration complete

[postgres@mypg01 data]$ repmgr service status

ID | Name | Role | Status | Upstream | repmgrd | PID | Paused? | Upstream last seen

----+--------+---------+-----------+----------+---------+-------+---------+--------------------

2 | mypg02 | primary | \* running | | running | 21059 | no | n/a

## 查看集群状态

repmgr -f /etc/repmgr.conf cluster show

ID | Name | Role | Status | Upstream | Location | Priority | Timeline | Connection string

----+--------+---------+-----------+----------+----------+----------+----------+---------------------------------------------------------

1 | mypg01 | primary | \* running | | default | 100 | 13 | host=mypg01 user=repmgr dbname=repmgr connect\_timeout=2

2 | mypg02 | standby | running | mypg01 | default | 100 | 13 | host=mypg02 user=repmgr dbname=repmgr connect\_timeout=2

这两个节点现在都已向repmgr注册，并且记录已复制到备用服务器。

## 查看流复制状态

连接到主节点并执行：

postgres=# SELECT \* FROM pg\_stat\_replication;

-[ RECORD 1 ]----+------------------------------

pid | 24637

usesysid | 10

usename | postgres

application\_name | mypg02

client\_addr | 192.168.137.130

client\_hostname |

client\_port | 36948

backend\_start | 2020-09-04 14:42:31.349007+08

backend\_xmin |

state | streaming

sent\_lsn | 0/3000288

write\_lsn | 0/3000288

flush\_lsn | 0/3000288

replay\_lsn | 0/3000288

write\_lag |

flush\_lag |

replay\_lag |

sync\_priority | 0

sync\_state | async

*在PostgreSQL 9.6中，您还可以使用视图pg\_stat\_wal\_receiver 检查备用数据库的复制状态。*

*postgres=# SELECT \* FROM pg\_stat\_wal\_receiver;*

*-[ RECORD 1 ]---------+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------*

*pid | 18504*

*status | streaming*

*receive\_start\_lsn | 0/15000000*

*receive\_start\_tli | 13*

*received\_lsn | 0/15000340*

*received\_tli | 13*

*last\_msg\_send\_time | 2020-06-14 11:18:37.935258+08*

*last\_msg\_receipt\_time | 2020-06-14 11:18:37.949053+08*

*latest\_end\_lsn | 0/15000340*

*latest\_end\_time | 2020-06-14 11:12:07.031845+08*

*slot\_name |*

*conninfo | user=repmgr passfile=/home/postgres/.pgpass connect\_timeout=2 dbname=replication host=mypg01 port=1921 application\_name=mypg02 fallback\_application\_name=walreceiver sslmode=disable sslcompression=1 target\_session\_attrs=any*

*请注意，该conninfo值是在recovery.conf中生成的值，与在repmgr.conf-设置的conninfo中设置的值略有不同;*

*除其他外，它将包含连接节点的名称application\_name。*

可以在主节点创建表对象，插入数据的方式验证流复制是否正常。

## **配置两节点相互信任**

步骤1: 分别在主备节点上使用postgres用户执行ssh-keygen 创建公钥和密钥

ssh-keygen -t rsa

# 然后一直按 [Enter] 键

ll /home/postgres/.ssh/

 # 在用户的家目录$home/.ssh/ 路径下就会有2个文件生成，id\_rsa私钥，id\_rsa.pub公钥

步骤2: 用 ssh-copy-id 把公钥复制到对端主机上

ssh-copy-id -i /home/postgres/.ssh/id\_rsa.pub [postgres@192.168.137.130](mailto:root@192.168.137.130)

# 将自己的id\_rsa.pub导入到远程主机192.168.137.130的/root/.ssh/authorized\_key 里面

[注: ssh-copy-id 把密钥追加到远程主机的 .ssh/authorized\_key 上.]

ssh-copy-id -i /home/postgres/.ssh/id\_rsa.pub [root@192.168.137.12](mailto:root@192.168.137.130)9或

ssh-copy-id [postgres@192.168.137.12](mailto:root@192.168.137.130)9

步骤3: 主备节点相互验证

systemctl restart sshd

ssh 192.168.137.130

ssh 192.168.137.129

# 就可以直接登录，不需要输入密码

## 手动切换验证

手动切换需在备节点执行以下命令

repmgr -f /etc/repmgr.conf standby switchover -U postgres --verbose

注意：/etc/repmgr.conf的读写权限和属主

[postgres@mypg02 ~]$ ll /etc/repmgr.conf

-rw-r--r-- 1 postgres postgres 163 Aug 26 17:01 /etc/repmgr.conf

问题1：

WARNING: following problems with command line parameters detected:

database connection parameters not required when executing STANDBY SWITCHOVER

NOTICE: executing switchover on node "mypg02" (ID: 2)

INFO: searching for primary node

INFO: checking if node 1 is primary

INFO: current primary node is 1

INFO: SSH connection to host "mypg01" succeeded

ERROR: unable to execute "repmgr" on "mypg01"

HINT: check "pg\_bindir" is set to the correct path in "repmgr.conf"; current value: (not set)

解决方法：添加设置变量pg\_bindir

pg\_bindir='/usr/local/pg103/bin'

以下正常手动切换

[postgres@mypg02 ~]$ repmgr -f /etc/repmgr.conf standby switchover -U repmgr --verbose

NOTICE: using provided configuration file "/usr/local/pg103/repmgr.conf"

WARNING: following problems with command line parameters detected:

database connection parameters not required when executing STANDBY SWITCHOVER

NOTICE: executing switchover on node "mypg02" (ID: 2)

INFO: searching for primary node

INFO: checking if node 1 is primary

INFO: current primary node is 1

INFO: SSH connection to host "mypg01" succeeded

INFO: archive mode is "off"

INFO: replication lag on this standby is 0 seconds

NOTICE: local node "mypg02" (ID: 2) will be promoted to primary; current primary "mypg01" (ID: 1) will be demoted to standby

NOTICE: stopping current primary node "mypg01" (ID: 1)

NOTICE: issuing CHECKPOINT on node "mypg01" (ID: 1)

DETAIL: executing server command "/usr/local/pg103/bin/pg\_ctl -D '/data/pg103/data' -W -m fast stop"

INFO: checking for primary shutdown; 1 of 60 attempts ("shutdown\_check\_timeout")

INFO: checking for primary shutdown; 2 of 60 attempts ("shutdown\_check\_timeout")

NOTICE: current primary has been cleanly shut down at location 0/15007588

NOTICE: promoting standby to primary

DETAIL: promoting server "mypg02" (ID: 2) using "/usr/local/pg103/bin/pg\_ctl -w -D '/data/pg103/data' promote"

waiting for server to promote.... done

server promoted

NOTICE: waiting up to 60 seconds (parameter "promote\_check\_timeout") for promotion to complete

INFO: standby promoted to primary after 0 second(s)

NOTICE: STANDBY PROMOTE successful

DETAIL: server "mypg02" (ID: 2) was successfully promoted to primary

INFO: local node 1 can attach to rejoin target node 2

DETAIL: local node's recovery point: 0/15007588; rejoin target node's fork point: 0/150075F8

NOTICE: setting node 1's upstream to node 2

WARNING: unable to ping "host=mypg01 user=repmgr dbname=repmgr connect\_timeout=2"

DETAIL: PQping() returned "PQPING\_NO\_RESPONSE"

NOTICE: starting server using "/usr/local/pg103/bin/pg\_ctl -w -D '/data/pg103/data' start"

NOTICE: NODE REJOIN successful

DETAIL: node 1 is now attached to node 2

INFO: node "mypg01" (ID: 1) is pingable

INFO: node "mypg01" (ID: 1) has attached to its upstream node

NOTICE: node "mypg02" (ID: 2) promoted to primary, node "mypg01" (ID: 1) demoted to standby

NOTICE: switchover was successful

DETAIL: node "mypg02" is now primary and node "mypg01" is attached as standby

NOTICE: STANDBY SWITCHOVER has completed successfully

**查看集群状态**

[postgres@mypg01 ~]$ repmgr -f /etc/repmgr.conf cluster show

ID | Name | Role | Status | Upstream | Location | Priority | Timeline | Connection string

----+--------+---------+-----------+----------+----------+----------+----------+-------------------------------------------------------------

1 | mypg01 | standby | running | mypg02 | default | 100 | 1 | host=mypg01 user=postgres dbname=postgres connect\_timeout=2

2 | mypg02 | primary | \* running | | default | 100 | 2 | host=mypg02 user=postgres dbname=postgres connect\_timeout=2

## 配置主备节点自动切换参数

vim /etc/repmgr.conf

failover='automatic'

promote\_command='repmgr standby promote -f /etc/repmgr.conf'

follow\_command='repmgr standby follow -f /etc/repmgr.conf'

*#promote\_command='/home/postgres/repmgr standby promote -f /etc/repmgr.conf'*

*#follow\_command='/home/postgres/repmgr/repmgr standby follow' -f /etc/repmgr.conf*

*failover参数有两个  
automatic：表示开启故障自动切换  
manual：不开启故障自动切换*

不开启故障自动切换，备节点检测到主节点故障后，备节点不会自动升级为主节点。

## 查看repmgrd服务状态

前面搭建好了简单的repmgr集群，这时查看集群和repmgr服务状态，可知repmgrd并未运行

[postgres@mypg02 data]$ repmgr -f /etc/repmgr.conf service status

ID | Name | Role | Status | Upstream | repmgrd | PID | Paused? | Upstream last seen

----+--------+---------+-----------+----------+-------------+-----+---------+--------------------

1 | mypg01 | primary | \* running | | not running | n/a | n/a | n/a

2 | mypg02 | standby | running | mypg01 | not running | n/a | n/a | n/a

## 主备节点开启集群进程repmgrd

*手动启动：*

*repmgrd -d -f /etc/repmgr.conf -p /tmp/repmgrd.pid*

*手动停止*

*kill `cat /tmp/repmgrd.pid`。*

repmgrd的PID文件

repmgrd默认会生成一个PID文件。

主节点repmgr日志：

[2020-09-03 13:37:13] [NOTICE] repmgrd (repmgrd 5.2dev) starting up

[2020-09-03 13:37:13] [INFO] connecting to database "host=mypg01 user=repmgr dbname=repmgr connect\_timeout=2"

INFO: set\_repmgrd\_pid(): provided pidfile is /tmp/repmgrd.pid

[2020-09-03 13:37:13] [NOTICE] starting monitoring of node "mypg01" (ID: 1)

[2020-09-03 13:37:13] [INFO] "connection\_check\_type" set to "ping"

[2020-09-03 13:37:13] [NOTICE] monitoring cluster primary "mypg01" (ID: 1)

[2020-09-03 13:37:13] [INFO] child node "mypg02" (ID: 2) is attached

备节点repmgr日志：

[2020-09-03 13:38:08] [NOTICE] repmgrd (repmgrd 5.2dev) starting up

[2020-09-03 13:38:08] [INFO] connecting to database "host=mypg02 user=repmgr dbname=repmgr connect\_timeout=2"

INFO: set\_repmgrd\_pid(): provided pidfile is /tmp/repmgrd.pid

[2020-09-03 13:38:08] [NOTICE] starting monitoring of node "mypg02" (ID: 2)

[2020-09-03 13:38:08] [INFO] "connection\_check\_type" set to "ping"

[2020-09-03 13:38:08] [INFO] monitoring connection to upstream node "mypg01" (ID: 1)

## 查看服务状态

[postgres@mypg01 ~]$ repmgr service status

ID | Name | Role | Status | Upstream | repmgrd | PID | Paused? | Upstream last seen

----+--------+---------+-----------+----------+---------+------+---------+--------------------

1 | mypg01 | primary | \* running | | running | 5047 | no | n/a

2 | mypg02 | standby | running | mypg01 | running | 4839 | no | 0 second(s) ago

## 自动切换验证

此时模拟主节点故障（直接执行关闭主节点）

pg\_ctl stop -D $PGDATA

备节点repmgr日志：

[2020-09-03 13:41:29] [WARNING] unable to ping "host=mypg01 user=repmgr dbname=repmgr connect\_timeout=2"

[2020-09-03 13:41:29] [DETAIL] PQping() returned "PQPING\_NO\_RESPONSE"

[2020-09-03 13:41:29] [WARNING] unable to connect to upstream node "mypg01" (ID: 1)

[2020-09-03 13:41:29] [INFO] checking state of node "mypg01" (ID: 1), 1 of 6 attempts

[2020-09-03 13:41:29] [WARNING] unable to ping "user=repmgr connect\_timeout=2 dbname=repmgr host=mypg01 fallback\_application\_name=repmgr"

[2020-09-03 13:41:29] [DETAIL] PQping() returned "PQPING\_NO\_RESPONSE"

[2020-09-03 13:41:29] [INFO] sleeping 10 seconds until next reconnection attempt

[2020-09-03 13:41:39] [INFO] checking state of node "mypg01" (ID: 1), 2 of 6 attempts

[2020-09-03 13:41:39] [WARNING] unable to ping "user=repmgr connect\_timeout=2 dbname=repmgr host=mypg01 fallback\_application\_name=repmgr"

[2020-09-03 13:41:39] [DETAIL] PQping() returned "PQPING\_NO\_RESPONSE"

[2020-09-03 13:41:39] [INFO] sleeping 10 seconds until next reconnection attempt

[2020-09-03 13:41:49] [INFO] checking state of node "mypg01" (ID: 1), 3 of 6 attempts

[2020-09-03 13:41:49] [WARNING] unable to ping "user=repmgr connect\_timeout=2 dbname=repmgr host=mypg01 fallback\_application\_name=repmgr"

[2020-09-03 13:41:49] [DETAIL] PQping() returned "PQPING\_NO\_RESPONSE"

[2020-09-03 13:41:49] [INFO] sleeping 10 seconds until next reconnection attempt

[2020-09-03 13:41:59] [INFO] checking state of node "mypg01" (ID: 1), 4 of 6 attempts

[2020-09-03 13:41:59] [WARNING] unable to ping "user=repmgr connect\_timeout=2 dbname=repmgr host=mypg01 fallback\_application\_name=repmgr"

[2020-09-03 13:41:59] [DETAIL] PQping() returned "PQPING\_NO\_RESPONSE"

[2020-09-03 13:41:59] [INFO] sleeping 10 seconds until next reconnection attempt

[2020-09-03 13:42:09] [INFO] checking state of node "mypg01" (ID: 1), 5 of 6 attempts

[2020-09-03 13:42:09] [WARNING] unable to ping "user=repmgr connect\_timeout=2 dbname=repmgr host=mypg01 fallback\_application\_name=repmgr"

[2020-09-03 13:42:09] [DETAIL] PQping() returned "PQPING\_NO\_RESPONSE"

[2020-09-03 13:42:09] [INFO] sleeping 10 seconds until next reconnection attempt

[2020-09-03 13:42:19] [INFO] checking state of node "mypg01" (ID: 1), 6 of 6 attempts

[2020-09-03 13:42:19] [WARNING] unable to ping "user=repmgr connect\_timeout=2 dbname=repmgr host=mypg01 fallback\_application\_name=repmgr"

[2020-09-03 13:42:19] [DETAIL] PQping() returned "PQPING\_NO\_RESPONSE"

[2020-09-03 13:42:19] [WARNING] unable to reconnect to node "mypg01" (ID: 1) after 6 attempts

[2020-09-03 13:42:19] [INFO] 0 active sibling nodes registered

[2020-09-03 13:42:19] [INFO] primary node "mypg01" (ID: 1) and this node have the same location ("default")

[2020-09-03 13:42:19] [INFO] no other sibling nodes - we win by default

[2020-09-03 13:42:19] [NOTICE] this node is the only available candidate and will now promote itself

[2020-09-03 13:42:19] [INFO] promote\_command is:

"repmgr standby promote -f /etc/repmgr.conf"

NOTICE: promoting standby to primary

DETAIL: promoting server "mypg02" (ID: 2) using "/usr/local/pg103/bin/pg\_ctl -w -D '/data/pg103/data' promote"

NOTICE: waiting up to 60 seconds (parameter "promote\_check\_timeout") for promotion to complete

NOTICE: STANDBY PROMOTE successful

DETAIL: server "mypg02" (ID: 2) was successfully promoted to primary

[2020-09-03 13:42:20] [INFO] 0 followers to notify

[2020-09-03 13:42:20] [INFO] switching to primary monitoring mode

[2020-09-03 13:42:20] [NOTICE] monitoring cluster primary "mypg02" (ID: 2)

可知备节点正确升级为主节点提供服务！！

## 查看集群状态

[postgres@mypg02 ~]$ repmgr service status

ID | Name | Role | Status | Upstream | repmgrd | PID | Paused? | Upstream last seen

----+--------+---------+-----------+----------+---------+------+---------+--------------------

1 | mypg01 | primary | - failed | ? | n/a | n/a | n/a | n/a

2 | mypg02 | primary | \* running | | running | 4839 | no | n/a

## 手动加入原主节点

以下操作并没有手动重启主节点，如果手动重启主节点，可能会发生双主脑裂现象。

[postgres@mypg01 data]$ repmgr -h mypg02 -U postgres -d postgres node rejoin

INFO: local node 1 can attach to rejoin target node 2

DETAIL: local node's recovery point: 0/300E720; rejoin target node's fork point: 0/300E790

NOTICE: setting node 1's upstream to node 2

WARNING: unable to ping "host=mypg01 user=postgres dbname=postgres connect\_timeout=2"

DETAIL: PQping() returned "PQPING\_NO\_RESPONSE"

NOTICE: starting server using "/usr/local/pg103/bin/pg\_ctl -w -D '/data/pg103/data' start"

NOTICE: NODE REJOIN successful

DETAIL: node 1 is now attached to node 2

[postgres@mypg01 data]$ repmgr service status

ID | Name | Role | Status | Upstream | repmgrd | PID | Paused? | Upstream last seen

----+--------+---------+-----------+----------+---------+-------+---------+--------------------

1 | mypg01 | standby | running | mypg02 | running | 24888 | no | 0 second(s) ago

2 | mypg02 | primary | \* running | | running | 21059 | no | n/a

## *脑裂场景*

*如果此时直接将原来的主节点重新启动会如何？*

*注意不要在生产环境这么操作，会引起脑裂问题！！！*

*[postgres@mypg01 ~]$ repmgr service status*

*ID | Name | Role | Status | Upstream | repmgrd | PID | Paused? | Upstream last seen*

*----+--------+---------+----------------------+----------+---------+------+---------+--------------------*

*1 | mypg01 | primary | \* running | | running | 5047 | no | n/a*

*2 | mypg02 | standby | ! running as primary | | running | 4839 | no | n/a*

*WARNING: following issues were detected*

*- node "mypg02" (ID: 2) is registered as standby but running as primary*

*[postgres@mypg02 ~]$ repmgr service status*

*ID | Name | Role | Status | Upstream | repmgrd | PID | Paused? | Upstream last seen*

*----+--------+---------+-----------+----------+---------+------+---------+--------------------*

*1 | mypg01 | primary | ! running | | running | 5047 | no | n/a*

*2 | mypg02 | primary | \* running | | running | 4839 | no | n/a*

*WARNING: following issues were detected*

*- node "mypg01" (ID: 1) is running but the repmgr node record is inactive*

*原主节点repmgr日志显示：*

*[2020-09-03 13:44:58] [DETAIL] attempted to connect using:*

*user=repmgr connect\_timeout=2 dbname=repmgr host=mypg01 fallback\_application\_name=repmgr options=-csearch\_path=*

*[2020-09-03 13:44:58] [WARNING] reconnection to node "mypg01" (ID: 1) failed*

*[2020-09-03 13:44:58] [WARNING] unable to ping "host=mypg01 user=repmgr dbname=repmgr connect\_timeout=2"*

*[2020-09-03 13:44:58] [DETAIL] PQping() returned "PQPING\_NO\_RESPONSE"*

*[2020-09-03 13:44:58] [ERROR] unable to determine if server is in recovery*

*[2020-09-03 13:44:58] [DETAIL] query text is:*

*SELECT pg\_catalog.pg\_is\_in\_recovery()*

*[2020-09-03 13:44:58] [WARNING] unable to determine node recovery status*

*[2020-09-03 13:45:00] [INFO] attempting to reconnect to node "mypg01" (ID: 1)*

*[2020-09-03 13:45:00] [INFO] reconnected to node "mypg01" (ID: 1)*

*INFO: set\_repmgrd\_pid(): provided pidfile is /tmp/repmgrd.pid*

*[2020-09-03 13:45:00] [NOTICE] reconnected to primary node after 160 seconds, resuming monitoring*

*[2020-09-03 13:45:02] [NOTICE] standby node "mypg02" (ID: 2) has disconnected*

*[2020-09-03 13:47:20] [INFO] monitoring primary node "mypg01" (ID: 1) in normal state*

*此时两边都可以创建表对象管理数据，主备节点生成两条时间线 timeline!!!*

## 原主节点恢复

方法一：参照新主节点克隆备节点

[postgres@mypg01 data]$ repmgr -h mypg02 -U repmgr -d repmgr -f /etc/repmgr.conf standby clone

NOTICE: destination directory "/data/pg103/data" provided

INFO: connecting to source node

DETAIL: connection string is: host=mypg02 user=repmgr dbname=repmgr

DETAIL: current installation size is 30 MB

ERROR: repmgr extension is available but not installed in database "repmgr"

HINT: check that you are cloning from the database where "repmgr" is installed

[postgres@mypg01 data]$ repmgr -h mypg02 -U postgres -d postgres -f /etc/repmgr.conf standby clone

NOTICE: destination directory "/data/pg103/data" provided

INFO: connecting to source node

DETAIL: connection string is: host=mypg02 user=postgres dbname=postgres

DETAIL: current installation size is 30 MB

INFO: replication slot usage not requested; no replication slot will be set up for this standby

NOTICE: checking for available walsenders on the source node (2 required)

NOTICE: checking replication connections can be made to the source server (2 required)

INFO: checking and correcting permissions on existing directory "/data/pg103/data"

NOTICE: starting backup (using pg\_basebackup)...

HINT: this may take some time; consider using the -c/--fast-checkpoint option

INFO: executing:

/usr/local/pg103/bin/pg\_basebackup -l "repmgr base backup" -D /data/pg103/data -h mypg02 -p 1921 -U postgres -X stream

NOTICE: standby clone (using pg\_basebackup) complete

NOTICE: you can now start your PostgreSQL server

HINT: for example: pg\_ctl -D /data/pg103/data start

HINT: after starting the server, you need to re-register this standby with "repmgr standby register --force" to update the existing node record

[postgres@mypg01 data]$ pg\_ctl start

waiting for server to start....2020-09-07 13:40:07.537 CST [45633] LOG: listening on IPv4 address "0.0.0.0", port 1921

2020-09-07 13:40:07.537 CST [45633] LOG: listening on IPv6 address "::", port 1921

2020-09-07 13:40:07.540 CST [45633] LOG: listening on Unix socket "/tmp/.s.PGSQL.1921"

...2020-09-07 13:40:11.570 CST [45634] LOG: database system was interrupted; last known up at 2020-09-07 13:37:20 CST

2020-09-07 13:40:11.624 CST [45635] FATAL: the database system is starting up

2020-09-07 13:40:11.653 CST [45636] FATAL: the database system is starting up

2020-09-07 13:40:11.669 CST [45634] LOG: entering standby mode

2020-09-07 13:40:11.707 CST [45634] LOG: redo starts at 0/4000060

2020-09-07 13:40:11.709 CST [45634] LOG: consistent recovery state reached at 0/4000168

2020-09-07 13:40:11.709 CST [45633] LOG: database system is ready to accept read only connections

2020-09-07 13:40:11.771 CST [45640] LOG: started streaming WAL from primary at 0/5000000 on timeline 6

done

server started

[postgres@mypg01 data]$ repmgr standby register --force

INFO: connecting to local node "mypg01" (ID: 1)

INFO: connecting to primary database

INFO: standby registration complete

NOTICE: standby node "mypg01" (ID: 1) successfully registered

[postgres@mypg01 data]$ repmgr service status

ID | Name | Role | Status | Upstream | repmgrd | PID | Paused? | Upstream last seen

----+--------+---------+-----------+----------+---------+-------+---------+--------------------

1 | mypg01 | standby | running | mypg02 | running | 24888 | no | 1 second(s) ago

2 | mypg02 | primary | \* running | | running | 21059 | no | n/a

扩展知识：物理复制槽

在没有启用 replication slots 的环境中，如果碰到 ERROR: requested WAL segment xxxx has already been removed 的错误，解决办法是要么提前开启了归档，要么重做slave，另外还需要在master上设置 wal\_keep\_segments 为更大的值。

主备断连的一种情况是因为主备lag持续增大造成备库需要的wal日志被主库清理，当然主库日志清理不止这一种情况，可能还会有vacuum等原因。那么我们可以通过物理复制槽来规避这个问题。

下面看看如何使用物理复制槽？

在主库创建物理复制槽：

SELECT \* FROM pg\_create\_physical\_replication\_slot('slot\_s1');

*查看复制槽*

*select \* from pg\_replication\_slots;*

*删除复制槽*

*select \* from pg\_drop\_replication\_slot('slot\_s1');*

备库使用该物理复制槽：

vi recovey.conf，新增如下配置：

primary\_slot\_name='slot\_s1'

再次加延迟进行压测：

*tc qdisc add dev bond0 root netem delay 1s*

我们观察主库的wal日志目录大小，发现目录已经超过1G，并且在持续增长，备库需要的日志在累积。

但是复制槽的使用一定要注意风险，一定要做好redo日志个数或者大小的监控，方式主备同步异常造成主库日志堆积。

参考链接：

<https://cloud.tencent.com/developer/article/1631594>

<https://www.scalingpostgres.com/tutorials/postgresql-replication-slots/>

|  |
| --- |
| # Assumes Replication is already setup as follows:  # Primary DB cluster called "main" on port 5432  # Replica DB cluster called "replica" on port 5433  # create replication slot on main  sudo su - postgres  psql -c "select \* from pg\_create\_physical\_replication\_slot('replica');"  psql -c "select \* from pg\_replication\_slots;"  # Update recovery.conf on replicanano /var/lib/postgresql/10/replica/recovery.conf  restore\_command = 'cp /var/lib/postgresql/pg\_log\_archive/replica/%f %p'  recovery\_target\_timeline = 'latest'  standby\_mode = 'on'  primary\_conninfo = 'user=rep\_user passfile=''/var/lib/postgresql/.pgpass'' host=''/var/run/postgresql'' port=5432 sslmode=prefer sslcompression=1 krbsrvname=postgres target\_session\_attrs=any'  archive\_cleanup\_command = 'pg\_archivecleanup /var/lib/postgresql/pg\_log\_archive/replica %r'  primary\_slot\_name = 'replica'  # Restart replica to start using the new slot  sudo systemctl restart postgresql@10-replica  tail -n 100 /var/log/postgresql/postgresql-10-replica.log  # check slot status on main  psql -c "select \* from pg\_replication\_slots;"  # insert data on main  psql test -c "insert into posts (id, title, content, type) values  (102, 'Intro to SQL Where Clause', 'Easy as pie!', 'SQL'),  (103, 'Intro to SQL Order Clause', 'What comes first?', 'SQL');"  # verify data added to replica  psql test -c "select \* from posts;" -p 5433  # Stop main cluster to simulate failure  sudo systemctl stop postgresql@10-main  # promote replica  sudo pg\_ctlcluster 10 replica promote  # verify replica is now a master / primary cluster  tail -n 100 /var/log/postgresql/postgresql-10-replica.log  # show replica cluster slots (now primary)  psql -c "select \* from pg\_replication\_slots;" -p 5433  # use this command to drop a slot  psql -c "select \* from pg\_drop\_replication\_slot('replica');" |

***方法二：pg\_rewind使主备节点时间线一致后，执行node rejoin重建主备关系***

关闭原主节点

[postgres@mypg01 data]$ pg\_ctl stop

***原主节点使用pg\_rewind参照新主节点修正时间线***

*pg\_rewind -D '/data/pg103/data' --source-server='host=mypg02 user=repmgr dbname=repmgr connect\_timeout=2 port=1921'-P --dry-run*

*pg\_rewind -D '/data/pg103/data' --source-server='host=mypg02 user=postgres dbname=postgres connect\_timeout=2 port=1921’-P*

*原主节点查看状态*

*[postgres@mypg01 data]$ repmgr service status*

*ID | Name | Role | Status | Upstream | repmgrd | PID | Paused? | Upstream last seen*

*----+--------+---------+----------------------+----------+---------+-------+---------+--------------------*

*1 | mypg01 | primary | ! running as standby | | running | 24888 | no | 0 second(s) ago*

*2 | mypg02 | primary | \* running | | running | 21059 | no | n/a*

*WARNING: following issues were detected*

*- node "mypg01" (ID: 1) is registered as an inactive primary but running as standby*

*指定备节点的跟随的主节点*

*repmgr standby follow -f /etc/repmgr.conf*

*启动原主节点*

*[postgres@mypg01 data]$ pg\_ctl start*

*waiting for server to start....2020-09-03 15:03:23.159 CST [7459] LOG: listening on IPv4 address "0.0.0.0", port 1921*

*2020-09-03 15:03:23.159 CST [7459] LOG: listening on IPv6 address "::", port 1921*

*2020-09-03 15:03:23.163 CST [7459] LOG: listening on Unix socket "/tmp/.s.PGSQL.1921"*

*2020-09-03 15:03:23.466 CST [7459] LOG: redirecting log output to logging collector process*

*2020-09-03 15:03:23.466 CST [7459] HINT: Future log output will appear in directory "log".*

*done*

*server started*

*原主节点状态*

*[postgres@mypg01 data]$ repmgr service status*

*ID | Name | Role | Status | Upstream | repmgrd | PID | Paused? | Upstream last seen*

*----+--------+---------+----------------------+----------+---------+------+---------+--------------------*

*1 | mypg01 | primary | ! running as standby | | running | 5047 | no | 1 second(s) ago*

*2 | mypg02 | primary | \* running | | running | 4839 | no | n/a*

*WARNING: following issues were detected*

*- node "mypg01" (ID: 1) is registered as an inactive primary but running as standby*

*将原主节点重新加入集群*

*[postgres@mypg01 data]$ repmgr -f /etc/repmgr.conf node rejoin -d 'host=mypg02 user=repmgr dbname=repmgr connect\_timeout=2 port=1921‘*

*INFO: timelines are same, this server is not ahead*

*DETAIL: local node lsn is 0/10088280, rejoin target lsn is 0/10088280*

*NOTICE: setting node 2's upstream to node 1*

*WARNING: unable to ping "host=mypg02 user=repmgr dbname=repmgr connect\_timeout=2 port=1921"*

*DETAIL: PQping() returned "PQPING\_NO\_RESPONSE"*

*NOTICE: starting server using "pg\_ctl -w -D '/data/pg103/data' start"*

*NOTICE: NODE REJOIN successful*

*DETAIL: node 2 is now attached to node 1*

*[postgres@mypg01 data]$*