# Patroni部署文档

目录

[Patroni部署文档 1](#_Toc104456010)

[软件包安装需求： 3](#_Toc104456011)

[安装包： 3](#_Toc104456012)

[Ip规划： 3](#_Toc104456013)

[安装步骤（所有节点）： 4](#_Toc104456014)

[1. 系统环境准备 4](#_Toc104456015)

[2. 配置sudo权限： 4](#_Toc104456016)

[3. Pg数据库安装配置： 4](#_Toc104456017)

[4. ETCD集群安装配置 7](#_Toc104456018)

[5. patroni集群安装配置 8](#_Toc104456019)

[6. 连接方式 12](#_Toc104456020)

## 软件包安装需求：

--all nodes

1. 标准配置 firewalld selinux chrony gcc libpq5-devel
2. postgreSQL数据库安装：使用源码安装或者yum 安装

postgresql12 postgresql12-server postgresql12-libs postgresql12-contrib

1. patroni软件需要 etcd python3 python3-devel

## 安装包：

Etcd(3.3.11)：



Patroni(2.12)：



Postgresql(12.10)：



Patroni安装依赖包：



## IP规划：

10.209.31.20 node1(postgres,etcd,partroni)

10.209.31.26 node2(postgres,etcd,partroni)

10.209.31.32 node3(postgres,etcd,partroni)

10.209.31.186 读写vip

## 安装步骤（所有节点）：

### 系统环境准备

关闭防火墙：

systemctl stop firewalld

systemctl disable firewalld

关闭selinux

sed -i 's/SELINUX=.\*/SELINUX=disabled/g' /etc/selinux/config

setenforce 0

时间配置：

yum install -y ntpdate  
ntpdate time.windows.com && hwclock -w

### 配置sudo权限：

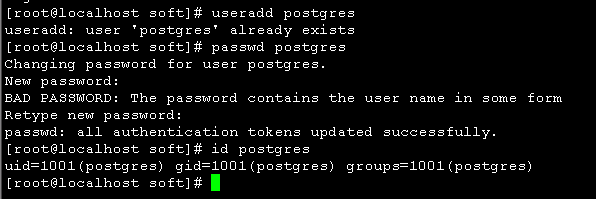
vi /etc/sudoers



### pg数据库安装配置（root操作）：

#### 上传并解压pg安装包：

tar -zxvf postgresql-12.0.tar.gz



#### 调整用户限制

vim /etc/security/limits.conf

\* soft nofile 655360

\* hard nofile 655360

\* soft nproc 655360

\* hard nproc 655360

\* soft memlock unlimited

\* hard memlock unlimited

\* soft core unlimited

\* hard core unlimited

\* soft stack unlimited

\* hard stack unlimited

#### 调整内核参数

vim /etc/sysctl.conf

fs.aio-max-nr=1048576

fs.file-max=76724600

fs.nr\_open=20480000

kernel.sem=4096 2147483647 2147483646 512000

kernel.shmall=5033164

kernel.shmmax=20615843030

kernel.shmmni=819200

net.core.netdev\_max\_backlog=10000

net.core.wmem\_max = 21299200

net.core.rmem\_max = 21299200

net.core.wmem\_default = 21299200

net.core.rmem\_default = 21299200

net.core.somaxconn=65535

net.ipv4.tcp\_max\_tw\_buckets=262144

net.ipv4.tcp\_tw\_reuse = 1

net.ipv4.tcp\_tw\_recycle = 1

net.ipv4.tcp\_keepalive\_time = 60

net.ipv4.tcp\_keepalive\_probes = 3

net.ipv4.tcp\_keepalive\_intvl = 20

net.ipv4.tcp\_retries1=5

net.ipv4.tcp\_syn\_retries=5

net.ipv4.tcp\_synack\_retries=5

net.ipv4.tcp\_retries2=12

net.ipv4.tcp\_rmem = 8192 250000 16777216

net.ipv4.tcp\_wmem = 8192 250000 16777216

net.ipv4.tcp\_syncookies = 1

net.ipv4.ip\_local\_port\_range=40000 65535

net.ipv4.tcp\_fin\_timeout=10

net.ipv4.tcp\_max\_syn\_backlog=65535

net.ipv4.tcp\_mem=8388608 12582912 16777216

net.ipv4.tcp\_timestamps=1

net.ipv4.tcp\_sack=1

vm.swappiness=0

vm.zone\_reclaim\_mode=0

vm.numa\_balancing=0

vm.extfrag\_threshold = 500

vm.mim\_free\_kbytes=1048576

vm.overcommit\_memory = 2

vm.overcommit\_ratio=90

vm.dirty\_ratio=10

vm.dirty\_background\_ratio=5

vm.dirty\_writeback\_centisecs=80

vm.mmap\_min\_addr=65536

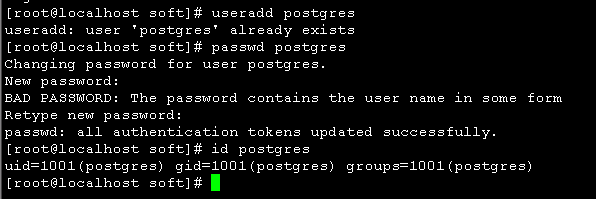
vm.nr\_hugepages=0

使用sysctl -p 生效。

#### 添加postgres 用户并修改密码

[root@localhost soft]# useradd postgres

[root@localhost soft]# passwd postgres



#### 编译安装：

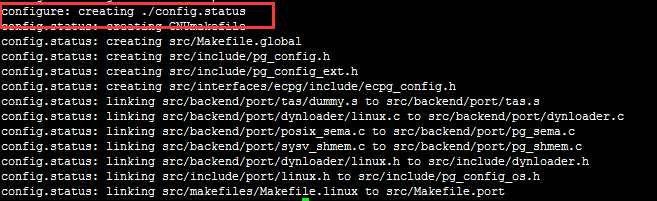
[root@localhost ~]# mkdir -p /home/data/pgsql/

[root@localhost ~]# mkdir -p /home/data/pgdata/

[root@localhost postgresql-12.10]#

./configure --prefix=/home/data/pgsql/ --with-perl --with-libxml --with-libxslt

最后几行出现 createing ./config.status 编译成功



[root@ localhost postgresql-12.10]# gmake && gmake install



#### 初始化：

使用patroni软件自动初始化数据库。

### ETCD集群安装配置（root操作）：

#### 上传etcd安装包，使用rpm安装服务：

rpm -ivh etcd-3.3.11-2.el7.centos.x86\_64.rpm

#### 配置软件参数：

vi /etc/etcd/etcd.conf

ETCD\_DATA\_DIR="/var/lib/etcd/etcd1" # 存放etcd数据的目录，自定义

ETCD\_LISTEN\_PEER\_URLS="监听URL，用于和其他节点通信

ETCD\_LISTEN\_CLIENT\_URLS="监听URL，用户客户端和SERVER进行通信

ETCD\_NAME="etcd1" # etcd名称，自定义

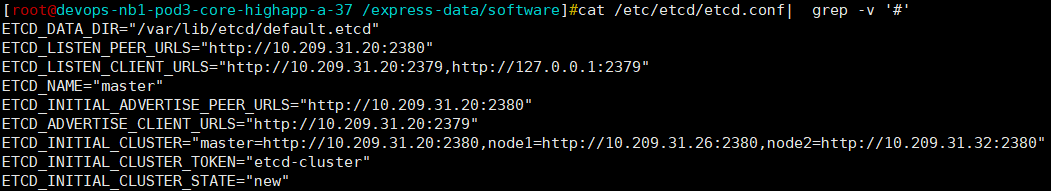
ETCD\_INITIAL\_ADVERTISE\_PEER\_URLS="告知集群其他节点，端口2380用于集群通信

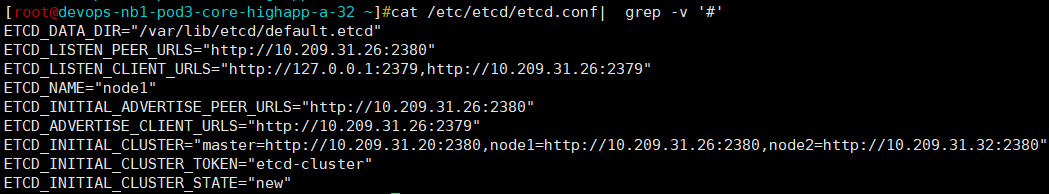
ETCD\_ADVERTISE\_CLIENT\_URLS="告知客户端自身的URL，TCP 2379端口用于监听客户端请求

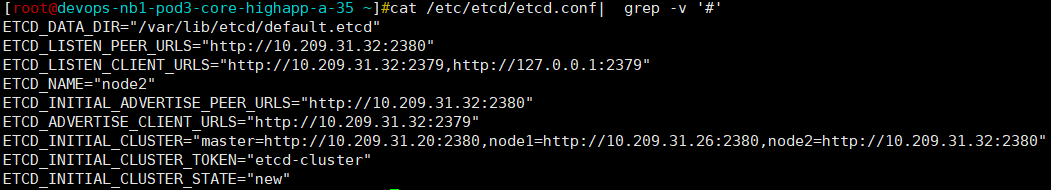
ETCD\_INITIAL\_CLUSTER="etcd1=定义了集群内所有成员

ETCD\_INITIAL\_CLUSTER\_TOKEN="etcd-cluster" # 集群ID，唯一标识

ETCD\_INITIAL\_CLUSTER\_STATE="new" # 集群状态，new为新创建集群，existing为已经存在的集群





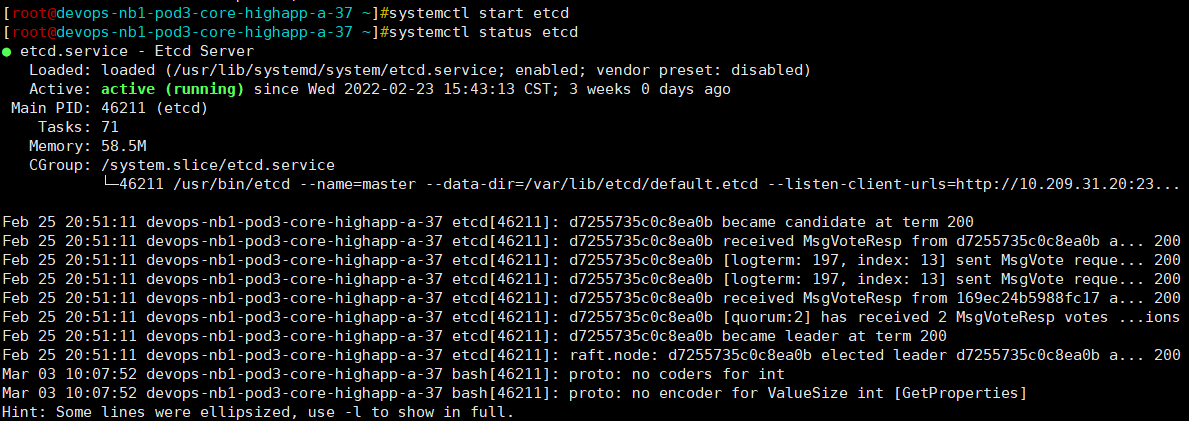


#### 启动etcd服务

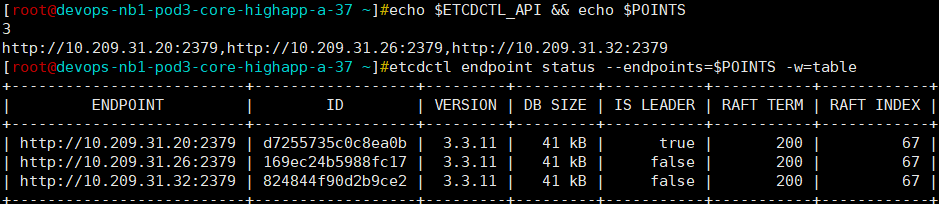
systemctl start etcd

systemctl status etcd（status没有报错就为正常启动）

systemctl enable etcd



#### 查看etcd成员状态



一定要注意选举周期(raft term )为相同的周期（数字）

### patroni集群安装配置（root操作）：

#### 上传patroni安装包以及依赖包，使用rpm安装服务：

rpm -ivh python3-cdiff-1.0-1.rhel7.noarch.rpm

rpm -ivh python36-psutil-5.6.7-1.el7.x86\_64.rpm

rpm -ivh python3-psycopg2-2.8.6-1.rhel7.x86\_64.rpm

rpm -ivh python3-ydiff-1.2-10.rhel7.noarch.rpm

rpm -ivh python36-PyYAML-3.13-1.el7.x86\_64.rpm

rpm -ivh python36-six-1.14.0-3.el7.noarch.rpm

rpm -ivh python36-prettytable-0.7.2-19.el7.noarch.rpm

rpm -ivh python36-dateutil-2.4.2-5.el7.noarch.rpm

rpm -ivh libtommath-0.42.0-6.el7.x86\_64.rpm

rpm -ivh libtommath-devel-0.42.0-6.el7.x86\_64.rpm

rpm -ivh libtomcrypt-1.17-26.el7.x86\_64.rpm

rpm -ivh python36-crypto-2.6.1-16.el7.x86\_64.rpm

rpm -ivh python3-etcd-0.4.5-20.rhel7.noarch.rpm

rpm -ivh python36-dns-1.16.0-1.el7.noarch.rpm

rpm -ivh python36-certifi-2018.10.15-5.el7.noarch.rpm

rpm -ivh patroni-etcd-2.1.2-1.rhel7.x86\_64.rpm

rpm -ivh python36-click-6.7-8.el7.noarch.rpm

rpm -ivh libpq5-14.1-42PGDG.rhel7.x86\_64.rpm

rpm -ivh python36-pysocks-1.6.8-7.el7.noarch.rpm

rpm -ivh python36-urllib3-1.25.6-2.el7.noarch.rpm

rpm -ivh patroni-2.1.2-1.rhel7.x86\_64.rpm

#### 配置软件启动参数：

vi /usr/lib/systemd/system/patroni.service

[Unit]

Description=Runners to orchestrate a high-availability PostgreSQL

After=syslog.target network.target etcd.service

[Service]

Type=simple

User=postgres

Group=postgres

EnvironmentFile=-/etc/patroni/patroni\_env.conf

ExecStartPre=-/usr/bin/sudo /sbin/modprobe softdog

ExecStartPre=-/usr/bin/sudo /bin/chown postgres /dev/watchdog

ExecStart=/usr/bin/env patroni /etc/patroni/patroni.yml

ExecReload=/bin/kill -s HUP $MAINPID

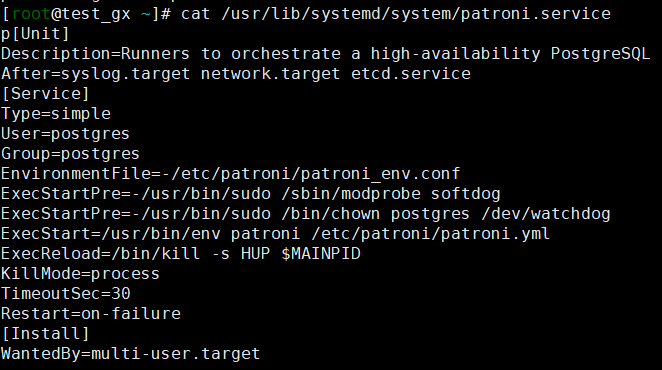
KillMode=process

TimeoutSec=30

Restart=on-failure

[Install]

WantedBy=multi-user.target



修改之后reload 配置：systemctl daemon-reload

#### 配置软件参数:

cat /etc/patroni/patroni.yml (文件格式是yaml文本，不支持使用tab，会报错)

scope: pgsql

namespace: /service/ #etcd中键值位置

name: pg20 #patroni名字，每个节点不一样

restapi:

listen: 0.0.0.0:8008 #保持默认，监听所有的8088端口

connect\_address: 10.209.31.20:8008 #本地连接通信

etcd3: #这里使用etcdv3，默认etcdv2，默认写入的键值都是不可见的

hosts: 10.209.31.20:2379,10.209.31.26:2379,10.209.31.32:2379 #etcd地址

bootstrap:

dcs:

ttl: 30

loop\_wait: 10

retry\_timeout: 300

maximum\_lag\_on\_failover: 1048576 #延迟超过设定值不允许failover

master\_start\_timeout: 300

synchronous\_mode: true #同步复制

synchronous\_mode\_strict:true

postgresql:

use\_pg\_rewind: true

use\_slots: true

parameters:

listen\_addresses: "0.0.0.0"

port: 5432

wal\_level: replica

hot\_standby: "on"

wal\_keep\_segments: 100

max\_wal\_senders: 10

max\_replication\_slots: 10

wal\_log\_hints: "on"

archive\_mode: "on"

archive\_timeout: 1800s

archive\_command: mkdir -p /express-data/pg12/wal\_archive && test ! -f /express-data/pg12/wal\_archive/%f && cp %p ../wal\_archive/%f

recovery\_conf:

restore\_command: cp /express-data/pg12/wal\_archive/%f %p

initdb: #初始化参数配置

- encoding: UTF8

- locale: C

- lc-ctype: zh\_CN.UTF-8

- data-checksums

pg\_hba: #流复制用户和链接白名单设置

- host replication postgres 10.209.31.0/24 md5

- host all all 0.0.0.0/0 md5

postgresql:

listen: 0.0.0.0:5432

connect\_address: 10.209.31.20:5432 # 连接pgsql服务的配置,这里不能使用127.0.0.1,pg\_basebackup需要远程连接主库进行在线复制

data\_dir: /express-data/pg12/pgdata # $PGDATA

bin\_dir: /express-data/pg12/pgsql/bin # $PGHOME/bin

authentication:

replication:

username: postgres

password: "1qaz@WSX.."

superuser:

username: postgres

password: "1qaz@WSX.."

rewind:

username: postgres

password: "1qaz@WSX.."

basebackup:

max-rate: 100M

checkpoint: fast

callbacks: # 本次配置没有使用haproxy+keepalived实现VIP切换和负载均衡，因为callbacks方式更快速，对系统资源消耗更小，操作更简单

on\_start: /bin/bash /etc/patroni/patroni\_callback.sh

on\_stop: /bin/bash /etc/patroni/patroni\_callback.sh

on\_role\_change: /bin/bash /etc/patroni/patroni\_callback.sh

watchdog: # 使用linux自带的软件watchdog监控patroni的服务持续性

mode: automatic # Allowed values: off, automatic, required

device: /dev/watchdog # watchdog设备，/dev/watchdog和/dev/watchdog0等同，可能存在兼容性区别

safety\_margin: 5

tags:

nofailover: false #是否执行自动切换

noloadbalance: false #是否开启负载均衡

clonefrom: false

nosync: false

（其他节点修改name，ip地址即可）

#### 创建patroni\_callbak脚本

# 脚本的开头传入了三个变量，但是在patroni.yml文件中我们并没有传入任何的变量，实际测试过程中发现由patroni服务默认传入三个变量

$1 - action, patroni触发的动作,stop/start/on\_role\_change/restart/reload

$2 - role, 当前节点的角色，master/{slave|replica}

$3 - scope, 作用范围，pgsql服务

# 当节点角色为主，使用ip addr命令绑定VIP地址

# 当节点角色为备，使用ip addr命令解绑VIP地址

$ vi /etc/patroni/patroni\_callback.sh

#!/bin/bash

readonly action=$1

readonly role=$2

readonly scope=$3

function usage() {

echo "Usage: $0 <on\_start|on\_stop|on\_role\_change> <role> <scope>"

exit 1

}

echo "this is patroni callback $action $role $scope"

case $cb\_name in

on\_stop)

sudo ip addr del 10.209.31.186/24 dev eth1 label eth1:1

;;

on\_start)

;;

on\_role\_change)

if [[ $role == 'master' ]]; then

# 绑定VIP

sudo ip addr add 10.209.31.186/24 brd 10.209.31.255 dev eth1 label eth1:1

# 监测VIP冲突,并屏蔽冲突的IP

sudo arping -q -A -c 1 -I eth1 10.209.31.186

else

sudo ip addr del 10.209.31.186/24 dev eth1 label eth1:1

fi

;;

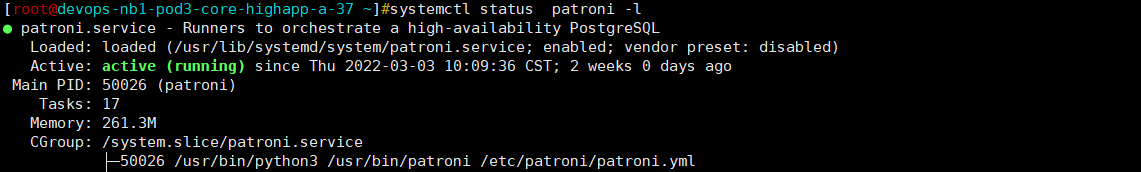
\*)

usage

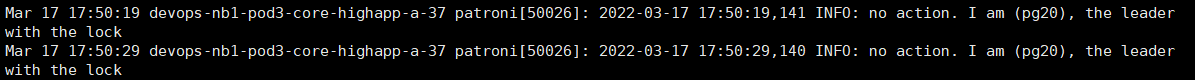
;;

esac

#### 启动patroni服务

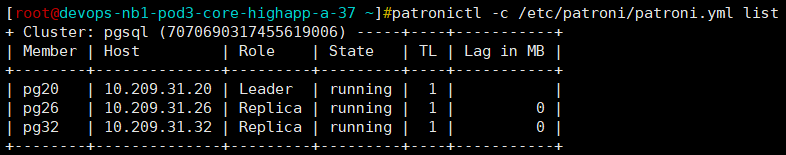


首次运行服务会初始化集群，不存在目录需要手动创建。显示到日志为角色状态时，即为正常。



#### 查看patroni集群状态

patronictl -c /etc/patroni/patroni.yml list



注意观察时间线（TL）是否一致。

### 连接方式

目前pgjdbc和libpq驱动可以在连接字符串中配置多个IP，由驱动识别数据库的主备角色，连接合适的节点。

#### JDBC：

JDBC的多主机URL功能全面，支持failover，读写分离和复制均衡。可以通过参数配置不同的连接策略。

连接主节点(实际是可写的节点)。当出现"双主"甚至"多主"连接第一个发现的可用的主节点- jdbc:postgresql://10.209.31.20:5432,10.209.31.26:5432,10.209.31.32:5432/postgres?targetServerType=primary

优先连接备节点，无可用备节点时连接主节点，有多个可用备节点时随机连接其中一个。

  - jdbc:postgresql://10.209.31.20:5432,10.209.31.26:5432,10.209.31.32:5432?targetServerType=preferSecondary&loadBalanceHosts=true

随机连接任意一个可用的节点

- jdbc:postgresql://10.209.31.20:5432,10.209.31.26:5432,10.209.31.32:5432/postgres?targetServerType=any&loadBalanceHosts=true

#### Libpq：

libpq的多主机URL功能相对pgjdbc弱一点，只支持failover。

  连接主节点(实际是可写的节点)

- postgres://10.209.31.20:5432,10.209.31.26:5432,10.209.31.32:5432/postgres?target\_session\_attrs=read-write

  连接任意节点

- postgres://10.209.31.20:5432,10.209.31.26:5432,10.209.31.32:5432/postgres?target\_session\_attrs=any

## 使用pg\_rman进行备份

#### 上传并解压安装包：

tar -zxvf pg\_rman-1.3.14-pg12.tar.gz



执行make && make install



#### 查看数据库设置

vi $PGDATA/postgresql.conf

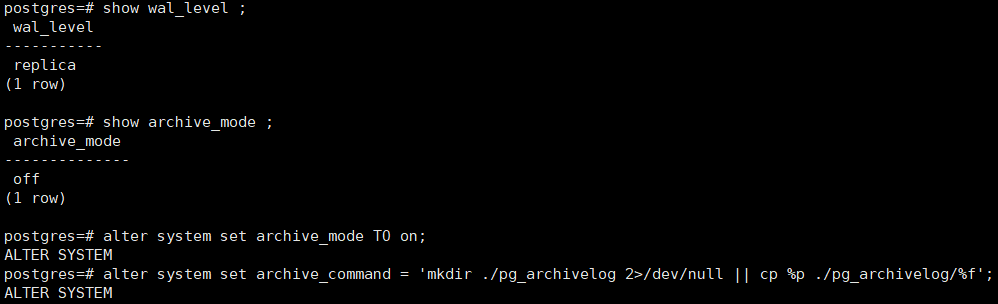
wal\_level = 'replica'

archive\_mode = on

archive\_command = 'mkdir ./pg\_archivelog 2>/dev/null || cp %p ./pg\_archivelog/%f'

archive\_timeout = 120

修改后需要重启数据库



#### 初始化pg\_rman

export BACKUP\_PATH=/home/postgres/pg\_rman\_bakdir/

export ARCLOG\_PATH=/database/pg12/pg\_archivelog/

pg\_rman -B $BACKUP\_PATH init



#### 编辑备份策略

修改pg\_rman.ini 文件

增加策略参数：

COMPRESS\_DATA=YES

KEEP\_ARCLOG\_FILES=7

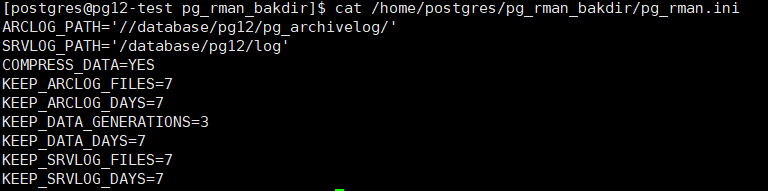
KEEP\_ARCLOG\_DAYS=7

KEEP\_DATA\_GENERATIONS=3

KEEP\_DATA\_DAYS=7

KEEP\_SRVLOG\_FILES=7

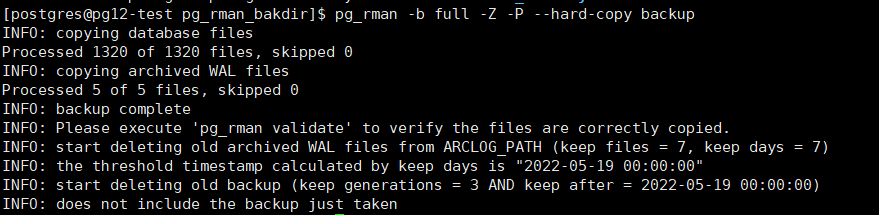
KEEP\_SRVLOG\_DAYS=7



#### 执行备份

全量备份：

pg\_rman -b full -Z -P --hard-copy backup



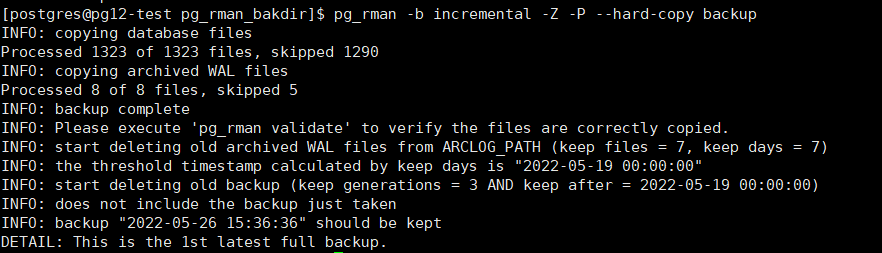
校验备份：

pg\_rman validate



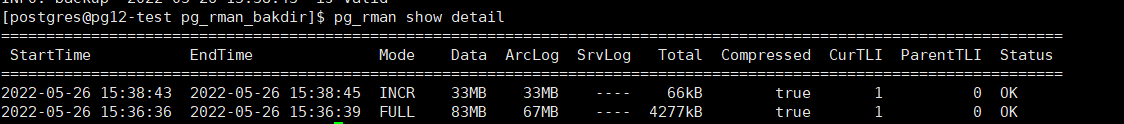
增量备份：

pg\_rman -b incremental -Z -P --hard-copy backup



查看备份：

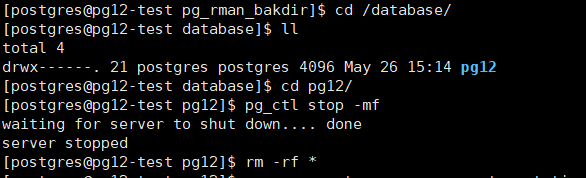
pg\_rman show detail



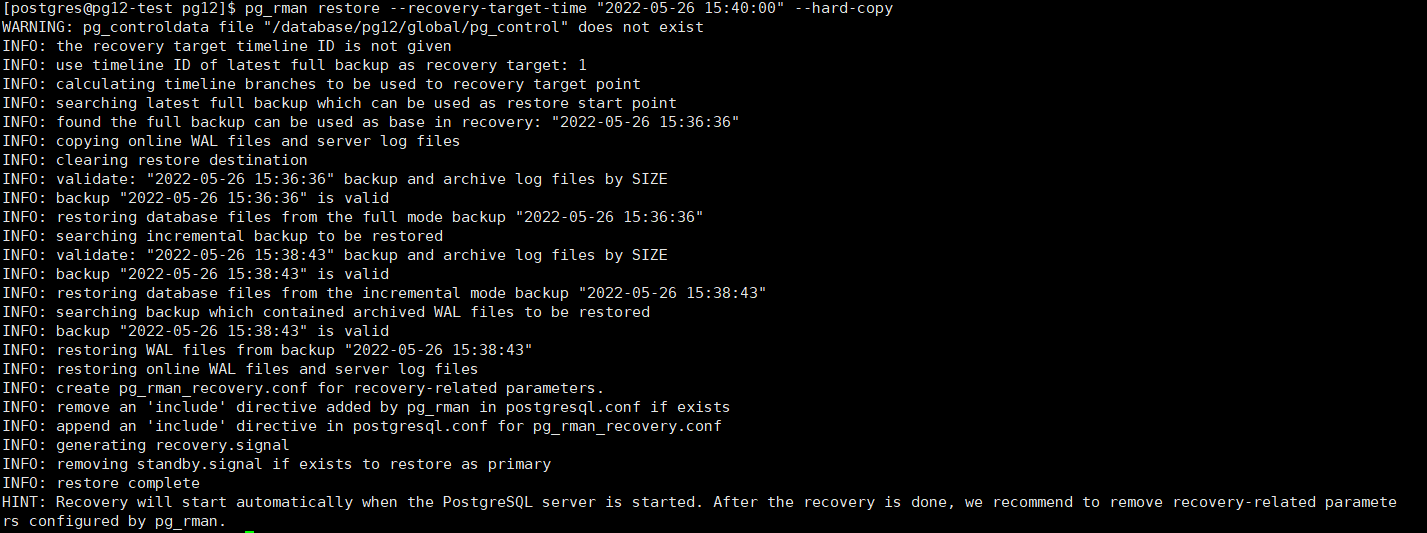
一次增量，一次全量。

#### 删除原库，进行恢复：

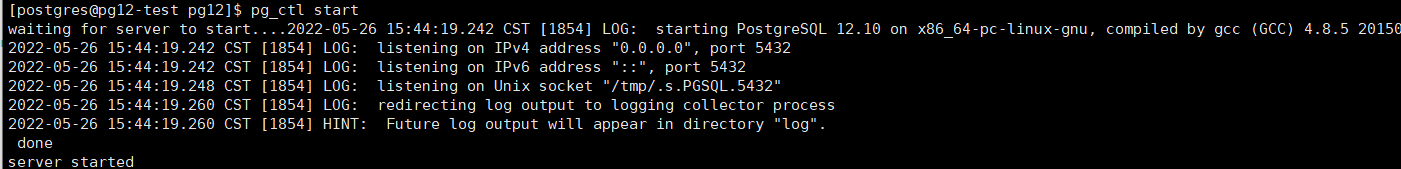
停止数据库并删除所有数据库文件：

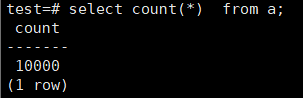


pg\_rman restore --recovery-target-time "2022-05-26 15:40:00" --hard-copy



启动数据库并校验数据：





数据恢复成功。