Postgres Skill Development on the Cloud

Venkat Valdmani

Linux Shell Commands

The Linux shell has several commands, often used by DBA to perform day to day Postgres Database Administration. The detailed usage of the commands is obtained by the man command.

man cat

NAME
cat - concatenate and print files
SYNOPSIS
cat [**-belnstuv**] [file ...]

cat file1
will print the contents of file1 to the standard output

1 cat

The post.txt has two lines below. With cat command, you can view the contents of the file.

cat post.txt

Michael Stonebraker created postgres in 1997.

Larry Ellison created Oracle in 1987

2 grep

The command grep is a very often used command, which searches for a string or number in a file. The grep command searches for a pattern in a file or in the output of a program.

The following file has 2 lines, where 1st line refers to Michael and 2-line refers to Larry. With the grep command, you can search for the occurrences of Michael Stonebraker

cat post.txt

```
$ cat post.txt

 Micheal Stonebraker create postgres in 1997.
Larry Ellison created Oracle in 1987
```

```
$ grep Micheal post.txt
 Micheal Stonebraker create postgres in 1997.
```

3 sed command

Sed command substitutes the string pattern. Following command substitutes lowercase c with C.

grep post.txt | sed ‘s/create/Created/g’

```
$ grep Micheal post.txt | sed 's/create/Created/g'
Micheal Stonebraker Created postgres in 1997.
```

4 ps command

When you connect to the Linux Operating System, the OS assigns a unique identification number called Process Identification (PID), to the connected user. The root is the first process to be created, which is assigned number 1, where other child processes are created from the parent process number1. The process command with -ef argument, with pipe and grep postgres command displays the following screen image. The parent process is the third column of the ps -ef command.

```
[postgres@node2 ~]$ ps -ef | grep postgres
postgres    3114       1  0 05:24 ?        00:00:00 /usr/pgsql-13/bin/postgres -D /appl/postgres/dev
postgres    3115    3114  0 05:24 ?        00:00:00 postgres: logger
postgres    3117    3114  0 05:24 ?        00:00:00 postgres: checkpointer
postgres    3118    3114  0 05:24 ?        00:00:00 postgres: background writer
postgres    3119    3114  0 05:24 ?        00:00:00 postgres: walwriter
postgres    3120    3114  0 05:24 ?        00:00:00 postgres: autovacuum launcher
postgres    3121    3114  0 05:24 ?        00:00:00 postgres: archiver last was 000000020000000000000B2
postgres    3122    3114  0 05:24 ?        00:00:00 postgres: stats collector
postgres    3123    3114  0 05:24 ?        00:00:00 postgres: logical replication launcher
```

As you notice in the above exhibit, parent process is 1(root) which is a parent of process number 3114(postgres server), which spawns child processes 3115,3117,3118,3119,3120,3121,3122 and

3123 for database background tasks. If you terminate process 3144, for any issues related to Postgres to restart the server, all child processes will terminate automatically.

5 nohup bg command

You can run a job in the background on the server, which allows you to disconnect from the terminal, while the job is running in the background, where the running status of job is not terminated when you disconnect from the server session. You can submit any job such as shell script in the operating system and may disconnect from your client machine.

Following is the command to submit a background job.

nohup ./load.sh &

6 fg command

The fg command changes the background jobs to run in the foreground.

7 shell redirection 1

The shell redirects the output to file. The redirection 1 is to display results in the output file.

./load.sh 1> load.txt &

[postgres@node2 ~]$ cat load.txt

[postgres@node2 ~]$ cat load.txt

processed …100000 rows

8 shell redirection 2

The shell 2 redirection is to capture the errors in the file

./load.sh 2> load.txt &

[postgres@node2 ~] $ cat load.txt

ERROR: could not extend file "base/13436/16396": No space left on device

HINT: Check free disk space.

9 Full nohup command

If you want to capture output of the jobs as well as errors, you can use the below command. As a matter of fact, most of the Linux jobs are run the below fashion, which allows you to disconnect from the terminal, as well as to capture successful and error status of the jobs.

nohup ./load.sh 2>&1 load.txt &

10 awk command

The awk command is a very powerful text processing tool in Linux that reads input data, either from file or from a program line by line, allowing you to specify patterns for data manipulation. It can print specific fields (columns) using variables like $1, $2, etc., representing the values in each line.

```
[postgres@node2 ~]$ cat post.txt
Michael Storebraker create postgres in 1997
Larry Elison created Oracle in 1987
Obama Was President of USA
[postgres@node2 ~]$ cat post.txt | awk '{print $1}'
Micheal
Larry
Obama
[postgres@node2 ~]$ cat post.txt | awk '{print $1$2}'
Michael Stonebraker
Larry Elison
ObamaWas
```

With awk command, you can search and print the field positions in the files or in the process status output. The following command displays the data directory of the postgres cluster.

```
$ ps -ef | grep -i postmaster | grep D | awk '{print $10}'
/var/lib/pgsql/13/data/
```

11 wc

The wc command counts the words in file or output of a program.

```
[postgres@node2 ~]$ wc post.txt
  3  17 108 post.txt
3 lines, 17 words with size of 108 bytes
[postgres@node2 ~]$ ls -l post.txt
-rw-r--r--. 1 postgres 108 Jul 25 23:33 post.txt
[postgres@node2 ~]$ wc -l post.txt
3 post.txt
```

12 echo

The echo command displays the screen output from the command. You have to specify the output values in double quotes.

[postgres@node2 ~]$ echo "Postgres Tutorial”

Postgres Tutorial

13 Files and File Systems

While working with PostgreSQL on Linux, you need to have a reasonable working knowledge of files, file systems, and its security access controls, which is called Access Control Lists.

Understanding the concepts about file permissions, ownership, and directory structures is essential for proper installation, configuration, and management of PostgreSQL.

In Linux, files are the basic units for storing data, which are organized within a file system. The file system provides direction to the Operating System to store, access and manage data on the storage devices. The storage for the file systems resides in storage arrays provisioned by vendors such as EMC or IBM. File systems in Linux are organized hierarchically using directories, forming a tree-like structure. Beginning with the top level "/root" directory, other directories branch out from it. Users can create directories, sub-directories, and can navigate through the file system by moving up and down the directory tree.

14 File permissions

The file security is governed by grant on files through Read, Write, and Execute permissions, with the octal command 4 for r(read), 2 for w(write), 1 for e(execute) and 0 for no access.

The permission is granted by chmod command, such as chmod 755 file name. Each file has three numbers, with the first number reserved for the file owner, the next number for the group owner, and the last number for others. A typical file is represented as 755, where the owner has read, write, and execute permissions, and others have read and execute permissions.

Following is the example for permission on test file, where user postgres has permission of 7 and group and others has the permission of 5.

You can long list the file with ls -l command, ls -l testfile.

-rwxr-r-x 1 postgres 0 Jul 16 18:20 testfile

The file system security access can be in the range starting with 100 to 777, where 1 translates to execute by owner and no permissions to group and others, and 0 stands to no permissions to group and users.

Note: In some situations, owner has 4 number representation for setiud command.

You can change the permission with the chmod command.

```
[postgres@node2 ~]$ ls -l post.txt
-rw-r--r--. 1 postgres 108 Jul 25 23:33 post.txt
[postgres@node2 ~]$ chmod 777 post.txt
[postgres@node2 ~]$ ls -l post.txt
-rwxrwxrwx. 1 postgres 108 Jul 25 23:33 post.txt
```

Every user is a part of a group, where the file system permissions are commonly shared among the members of the group. For instance, postgres user belongs to postgres group.

This account is used for your general system activities and is separate from the "postgres" account. You should ensure that no other regular users are added to the "postgres" group. The "postgres" group should be dedicated only to the "postgres" user account, as it is intended for system-level database management purposes. Allowing other users into this group could potentially compromise the security and integrity of the PostgreSQL installation.

-rwxr-xr-x 1 postgres 0 Jul 16 18:20 testfile

15 mkdir

The first step after connecting to Linux is to create directories for storage data in an organized manner. The command to create a directory is mkdir. With -p option you can create directories

recursively. The following example is to create directories such as backup/yearly/monthly/weekly with mkdir -p command.

```
$ mkdir -p backup/yearly/monthly/weekly
```

16 pwd

The pwd command prints the path of the current working directory.

```
$ pwd
/appl/postgres
```

17 cd command

With cd command you can navigate between directories. With the command cp backup, you can navigate to backup command.

```
/appl/postgres
$ cd scripts
```

18 touch command

With touch command you can create an empty file

```
$ echo "Test data" > backupfile
$ cat backupfile
```

19 cp command

```
$ cp backupfile backupfile_old
```

With cp command, you can copy from one directory to another directory or in same directory.

20 mv command

With mv command you can move one file or directory to another name

```
$ mv backupfile_old backup_original
```

21 list command

The ls is the most used command to obtain attributes of the file such as owner, file creation time, and the size. The ls -l is for log listing of the file.

21.1 ls -l

```
$ ls -l
total 40664
-rwxr-xr-x. 1 postgres postgres    24246 Apr 13 04:25 advanced.sh
-rw-r--r--. 1 postgres postgres       10 Apr 16 02:45 backupfile
-rw-r--r--. 1 postgres postgres       10 Apr 16 02:46 backup_original
```

The command ls -ltr is to display files created by date.

21.2 ls -ltr

```
-rwxr-xr-x. 1 postgres postgres    24246 Apr 13 04:25 advanced.sh
-rw-r--r--. 1 postgres postgres       10 Apr 16 02:45 backupfile
-rw-r--r--. 1 postgres postgres       10 Apr 16 02:46 backup_original
```

21.3 ls -R

The command -R displays the Recursive files in or directories.

```
$ ls -R
.:
advanced.sh  backupfile       bur.sh       envmon.conf.new  main.sh         redshift.sh
backup       backup_original  envmon.conf  envmon.conf.org  my_scripts.cvf  salesq101242817.gz

./backup:
books.dump.20231211_20_43    finance.dump.20240412_05_46        postgres.dump.20231211_19_26  sales.dump.20231219_02_02
finance.dump.20240412_05_31  finance.dump.20240412_06_07        sales1.dump.20240124_03_18    sales.dump.20231231_18_43
finance.dump.20240412_05_34  full_database.dump.20240128_02_42  sales.dump.20231211_19_27     salesq1.dump.20240128_02_45
```

21.4 ls -a hidden files

There are hidden files in the Operating System with. prefix. With ls -a command, you can display the hidden files. The following is the example to list .bash files, which are executed during the user login process.

```
$ ls -a  | grep .bash
.bash_history
.bash_profile
```

With list command you can list files with several options such as by creation time etc

22 chown

The user is the absolute owner of the file, whereas to grant access to groups and others, you have to user chown command on a file. In the following example, the user postgres with group postgres is granted full control on the directory /appl/postgres

```
chown postgres:postgres /appl/postgres
```

23 find command.

The command find is especially useful to work on several maintenance activities.

The find command with -xdev and -size flags, identified files large than 1M blocks. Depending on the nature of the file, or requirement, you can delete or move the file to a different location.

```
$ find . -xdev -size +100000
./16/data/base/5/16393
```

Another useful command is to find files that were created 30 minutes ago.

```
$ find . -type f -mmin -30 -exec ls -lrt {} \;
-rw-r--r--. 1 postgres postgres 10 Apr 16 02:45 ./scripts/backupfile
-rw-r--r--. 1 postgres postgres 10 Apr 16 02:46 ./scripts/backup_original
```

With -exec command, you can execute all tasks such as listing or removing the files. In the below example, in the backup directory delete all files created 30 minutes ago.

```
$ find . -type f -mmin -30 -exec rm {} \;
$ find . -type f -mmin -30 -exec ls -lrt {} \;
```

With the find command you can find the file attributes such as file creation time. You can remove old log or trace files with the find command.

24 tail command

With tail command, you can view the last lines of the file.

```
$ tail postgresql-Tue.log
2024-04-16 02:17:34.496 UTC [1028] LOG:  listening on IPv6 address "::1", port 5432
2024-04-16 02:17:34.496 UTC [1028] LOG:  listening on IPv4 address "127.0.0.1", port 5432
2024-04-16 02:17:34.503 UTC [1028] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
2024-04-16 02:17:34.514 UTC [1028] LOG:  listening on Unix socket "/tmp/.s.PGSQL.5432"
```

With the -f flag, you can monitor the status of a file that is being constantly changed.

```
$ tail -f postgresql-Tue.log
2024-04-16 02:17:34.496 UTC [1028] LOG:  listening on IPv6 address "::1", port 5432
2024-04-16 02:17:34.496 UTC [1028] LOG:  listening on IPv4 address "127.0.0.1", port 5432
2024-04-16 02:17:34.503 UTC [1028] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
2024-04-16 02:17:34.514 UTC [1028] LOG:  listening on Unix socket "/tmp/.s.PGSQL.5432"
2024-04-16 02:17:34.524 UTC [1045] LOG:  database system was interrupted; last known up at 2024-04-13 08:06:00 UTC
2024-04-16 02:17:34.582 UTC [1045] LOG:  database system was not properly shut down; automatic recovery in progres
2024-04-16 02:17:34.584 UTC [1045] LOG:  redo starts at 0/15D6030
2024-04-16 02:17:34.584 UTC [1045] LOG:  invalid record length at 0/15D6118: wanted 24, got 0
2024-04-16 02:17:34.584 UTC [1045] LOG:  redo done at 0/15D60E0
2024-04-16 02:17:34.597 UTC [1028] LOG:  database system is ready to accept connections
2024-04-16 03:05:47.079 UTC [2058] ERROR:  syntax error at or near "alskdfjlsdjf" at character 1
2024-04-16 03:05:47.079 UTC [2058] STATEMENT:  alskdfjlsdjf;
```

25 mknod and mkfifo command

The FIFO stands as first in and first out, a named pipe file used for inter process communication With mkfifo command, one process can read whereas other process can write to same file, which is primarily used in backup of large database along with compression. The first step is to create a file with mkfifo command, which is used to read as well as compress in the background. The file created by mkfifo is used as backupfile from pg_dump backup utility. As you can read and write to fifo, you can write as well as compress the file in the background. Following is the simple script to backup and compress at the same time. After completion of the backup, the fifo file is deleted.

```
export bkpath=/appl/postgres/backup
export dd=$(date +%y%m%d)
export PGHOME=/usr/pgsql-13/bin
```

```
export db=sales
 mkfifo $bkpath/$db.dump.$dd
    gzip < $bkpath/$db. dump.$dd > $bkpath/$db.dump.$dd.gz &
    $PGHOME/pg_dump -f $bkpath/$db.dump.$dd $db
 ret=$?
    echo $ret
   if [ $ret -eq 0 ]
     then
       rm $bkpath/$db.dump.$dd
     else
        mailx -s "check   backup error from `host`"
   fi
```

You can trace the shell script with -x command.

Following is the shell script running with -x option

```
[postgres@node2 ~]$ sh -x bk.sh
+ export bkpath=/appl/postgres/backup
+ bkpath=/appl/postgres/backup
++ date +%y%m%d
+ export dd=230726
+ dd=230726
+ export PGHOME=/usr/pgsql-13/bin
+ PGHOME=/usr/pgsql-13/bin
+ export db=sales
+ db=sales
+ mkfifo /appl/postgres/backup/sales.dump.230726
+ /usr/pgsql-13/bin/pg_dump -f /appl/postgres/backup/sales.dump.230726 sales
+ gzip
+ ret=0
+ echo 0
0
+ '[' 0 -eq 0 ']'
+ rm /appl/postgres/backup/sales.dump.230726
[postgres@node2 ~]$ ls -ltr /appl/postgres/backup/
total 312
-rw-r--r--. 1 postgres postgres 319111 Jul 26 19:29 sales.dump.230726.gz
```

26 open files

The Linux process opens files for read, write to perform operations on them. The open file details are noticed in lsof command. The common is useful to unmount the file system. If any file is open in the file system, mount operation will be halted. Following is the list of open file details of running postgres cluster.

```
-bash-4.2$ lsof | grep -i postgres
postgres  3645            postgres  cwd       DIR              202,1      4096 10054046 /var/lib/pgsql/data
postgres  3645            postgres  rtd       DIR              202,1       268       96 /
postgres  3645            postgres  txt       REG              202,1   5536408  5262883 /usr/bin/postgres
postgres  3645            postgres  DEL       REG                0,5            557056 /SYSV0052e2c1
```

27 list file type

With file command you can identify the type of the file, whether it is plain text or executable file.

```
$ file /usr/pgsql-13/bin/psql
/usr/pgsql-13/bin/psql: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-6
4.so.2, BuildID[sha1]=7a2f35f388972c5fbd592db590f8c8f229d7b5fb, for GNU/Linux 3.2.0, stripped
```

28 strings command

The strings command displays the characters from the binary file.

Following is the example:

```
$ strings bk.tar | head -10
postgresql-Fri.log
0000600
0000032
0000032
00000001541
14606143013
014747
ustar
postgres
postgres
```

29 od command.

The od utility used for octal, decimal, or hexadecimal dump of binary files

Following is the output of text file.

```
$ cat tt.txt
database system is shut down
```

With od -c you can identify the octal format of the file.

```
$ od -c tt.txt
0000000   d   a   t   a   b   a   s   e       s   y   s   t   e   m
0000020   i   s       s   h   u   t       d   o   w   n  \n
0000035
```

30 split files

You can split file into many small pieces, for management for large files for export or copy operations. You can split the file with the awk command.

```
$ awk '{print > "bk" (NR % 4) +1 }' bk.tar
$ ls -ltr bk*
-rw-r--r--. 1 postgres postgres 20480 Apr 16 03:09 bk.tar
-rw-r--r--. 1 postgres postgres  1913 Apr 16 03:17 bk4
-rw-r--r--. 1 postgres postgres 10193 Apr 16 03:17 bk3
-rw-r--r--. 1 postgres postgres  1740 Apr 16 03:17 bk1
-rw-r--r--. 1 postgres postgres  6635 Apr 16 03:17 bk2
```

31 compress

The compress command gzip is to compress the datafile.

```
$ gzip bk.tar
$ ls -l bk.tar*
-rw-r--r--. 1 postgres postgres 1371 Apr 16 03:09 bk.tar.gz
```

32 tar command

The command tar bundles several files and directories into a single file.

```
$ tar -cvf bk.tar *.log
```

```
$ tar -cvfz bk.tar.gz *.log
bk.tar.gz
```

The above command compresses along with tar the file.

33 mount

The mount command links the file system name to a directory. AWS provides an elastic file system to mount from different machines or nodes. In the following example, the file system is mounted on multiple computers, which serves as a network file system.

```
mount -t nfs4 -o

nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport

fs-0d561ffb72d2e6f79.efs.us-east-amazonaws.com: /efs
```

34 dd command

The dd command is used to write and write the raw device files. The flag reads from the device and writes to a file.

```
$ dd if=/dev/xvda4 of=/tmp/a4.txt bs=1024 count=10
10+0 records in
10+0 records out
10240 bytes (10 kB, 10 KiB) copied, 0.000199447 s, 51.3 MB/s
```

35 strings

The strings command displays data stored in binary format which helps in identifying contents of datafile.

```
$ strings /tmp/a4.txt
XFSB
root
XAGF
XAGI
XAFL
AB3B
AB3C
```

36 /dev/zero

The command /dev/zero erases the contents of the raw device. Following is the example: (1) with dd command output the 4.1 MB contents of /dev/xvdf to /tmp/a.txt file and read the contents with (2) string command, (3) zero the device file /dev/xvdf with /dev/zero command , (4) read the contents of /dev/xvdf output into /tmp/a.txt with strings command, and (5) Read contents from /tmp/a.txt. You will use this command to erase the contents from disk during installation if you determine to reinstall due to errors while installing Postgres Cluster in EC2/AzureVM environments.

```
df -h | grep /postdata
/dev/xvdf        64Z   64Z  1.9G 100% /postdata
# (1) dd if=/dev/xvdf of=/tmp/a.txt bs=1024 count=4024
4024+0 records in
4024+0 records out
4120576 bytes (4.1 MB) copied, 0.0220823 s, 187 MB/s
#(2)  strings /tmp/a.txt
/postdata
# (3) dd if=/dev/zero of=/dev/xvdf count=1024 bs=8192
1024+0 records in
1024+0 records out
8388608 bytes (8.4 MB) copied, 0.0053907 s, 1.6 GB/s
# dd if=/dev/xvdf of=/tmp/a.txt bs=1024 count=4024
4024+0 records in
4024+0 records out
4120576 bytes (4.1 MB) copied, 0.010616 s, 388 MB/s
# strings /tmp/a.txt
#
```

37 /dev/null

The command /dev/null empties the content of file, which is used often to purge large or trace files

```
$ ls -l postgresql-Tue.log
-rw-------. 1 postgres postgres 1220 Apr 16 03:05 postgresql-Tue.log
$ cat /dev/null > postgresql-Tue.log
$ ls -l postgresql-Tue.log
-rw-------. 1 postgres postgres 0 Apr 16 03:32 postgresql-Tue.log
```

38 head command

The head command displays top 2 lines in the file

```
[postgres@node2 ~]$ head -2 nohup.out
ERROR:  relation "pg_users" does not exist
LINE 1: create table t1 as select * from pg_users;
^
```

39 tail command

The tail command displays the bottom 2 lines in the file.

```
[postgres@node2 ~]$ tail -2 nohup.out
ERROR:  could not extend file "base/13436/16402": No space left on device
HINT:  Check free disk space.
```

40 diff command

The diff command compares the differences between two files, with “>” symbol for non matching word and “<” symbol for matching word.

```
[postgres@node2 ~]$ cat post.txt
Micheal Storebraker create postgres in 1997
Larry Elison createds Oracle in 1987
Obama Was President of USA.         --> Additional word
[postgres@node2 ~]$ cat post1.txt
Micheal Storebraker create postgres in 1997
Bill Gates created Microsoft in 1987
[postgres@node2 ~]$ diff post.txt post1.txt
2,3c2
< Larry Elison createds Oracle in 1987
< Obama Was President of USA
```

```
> Bill Gates created Microsoft in 1987
[postgres@node2 ~]$ diff post1.txt post.txt
2c2,3
< Bill Gates created Microsoft in 1987
> Larry Elison createds Oracle in 1987
> Obama Was President of USA
```

41 Symbolic Links

With Symbolic links, you can create alias for files required in certain programs to have the same file name in different directories. The links are created in two methods: 1) soft method, 2) Hard method. With the soft link method, two inodes are created one for the original and another for the copy file, whereas with hard link the same inode is shared by two files.

Following is the example:

1) create a file called tt.

touch tt

2) Check the inode of the file.

ls -i tt
7799288 tt

3) Create a soft link.

ln -s tt tt1

4) Check the inode created by softlink

ls -i tt1
7799373 tt1

5) Create hard link.

ln tt tt2

6) Check the inode of hardlink.

ls -i tt2
7799288 tt2

```
$ touch tt
$ ls -i tt
8388740 tt
$ ln -s tt tt1
$ ls -i tt1
11610864 tt1
$ ln tt tt2
$ ls -i tt2
8388740 tt2
```

42 File and Disk Usage commands

To manage PostgreSQL on Linux, you have to understand several space usage commands.

41 du command

The du (disk usage) is the most common command to check the space file system space utilization. The command is du -sh .

[postgres@node2 dev]$ du -sh /appl/postgres/dev

6G /appl/postgres/dev

[postgres@node2 dev]$ du -sh * /appl/postgres/dev | grep G

1.4G base

1.1G pg_wal

[postgres@node2 dev]$ du -sh * /appl/postgres/dev | grep K

4.0K backup_label.old

268K backup_manifest

42 df command

[postgres@node2 /]$ df -h /appl/postgres

Filesystem Size Used Avail Use% Mounted on

/dev/xvda2 10G 7.0G 3.1G 70% /

43 variables

You can assign values to variables. Following is the example of assign values to date variable with Year, Month and Date values from date system variable.

[postgres@node2 ~]$ date=$(date +%Y%m%d) .

[postgres@node2 ~]$ echo $date

20230724