Error faced

Error 1:-
postgres@mdcxuatdb47:~$ psql
psql: error while loading shared libraries: libpq.so.5: cannot open shared object file: No such file or directory

sol:-
export PATH=/usr/lib/pgsql-15.6/bin:$PATH
export PGDATA=/mars_db/postgresql-15.6/data
export PGPORT=5432
export PGDATABASE=postgres
#`export PGHOST=/tmp
export LD_LIBRARY_PATH=/usr/lib/pgsql-15.6/lib:$LD_LIBRARY_PATH #

==========

Error 2:-

ssl not configure:-

sol:-
openssl req -new -x509 -days 3650 -key server.key -out server.crt -subj
"/C=ID/CN=192.168.150.200"

openssl x509 -req -signkey server.key -out server.crt -days 365

==========

Error 3:-

WARNING: database "policy_engine" has a collation version mismatch
DETAIL: The database was created using collation version 2.17, but the operating system provides version 2.28.
HINT: Rebuild all objects in this database that use the default collation and run ALTER DATABASE policy_engine
REFRESH COLLATION VERSION, or build PostgreSQL with the right library version.

solution:- You are now connected to database "policy_engine" as user "postgres".
policy_engine=# ALTER DATABASE policy_engine REFRESH COLLATION VERSION;
NOTICE: changing version from 2.17 to 2.28
ALTER DATABASE

Note:- 1.A collation version refers to the specific version of these sorting and comparison rules provided
by the operating system (OS) at the time the database was created or modified.

2.The collation version is critical for maintaining consistency in text comparisons and sorting. When the collation rules
provided by the OS change, it's important to update or rebuild database objects that rely on those rules (like indexes).

==========

Error 4:-

/usr/lib/pgsql-15.6/bin/psql: error while loading shared libraries: libpq.so.5: cannot open shared object file: No such file or directory

If the libpq.so.5 library exists but isn't found by PostgreSQL tools, you need to tell the system where to look for it:
tempory changes:-

export LD_LIBRARY_PATH=/usr/lib/pgsql-15.6/lib:$LD_LIBRARY_PATH

/usr/lib/pgsql-15.6/bin/psql --version

make perment chenges:-
1.vi .bash_profile

export LD_LIBRARY_PATH=/usr/lib/pgsql-15.6/lib:$LD_LIBRARY_PATH

2. source .bash_profile

==========
==========
Error 5:-
barman error:-

Both root and postgres are using /usr/local/bin/barman, but they are still showing different versions of Barman

1.Verify Python Version Used by Barman:-

/usr/local/bin/barman --version
/usr/local/bin/barman -V
/usr/local/bin/barman --help | grep Python ----------> excute on both postgres & barman

2.Check the Python Interpreter Used by Barman:-

head -n 1 /usr/local/bin/barman

result:- #!/usr/bin/python3

3.: Identify Python Packages for Barman:-

/usr/bin/python3 -m pip show barman ----------> excute on both postgres & barman

==========
==========
Error 6:-

PostgreSQL service did not start through systemctl :-

sudo setenforce 0
sudo systemctl restart postgresql-10

if start then used below command:-

sudo semanage permissive -a postgresql_t

==========
==========
Error 7:-

[root@localhost ~]# rpm -ivh postgresql13-devel-13.18-1PGDG.rhel9.x86_64.rpm
warning: postgresql13-devel-13.18-1PGDG.rhel9.x86_64.rpm: Header V4 RSA/SHA256 Signature, key ID 08b40d20: NOKEY
error: Failed dependencies:
perl(IPC::Run) is needed by postgresql13-devel-13.18-1PGDG.rhel9.x86_64

sol:-

yum install perl
yum install perl-CPAN
cpan IPC::Run

==========
==========
Error 8:-

+ /usr/lib/pgsql-15.6/bin/pg_basebackup -U postgres -h /tmp -p 4702 -D /backup/PostgreSQL_Base_Backup_18-02-25-043609 -l Tue Feb 18 04:36:09 IST 2025 -P -Ft -z -R
+ echo -e \nEnd Time: Tue Feb 18 04:36:09 IST 2025
pg_basebackup: error: connection to server on socket "/tmp/.s.PGSQL.4702" failed: FATAL: no pg_hba.conf entry for replication connection from host "[local]", user "postgres", no encryption

sol:-

vi pg_hba.conf

local replication all trust

Error 9:-

configure: error: library 'xml2' (version >= 2.6.23) is required for XML support

For RHEL, CentOS, Rocky Linux, AlmaLinux:

sudo dnf install libxml2-devel -y

For Ubuntu, Debian:

sudo apt update
sudo apt install libxml2-dev -y

For SUSE (openSUSE, SLES):

sudo zypper install libxml2-devel

After installation, verify the package:

pkg-config --modversion libxml-2.0

==========

Error 10:-

configure: error: library 'xslt' is required for XSLT support

For RHEL, CentOS, Rocky Linux, AlmaLinux:

sudo dnf install libxslt-devel -y

For Ubuntu, Debian:

sudo apt update
sudo apt install libxslt1-dev -y

For SUSE (openSUSE, SLES):

sudo zypper install libxslt-devel

After installation, check if libxslt is correctly installed:

pkg-config --modversion libxslt

Error11:-

When we upgrade postgresql if password set we faced below error:-

[postgres@MCVD41S01023 ~]$ /usr/lib/pgsql-16.7/bin/pg_upgrade -d /data/postgres_14 -D /data/pgsql_16 -b /usr/pgsql-14/bin -B /usr/lib/pgsql-16.7/bin -c
Performing Consistency Checks

Checking cluster versions ok

connection to server on socket "/var/lib/pgsql/.s.PGSQL.50432" failed: fe_sendauth: no password supplied

could not connect to source postmaster started with the command:
"/usr/pgsql-14/bin/pg_ctl" -w -l "/data/pgsql_16/pg_upgrade_output.d/20250313T232957.377/log/pg_upgrade_server.log" -D "/data/postgres_14" -o "-p 50432 -b -c listen_addresses='' -c unix_socket_permissions=0700 -c unix_socket_directories='/var/lib/pgsql'" start
Failure, exiting

In that case used below sloution:-

Solution 1: Use PGPASSWORD Environment Variable

Run pg_upgrade with the PGPASSWORD environment variable:-

```
export PGPASSWORD='your_postgres_password'
/usr/lib/pgsql-16.7/bin/pg_upgrade \
  -d /data/postgres_14 \
  -D /data/pgsql_16 \
  -b /usr/pgsql-14/bin \
  -B /usr/lib/pgsql-16.7/bin \
  -c
```

Solution 2: Use ~/.pgpass **File** (Recommended for Security)

Create a .pgpass file in the **home directory**:

```
echo "localhost:50432:*:postgres:your_postgres_password" > ~/.pgpass
chmod 600 ~/.pgpass
```

Then run pg_upgrade as usual:

```
/usr/lib/pgsql-16.7/bin/pg_upgrade \
  -d /data/postgres_14 \
  -D /data/pgsql_16 \
  -b /usr/pgsql-14/bin \
  -B /usr/lib/pgsql-16.7/bin \
  -c
```

==========

==========

Error12:-

pg_basebackup: error: could not receive data from WAL stream: server closed the connection unexpectedly
This probably means the server terminated abnormally before or while processing the request.
pg_basebackup: error: background process terminated unexpectedly
pg_basebackup: removing contents of data directory "/data/patroni"

Solution:-

1. Increase WAL Sender Timeout

```
wal_sender_timeout=10min
```

2. Enable Replication Slots (Prevents WAL Deletion):-

WAL files might be deleted **before** pg_basebackup can stream them. Use a **replication slot** to prevent this.

On the primary, create a **physical replication slot**:

SELECT * FROM pg_create_physical_replication_slot('standby_slot');

Then, modify your pg_basebackup command to use it:

pg_basebackup -D /data/patroni -h <primary-ip> -p 5432 -U postgres -P -Xs -R --checkpoint=fast --slot=standby_slot

3.Increase checkpoint_timeout and wal_writer_delay:-

To reduce WAL file pressure, increase:

```
ALTER SYSTEM SET checkpoint_timeout = '30min';
ALTER SYSTEM SET wal_writer_delay = '200ms';
SELECT pg_reload_conf();
```

This prevents frequent WAL segment removal.

4. Tune archive_timeout for WAL Archive Stability:-

If WAL archiving is enabled (archive_mode = on), set:

```
ALTER SYSTEM SET archive_timeout = '300s';
SELECT pg_reload_conf();
```

This prevents WAL segments from being archived too quickly.

ERROR 13:-

```
postgres=# \l
ERROR:  column d.daticulocale does not exist
LINE 8:     d.daticulocale as "ICU Locale",
                ^
HINT:  Perhaps you meant to reference the column "d.datlocale".
postgres=#
```

Sol:-

This typically happens when the `psql` client version is **newer than** the PostgreSQL **server version**, and the client is trying to access columns (like `d.daticulocale`) that exist only in newer PostgreSQL versions (e.g., PostgreSQL 16+).

The `\l` command internally runs a query on `pg_database`, and starting in PostgreSQL 16, new columns like `daticulocale` and `datcollversion` were added.

----------Update .bash_profile----------

ERROR 13:-

If user still connected database and you want to rename database because user connected to database it will not allowed to rename database. To rename database used bleow steps

Sol:-

Step 1:- Temp disallow new connections and terminate exiting one

```
-- Prevent new connections:
ALTER DATABASE ecgc_neia WITH ALLOW_CONNECTIONS = false;

-- Terminate existing connections:
SELECT pg_terminate_backend(pid)
FROM pg_stat_activity
WHERE datname = 'ecgc_neia';
```

Step 2:-

```
ALTER DATABASE ecgc_neia RENAME TO ecgc_neia_old_11082025;
```

Step 3:-

```
ALTER DATABASE ecgc_neia_old_11082025 WITH ALLOW_CONNECTIONS = true;
```

==========

Error 14:-

Reason for "ERROR: canceling statement due to conflict with recovery"
The error "canceling statement due to conflict with recovery" occurs when a query running on a PostgreSQL standby server conflicts with the WAL replay process. This is a normal behavior in a Hot Standby setup, where read queries on the standby must not block the WAL application.

Causes of the Error

- Long-running queries on the standby
- If a query on the standby reads data that the primary server has modified, and WAL replay requires access to those rows or indexes, PostgreSQL cancels the query to allow WAL replay to continue.

Solution:-

- **Enable hot_standby_feedback:-**

1. This setting prevents queries on the standby from being canceled due to vacuuming or HOT updates on the primary.
- pro & cons of this parameter as follows:-

Pros:-
Prevents VACUUM from removing old tuples needed by standby queries.
Reduces query cancellations.

Cons:-
Increases bloat on the primary server since VACUUM must retain dead tuples longer.
May cause table and index bloat, requiring aggressive autovacuum tuning.

- **max_standby_streaming_delay:-**

- This allows standby queries to run longer before getting canceled.

pro & cons of this parameter as follows:-

Pros:-
Ensures standby queries complete before WAL changes overwrite required tuples.
Useful for historical read replicas.

cons:-
Causes intentional replication lag, making standby outdated.
Not useful if real-time failover is required.

Note:- On standby server we have to add these parameter.