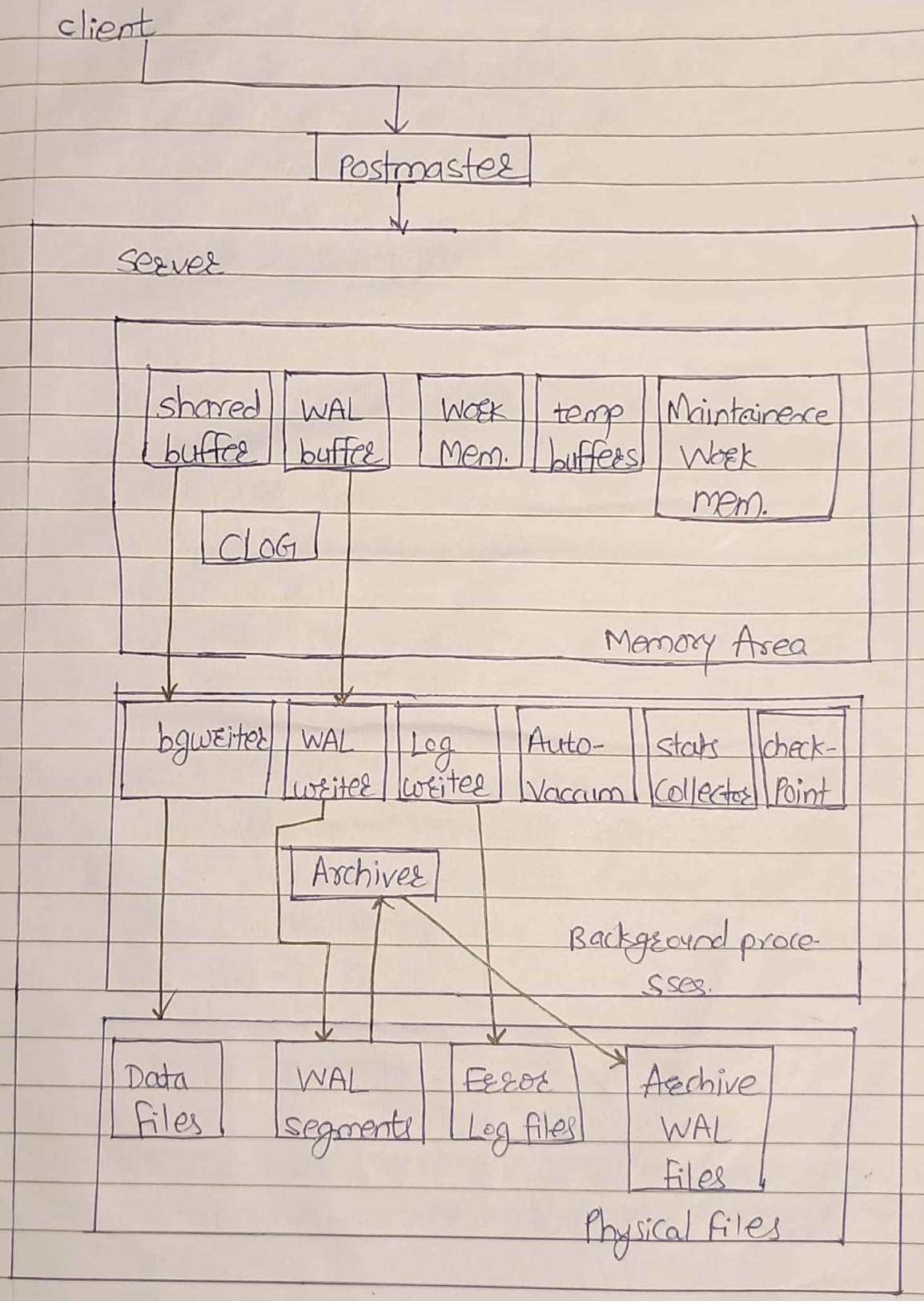
PostGRes

Architecture



Scanned with CamScanner

Shared buffer :- stores data pages in memory. whenever a query needs to access data, it check if required pages are already in the shared buffer. Then if data is present, data is returned from this cache without any disk access.

when a query modifies data in a Postgresql database, the changes are initially made in the shared buffer & this modified pages are called dirty pages.

These dirty pages then written to data files by bgwriter.

WAL_buffer :- WAL buffers are a shared memory used to temporarily store the WAL records generated during database transactions.

These WAL buffers are flushed to WAL segments by WAL writer.

CLOG buffer :- It stores the commit status of every transaction in the database.

Scanned with CamScanner

work_mem. :- It allocates the memory for DDL & DML transactions. Also, for sorting operations.

temp_buffers :- stores the temporary tables.

(maint_work_mem)
aint_work_mem :- It allocates the memory for maintainance operations like VACCUM, ANALYZE, REINDEX.

bgwriter :- This background process helps to send dirty pages from shared buffer to data files.

WAL_writer :- Helps to send WAL buffers (WAL Records) to WAL segments.

Log writer :- Helps to create errors logs in error log files.

stats collector :- Records & collect the server related activity.

checkpoint :- This process updates the data pages to the data files every 5 min which is by default. It also ensures that data pages are transferred to data files.

Archiver :- This process helps to archive the WAL files.

Scanned with CamScanner

Autovaccum :- Responsible for performing vaccum & analyze operations.

Autovaccum reclaims the space used by dead tuples.

Data files :- These are the disks where actual physical data is stored.

WAL segments :- Files that stores transactional changes & which are used for recovery, PITR, Replication.

Error log files :- These files captures every type of error messages, activities, warnings, notices & other type of log entries using the background process ~~archiver~~. log writer.

Archive WAL files :- Basically, these are the backup files of WAL segments.

Also, used in the (DR Drill.) making of standby database.

Scanned with CamScanner

* How 'SELECT' query runs in Postgresql?

→ - when a 'SELECT' query runs in PostgreSQL, it first goes through the parser, which checks the syntax of the query.

- Then, the optimizer creates the most efficient execution plan based on indexes and data distribution. ~~The~~

- Then, the query is executed by the Executor.

- PostgreSQL checks the shared buffer for the data. If its not there, it fetches from the data files on disk.

- Operations like sorting, joining use work_mem & if it exceeds memory limits, it may use temporary files.

- In this way, user gets the data when he runs any 'SELECT' query.

(Background processes like bg_writer and WAL_writer handle data writes but are not relevant to 'SELECT' queries. - Note)

FOR EDUCATIONAL USE

Scanned with CamScanner

* Installation of postgresql server & types.

Types & steps →

i) using RPM package

for Debian/Ubuntu : apt-get
for Redhat / CentOS : yum install

Step ⇒ • Go to postgresql.org.
Download the rpm package & install on the linux server from root user.
• Run initdb command.
• Enable postgresql server
• start postgresql server

ii) using Source code

step ⇒ • Download the source code tar file & extract the tar file.
• create postgres user & required directories.
• Configure PostgreSQL (with port)
• Build postgreSQL using 'Make' command.
• Install postgresql using 'Install' Command.
• change binary owner to postgres
• From postgres user, initialize data directory.
• start postgresql server

Scanned with CamScanner

- set the environment variables or create bash profile & then connect using psql.

iii) cloud based - Installation

- Amazon RDS for postgreSQL
- Microsoft Azure Database for PostgreSQL
- Google cloud SQL for PostgreSQL.

iv) Binary installation

step ⇒
- Download the binary installer provided by EDB.
- Run the installer using ./
- Graphical interface window will open, choose where postgresql will be installed.
- specify data directory & set postgres user password.
- specify port & complete the installation.
- Then initialize the database.
- Finally, Enable & start the postgresql server.
- set environment variables or create the bash profile.
- And then connect using psql.

Scanned with CamScanner

* Vaccum & its types.

Vaccum is a critical maintenance operation that helps in reclaiming free space occupied by dead tuples (dead rows) in the tables. which is also called bloat in the table.

Types ⇒ 1) Standard Vaccum (plain Vaccum)

There is autovaccum background process that runs automatically based on configuration parameters.

This vaccum does not lock the table for read or write operations, it allows access to the table.

It reclaims space occupied by dead tuples & updates the statistics for query planning.

Reclaimed space is reused for future updates or inserts.

Commands :→

```
vaccum ;
(or)
vaccum table_name ;
(or)
vaccum schema_name.table_name;
```

Scanned with CamScanner

2) Vaccum FULL

Reclaims the space by removing dead tuples & it locks the table.

It reorganize the table & due to locking, no access to read or write table.

Commands :→

```
Vaccum FULL;
(or)
Vaccum FULL table_name;
```

(Note:- use 'pg_stat_user_tables' view to the check & verify the reclaimed space.

Also, check table size before & After vaccum operations.

```
SELECT relname AS table_name,
    pg_size_pretty (pg_total_relation_size (relid))
        AS total_size FROM
            pg_stat_user_tables ORDER BY
                pg_total_relation_size (relid)
                    DESC;)
```

(Note :- configure below parameters of autovaccum in postgre-sql.conf)

autovaccum = on, autovaccum_vaccum_threshold = 50, autovaccum_naptime = 1 min, autovaccum_vaccum_scale_Factor = 0.2)

Scanned with CamScanner

* VACCUM ANALYZE:-

vaccum analyze performs a vaccum & then gather statistics to help the planner generate optimal query plans.

commands :→

Vaccum ANALYZE;

* Vaccum VERBOSE :-

Provides detailed output about what the vaccum process is doing.

Commands :→ VACCUME VERBOSE;

* SQL Tuning Query (Performance tuning)

1) Reasons for slow Running queries & how would you troubleshoot them.

Reason → It includes
- Missing or insufficient indexes
- Poorly written SQL queries
- outdated statistics
- large data volumes
- High CPU load, I/O Load.

Troubleshoot → • Check the query execution plans using EXPLAIN or EXPLAIN ANALYZE to identify bottlenecks.

Scanned with CamScanner

- Look for missing index using 'pg_stat_user_indexes
- Use VACCUM ANALYZE to view the table statistics.
- check CPU, memory, I/O status using TOP command, vmstat, iostat command.
- Again, check for slow running queries using 'pg_stat_statements' view.
- check for active connections.

(Note:- 'pg_stat_statements' view tracks/shows execution statistics for all SQL queries.

Columns in 'pg_stat_staments'

queryid, - The actual SQL query text.
calls, - No. of times query executed.
total_time, - Time spend in executing query.
min_time, - Shortest execution time of query.
max_time, - Longest execution time of query.
mean_time, - Average execution time
stddev_time, - Standard deviation of query's exe. time
rows, - Total no. of rows returned by query.
shared_blks_hit - No. of shared buffer cache hits.
shared_blks_read - No. of shared buffer reads.
temp_blks_written, - No. of temp blocks written
)

Scanned with CamScanner

2) what is EXPLAIN ANALYZE ? & how do you use it ?

'Explain Analyze' provides actual execution plan of a SQL query, showing how PostgreSQL executes the query step-by-step.

ex:- EXPLAIN ANALYZE <SQL query> ;

(Note:- In output, we get output of parameters,
seq scan, Filter,
Rows Removed by filter,
planning time, execution time)

3) what is the importance of 'maintenance_work_mem' and 'work_mem' in query tuning?

work_mem :→ Controls the amount of memory used for sorting operations, hash tables, ORDER BY, GROUP BY, operations.

Maintenance_work_mem :→ used for maintenance operations like vaccum, indexing etc.

4) How to check long running & Reasons of long running & also how to troubleshoot them.

Scanned with CamScanner

Reasons :→
- Missing or insufficient indexes.
- Large data volumes.
- Blocking issues. (by long running)
- High Load (CPU, I/O)

Troubleshoot :→
- Use EXPLAIN or EXPLAIN ANALYZE to check query execution plan.
- Look for missing index.
- Use VACCUM ANALYZE to view table statistics.
- Check for any blocking query.
- Kill any unwanted process / pid that causing server performance.

(Note:- using 'pg_stat_activity' view, we can check for long running queries. Also, we can check active sessions / inactive sessions.

Columns → pid, usename, application_name, client_addr, state, query, state_change, now()-state change)

'pg_stat_activity' provides information about the current activity of each connection to the database. It tracks active queries, session details, state of connections, etc.

Scanned with CamScanner

MVCC (Multi-Version Concurrency Control)

→ MVCC mechanism allows (you) multiple transactions to access the database simultaneously without conflicting with each other.

MVCC allows multiple transactions to access the same database objects concurrently, without locking them.

MVCC reduces contention & deadlocks, also improves performance, consistency & data integrity.

ex:- Let's take an example of an ATM machine. If Concurrency (MVCC) is not applied, in this case, different users won't be able to access their accounts and draw money at the same time.
Whereas if Concurrency control (MVCC) is applied, then multiple users can do so easily.

FOR EDUCATIONAL USE

Scanned with CamScanner

Table Partitioning

- Table partitioning is a method of dividing a large table into smaller managable pieces called 'Partitioning'.

- Partition can help improve performance & simplify data management.

- Maintenance tasks like indexing, vaccuming are easier because they can be done on individual partition.

Types -

① Range Partitioning
Data is divided based on 'RANGE OF VALUES'.

② List Partitioning
Data is divided based on 'LIST VALUES'.

③ Hash Partitioning
Data is divided based on a 'HASH FUNCTION' for even distribution.

(A hash function is a mathematical function that takes an input & returns a fixed size string or number, known as hash value.)

FOR EDUCATIONAL USE

Scanned with CamScanner

What is Indexing & Types?

Indexing is a technique to speed up the retrieval of rows from a table by creating a data structure/Index for faster searching.

An Index helps the database find the required rows without scanning the entire table, thus improving query performance.

Types-

① B-Tree index

B-Tree index handles equality & range queries on data that can be sorted into some ordering.

'< <= = >= >'

ex:- SELECT * FROM table WHERE Column >= 10;

② Hash index

Hash index used for equality comparisons only.

ex:- SELECT * FROM table WHERE column = 'value';

③ GIN (Generalized inverted index)

Used for indexing complex data types like arrays, JSON, & full-text search.

FOR EDUCATIONAL USE

ex:- SELECT * FROM table WHERE Column @> {"key":"value"};

Scanned with CamScanner

④ GiST (Generalized search Tree)
Used for (flexible Index) geometric & full text-Search.
ex:- SELECT * FROM table WHERE Column <-> point (10,20);

⑤ SP-GiST (Space-partitioned Generalized search tree)

- Useful for partitioning data types like ranges and points.

⑥ BRIN (Block Range Index)

- Efficient for large tables with ordered data like timestamps.

ex:- SELECT * FROM table WHERE date BETWEEN '2023-01-01' AND '2023-01-31';

* SQL JOINS

- SQL Joins are used to combine rows from two or more tables.
- They allow you to return the data from multiple tables in a single query.

CROSS JOINS → Returns Cartesian product of both tables.
INNER JOINS → only rows that matched in both tables.
LEFT JOINS → all rows from left table, matched from right
RIGHT JOINS → all rows from right table, matched from left
FULL JOINS → Combines results of both LEFT & RIGHT (all possible combinations of rows from tables) JOINS.

FOR EDUCATIONAL USE

Scanned with CamScanner

Tablespaces in PostgreSQL

Tablespace in PostgreSQL is a storage location on the file system/server that allows you to define where the database objects like tables, indexes, schemas are stored.

ex:- (i) To create a tablespace

```
CREATE TABLESPACE edb LOCATION '/path/to/dir';
```

(ii) To assign a tablespace to that table

```
ALTER TABLE table_name SET TABLESPACE
                    tablespace_name;
```

(iii) Create a table & assign a tablespace

```
CREATE TABLE table_name (id int, name
                    VARCHAR (50))
                    TABLESPACE tablespace_name;
```

System tablespaces:-

There are two default tablespaces in PostgreSQL.

(i) pg_default → default location of database objects. Present in 'base' folder in data_directory in postgresql server.

(ii) pg_global → This tablespace includes system catalogs & metadata tables of databases. Present in 'global' folder in data_directory in postgresql server.

FOR EDUCATIONAL USE

Scanned with CamScanner

* Migration - steps & how to do it?

- To migrate a postgreSQL server to AWS, As a DBA, I would first evaluate current postgreSQL version, database size, extensions, configurations & dependencies of the server.

- Then, I will plan the migration strategy like downtime, complexity, server configuration for the AWS. (Or) Amazon RDS for postgreSQL.

- I will take the backup of small databases using 'pg_dump' & for large databases 'pg_basebackup'.

- Next, I will create the RDS instance for postgreSQL server & configure the parameters like 'work_mem', 'shared_buffers', & 'maintenance_work_mem'.

- Once the environment is setup, I will transfer the 'pg_dump' backup to AWS. and will restore it using 'pg_restore'.

- After migrating the data, I will validate data integerty, schema consistency. I will tell application team to switchover the application to new AWS server. I will focus on performance tuning, setting automatic backups, monitoring alerts, scheduling scripts after migration.

Scanned with CamScanner

Minor Version Upgrade steps from 15.0 to 15.1.

Step1 :- Backup your Database. You can use pg_dumpall or pg_basebackup.

pg_dumpall -U postgres -f /path/to/backup.sql

Step2 :- Stop the postgresql services

systemctl stop postgresql-15

Step 3 :- update the yum repositary

yum update postgresql15*

Step 4 :- Install the package

yum install postgresql15-server-15.1
postgresql15-15.1
(Note:- This will install the Postgresql-15.1)

(or)

if you want to upgrade to latest minor version automatically use,

yum install postgresql15-server postgresql15-contrib

Step-5 :- Start the server
systemctl start postgresql-15

FOR EDUCATIONAL USE

Step-6 :- Verify the upgrade. (SELECT version();)

Scanned with CamScanner

Upscale Activity / Downscale activity parameters in postgreeSQL.conf file.

→ ex:-

max_connections = 2000
shared_buffers = 4GB
effective_cache_size = 12 GB
maintenance_work_mem = 1 GB
checkpoint_completion_target = 0.9
wal_buffers = 16 MB
default_statistics_target = 100
random_page_cost = 1.1
effective_io_concurrency = 200
work_mem = 524 KB
huge_pages = off
min_wal_size = 1 GB
max_wal_size = 4 GB
max_worker_processes = 8
max_parallel_workers_per_gather = 4
max_parallel_workers = 8
max_parallel_maintenance_workers = 4

(i) max_connections
Defines maximum no. of concurrent connections to the postgreSQL database.

(ii) shared_buffers
Determines how much memory PostgreeSQL uses for caching data.

FOR EDUCATIONAL USE

Scanned with CamScanner

(Note:- 25% of total RAM (memory) is allocated to shared_buffers)

(iii) effective_cache_size

This parameter estimates how much memory is used by query planner to calculate the cost of different query execution plans.

Also, the effective_cache_size helps the database decide whether to use an index.

Roughly, set of effective_cache_size should be 75% of the RAM (Total).

Increasing the effective_cache_size parameter can improve query performance.

(iv) maintenance_work_mem

Specifies the maximum amount of memory to be used by maintenance operations, such as VACCUM, CREATE INDEX, EXPLAIN, ANALYZE, & ALTER TABLE, etc.

(v) checkpoint_completion_target

Specifies the fraction of time between checkpoints.

ex:- 0.9

This means PostgreSQL will aim to complete the checkpoint 90% of the way through the checkpoint interval.

This helps to distribute disk I/O load.

Sundaram
FOR EDUCATIONAL USE
Scanned with CamScanner

(vi) wal_buffers

specifies the amount of memory used for writing of WAL data before it is flushed to disk.

(Note: The default setting is usually 16 MB).

(vii) max_wal_size

specifies the maximum size of WAL files that PostgreSQL will retain before performing a checkpoint.

(viii) max_worker_processes

specifies the maximum no. of background worker processes that the system can support. ex:- 8 (8 background worker processes will start).

(ix) max_parallel_workers_per_gather

specifies the maximum no. of workers that can be started by a single Gather node during a parallel query. ex:- 4 (4 worker processes to be used for parallel query).

(x) max_parallel_workers

specifies the maximum no of parallel worker processes that the system can support. ex:- 8 (8 parallel workers can be used across all queries).

(xi) max_parallel_maintenance_workers

specifies max. no. of workers that can be started by a parallel maintenance operation. ex:- 4 (4 workers starts for maintenance operation)

FOR EDUCATIONAL USE

Scanned with CamScanner

* DR Drill :- (Repmgr)

Pre-requisites :-

On DR-Server ⇒

Run, i) repmgr cluster crosscheck
(verifies current status of all nodes in the replication cluster)

ii) repmgr cluster show
(provides summary of each node like id, role, status, Location, Connection string, etc.)

iii) repmgr standby switchover --dry-run
(To verify that the server is ready to switchover)

(Note:- If dry run command runs successfully, then go for next steps)

Above, are the pre-requisites must be done on standby server during start of the DR Drill activity.

Scanned with CamScanner

Switchover :-

On Primary server ⇒

- Disable the archive_command in postgresql.conf file and take the reload of the server.

After disabling archive_command Run,

SELECT pg_reload_conf();

On DR server ⇒

- Go for switchover & enable the archive_command & take reload of the server.

Run,

repmgr standby switchover

then, enable archive_command & take the reload of the server using

SELECT pg_reload_conf();

Post-DR Drill :→
(switchover)

Check replication status & check status of archive_command on both the servers.

Scanned with CamScanner

* DR-Drill - (Switchback activity) :-

Pre-requisites -

On new DR server ⇒

Run, i) repmgr cluster crosscheck

ii) repmgr cluster show

iii) repmgr standby switchover -dry-run

(Note:- If dry run command runs successfully then go for the next steps).

Switchback -

On new primary server ⇒

- Disable the archive_command & take reload of the server.

On new DR server ⇒

- Go for switchover & enable the archive_command & take reload of the server.

Run, i) repmgr standby switchover

& ii) Enable archive_command & take reload.

& then check replication status.

Scanned with CamScanner

* shell / bash scripting (shell & bash).

Shell scripting ⇒ scripts that can be executed by any unix shell.
extension is '.sh'

when creating shell script, it includes '#!/bin/sh' at the starting of the script. which ensures that script runs using /bin/sh (bourne shell).

Bash scripting ⇒ scripts that can be executed by Bash shell.

when creating bash script, it includes '#!/bin/bash' at the starting of the script. which ensures that script runs using /bin/bash (bourne again shell (or) bash shell).

Key difference betn shell scripting & bash scripting.
- shell scripts can target any shell. (Zsh, ksh, etc).
- Bash scripts can target only the bash shell.

- Shell scripts used for various shell environments.
- Bash scripts commonly used for Linux systems.

Scanned with CamScanner

Difference betn shell scripting & Bash scripting

Shell script	Bash script
Script that can be executed in any shell.	Script that can be executed in bash shell only.
shell has fewer features as compared to bash.	Bash has more features as compared to shell.
Shell is considered the original Unix shell.	Bash is an acronym for Bourne Again Shell.
we can use shebang, #!/bin/sh	We can use shebang, #!/bin/bash
Shell may be one of Korn, C Shell, Bourne, Bash.	Bash script is one form of shell script.

Sundaram®

FOR EDUCATIONAL USE

Scanned with CamScanner

How do you schedule a Bash script to run postgresql maintenance task like VACCUM & ANALYZE?

→ (i) Create the bash script using vi editor.

Enter/insert, DB credentials, VACCUM & ANALYZE command, also mention log file location & log file name, & then save the file.

(ii) Give executable permissions to the script.

(iii) schedule the script in crontabs using 'Crontab -e' according to your requirement.

Scanned with CamScanner

* pgadmin, pgbouncer, pgrepack.

pgadmin ⇒ - pgadmin is an open-source tool for administration & management for postgresql database.
- It provides a graphical interface to interact with postgreSQL databases.
- It allows you to write, execute & analyze SQL queries.
- It manages database roles, permissions & security setting.

pgbouncer ⇒ - pgbouncer is a lightweight connection pooler for postgreSQL.
- It is used to manage & optimize database connections.
- It distributes client connections across multiple PostgreSQL servers to balance the load.

pg_repack ⇒ - pg_repack is an extension & utility for PostgreSQL that helps to reclaim space and optimize database performance by reorganizing tables & indexes.

- Index Rebuilding, table reorganization, monitoring tables for bloat, these are the key features.

Scanned with CamScanner

* Upgradation

Minor version upgrade ⇒

- Backup the data/ Backup the databases.
- Check compatibility of current postgreSQL version.
- Download the latest minor version, & install the new minor version using rpm package. (using yum).
- Stop the server & upgrade the data_directory using pg_upgrade
- Start the postgreSQL server & verify the minor version upgrade using command

  psql -V
- Remove old version using command

  yum remove postgreSQL-<old version>

Major version upgrade ⇒

- Backup the data using pg_dumpall, pg_dump or pg_basebackup.
- Check compatability in release notes for new version of postgresql to identify any changes, configuration changes or incompatible changes.
- Download the new version from website & install it using yum/apt.
- Stop the server & upgrade the old & new binary directory/path and data directory.

Scanned with CamScanner

- use pg_upgrade utility.

ex:-

```
pg_upgrade -b /old/bin/path -B /new/bin/path
           -d /old/data_dir -D /new/data_dir
```

- start the postgresql server.
- Then Run 'analyze_new_cluster.sh' script which is automatically generated by pg_upgrade & should be run to optimize performance by updating stats on new cluster.
- After this remove the old version by yum remove.
- verify new version using 'psql -V'
- update the configuration files & reload the server.

** Backup and Recovery

PgBackrest - pgbackrest is a powerful backup tool (and Restore tool) for postgreSQL that provides features like compression, encryption and parallel processing.

steps:-

pgbackrest installation steps:-

- Download the pgbackrest package (rpm) on your server.
- Install the package using 'yum' / 'apt-get'.
  yum install -y pgbackrest
- Create required backup directory
  /var/lib/pgbackrest & give permissions.
- Edit pgbackrest.conf file which is located at '/etc/pgbackrest/pgbackrest.conf' and add the entry of global path & installed server directory_data. & port.
- Edit archive_command in postgresql.conf file & put archive_mode = on.
- Reload the postgresql server.
- Create the stanza of the server.
  pgbackrest --stanza= stanza_name --log-level-console-info stanza= create

& then validate stanza using check command.

--log-level-console = info (It displays errors, warnings, informational messages of the stanza backup, status.)

(i) perform a full Backup

pgbackrest --stanza= stanza_name --type=FULL --log-level-console= info backup

Scanned with CamScanner

ii) Restore database using pgbackrest

- Stop the postgresql Server
- pgbackrest --stanza= stanza_name --delta
  --log-level-console= info
  restore
- Start the postgresql server.

(Note:- '--delta' will only restore the files that have changed since the last backup.)

--delta speeds up the recovery, minimizes the downtime & maintains existing data.

iii) Incremental Backup

pgbackrest --stanza=stanza-name --type=incr
--log-level-Console=info
backup

iv) PITR (Point in time recovery)

- You will have last full/incr backup. Take that info using 'pgbackrest --stanza= stanza_name info
- Stop the server & note down the timestamp.
- Run, pgbackrest --stanza= stanza_name --delta
  --type=time --target= 'timestamp'
  --log-level-console=info restore
- Start the server.

Scanned with CamScanner

* pg_dump →

Can take backups of entire database, tables, schemas, data-only, etc.

(i) entire database

```
pg_dump -U username -d dbname -p portname/no.
        -f /path/to/backup.sql
```

-f : directs that backup should be taken in .sql file.

(ii) table backup

```
pg_dump -U username -d dbname -p portno.
        -t tablename -f /path/to/tablebkp.sql
```

(iii) schema-only

```
pg_dump -U username -d dbname -p portno.
        --schema-only -f /path/to/schemabkp.sql
```

(iv) data-only

```
pg_dump -U username -d dbname -p portno. --data-only
        -f /path/to/databkp.sql
```

Scanned with CamScanner

(v) in directory format (backup of a database)

```
pg_dump -U username -d dbname -p port no.
        -Fd c -f /path/to/dbkp.dump
```

-F: specify the format
d: directory format
c: custom format

(vi) Using nohup (backup of a database)

```
nohup pg_dump  -||-  2>&1 &
```

2: standard error
1: standard output

(Basically '2>&1' means redirect the standard error to the same location as standard output.)

&: at the end, in the command this backup process will run in the background.

(vii) using nohup & logfile (backup of database)

```
nohup pg_dump -||- /path/to/bkp.sql > /path/to/logfile.log 2>&1 &
```

The output of progress of backup & errors will show in this logfile.

Scanned with CamScanner

* pg_dumpall →

i) To take the backup of all databases & objects, we can use pg_dumpall.

```
pg_dumpall -U username -p portno. -f /path/to/alldbbkp.sql
```

ii) To take the backup of only roles and permissions.

```
pg_dumpall -U username -p portno. --roles-only -f /path/to/rolesbkp.sql.
```

for, restore these roles, Run,

```
psql -U username -p portno. -f /path/to/rolesbkp.sql
```

* pg_restore →

i) Restore a database

```
nohup pg_restore -U username -d dbname -p portno -j 4 -v /path/to/dbbkp.sql > /path/to/logfile.log 2>&1 &
```

-j 4 : uses 4 parallel jobs for restore process
By using parallel jobs, pg_restore take multi-core processors to restore the database.

Scanned with CamScanner

ii) Restore a table

```
pg_restore -U username -d dbname -p portno.
           -t tablename /path/to/tablebkp.sql
```

* pg_basebackup

pg_basebackup performs a full physical backup of entire database cluster including all the data files, configuration files & WAL files.

i) Take full backup

```
pg_basebackup -U username -p port -h hostname
              -D /path/to/backup -Ft -z -P
```

-Ft : backup in tar format
-z : compresses the backup
-P : Shows the progress of the backup.

(Note:- To take backup with WAL files, use -Ft -z -P -X fetch)

This is the tool for postgreSQL database, which is essential for disaster recovery & data protection.

Scanned with CamScanner

What is Replication? Types & steps to configure.

Types:-

① Streaming Replication-

- It is nothing but continuous data transfer of WAL files from primary to standby.
- It supports both sync & async modes of replication.

② Synchronous Replication -

A mode of streaming replication where data has been written to standby server before completing the transactions.

③ Asynchronous Replication-

A mode of streaming replication where primary server does not wait for the standby server to confirm the receipt of WAL files. & it provides better performance.

④ Logical Replication -

This replication allows for the replication of a specific database objects.

Sundaram
FOR EDUCATIONAL USE
Scanned with CamScanner

Def^n:- Replication refers to the process of copying and maintaining database objects such as tables, schemas, & data across multiple servers.
Replication ensures that changes made in one postgresql primary server are propagated to others. (servers)

setup (configuration steps) :-

Install postgresql on standby server but don't initialize the database on standby server.

On Primary :-

Change the parameters like listen_addresses, WAL_Level, WAL_keep_size, hot_standby, hot_standby_feedback, track_commit_timestamp, max_WAL_senders, archive_mode.

(A) Listen_addresses = '*'
This parameter specifies which IP addresses postgreSQL server will listen for incoming connections.

(B) WAL_Level = replica
It supports streaming replication & keep it in sync.

Sundaram
FOR EDUCATIONAL USE
Scanned with CamScanner

(C) wal_keep_size
This parameter specifies the size of WAL files to keep in pg_WAL directory.

(D) hot_standby = on
This parameter enables the standby server to accept read-only queries.

(E) hot_standby_feedback = on
This parameter allows standby server to send feedback to the primary server.

(F) track_commit_timestamp = On
This parameter tracks the commit timestamp of every transactions.

(G) max_WAL_senders
This parameters defines maximum no. of connections from standby servers for streaming replication.

(I) archive_mode = on (optional)
If you are using WAL archiving alongside streaming replication, you must enable this parameter ON.

So, all these parameters set on primary server.

Sundaram
FOR EDUCATIONAL USE
Scanned with CamScanner

Now, Add the pg_hba entry for the standby server in pg_hba.conf file on primary server.

On standby :-

- Install postgresql server but don't initialize database.

- Enable postgresql server (from root user)

  systemctl enable postgresql-{version}

- Then, take the backup (from) of primary server on standby server using pg_basebackup using command

/pg_basebackup/directory -D /data_directory/path
-X stream -c fast -h (I.P. of primary)
-p portno. -R -P

/pg_basebackup/directory
This is the base backup directory of the standby server where the backup from the primary will be stored.

Sundaram

FOR EDUCATIONAL USE

Scanned with CamScanner

-D /data_directory /path
specifies the directory (data) on the standby server where the backup should be placed, where postgreSQL instance will be stored.

-X stream
This parameter helps to send /stream WAL files from primary to standby server.

-c fast
This parameter helps to fast the process of backup & also controls the checkpoint.

-h (I.P. of primary server)
specifies the hostname (or) I.P. address of the primary server from which the backup is being taken.

-p portnumber
specifies the port no. of primary server.

-R
It automatically creates a recovery.conf file in data directory in versions before 12. & in postgreSQL version 12 & later, standby.signal file is created for streaming replication.

Sundaram
FOR EDUCATIONAL USE
Scanned with CamScanner

-P
Displays the progress of the backup.

After completing backup process, start the postgresql server on standby server if your streaming replication is ready.

systemctl start postgresql-{version}

⊙ How to check replication lag? & if there is replication lag what will you do & how you will troubleshoot?

→
- Using 'pg_stat_replication' view on primary server, i can check replication lag.

- Also, using 'pg_last_wal_receive_lsn()' & 'pg_last_wal_replay_lsn()' views on standby server, i can check replication lag.

pg_last_wal_receive_lsn()
The last WAL location received by the standby.

pg_last_wal_replay_lsn()
The last WAL replayed on the standby.

Sundaram
FOR EDUCATIONAL USE
Scanned with CamScanner

- Also, using 'pg_stat_wal_receiver' view on standby server, i can replication lag.

- Using monitoring tools like (pgBadger, Redgate, pg_stat_statements).

Troubleshooting replication lag,

First, As a DBA, I will check the replication lag using 'pg_stat_replication' view on the primary server. In which, write lag indicates delay in WALs being sent to standby, flush lag indicates delay in writing WALs to disk on standby & replay lag indicates delays in applying/replaying WAL on the standby. I will check network issues betn primary & standby using ping or traceroute, also i will check CPU & memory on the standby server. I will check for long running queries on standby server using pg_stat_activity which can cause lagging issues. I will review the replication parameters in postgresql.conf files on both servers. I will check archive_command, archive_mode & will also check wal_keep_size parameter is configured properly. I will check disk space on both servers. I will use monitoring tools like Redgate, pgBadger, to check the server performance & what solutions I can apply to remove the lag.

FOR EDUCATIONAL USE

Scanned with CamScanner

* Important postgreSQL extensions:-

i) PL/pgSQL - Provides a procedural language for writing complex functions and triggers within postgreSQL.

ii) pg_stat_statements - Tracks the execution statistics of all SQL statements.

iii) pgAudit - Provides detailed logging of database activity.

iv) pgcrypto - offers cryptographic functions for encrypting & decrypting data as well as hashing.

v) pg_trgm - Supports text search & similarity queries using trigrams. for fast similarity searching

vi) pgvector - provides support for vector similarity searching which is useful for AI & ML applications

vii) pg_partman - Management of table partitioning.

viii) PostGIS - Enable geographic objects & spatial queries.

ix) postgres_fdw - Allows postgreSQL to query & join data from other postgreSQL databases.

x) hstore - Enable storage of key-value pairs within a single PostgreSQL column, useful for semi-structured data.

Scanned with CamScanner

data_directory folders :→

i) base - Contains actual database files & data directories for individual databases.

ii) global - Contains clusterwide tables like pg_database, pg_tables, pg_index etc.

iii) log - stores server log files.

iv) current_logfiles - contains current active log files.

v) PG_VERSION - contains postgresql version number.

vi) pg_twophase - stores files related to two-phase commit transactions.

vii) pg_stat_tmp - stores temporary statistics files used by PostgreSQL for monitoring & performance.

viii) pg_snapshots - Contains files related to transaction snapshots.

ix) pg_serial - Holds information related to serializable transaction isolation level.

x) pg_replslot - stores files related to replication slots, which manage logical replication.

Scanned with CamScanner

xi) pg_notify - Contains files related to PostgreSQL's notification system.

xii) pg_multixact - stores data related to multi-transaction identifiers used in multi-version concurrency control.

xiii) pg_dynshmem - Contains files for dynamic shared memory segments.

xiv) pg_commit_ts - Holds information on commit timestamps.

xv) postgreSQL.conf - Main configuration file for PostgreSQL server settings.

xvi) postgreSQL.auto.conf - Contains (or) it is a file used for storing configuration parameters that are set by ALTER SYSTEM.

xvii) pg_ident.conf - Controls client authentication & specify who can connect to which database.

xviii) pg_hba.conf - Defines user mapping rules for authentication.

xix) pg_xact - stores files related to transaction state & logs.

Scanned with CamScanner

xx) pg_WAL - Contains WAL files.

xxi) pg_subtrans - stores information related to subtransactions.

xxii) pg_stat - Contains files related to runtime statistics.

xxiii) pg_logical - stores files related to logical replication if needed.

xxiv) postmaster.opts - Contains options passed to the PostgreSQL server process on startup.

xxv) Postmaster.pid - stores the process ID of the running PostgreSQL server.

Scanned with CamScanner

* There are multiple client Aunthentication Methods in pg_hba.conf file?

1) trust

This method allows any user to connect without any authentication. Used for local connections where security is not a concern.
ex:- Development Environments.

2) md5

It is password authentication method based on MD-5 hashed password.

When a client hashes a password using MD5 & sends the hashed password to the server, the server compares that hashed password in the pg_authid table & gives permission to user to connect to the server.

MD5 hashed password is a password that has been processed using MD5 (Message Digest Algorithm 5) hash function.

3) scram-sha-256

It is password authentication method using SCRAM (salted Challenge Response Authentication mechanism) protocol with SHA-256 hashing.

It is the most secured method available in postgresql.

Sundaram
FOR EDUCATIONAL USE
Scanned with CamScanner

4) peer
This method is used for local connections.
ex:- its suitable for local connections on Linux.

5) ident
Used for TCP/IP connections, where the PostgreSQL server contacts an ident service running on the client machine to determine the client's operating system username.

6) Idap
(Ident Authentication Daemon Protocol)
It allows PostgreSQL to authenticate a connection based on the Unix User's identity. The database will check the OS user on the client machine & map it to a PostgreSQL role for access.

7) cert
The client must present a valid SSL certificate issued by certificate Authority (CA).
Then, the connection to the database is possible for that client.

8) PAM
(Pluggable Authentication Module) The pam method uses the PAM framework to authenticate PostgreSQL users.

Sundaram
FOR EDUCATIONAL USE
Scanned with CamScanner

* Pgbadger-

which parameters we set for enabling pgbadger reports? (in postgreSQL.conf file)

→ 1) Logging_collector =on
(enables collection of log messages)

2) log_directory
log=directory = 'pg_log' (or) any other specify directory where logs are stored.

3) log_filename
Defines naming pattern for log files.

4) log_rotation_size
Defines max. size of a log file before it is rotated.

5) log_statement
log_statement = all (for all detailed logs)

6) log_min_duration_statement
specify min. execution time for a statement to be logged.

7) log_rotation_age
sets the max. age of a log file before it is rotated.

Scanned with CamScanner

Scenario-Based Questions (for L2 DBA)

1) Role of WALs in PostgreSQL?
→ WAL ensures durability and enables crash recovery.

2) what if pg_xlog directory gets full?
→ Clear space or archive WAL files.

3) How would you handle performance issues where multiple queries are causing lock contention?
→ I will use 'pg_stat_activity' view & kill long-running queries.

4) How connection pooling works in PostgreSQL?
→ use 'pgBouncer' or 'pgpool' to manage database connections efficiently.

5) How would you configure PostgreSQL to use a custom data directory during installation?
→ I will specify custom directory while initialize data directory using command
initdb -D /custom/directory

6) Difference betn package installation & source code installation?
→ source code offers flexibility, but any dependcies & updates you have to do it manually. while Package installation is easier & faster to install & any dependencies automatically get installed. Also, updates are easy.

Sundaram
FOR EDUCATIONAL USE
Scanned with CamScanner

7) How would you install multiple versions of PostgreSQL on the same server?

→ Use Docker or configure different data directories & port for every version.

8) How would you setup roles & grant different privileges to each role?

→ I will create the roles. I will grant the necessary privileges to that roles. And then I will grant that role to the users according to the requirement.

9) How do you audit user activity in PostgreSQL?

→ I will use the pgaudit extension.
I will install pg_audit extension, then enable it in postgresql.conf file. then I will audit events that need to log like SELECT, INSERT.
(use ALTER ROLE or ALTER DATABASE to set audit parameters).

Also, I can use logging by setting parameters like log_statements, log_connections, log_disconnections, log_line_prefix in postgreSQL.conf file. The Logs can be found in the specified log directory.

Also, I can use pgBadger that can analyze logs & provide detailed reports on user activity & performances.

Sundaram
FOR EDUCATIONAL USE
Scanned with CamScanner

10) How do you revoke a user's access without dropping the user entirely?

→ I will use 'REVOKE ALL' to remove permissions while keeping the user.

11) How would you troubleshoot failed login attempts for a user?

→ I will check user's permissions, pg_hba.conf file and the logs for any errors.

12) How would you move a table from one schema to another?

→ I will use 'ALTER TABLE table_name SET SCHEMA schema_name_new;'

13) what is the relation betn schemas & users?

→ users can access multiple schemas. Schemas contains database objects like tables, views, indexes & functions. Schemas allow multiple users to create database objects.

users represent a database user or a group of users. If a role has 'USAGE' privilege on a schema, then it can access objects in that schema. roles manage access on schema for the users.

14) what steps would you take to secure a PostgreSQL database in a cloud environment?

→ I will use SSL, encryption & network level security control

(Secure Socket Layer) - a secured encrypted link between a server & a client.

Sundaram FOR EDUCATIONAL USE

Scanned with CamScanner

15) How do you implement role-based access control in PostgreSQL?

→ I will grant specific privileges to the role. & then grant that role to the users to access the only necessary resources.

16) How you will configure & monitor audit log?

→ Enabling logging using log_statement, log_duration or using pgaudit extension.

17) How would you implement table partitioning to improve query performance?

→ By partitioning large tables based on time ranges & key values.

18) what is the difference betn range partitioning & list partitioning in PostgreSQL?

→ Range partitioning divides data by ranges & list partitioning divides data by specific values.

19) How do you backup and restore partitioned tables?

→ I will use pg_dump to back up partitions and restore individually if needed.

20) How do you determine which columns need indexing in a large table?

→ Use 'EXPLAIN' or monitor slow queries to identify columns which need indexing.

Sundaram

FOR EDUCATIONAL USE

Scanned with CamScanner

21) How do you monitor & identify unused or redundant indexes?

→ use 'pg_stat_user_indexes' & 'pg_stat_all_indexes' views to check statistics.

22) what impact does index bloat have on query performance, and how do you fix it?

→

23) when would you use a VACCUM FULL versus a VACCUM REGULAR?

→ VACCUM FULL reclaims all space but locks the table, a user can't have read or write access.
VACCUM REGULAR reclaims all space but does not locks the table, a user can have access on that table.

24) How do you monitor Autovaccum processes & ensure they run optimally?

→

25) what happens if a VACCUM operation is interrupted?

→ The process restarts or rolls back if interrupted.

26) How do you configure PostgreSQL to generate logs compatible with pgbadger?

→ use log_line_prefix & enable detailed query logging.

Sundaram
FOR EDUCATIONAL USE
Scanned with CamScanner

27) what is the difference between postgresql.conf & postgresql.auto.conf file.

→ Postgresql.conf file is the main configuration file in postgreSQL, where you set all server parameters. while,

The PostgreSQL.auto.conf file, is automatically generated by PostgreSQL when you use the ALTER SYSTEM command to change configuration parameters.

28) If I have a full database backup on source server, can i restore particular table or tables from that backup file to target server?

(without -Fc)

→ Yes, If I have taken backup using pg_dump in plain text format, I can open the dump file & extract that table into a seperate file & then i will restore that specific table to the target server.

& also, If I have taken backup in custom format (using -Fc option), I can directly take table restoration from that dumpfile using pg_restore.

pg_restore -h hostname -d dbname -P port -U username -t table_name /path/to/your/dumpfile.

Sundaram
FOR EDUCATIONAL USE

Scanned with CamScanner

29) what is mean by plain text format & custom format backup in Postgresql's pg_dump command? what are extensions of backup file?

1. plain text format

The plain text format is a human-readable SQL script containing SQL statements. This format can be directly executed with psql to restore the database.

ex:- pg_dump -h hostname -U username -p port -d dbname > backup.sql

(or)

pg_dump -h hostname -U username -p port -d dbname -f /path/to/backup.sql

2. Custom format (-Fc)

The custom format is a binary format. It allows you to restore specific tables, schemas or other objects using pg_restore. This format cannot be viewed or edited.

ex:- pg_dump -U username -d dbname -p port -h hostname -Fc -f /path/to/backup.dmp

-Fc → custom format

-f → specifies the output file for the backup.

(Note:- extensions like .dump, .backup, .bkp, .pgdump) used in custom format backup)

Sundaram FOR EDUCATIONAL USE

Scanned with CamScanner

30) There are two databases in my postgresql server named db1 & db2. If I created a user 'ravi' in postgres database. will it be created in db1 & db2 databases.

→ users in postgresql are cluster-wide objects, & not database-specific objects.
Hence, 'ravi' user will be created in both db1 & db2 databases.

(Note:- even though the user exists in the entire cluster, they won't automatically have access to all databases. You need to grant the required privileges to the user).

31) I have two databases db1 & db2, The 'ravi' user is created by me. and I want to grant only 'SELECT' or read permissions to 'ravi' for db1 while I want to grant 'SELECT', 'INSERT', 'UPDATE' & 'DELETE' to 'ravi' for db2. How I can do it?

→ for db1, 'ravi' user will have connect, usage & SELECT on the public schema and tables.

for db2, 'ravi' user will have connect, usage, SELECT, INSERT, UPDATE and DELETE on the public schema and tables.

Sundaram
FOR EDUCATIONAL USE
Scanned with CamScanner

for db1,

GRANT CONNECT ON DATABASE db1 TO ravi;
GRANT USAGE ON SCHEMA public TO ravi;
GRANT SELECT ON ALL TABLES IN SCHEMA public TO ravi;

for db2,

GRANT CONNECT ON DATABASE db2 TO ravi;
GRANT USAGE ON SCHEMA public TO ravi;
GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA public TO ravi;

32) How to upgrade postgreSQL version 7 to postgreSQL version 15?

→ we can't directly upgrade version 7 to version 15. As, for that, we have to do sequential upgrades like from 7 to 8, 8 to 9, 9 to 10, 10 to 11, & so on until you reach 15.

version 7 does not support 'pg_upgrade' tool & from version 8 & so on 'pg_upgrade' tool is supported. So, we will first upgrade version 7 to 8 using pg_dump & pg_restore. Then, we will upgrade version 8 to 9, 9 to 12, 12 to 15 directly. Because, pg_upgrade allows you to directly take these jumps from version to version which minimizes steps, slowness. Also, pg_upgrade support direct major version upgrades.

Sundaram
FOR EDUCATIONAL USE
Scanned with CamScanner

33) pg-upgrade command.

→ ex:- upgrading version 9 to 12.

```
pg_upgrade -b /path/to/postgresql-9.6/bin -B
           /path/to/postgresql-12/bin
           -d /path/to/old/data_directory -D
           /path/to/new/data_directory -K
```

Here,

-K : Specifies or tells pg_upgrade (that) tool not to delete the old cluster's data files after a successful upgrade.

-K ensures that the old cus cluster's data files are kept for safety.

ex:- version 9 to 12.

```
pg_upgrade -b /path/to/postgresql-96/bin -B
           /path/to/postgresql-12/bin -d /path/to/old/
           data_directory -D /path/to/new/data_directory
           -O '-c config_file=/path/to/old/postgresql.conf'
           -O '-c config_file=/path/to/new/postgresql.conf'
```

Here, -O '-c config_file=/path/to/old/postgresql.conf', it tells pg_upgrade to apply defined parameters in old config. file to new upgrade file.

-O '-c config_file=/path/to/new/postgresql.conf', it directs the pg_upgrade to use specified Configuration file for new cluster, allowing custom setting for new environment.

Sundaram FOR EDUCATIONAL USE

Scanned with CamScanner

4) what methods does PostgreSQL provides to create a new database?

→ (i) using createdb (on server level)

Createdb dbname

(ii) on database level

Create database dbname;

Scanned with CamScanner