1. `wal_level = replica`

- **Purpose**: The `replica` level is used for physical replication (streaming replication) and point-in-time recovery (PITR).
- **Use case**: If you're setting up a physical standby server (for high availability) or performing backup and restore operations.
- **What is logged**:
  - Enough WAL data to ensure that a standby server can exactly replicate the primary server.
  - It logs all changes to the database but in a format designed for exact block-level replication.
- **Impact on performance**: This level logs more information than `minimal` but is still optimized for performance since it only writes the data needed for physical replication.

Best suited for:

- Physical streaming replication.
- Point-in-time recovery (PITR).
- Disaster recovery solutions that rely on exact block replication.

2. `wal_level = logical`

- **Purpose**: The `logical` level is used for logical replication, which allows replicating subsets of data (specific tables, rows, or even specific changes) to another server, and for using logical decoding (e.g., decoding WAL logs for change data capture (CDC)).
- **Use case**: This level is necessary when using logical replication or features like logical decoding, which are often used in data integration, replication of partial data (e.g., specific tables), or in cases where you want to capture data changes in real time.
- **What is logged**:
  - All the information needed for logical replication, which includes changes to individual rows in a format that can be used by external systems to replay these changes.
  - More detailed logging than `replica`, including changes at the row level with additional metadata needed for logical decoding.
- **Impact on performance**: Since it logs more detailed information than `replica`, there can be a higher performance overhead. However, it enables powerful features like selectively replicating specific changes to a downstream system.

Best suited for:

- Logical replication.
- Streaming real-time data changes to an external system (change data capture).
- Database migrations or replication of subsets of data.
- Integration with external systems (e.g., event-driven architectures).

Key Differences:

Feature	`replica`	`logical`
Type of Replication	Physical replication (entire database)	Logical replication (partial or entire database)
Level of Detail	Logs enough information for exact replication	Logs detailed row-level changes with extra metadata
Use Case	Standby servers, PITR	Logical replication, change data capture
Performance Impact	Lower than `logical` due to less detailed logging	Slightly higher due to detailed logs for logical decoding
Replication Granularity	Whole database replication	Specific tables, rows, or even individual changes

Summary:

- Use **`wal_level = replica`** if you are only interested in physical replication or PITR.
- Use **`wal_level = logical`** if you need logical replication, logical decoding, or change data capture.

Logical replication provides more flexibility but comes at a higher performance cost compared to physical replication.

VACUUM

Purpose: Removes dead rows and marks space available for reuse.
Space Handling: Does not return space to the operating system.
Space Reclamation: Space is reclaimed only if obsolete rows are at the end of a table.

VACUUM FULL

Purpose: More aggressive than regular VACUUM.
Compaction: Compacts tables by creating a completely new version of the table without dead space.
Time and Resources: Takes more time and requires extra disk space for the new copy of the table until the operation completes.

Creating **vacuum_test** table with data

```
postgres=# CREATE TABLE vacuum_test (id INT PRIMARY KEY, name
VARCHAR NOT NULL);

CREATE TABLE

postgres=# insert into vacuum_test select
generate_series(1,1000000),md5(generate_series(1,1000000)::text);
```

```
INSERT 0 1000000

postgres=#  \dt+

                              List of relations

 Schema |    Name     | Type  |  Owner   | Persistence |  Size  |
Description

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

 public | color       | table | postgres | permanent   | 16 kB  |

 public | employee    | table | postgres | permanent   | 104 kB |

 public | student     | table | postgres | permanent   | 16 kB  |

 public | test        | table | postgres | permanent   | 75 MB  |

 public | vacuum_test | table | postgres | permanent   | 65 MB  |

(5 rows)
```

vacuum_test Table size is 65MB

Deleting the half of the data

```
postgres=# delete from vacuum_test where id < 500000;

DELETE 499999
```

After deleting also space is same 65MB , it has not reduced, Space is not release from the table

```
postgres=#  \dt+

                              List of relations

 Schema |    Name     | Type  |  Owner   | Persistence |  Size  |
Description

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

 public | color       | table | postgres | permanent   | 16 kB  |

 public | employee    | table | postgres | permanent   | 104 kB |

 public | student     | table | postgres | permanent   | 16 kB  |

 public | test        | table | postgres | permanent   | 75 MB  |

 public | vacuum_test | table | postgres | permanent   | 65 MB  |

(5 rows)
```

```
postgres=# select schemaname,relname,n_dead_tup,last_autovacuum from
pg_stat_user_tables where relname='vacuum_test';
 schemaname |   relname   | n_dead_tup |         last_autovacuum
------------+-------------+------------+---------------------------------
-----
 public     | vacuum_test |          0 | 2024-09-21
15:08:46.55958+05:30
(1 row)
```

Before Full Vacuum file path

```
postgres=# select pg_relation_filepath('vacuum_test'::regclass);
 pg_relation_filepath
----------------------
 base/14448/16445
(1 row)
```

```
postgres=# \dt+ vacuum_test
                              List of relations
 Schema |    Name     | Type  |  Owner   | Persistence | Size  |
Description
--------+-------------+-------+----------+-------------+-------+----
---------
 public | vacuum_test | table | postgres | permanent   | 65 MB |
(1 row)
`
```

postgres=# vacuum FULL vacuum_test;

Table size and path has changed after vacuum FULL:

```
postgres=#  \dt+ vacuum_test
                              List of relations
 Schema |    Name     | Type  |  Owner   | Persistence | Size  |
Description
--------+-------------+-------+----------+-------------+-------+----
---------
 public | vacuum_test | table | postgres | permanent   | 33 MB |
(1 row)

postgres=#  select pg_relation_filepath('vacuum_test'::regclass);

 pg_relation_filepath

----------------------

 base/14448/16455

(1 row)
```