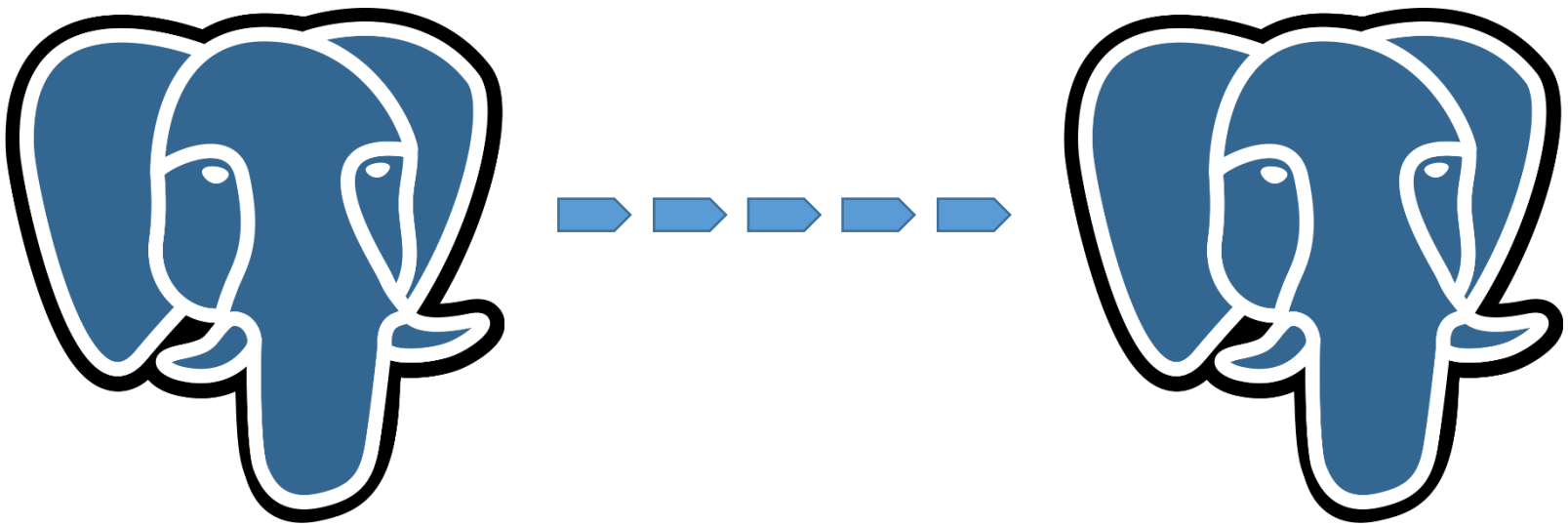
Built-in Physical and Logical Replication in Postgresql



Fırat Güleç

- Company
  Hepsiexpress

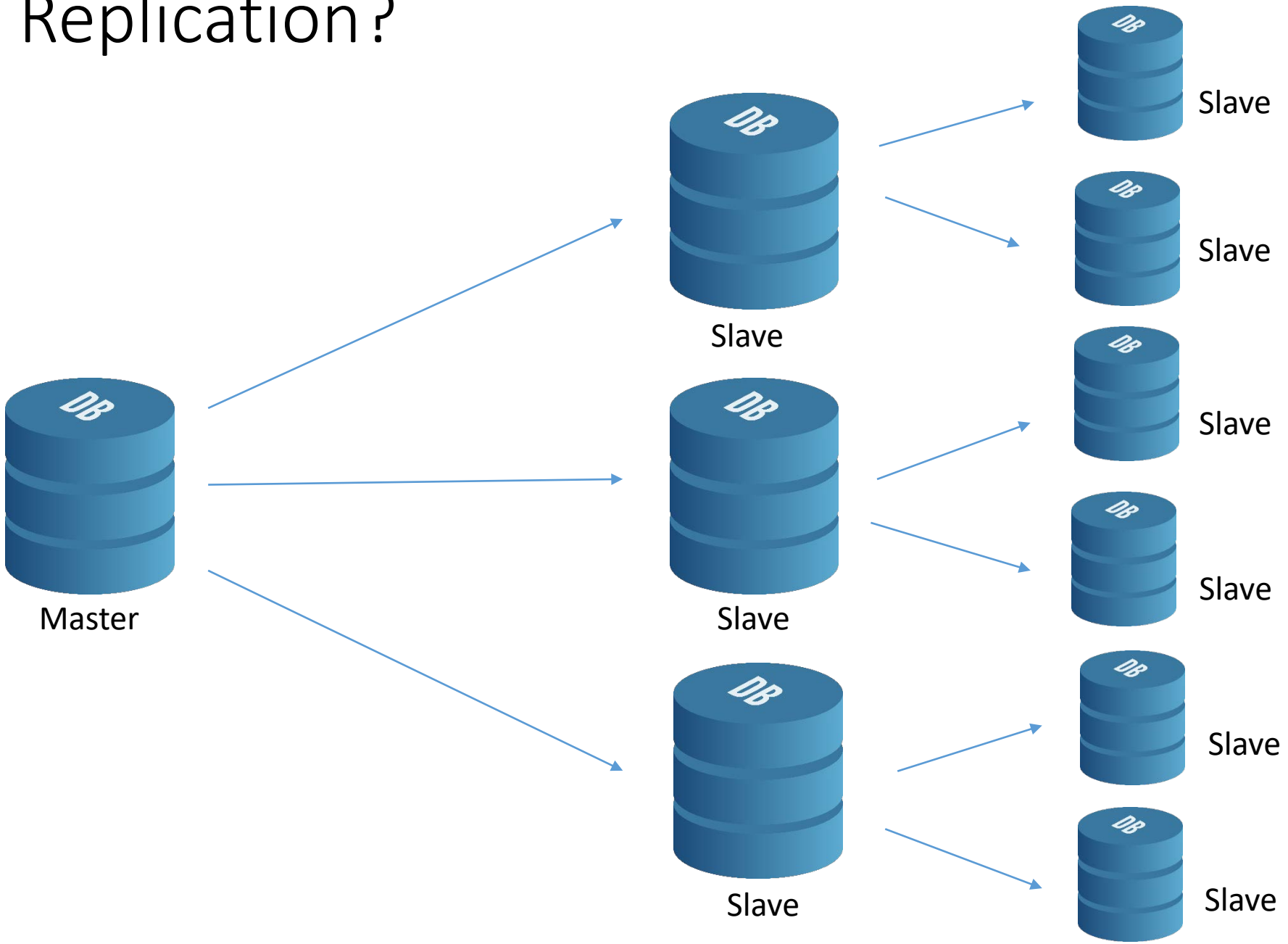
- Now
  Infrastructure & Database Manager
  Member of Postgresql Europe
  Contributing to Open Source Community

- Next Talks
  Austria PGDay 2019, September 6th, 2019

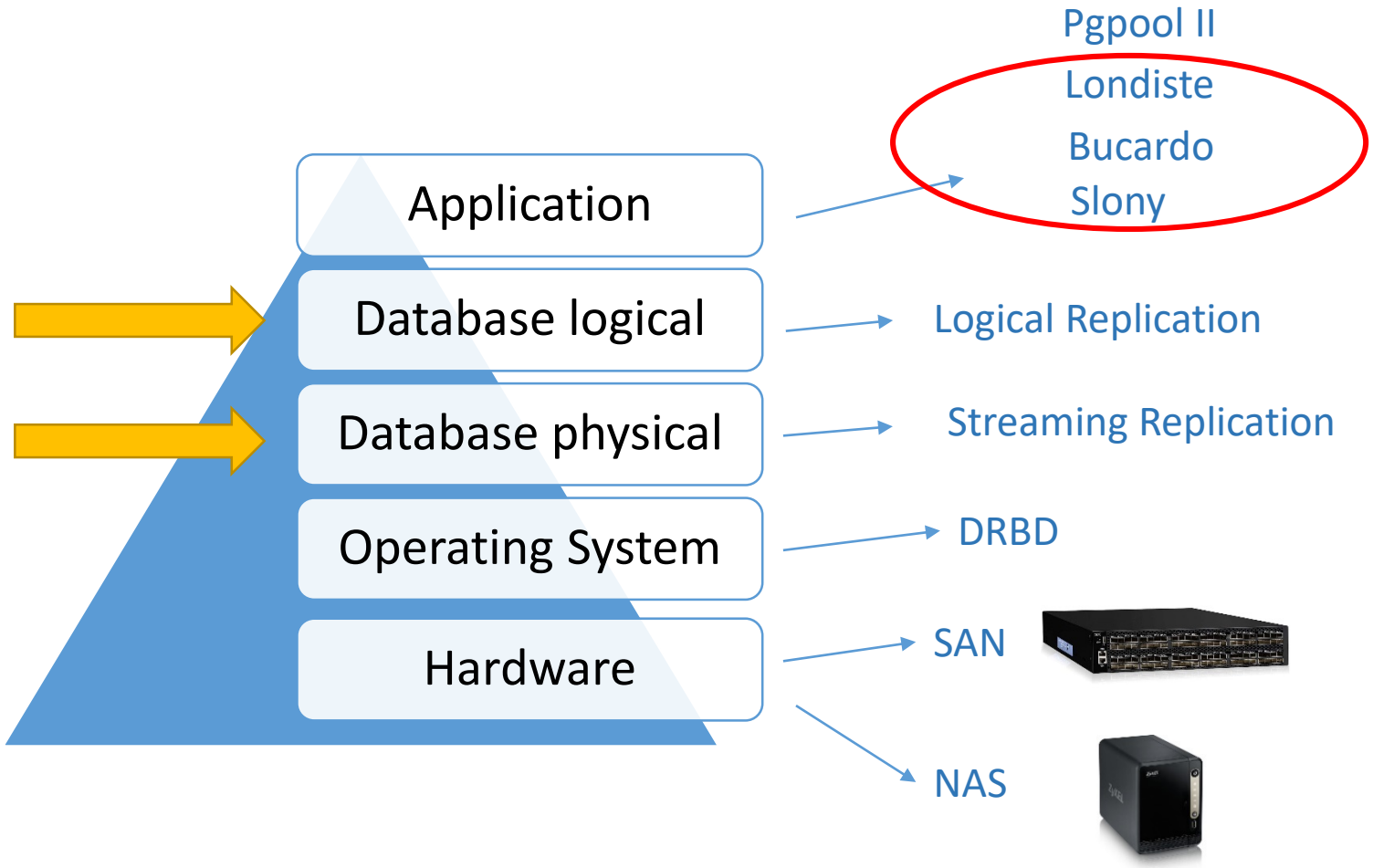
Agenda

- What is Replication?
- Why do we need Replication?
- How many replication layers do we have?
- Understanding milestones of built-in Database Physical Replication.
- What is the purpose of replication? and How to rescue system in case of failover?
- What is Streaming Replication and what is its advantages? Async vs Sync, Hot standby etc.
- How to configurate Master and Standby Servers? And What is the most important parameters? Example of topoloji.
- What is Cascading Replication and how to configurate it? Live Demo on Terminal.
- Quorum Commit for Sync Replication etc.
- What is Logical Replication coming with PostgreSQL 10? And What is its advantages?
- What is the purpose of Logical replication?
- How to set up Logical Replication and What are its benefits?
- Limitations of Logical Replication
- Logical Replication vs Physical Replication in detail.
- 10 Questions quiz and giving some gifts to participants according to their success.

What is Replication?



Replication Layers



High Availability, Load Balancing, and Replication Feature Matrix

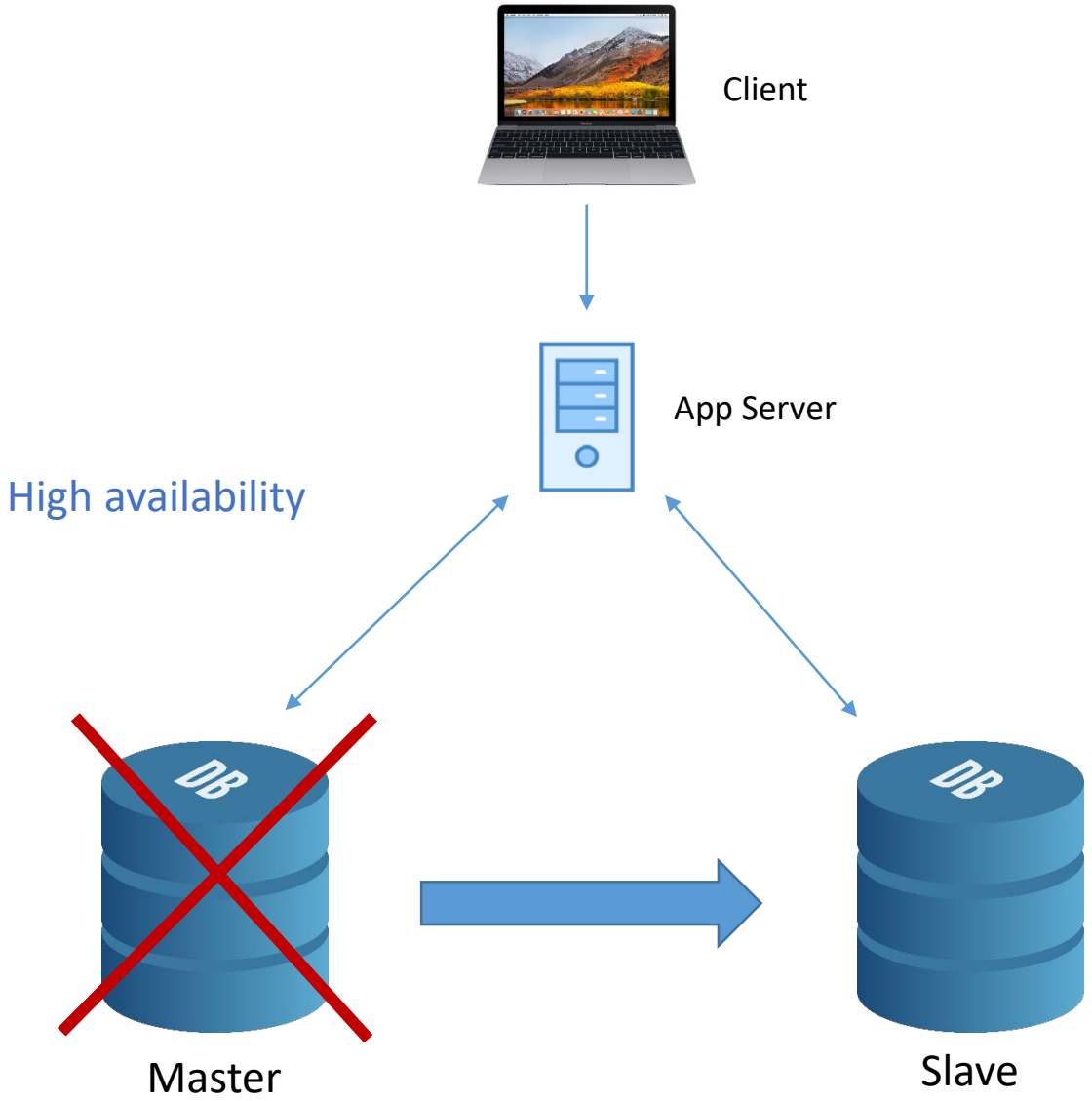
Table 26.1. High Availability, Load Balancing, and Replication Feature Matrix

Feature	Shared Disk Failover	File System Replication	Write-Ahead Log Shipping	Logical Replication	Trigger-Based Master-Standby Replication	Statement-Based Replication Middleware	Asynchronous Multimaster Replication	Synchronous Multimaster Replication
Most common implementations	NAS	DRBD	built-in streaming replication	built-in logical replication, pglogical	Londiste, Slony	pgpool-II	Bucardo	
Communication method	shared disk	disk blocks	WAL	logical decoding	table rows	SQL	table rows	table rows and row locks
No special hardware required		•	•	•	•	•	•	•
Allows multiple master servers				•		•	•	•
No master server overhead	•		•	•		•		
No waiting for multiple servers	•		with sync off	with sync off	•		•	
Master failure will never lose data	•	•	with sync on	with sync on		•		•
Replicas accept read-only queries			with hot standby	•	•	•	•	•
Per-table granularity				•	•		•	•
No conflict resolution necessary	•	•	•		•			•

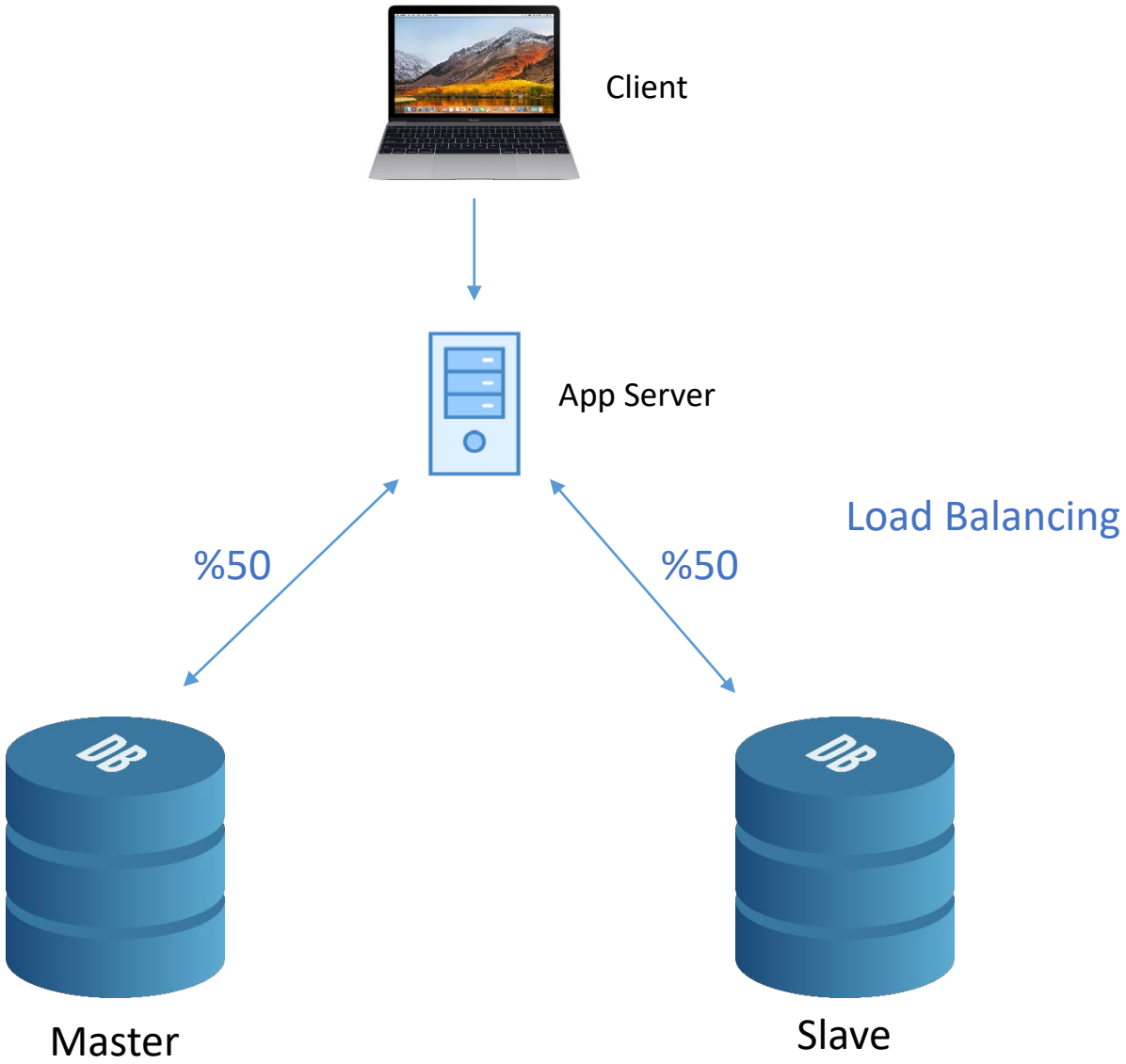
https://www.postgresql.org/docs/11/different-replication-solutions.html

Pyhsical Replication

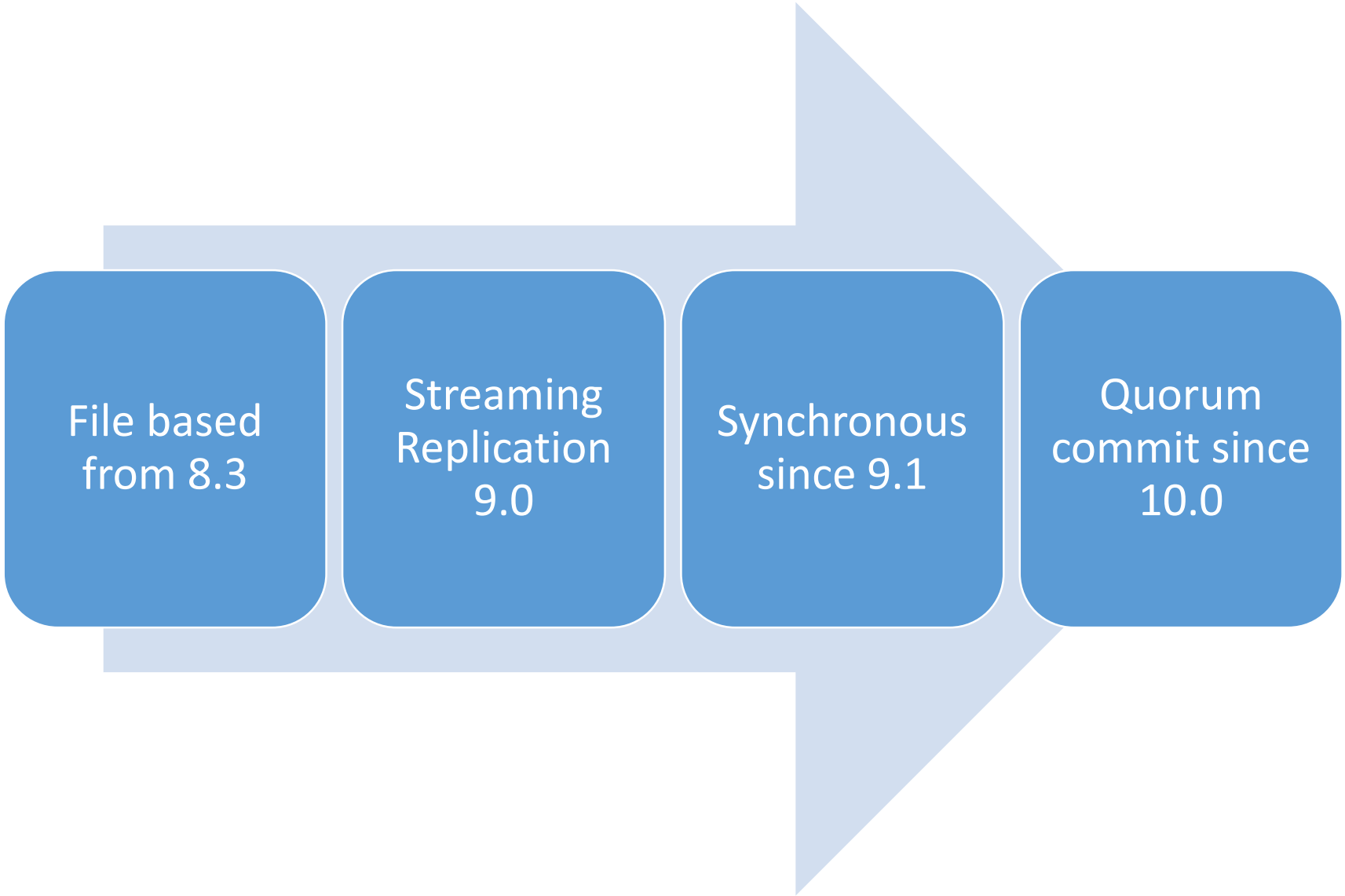
The purpose of Streaming Replication?



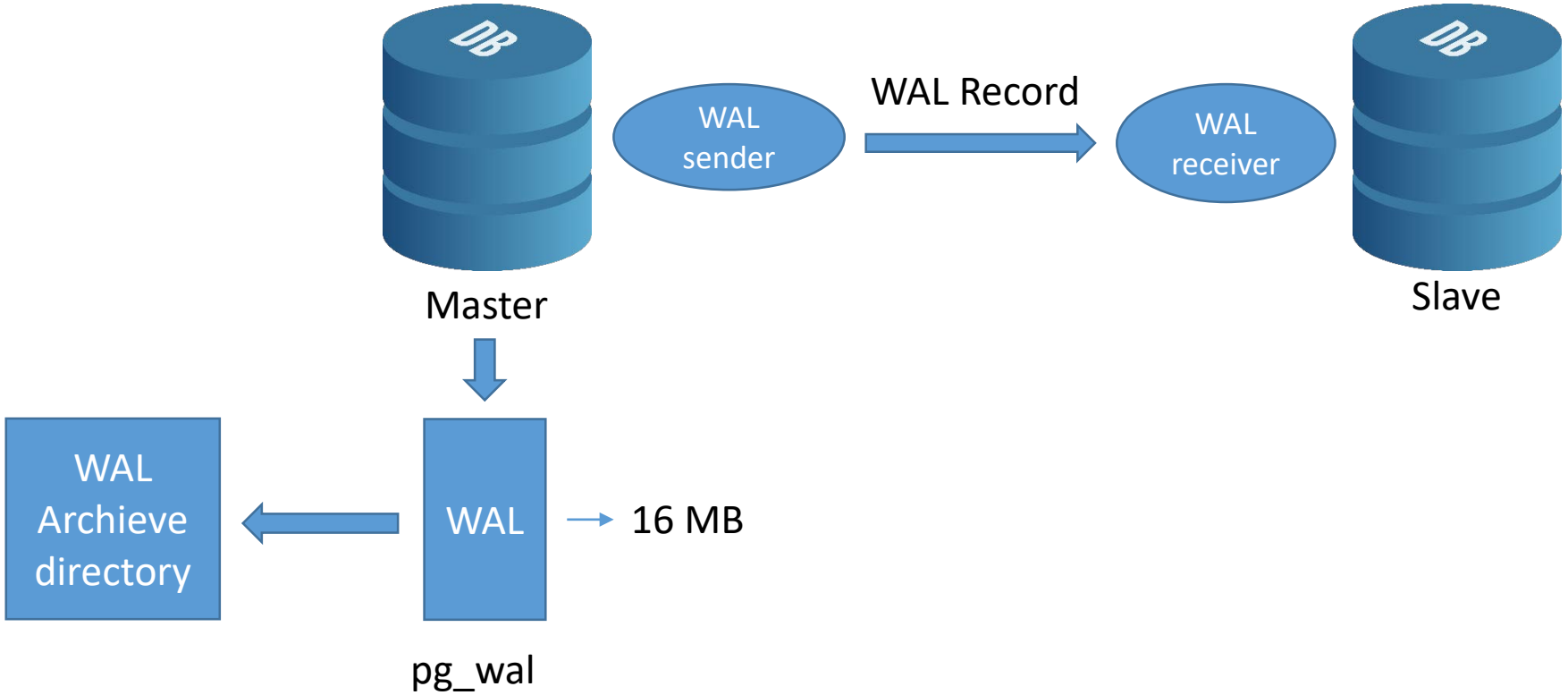
The purpose of Streaming Replication?



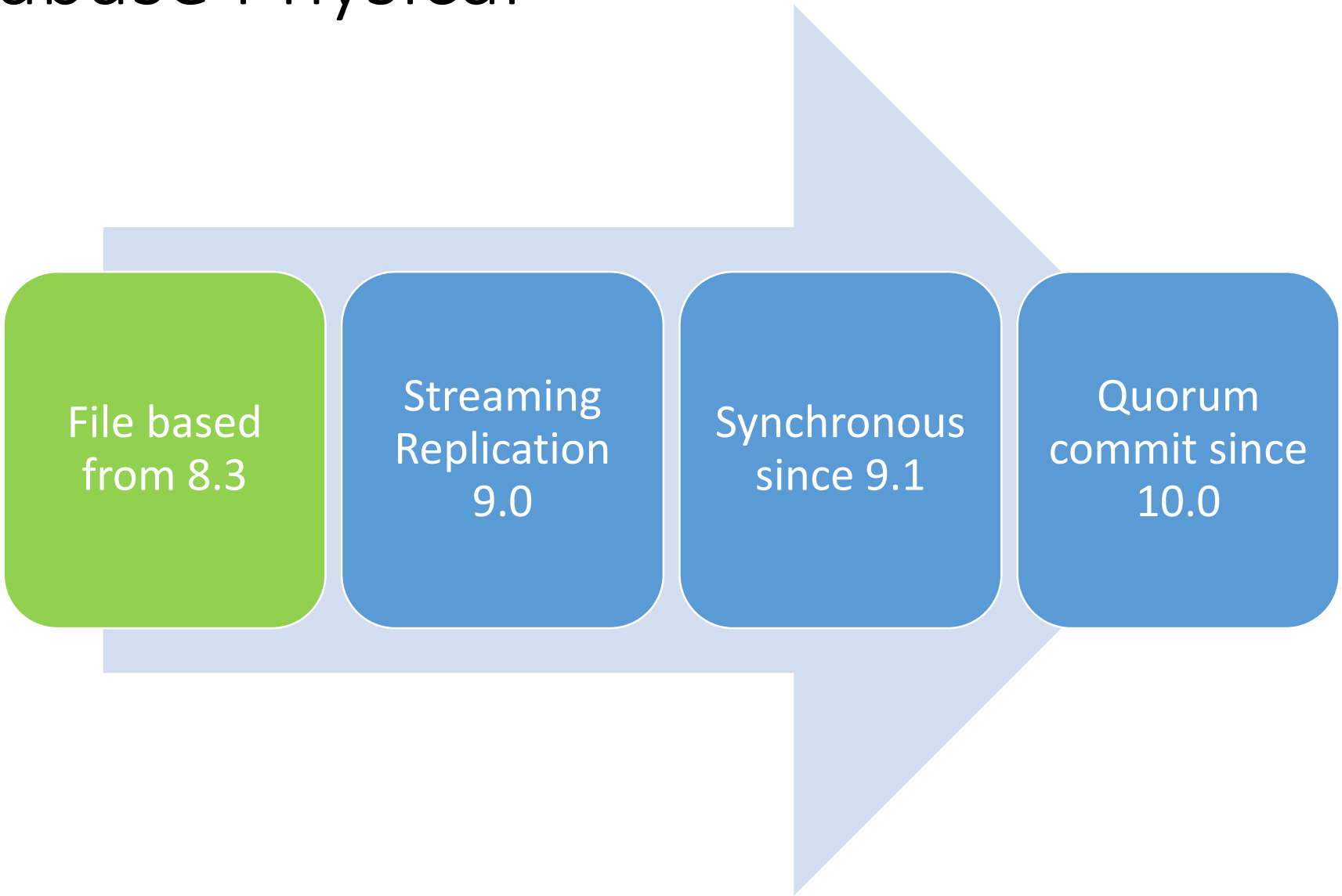
Database Physical Replication Milestones



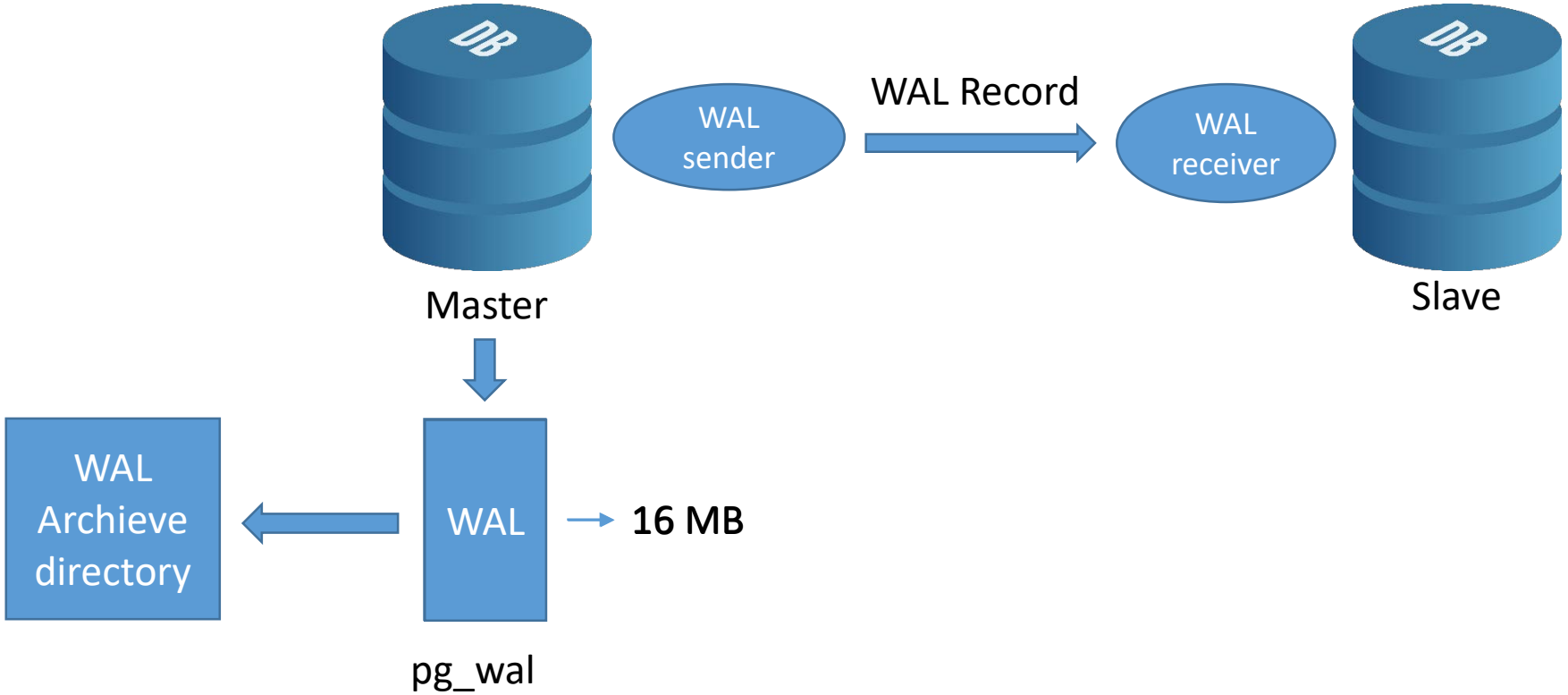
Architecture of Streaming Replication



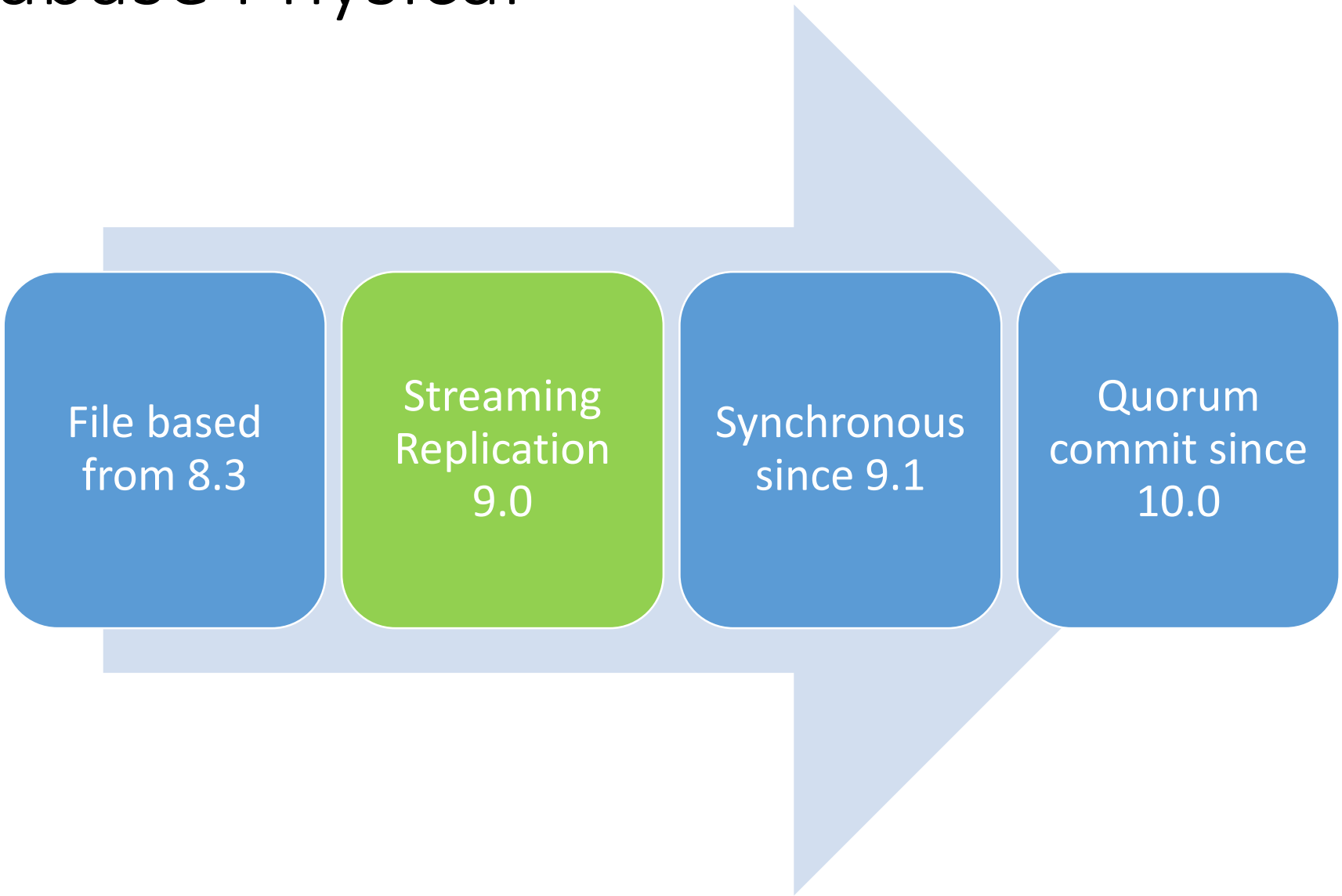
Database Physical



File based Replication from Postgresql 8.3



Database Physical



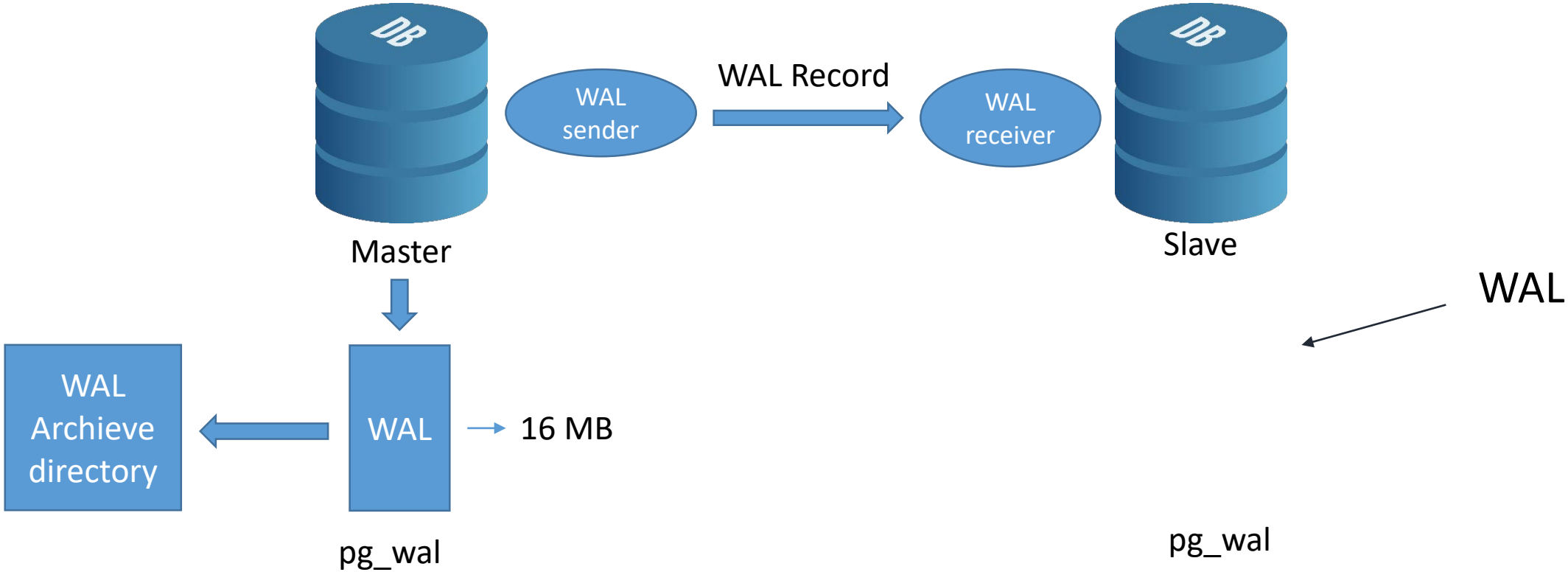
Streaming Replication

Record-based log shipping

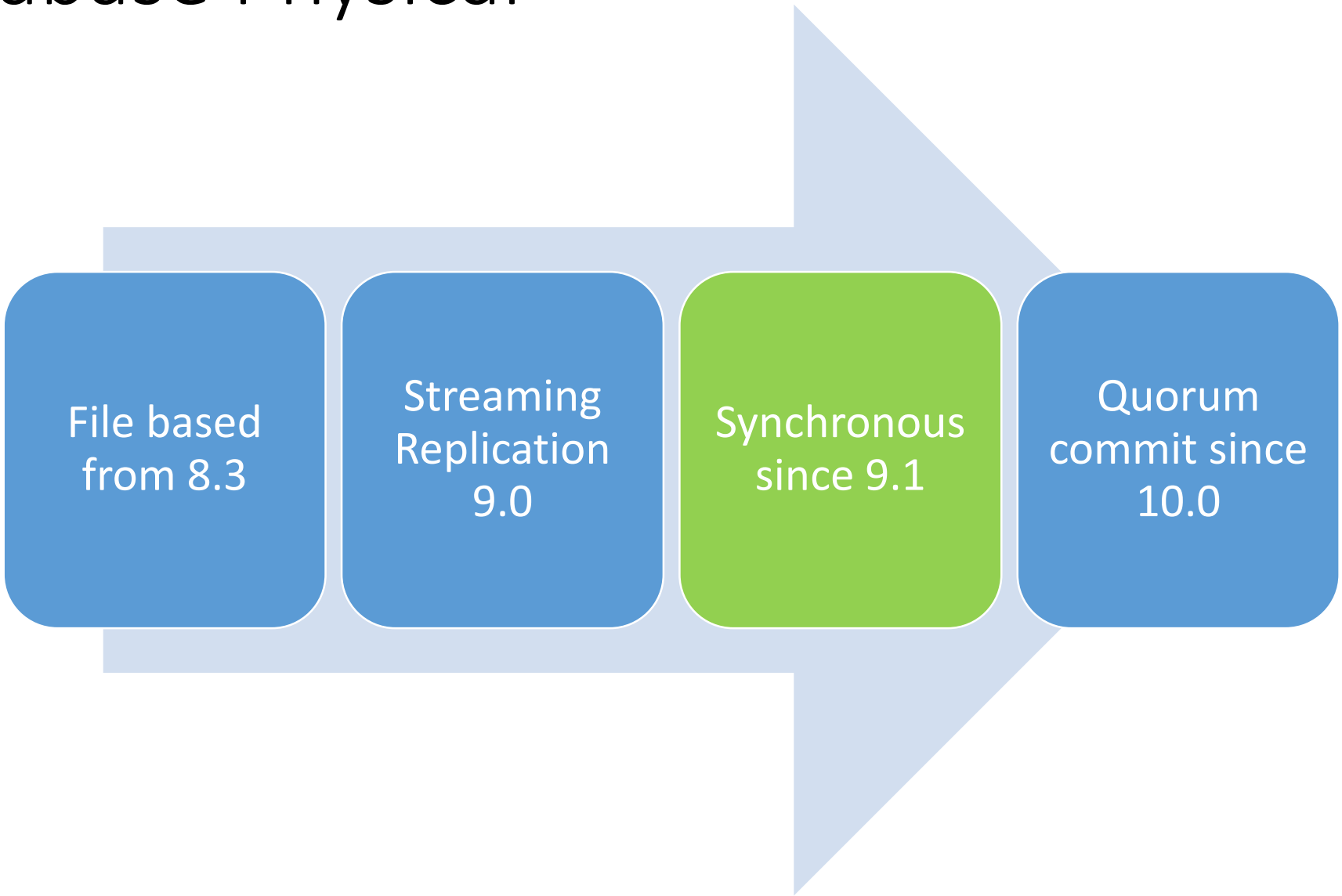
The primary and standby servers so that they are as similar as possible

1- Major PostgreSQL release levels is not possible

2- 32-bit to a 64-bit system will not work.



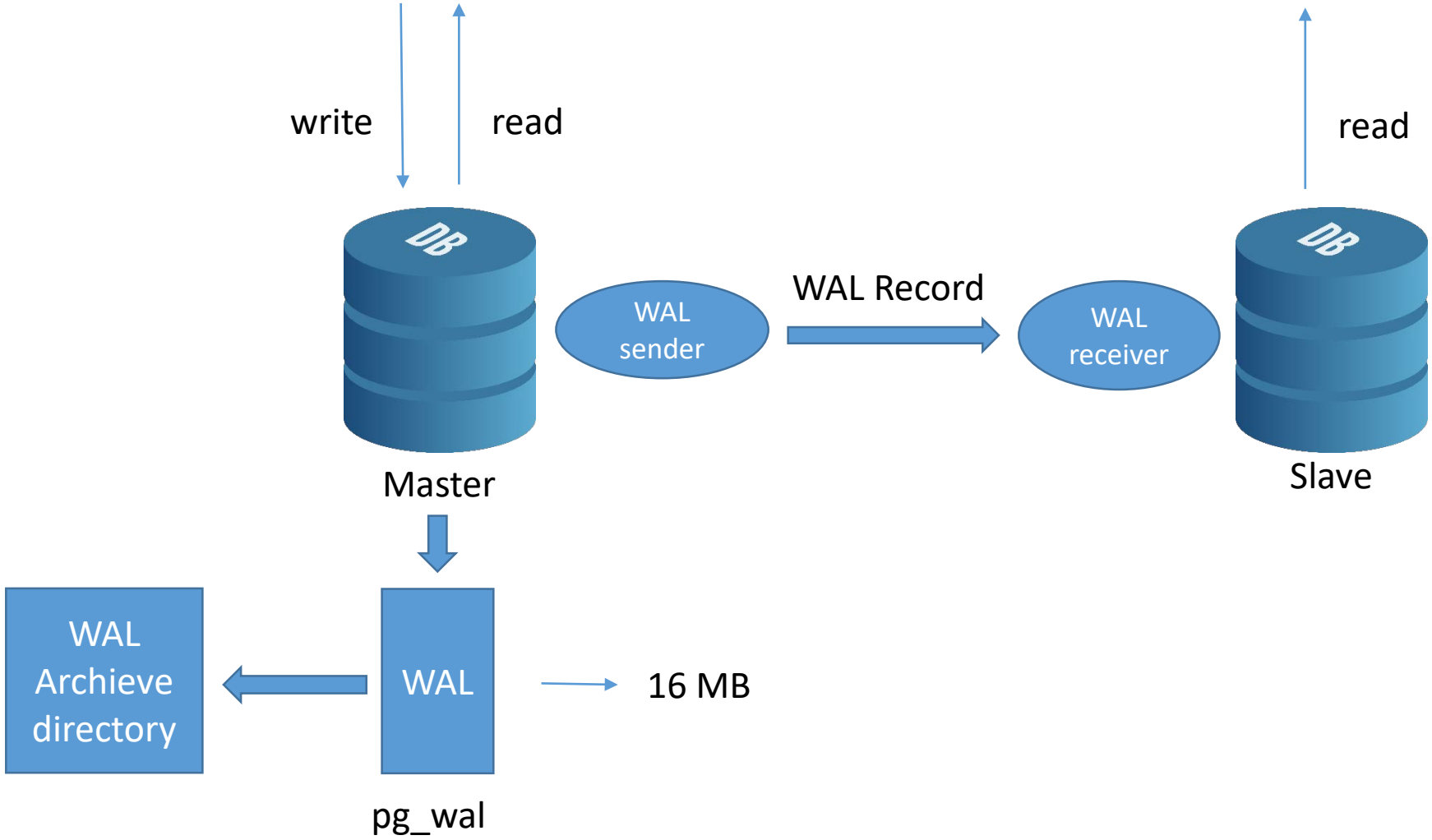
Database Physical



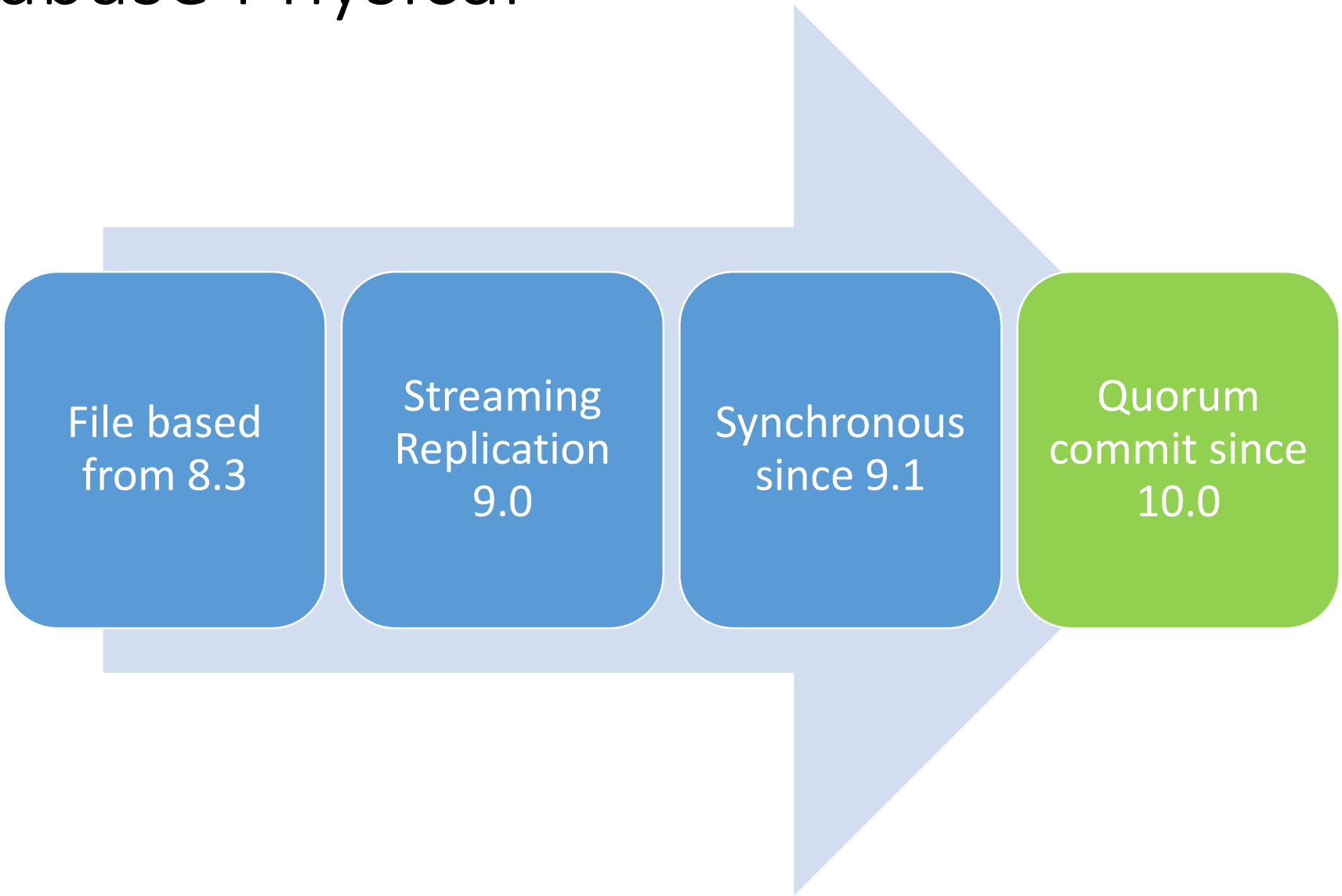
Streaming Replication

1- Async vs Sync

2- Hot Standby or Warm Standby?

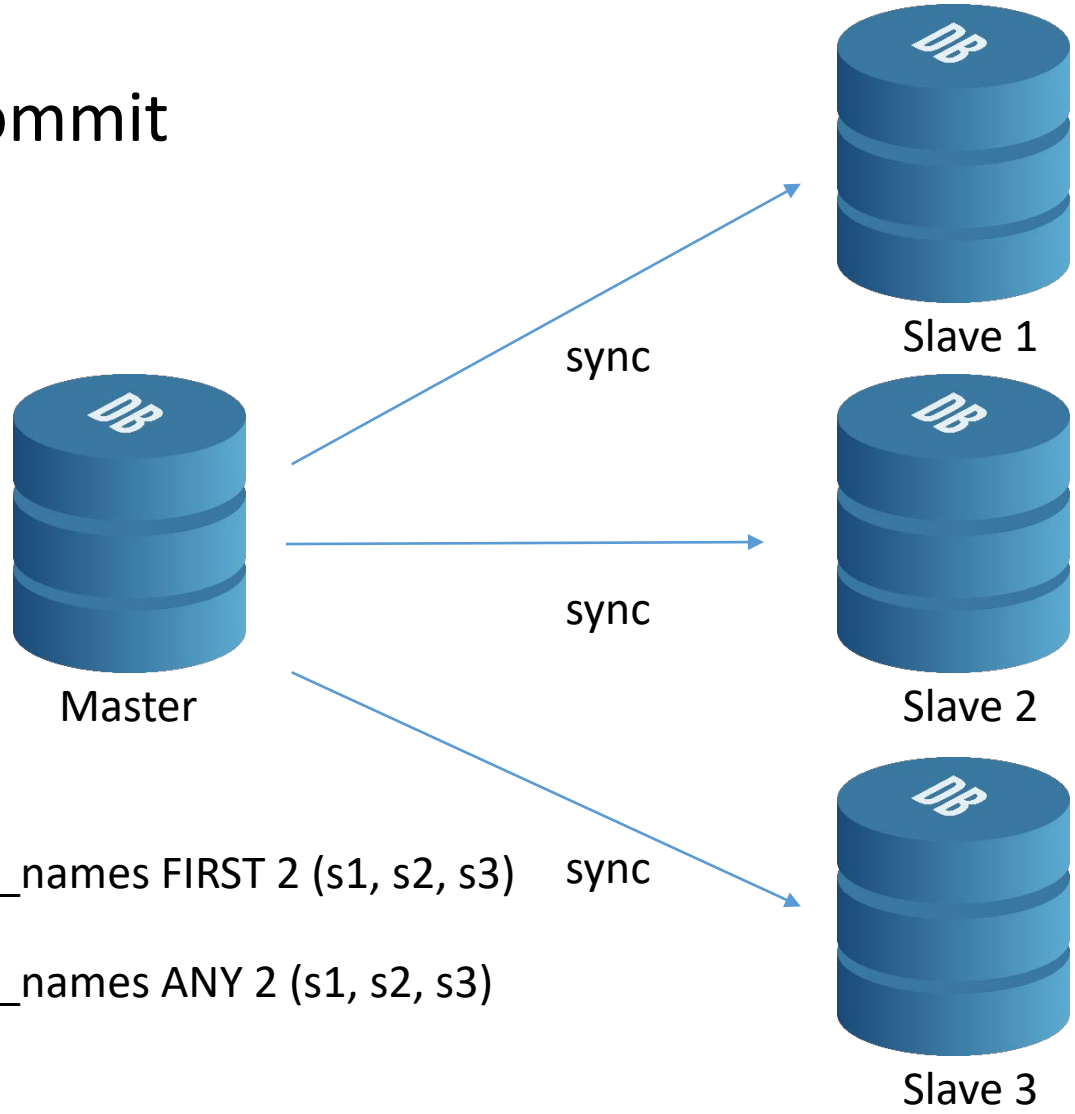


Database Physical



Quorum Commit for Sync Replication

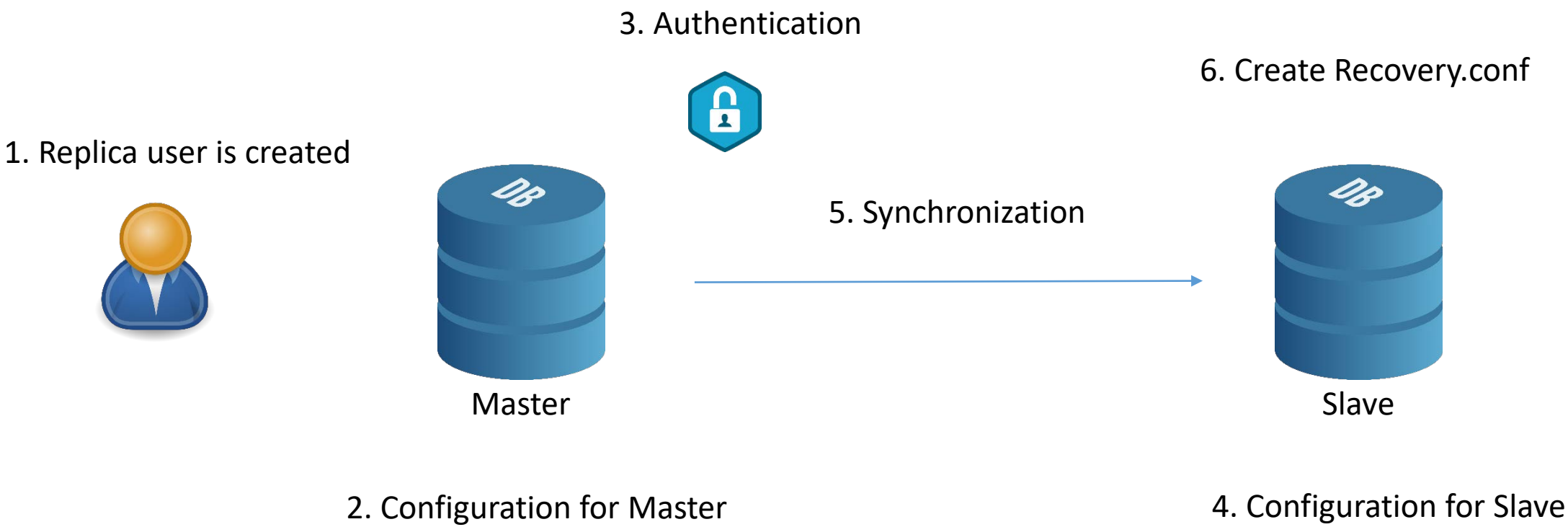
- 10.0: Quorum Commit



10.0→synchronous_standby_names FIRST 2 (s1, s2, s3)

10.0→synchronous_standby_names ANY 2 (s1, s2, s3)

6 Steps for Streaming Replication



Hot Standy - postgresql.conf

- wal_level → determines how much information is written

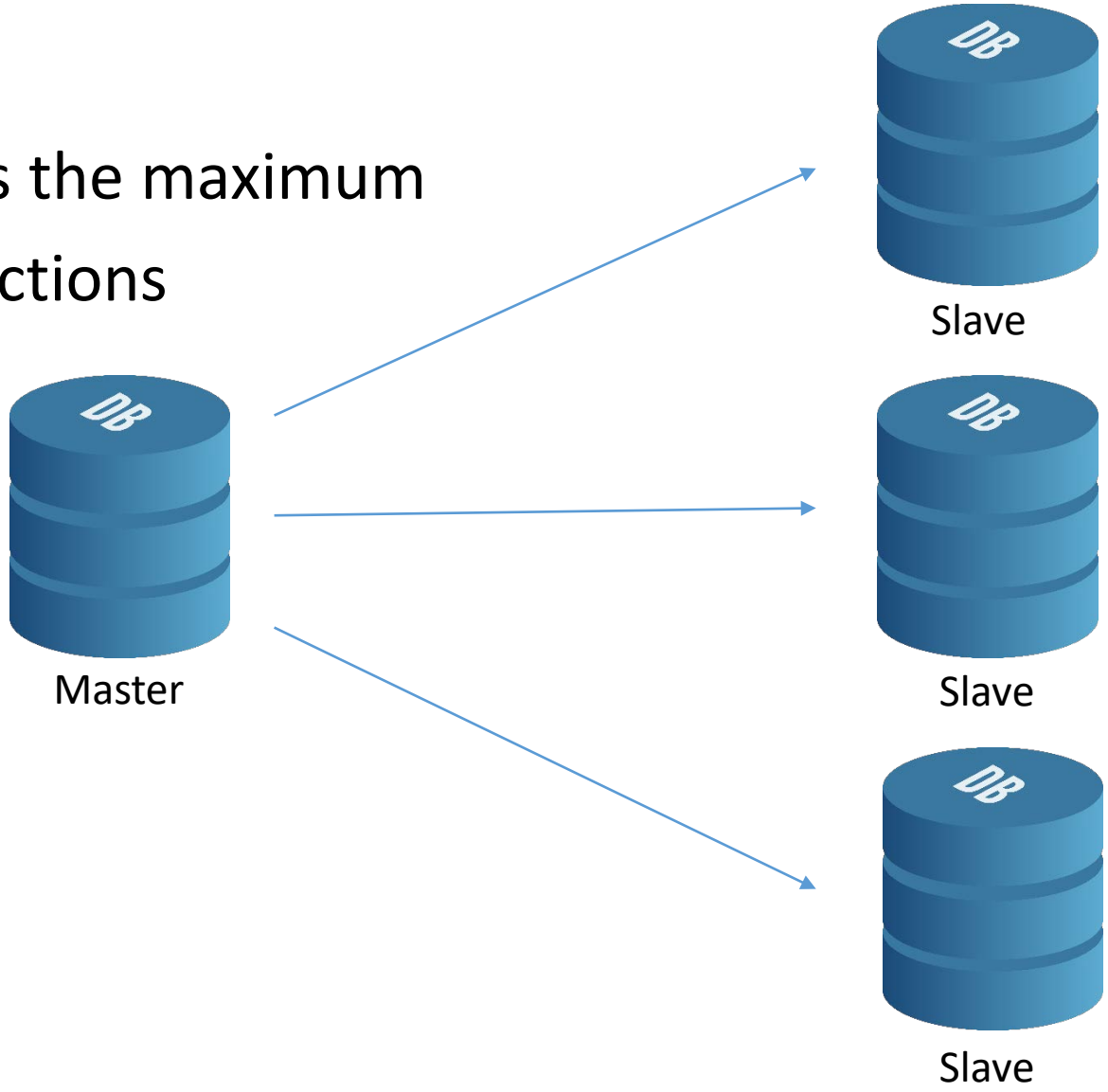
wal_level=‘minimal’

wal_level=‘replica’

wal_level=‘logical’

max_wal_senders

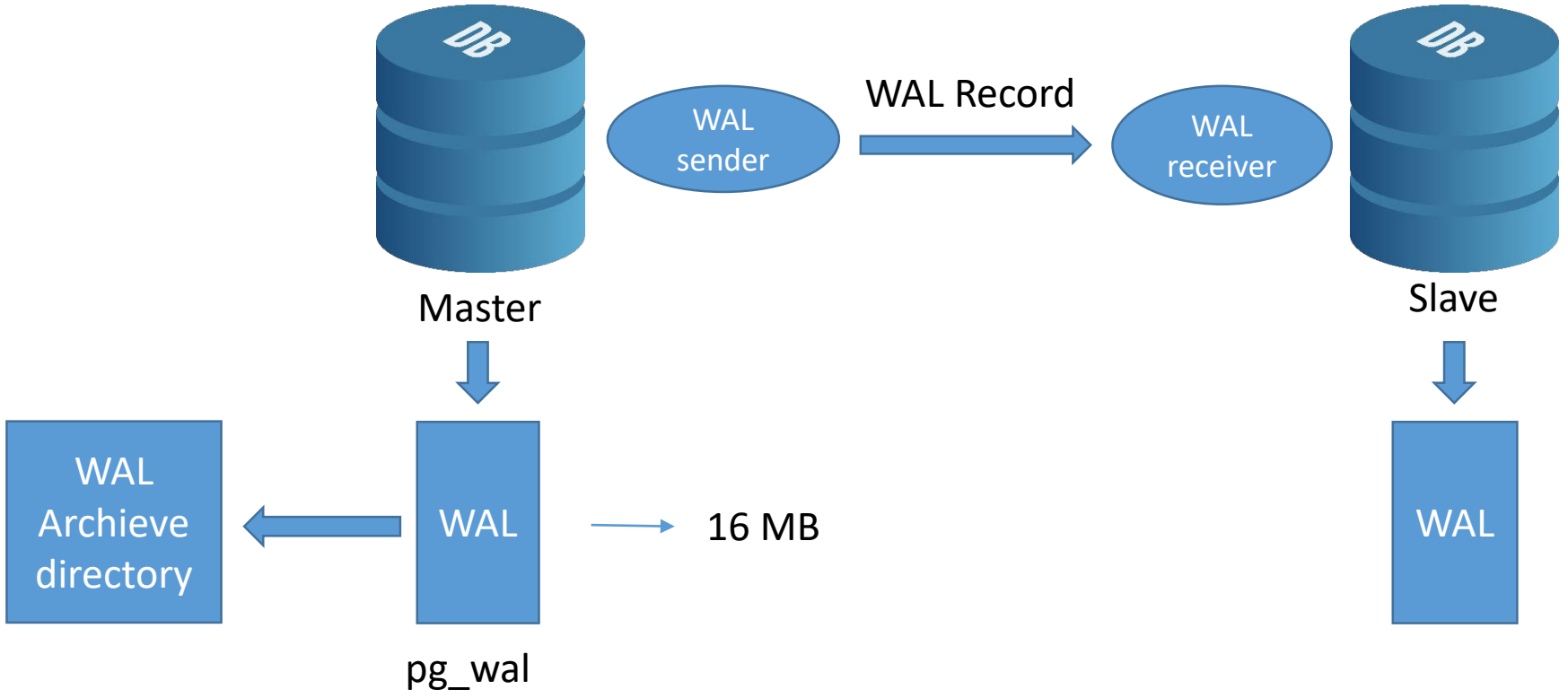
- max_wal_senders= specifies the maximum number of concurrent connections



wal_keep_segments

```
-rw------- 1 postgres postgres 16777216 May  2 14:51 0000000100000519000000BC
-rw------- 1 postgres postgres 16777216 May  2 14:51 0000000100000519000000BB
-rw------- 1 postgres postgres 16777216 May  2 14:51 0000000100000519000000BF
-rw------- 1 postgres postgres 16777216 May  2 14:51 0000000100000519000000BA
-rw------- 1 postgres postgres 16777216 May  2 14:51 0000000100000519000000BE
-rw------- 1 postgres postgres 16777216 May  2 14:51 000000010000051900000073
-rw------- 1 postgres postgres 16777216 May  2 14:57 000000010000051900000074
-rw------- 1 postgres postgres 16777216 May  2 14:57 000000010000051900000075
-rw------- 1 postgres postgres 16777216 May  2 14:57 000000010000051900000076
-rw------- 1 postgres postgres 16777216 May  2 14:57 000000010000051900000077
-rw------- 1 postgres postgres 16777216 May  2 14:57 000000010000051900000078
-rw------- 1 postgres postgres 16777216 May  2 14:57 000000010000051900000079
-rw------- 1 postgres postgres 16777216 May  2 15:18 0000000100000519000007A
-rw------- 1 postgres postgres 16777216 May  2 15:18 0000000100000519000007B
-rw------- 1 postgres postgres 16777216 May  2 15:18 0000000100000519000007C
-rw------- 1 postgres postgres 16777216 May  2 15:18 0000000100000519000007D
-rw------- 1 postgres postgres 16777216 May  2 15:18 0000000100000519000007E
-rw------- 1 postgres postgres 16777216 May  2 15:18 0000000100000519000007F
-rw------- 1 postgres postgres 16777216 May  3 10:47 000000010000051900000080
-rw------- 1 postgres postgres 16777216 May  3 13:46 000000010000051900000081
-rw------- 1 postgres postgres 16777216 May  3 13:46 000000010000051900000082
-rw------- 1 postgres postgres      364 May  3 13:46 000000010000051900000082.00000028.backup
-rw------- 1 postgres postgres 16777216 May  3 17:30 000000010000051900000083
-rw------- 1 postgres postgres 16777216 May  3 17:40 000000010000051900000084
-rw------- 1 postgres postgres 16777216 May  3 17:50 000000010000051900000085
-rw------- 1 postgres postgres 16777216 May  3 18:00 000000010000051900000086
-rw------- 1 postgres postgres 16777216 May  3 18:10 000000010000051900000087
-rw------- 1 postgres postgres 16777216 May  3 18:20 000000010000051900000088
drwx------ 2 postgres postgres  1073152 May  3 18:20 archive_status
-rw------- 1 postgres postgres 16777216 May  3 18:22 000000010000051900000089
root@dfast-postgresql:/var/lib/postgresql/11/main/pg_wal#
```

wal_keep_segments



1-Replication User for Master

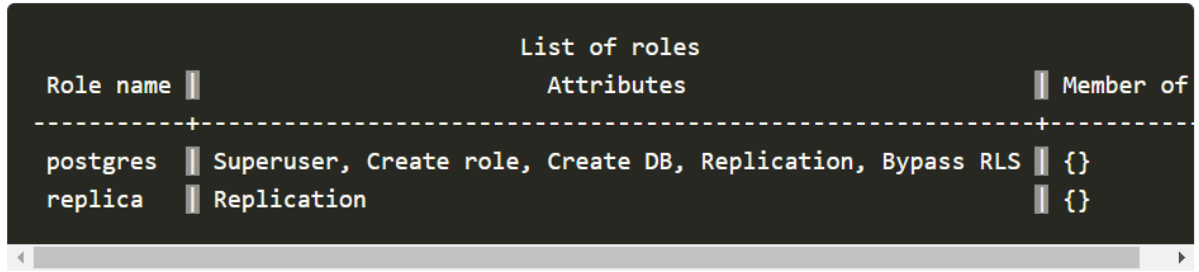
- sudo -u postgres psql

Next, create a new user and role with the following command:

- postgres=#CREATE USER replica REPLICATION LOGIN ENCRYPTED PASSWORD ‘*******';

postgres=#\du

- You should see the following output:



2-Hot Standby Configuration for Master

in postgresql.conf

- wal_level=replica
- wal_keep_segment=20
- max_wal_sender=3
- archieve_mode=on
- archive_command = 'test ! -f /var/lib/postgresql/pg_log_archive/%f && cp %p /var/lib/postgresql/pg_log_archive/%f'

3-pg_hba.conf configuration for Master

For authentication:

host replication replica 10.90.82.61/32 md5

4-Hot standy configuration for slave

In Postgresql.conf

- hot standby=on

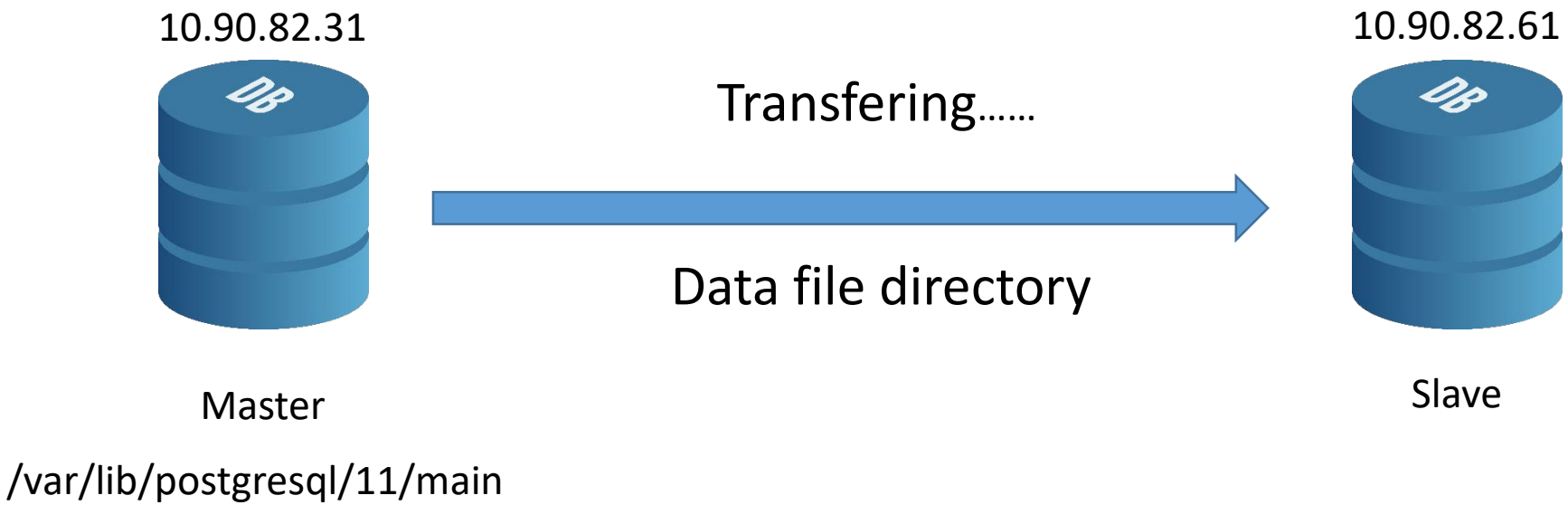
Below configuration in case of fail over

- archive_mode = on
- archive_command = 'test ! -f /var/lib/postgresql/pg_log_archive/%f && cp %p /var/lib/postgresql/pg_log_archive/%f'
- wal_keep_segment=20
- max_wal_sender=3

5-Syncronize Data from Master Server to Slave Server

On the slave server, stop the postgresql service:

- sudo systemctl stop postgresql and move existing data folder.
- pg_basebackup -h 10.90.82.31 -D /var/lib/postgresql/11/main/ -P -U replica --wal-method=fetch



6-Recovery.conf file on standby

Datafile Directory→/var/lib/postgresql/11/main

```
standby_mode = 'on'
primary_conninfo = 'host=192.168.1.110 port=5432 user=replica
password=[redacted]'
restore_command = 'cp //var/lib/postgresql/9.4/main/archive/%f %p'
trigger_file = '/tmp/postgresql.trigger.5432'
```

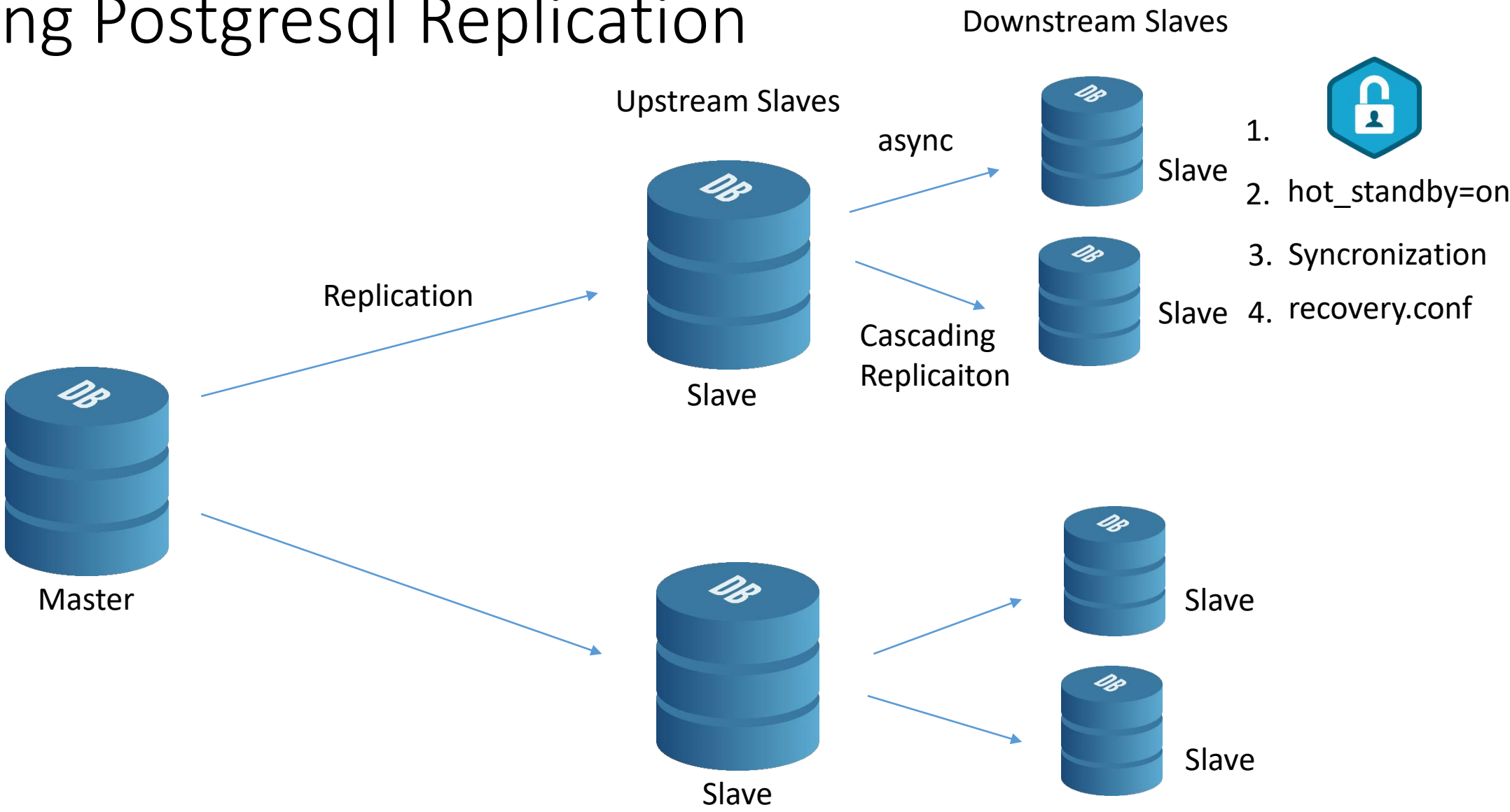
Test Replication

Streaming Replication is DONE

Command→psql -x -c "select * from pg_stat_replication;"

```
Password for user postgres:
-[ RECORD 1 ]----+------------------------------
pid              | 21027
usesysid         | 1426344
usename          | replica
application_name | walreceiver
client_addr      | 10.70.82.61
client_hostname  |
client_port      | 34526
backend_start    | 2019-05-08 18:21:49.737654+03
backend_xmin     |
state            | streaming
sent_lsn         | 53D/C17D3CE8
write_lsn        | 53D/C17D3CE8
flush_lsn        | 53D/C17D3CE8
replay_lsn       | 53D/C17D3CE8
write_lag        | 00:00:00.000554
flush_lag        | 00:00:00.002002
replay_lag       | 00:00:00.002004
sync_priority    | 0
sync_state       | async
```

Cascading Postgresql Replication



max_standby_archive_delay for standby

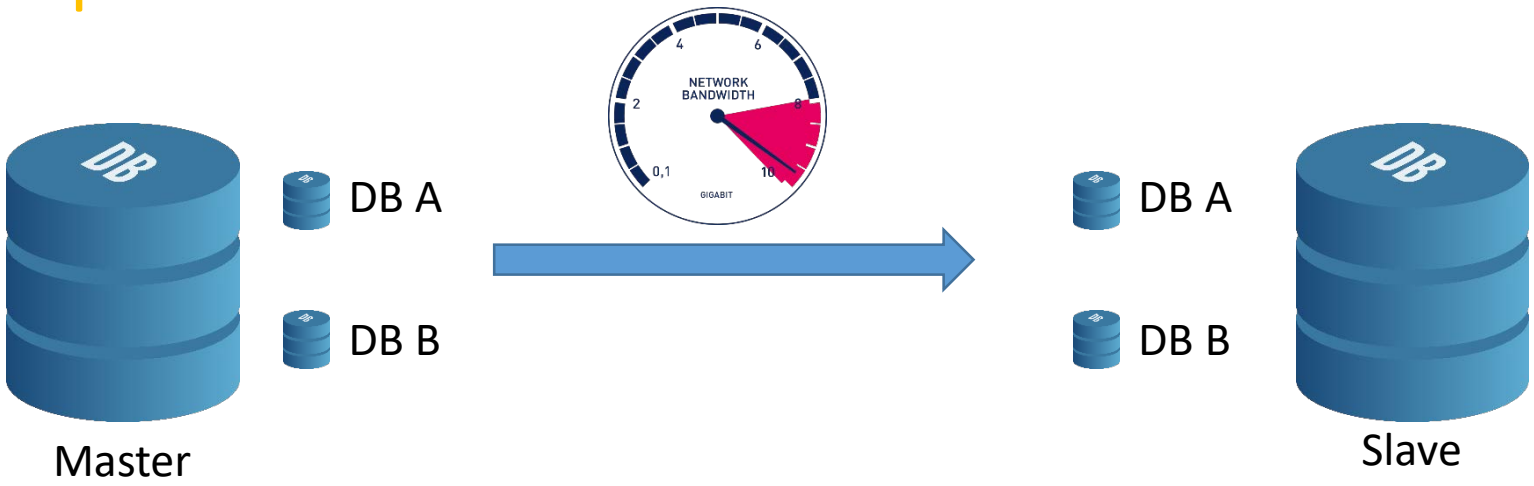
```
max_standby_archive_delay = 180s        # max delay before canceling queries
                                        # when reading WAL from archive;
                                        # -1 allows indefinite delay
max_standby_streaming_delay = 180s      # max delay before canceling queries
                                        # when reading streaming WAL;
                                        # -1 allows indefinite delay
```

Logical Replication

What is Logical Replication?

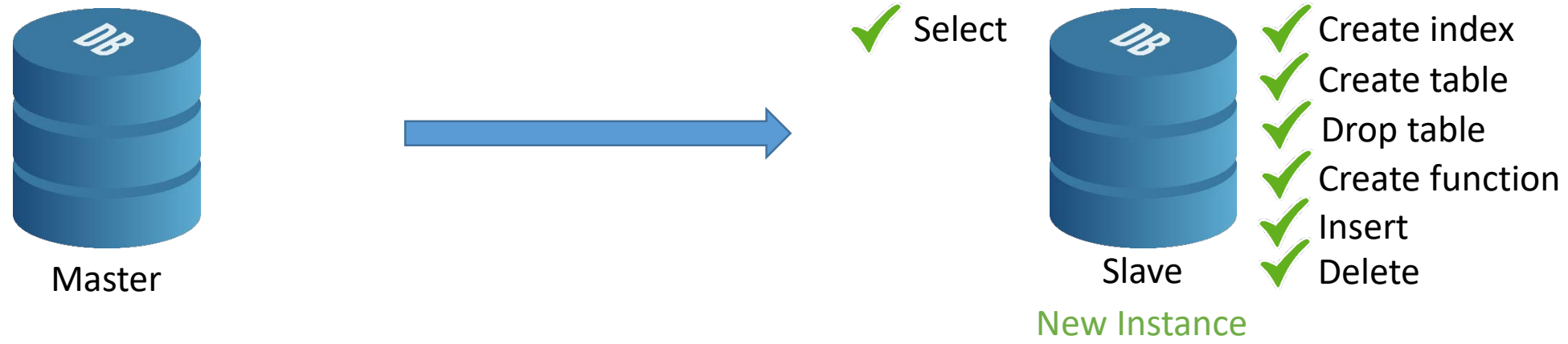
Logical replication

Streaming replication

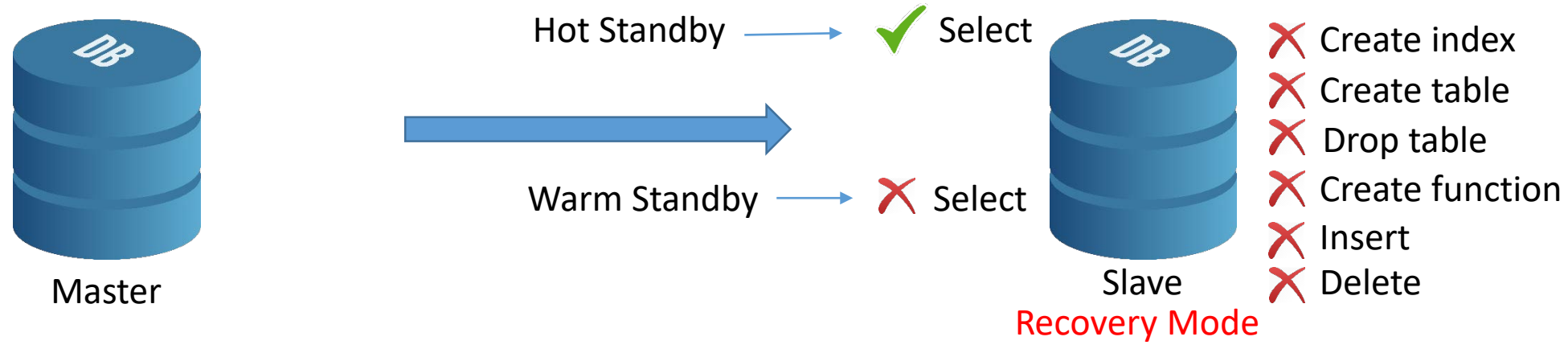


What is Logical Replication?

Logical replication

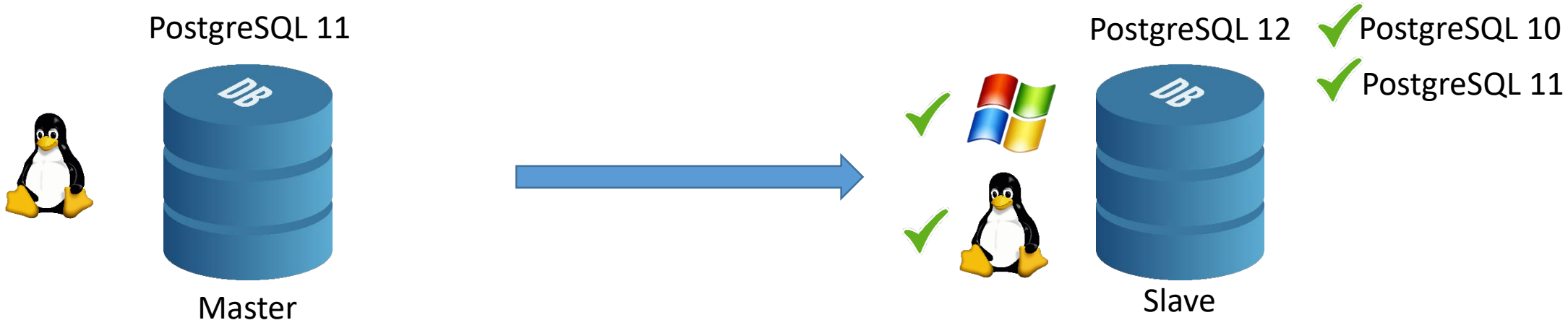


Streaming replication

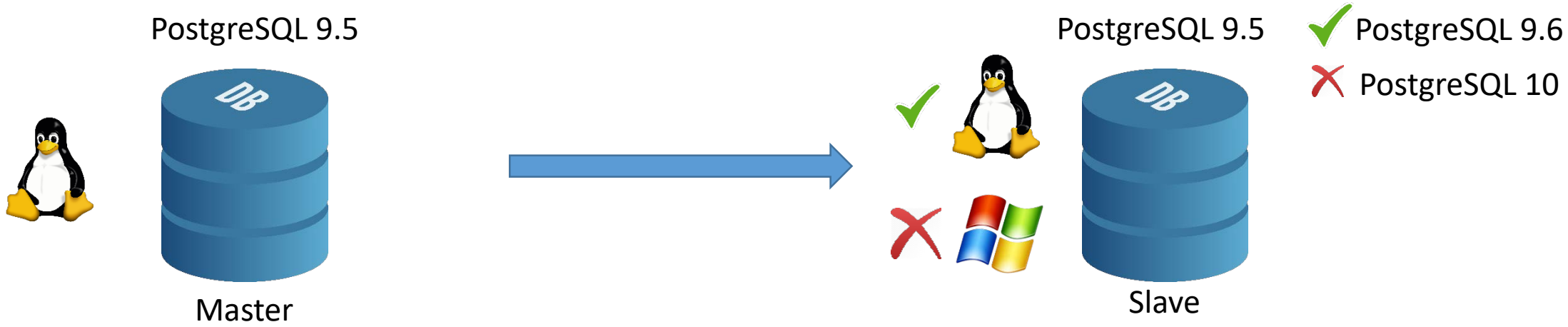


What is Logical Replication?

Logical replication

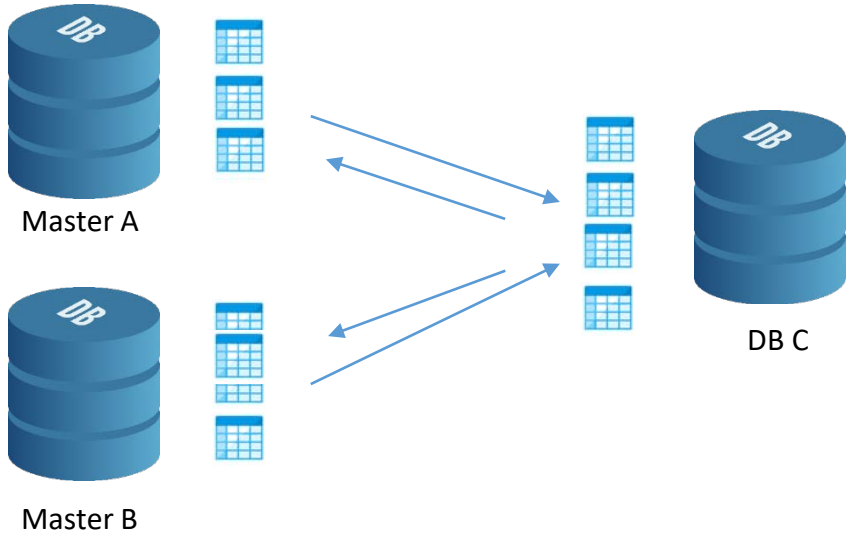


Streaming replication

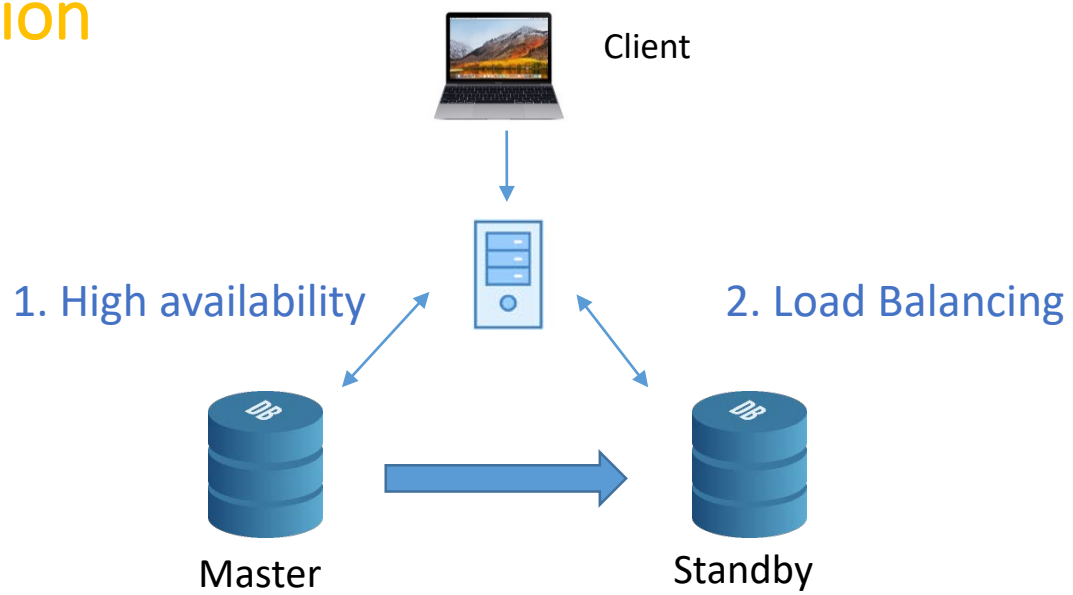


Expected use cases of Logical replication

2. Analytical Purpose of database

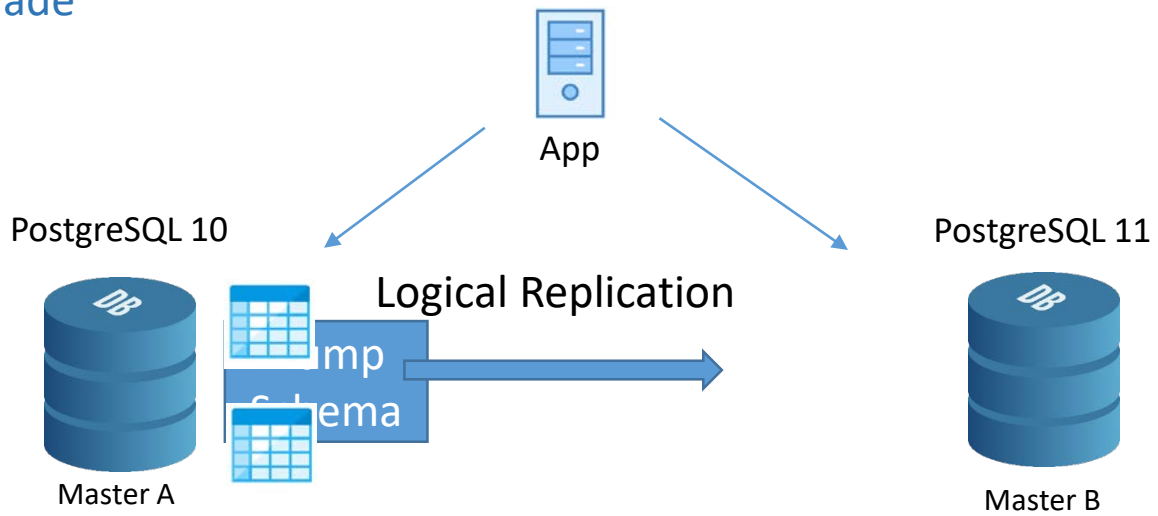


Streaming replication

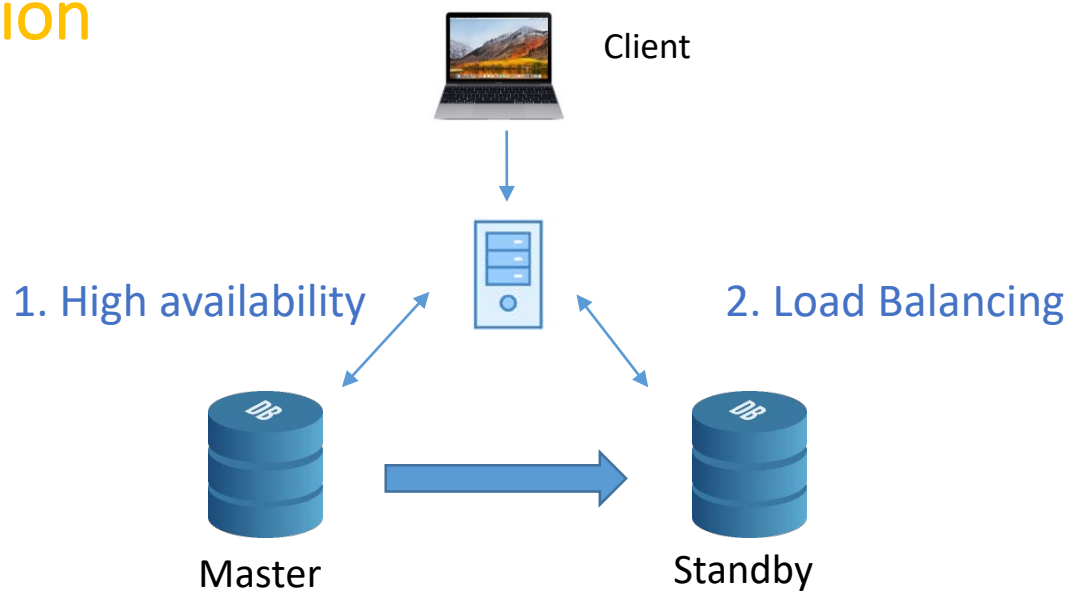


Expected use cases of Logical replication

3. Online Upgrade

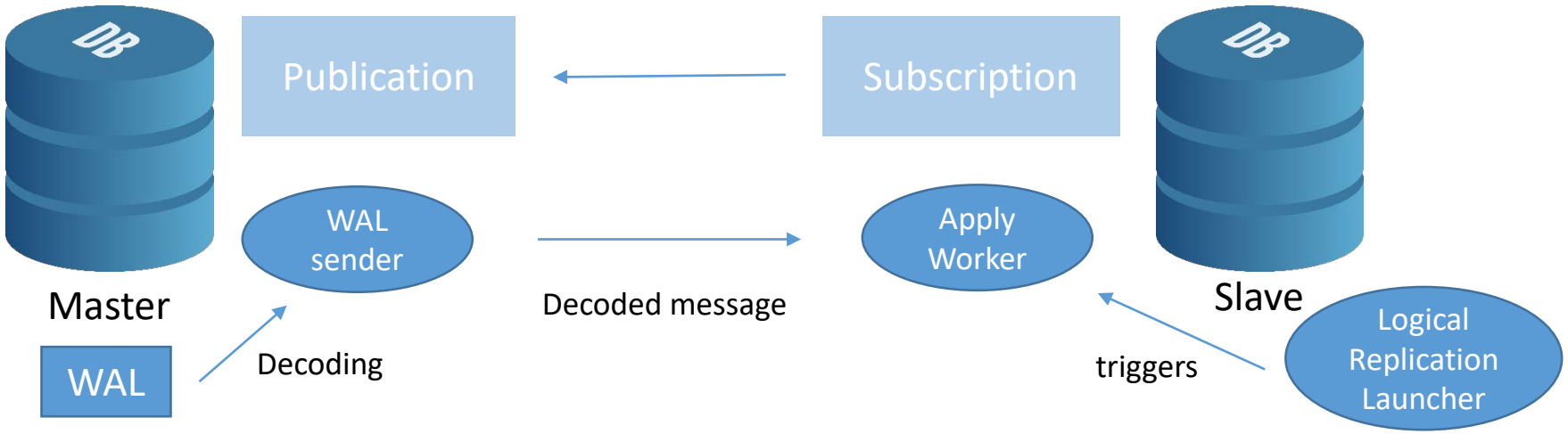


Streaming replication

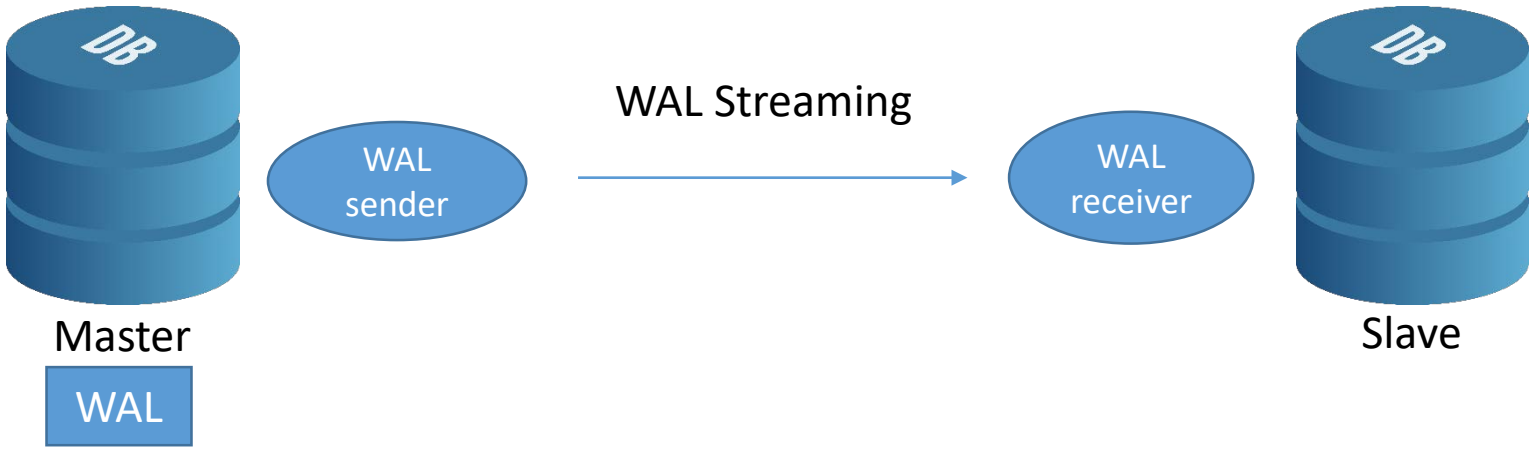


Architecture

Logical replication

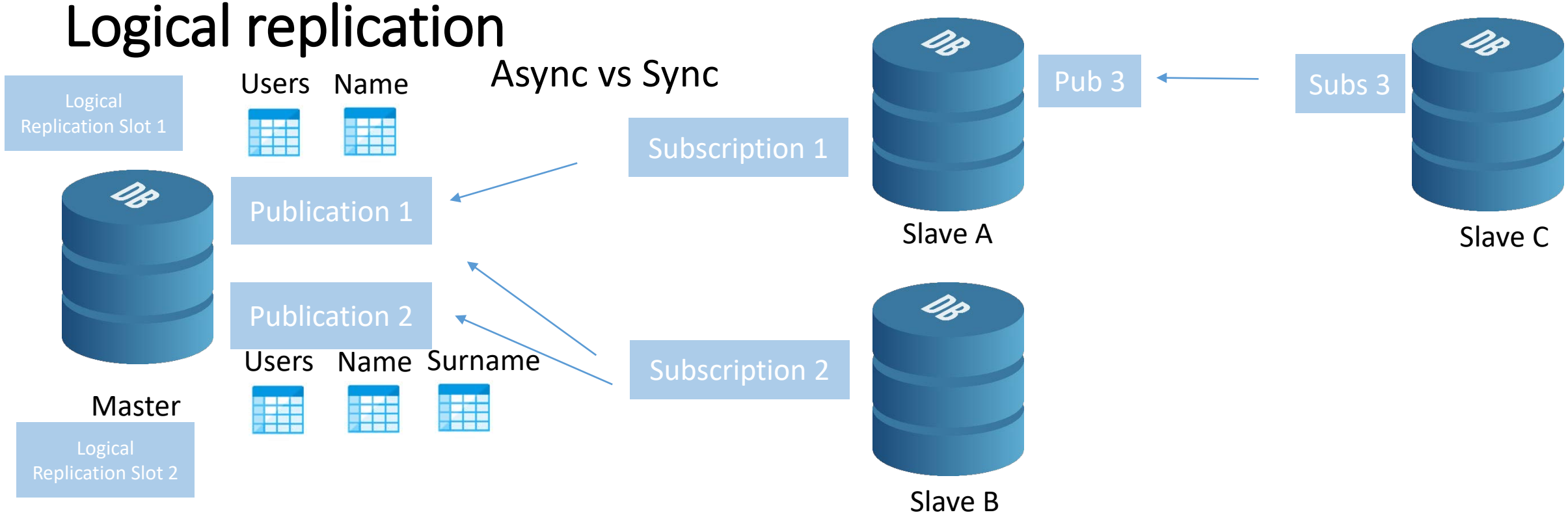


Streaming replication

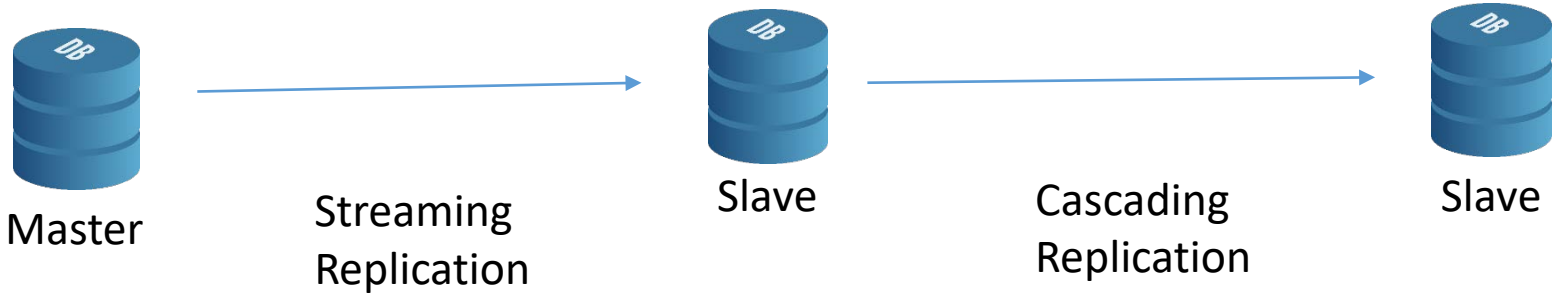


What is Publication&Subscription?

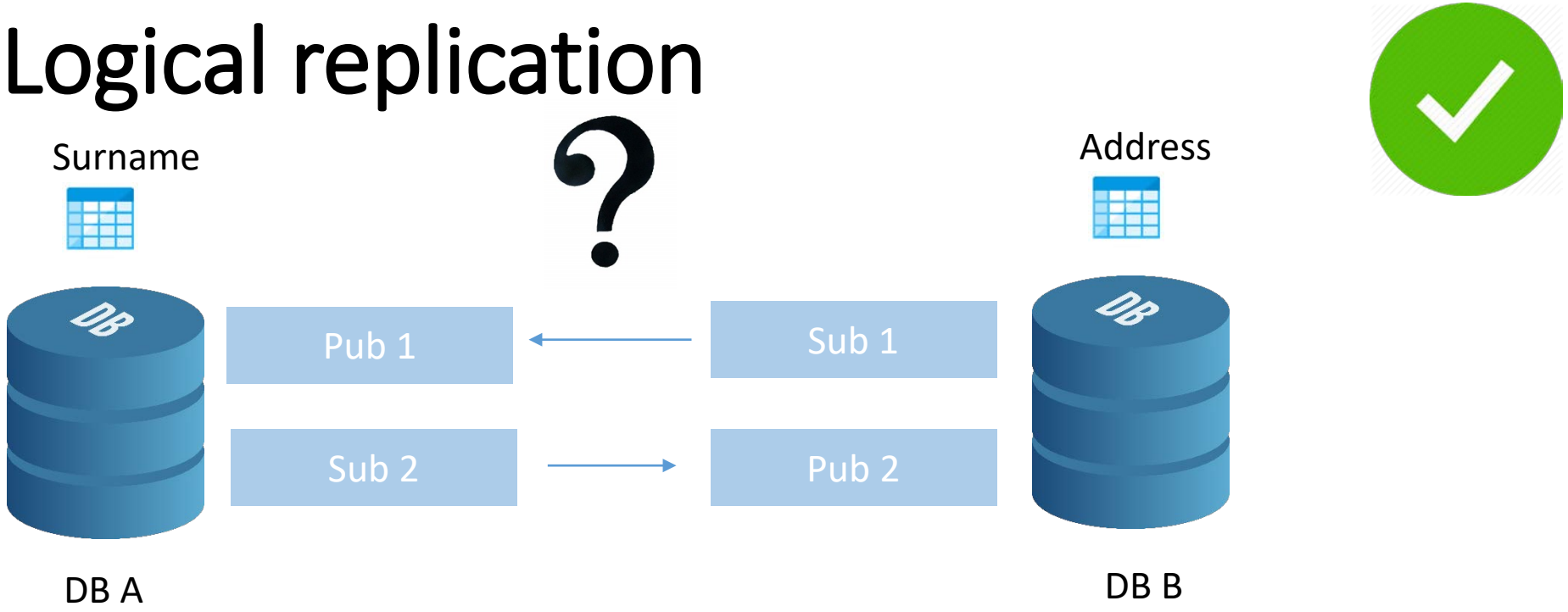
Logical replication



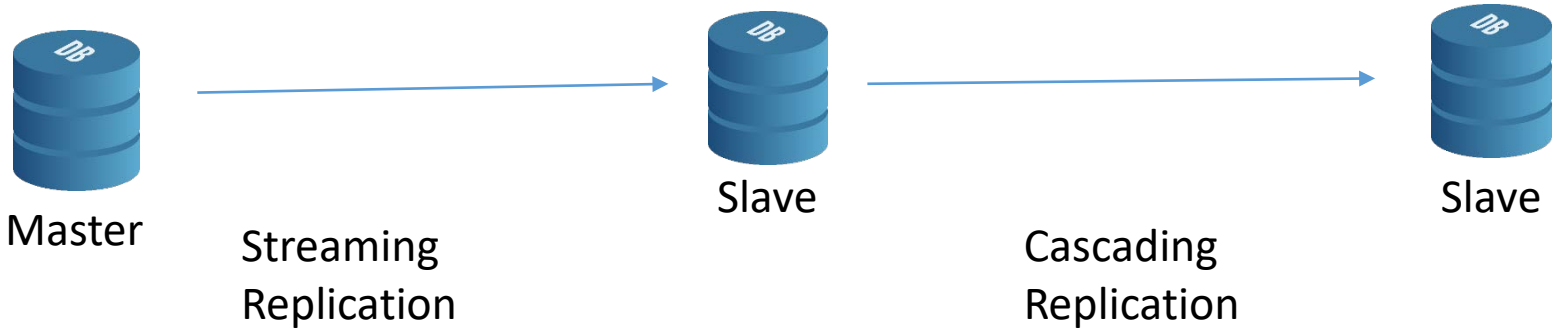
Streaming replication



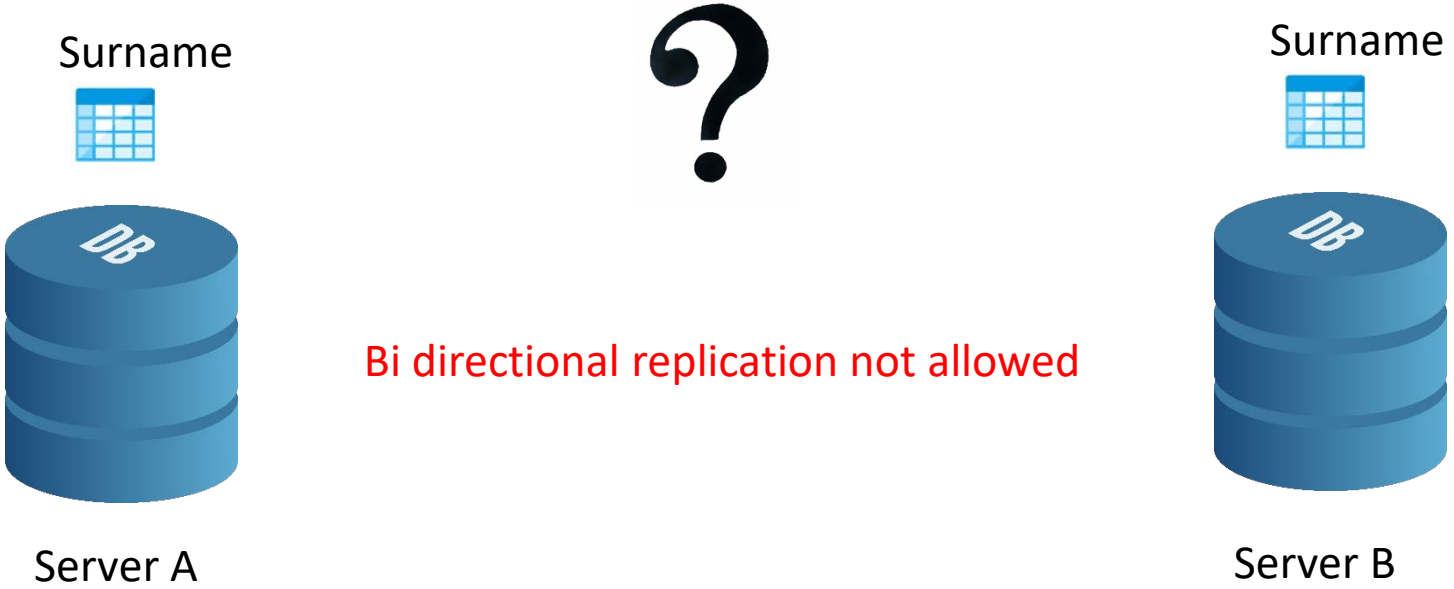
Logical replication



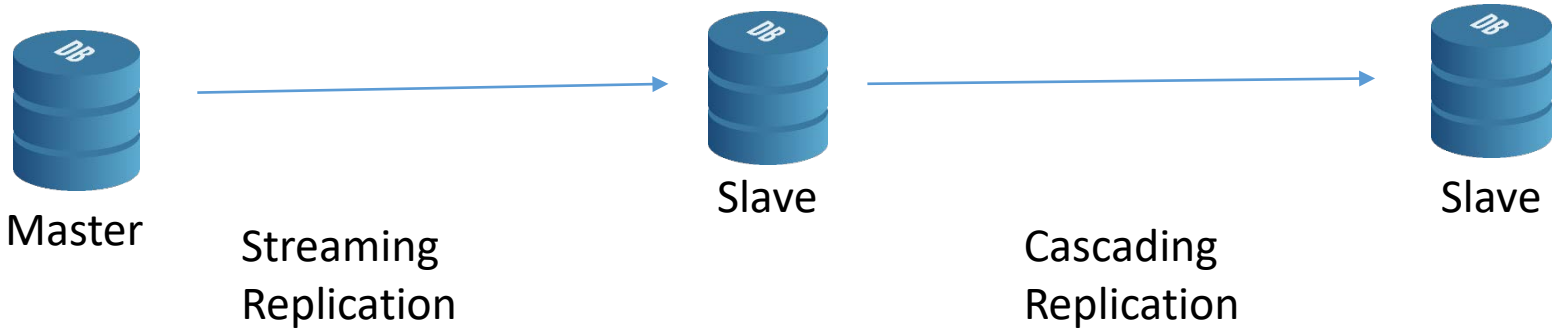
Streaming replication



Logical replication



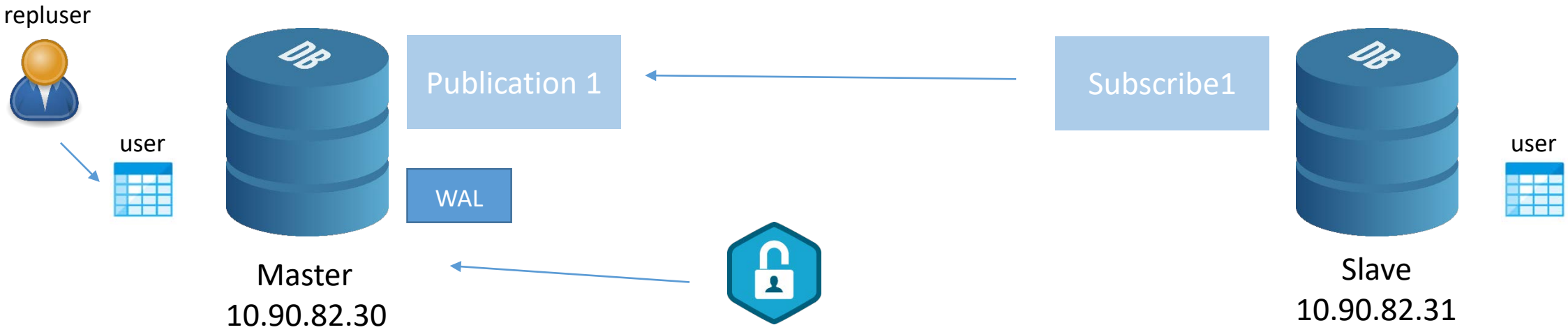
Streaming replication



Logical Replication Limitations in 11.0

- does not replicate schema and DDL
- does not replicate sequences
- does not replicate Large Objects
- Replication is only possible from base tables to base tables

Logical replication with PostgreSQL 11



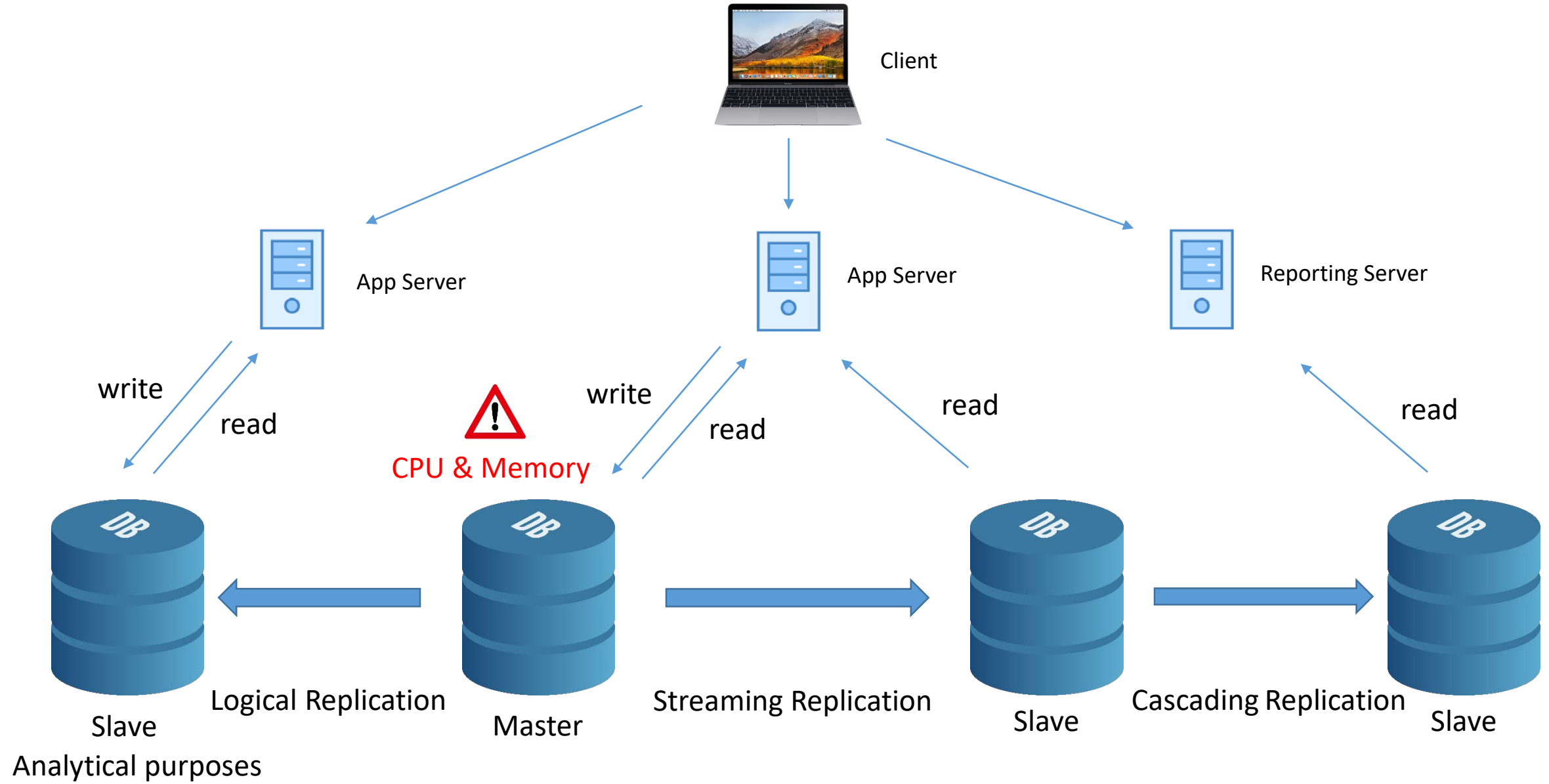
Pg_hba.conf→ host all repluser 10.90.82.31/0 md5

1. wal_level = logical
2. Connected to db with postgres →\c database1
3. CREATE TABLE user (user_name text PRIMARY KEY, full_name text);
4. CREATE PUBLICATION Publication1 for table user;
5. CREATE ROLE repluser WITH LOGIN PASSWORD 'admin123' REPLICATION ;
6. GRANT SELECT ON public.delivery TO repluser;
7. GRANT USAGE ON SCHEMA public TO repluser;

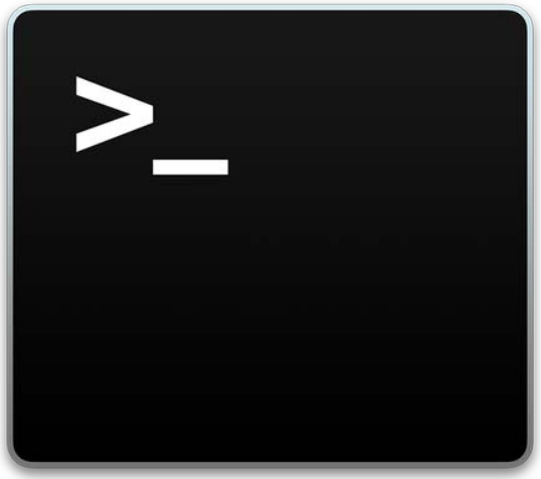
1. Connected to db with postgres →\c database2
2. CREATE TABLE user (user_name text PRIMARY KEY, full_name text);
3. CREATE SUBSCRIPTION Subscribe1 CONNECTION 'host=10.90.82.31 dbname=database1 user=repluser password=admin123' PUBLICATION Publication 1 ;

Logical Replication is DONE

Topology(Logical & Pyhsical Replication)



Monitoring on Terminal



Master

1- pg_stat_replication
2- pg_replication_slots
3- pg_publication_tables

Slave(Logical)

1- pg_subscription
2- pg_stat_subscription
3- pg_subscription_rel

Slave(Physical)

1- pg_stat_wal_receiver

https://kahoot.it/

Hepsiexpress & Horizon 2020

We are looking for partners!

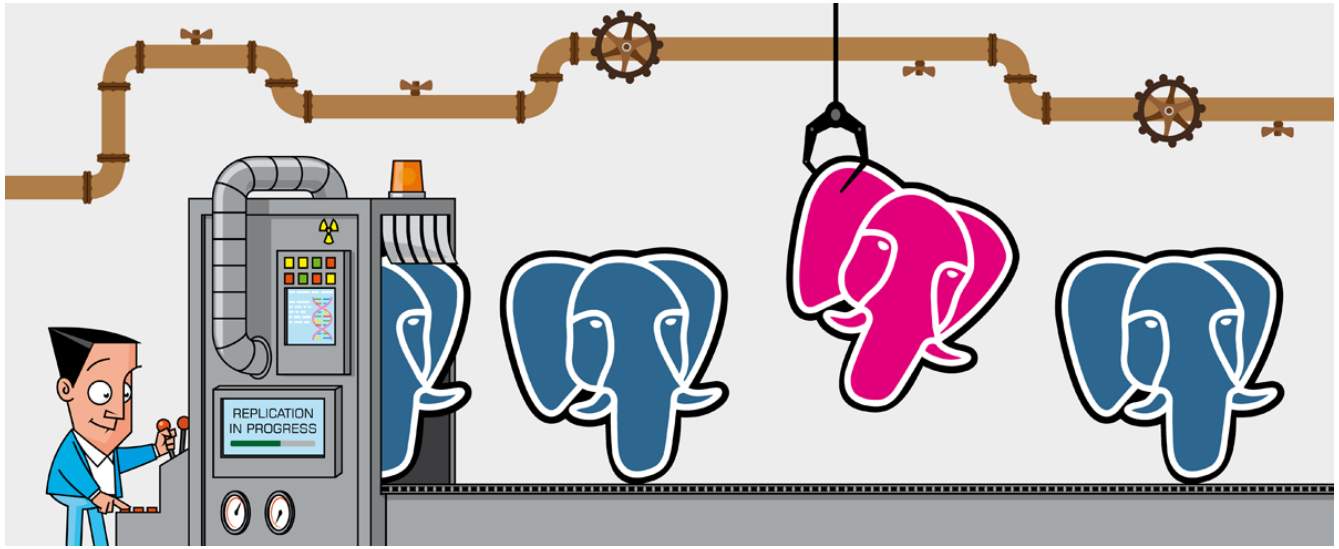
2018 Applications

ISA: Innovative logistics Solutions and data framework for the on-demand economy

Focus Areas for New Projects

- Logistics Research & Innovation
- Route Optimisation
- Natural Language Processing
- Open Source Softwares

Thank you



Fırat Güleç