Get It Back

A New PostgreSQL Admin's Guide to
Redundancy and Recovery

Why should I listen to Josh?

- Dev&&Ops at End Point Corporation

- Postgres for (mumble)-teen years
- Or don’t?

Backing Up Postgres

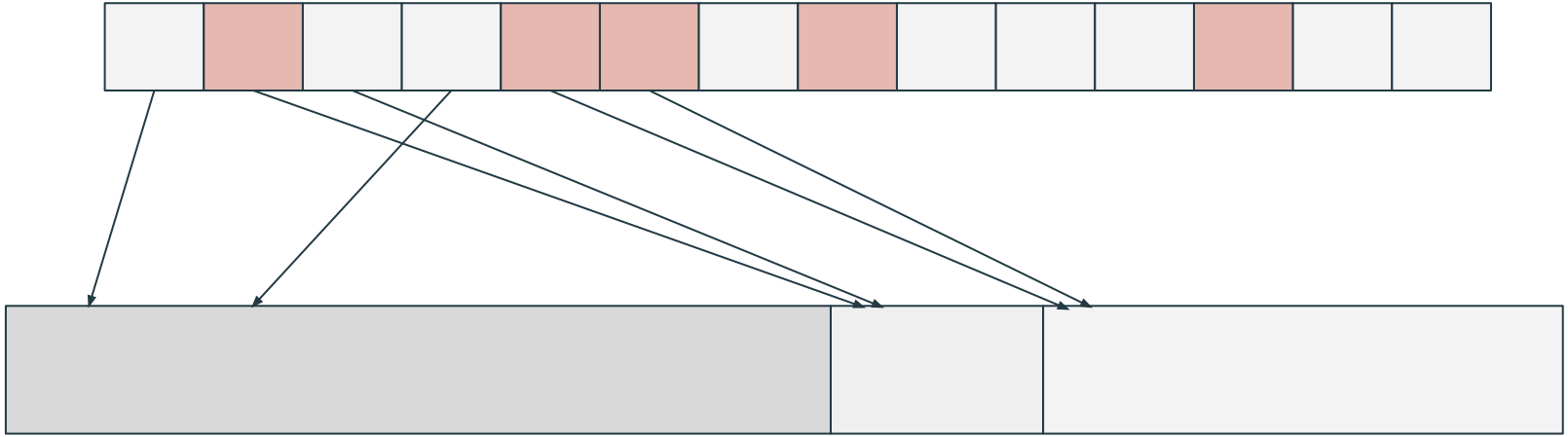
Backing Up Postgres - Don’t Do This

```
$ while :
>     cp -r /var/lib/postgresql /var/lib/postgresql.bak
>     sleep 3600
> done
```

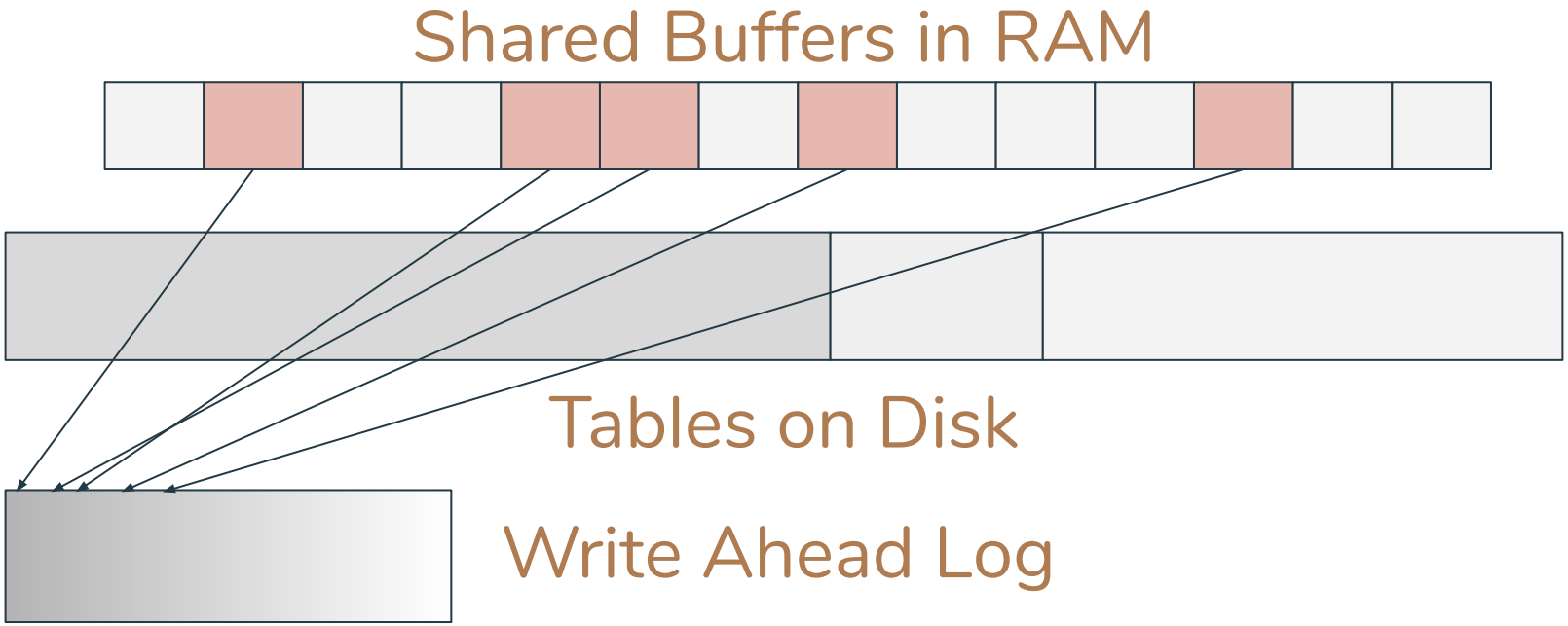
Backing Up Postgres - Don’t Do This

```
$ while :; do
>     cp -r /var/lib/postgresql /var/lib/postgresql.bak
>     sleep 3600
> done
```

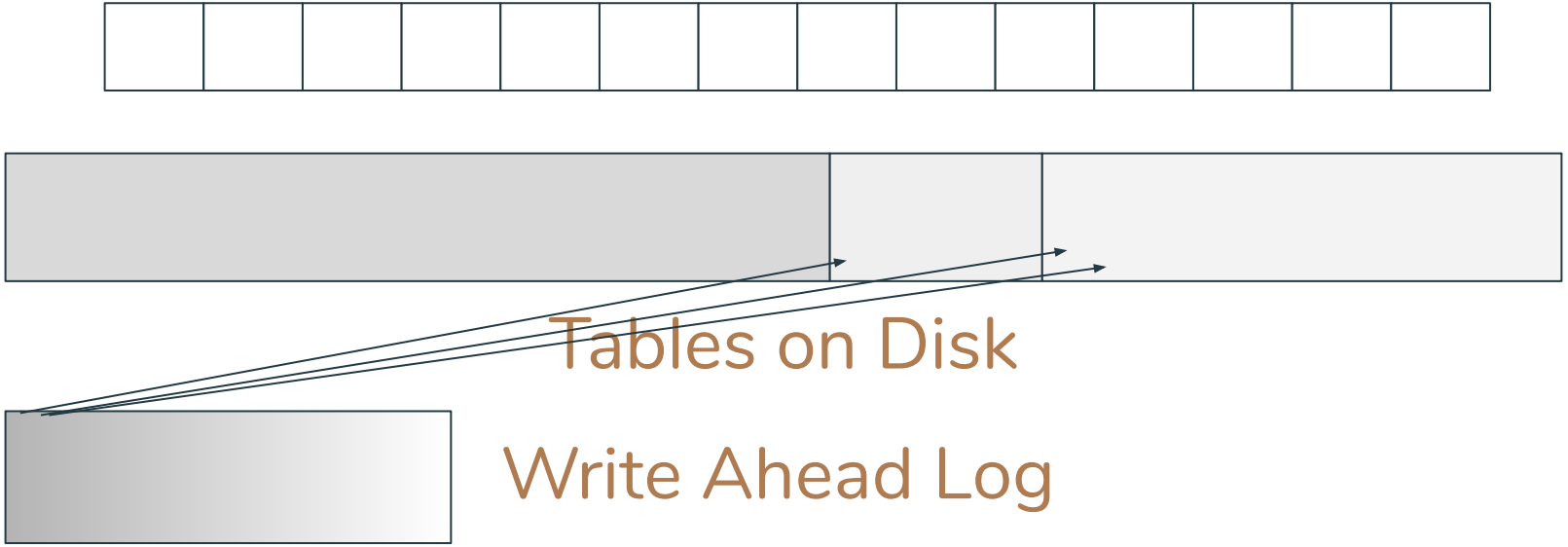
Shared Buffers in RAM

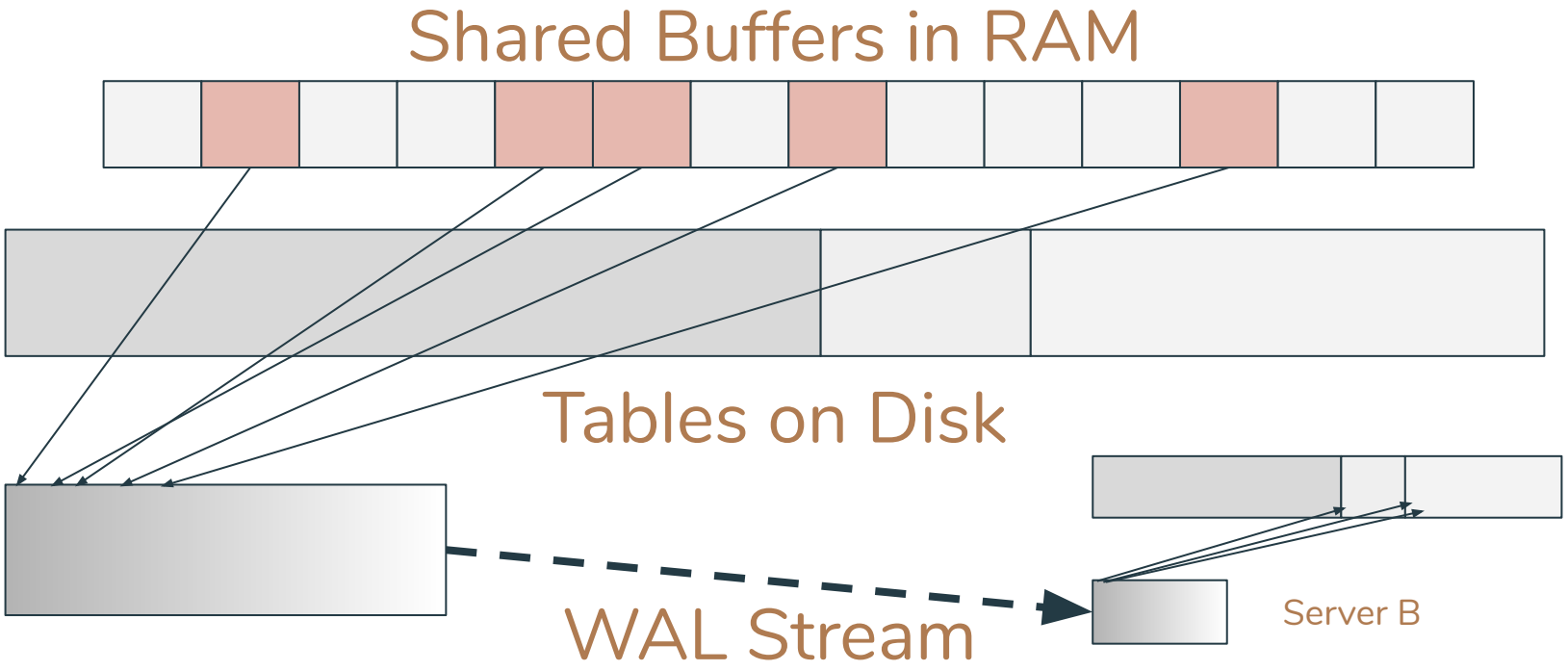


Tables on Disk


Shared Buffers in RAM
Tables on Disk
Write Ahead Log

In Recovery




Shared Buffers in RAM
Tables on Disk
WAL Stream
Server B

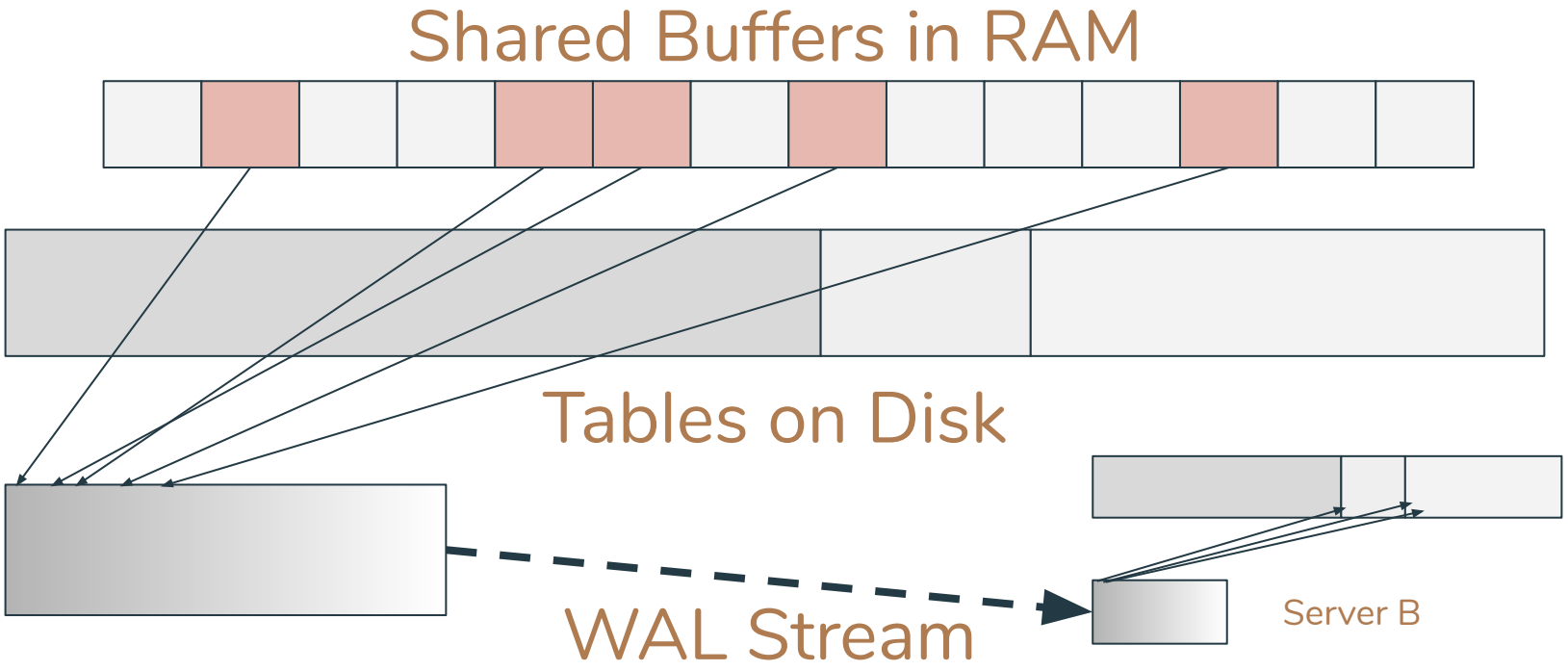
Replication

Replication

- Physical, aka Streaming Replication
- Logical Replication
  - Sort of built-in, >= 9.4
  - Trigger-based (any version)

Replication

- Physical, aka Streaming Replication
- Logical Replication
  - Sort of built-in, >= 9.4
  - Trigger-based (any version)
- “Your Herd of Elephants” … Tomorrow, 12:30


Shared Buffers in RAM
Tables on Disk
WAL Stream
Server B

Let’s Get Physical

I Lied

You can `cp -r`

Kind of…

… With some specific settings in place..

And only when you tell Postgres.

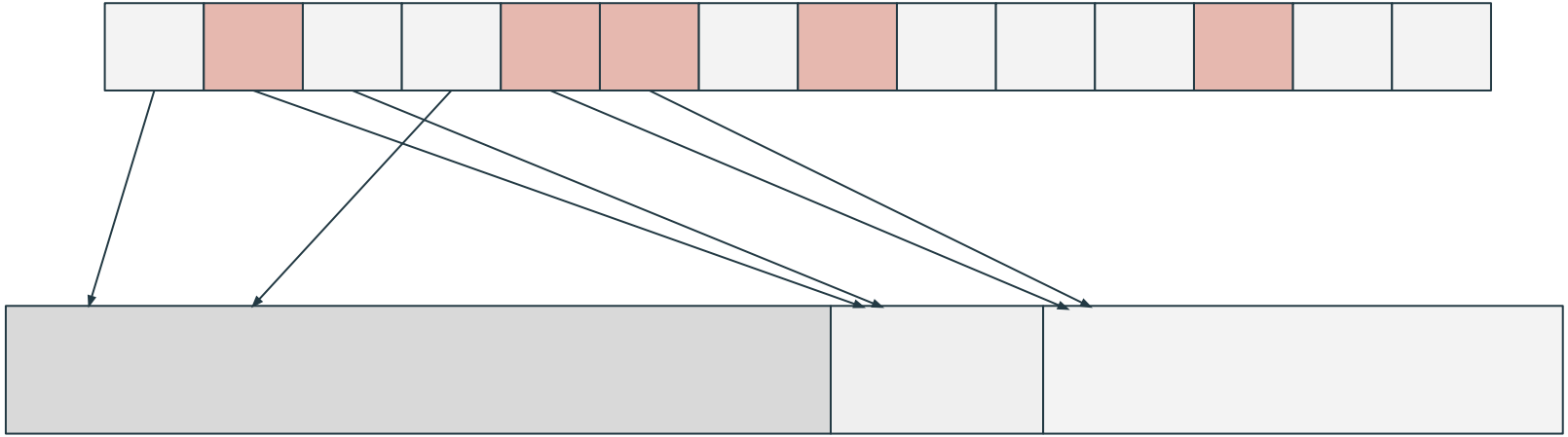
Anatomy of a Base Backup

1. Must have configured:
    a. wal_level = replica (or logical)
    b. archive_mode = on
    c. archive_command = 'cp %p /var/wal_archive/%f'
2. SELECT pg_start_backup('label');
3. cp -r (or, something more respectable)
4. SELECT pg_stop_backup();

Anatomy of a Base Backup

1. pg_start_backup('foo')
   a. Set a start marker
   b. Checkpoint….
   c. Return
2. Now we can copy at our leisure
3. pg_stop_backup()
   a. Wait for all archival to complete
   b. Clean up

Shared Buffers in RAM



Tables on Disk

Anatomy of a Base Backup

1. pg_start_backup('foo')
   a. Set a start marker
   b. Checkpoint….
   c. Return
2. Now we can copy at our leisure
3. pg_stop_backup()
   a. Wait for all archival to complete
   b. Clean up

Or...

$ pg_basebackup

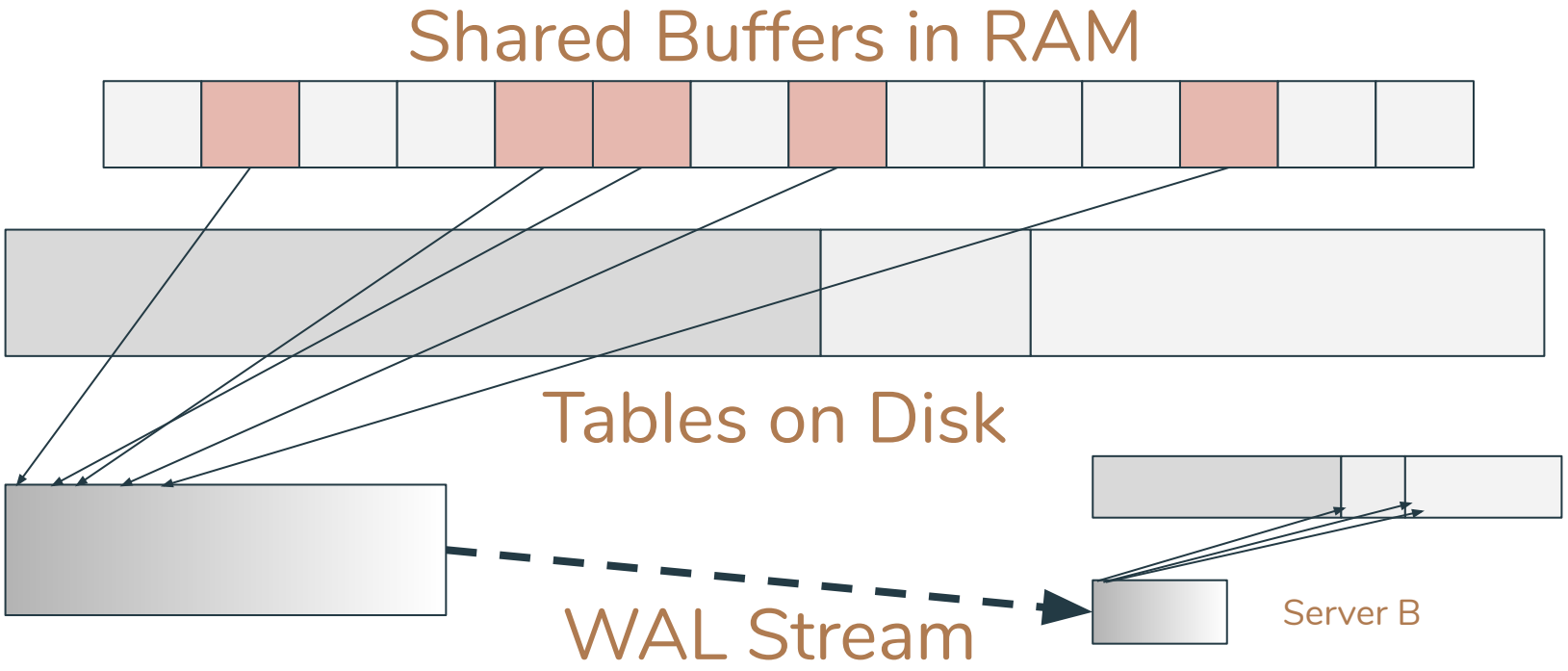
Uses the replication protocol..

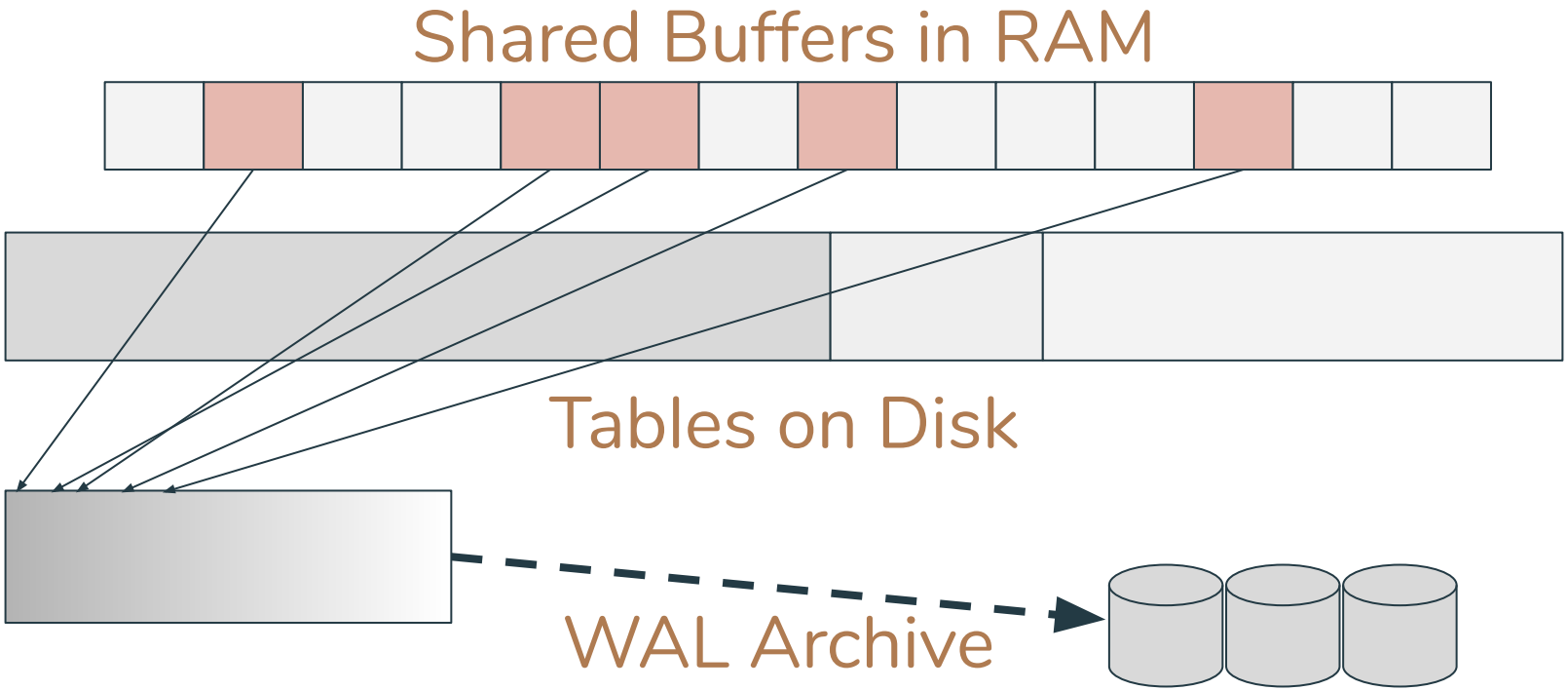
Connection must have REPLICATION permission

&&

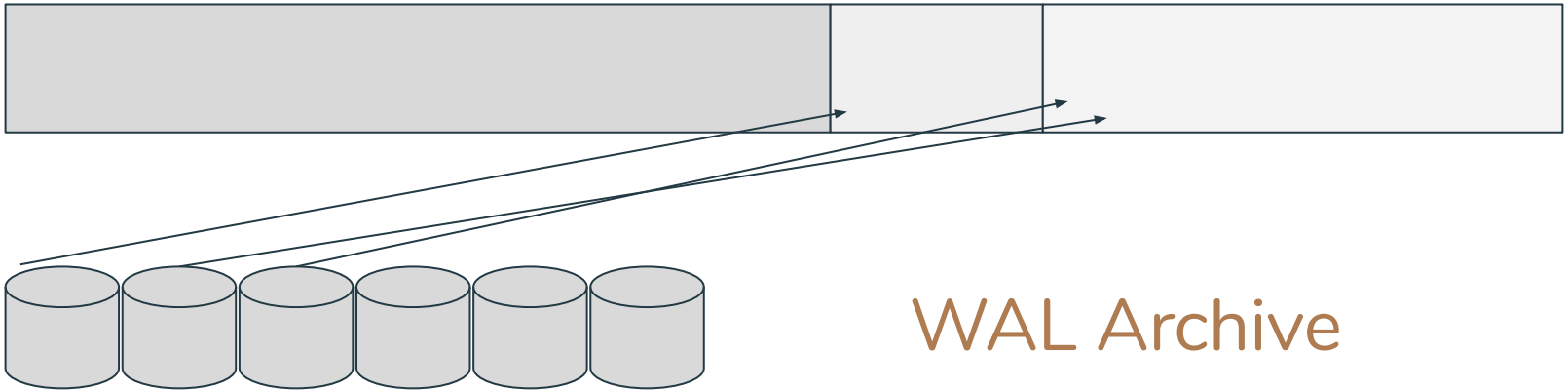
max_wal_senders must have room.

So...?


Shared Buffers in RAM
Tables on Disk
WAL Stream
Server B


Shared Buffers in RAM
Tables on Disk
WAL Archive

Base Backup

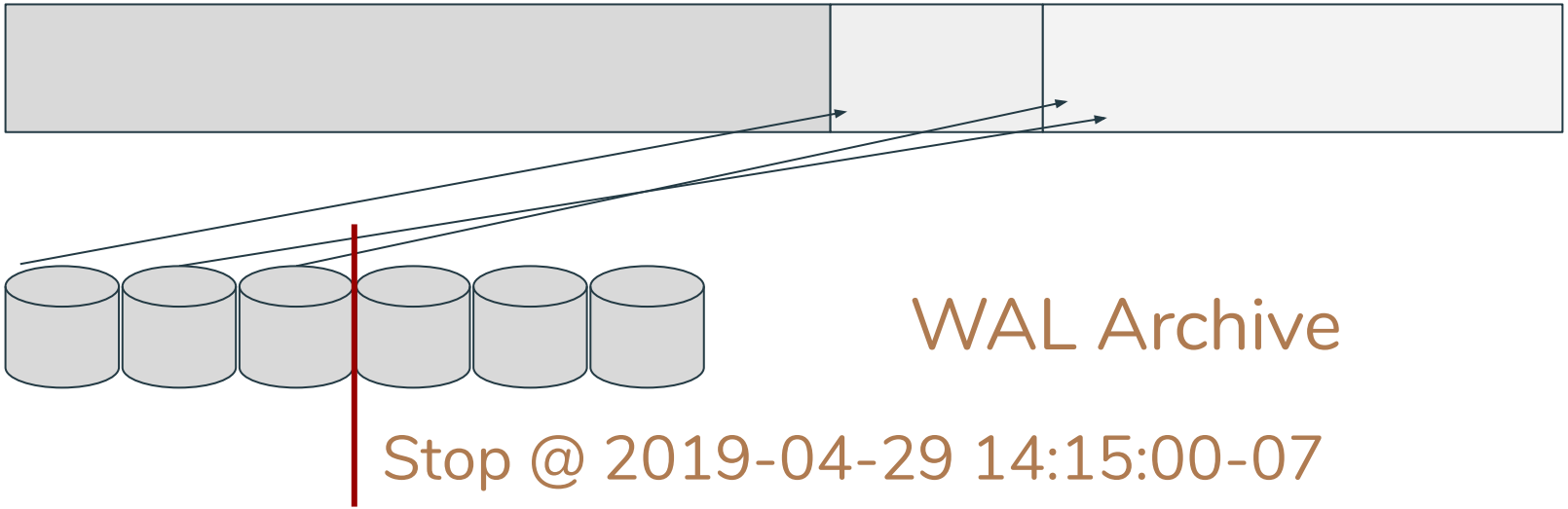


WAL Archive

PITR

Point-In-Time Recovery

Base Backup



WAL Archive

Stop @ 2019-04-29 14:15:00-07

Now that I've told you all that...

- pgBackRest
- Barman
- PGHoard
- WAL-E
- WAL-G
- (etc, etc)

Look for…

- Cloud storage integration
- Encryption
- Management interfaces
- Storage efficiency
- Retention
- Scheduling...

Logical Backups with pg_dump

pg_dump

- Generates a series of SQL statements to rebuild a database

pg_dump

- Generates a series of SQL statements to rebuild a database

- Works over a normal Postgres connection
- Works with the permissions it has, no special settings
- Selective, single database, or parts of it
- Output is text

pg_dump

- Generates a series of SQL statements to rebuild a database

- Works over a normal Postgres connection
- Works with the permissions it has, no special settings
- Selective, single database, or parts of it
- Output is text
- pg_dumpall == pg_dumpall --globals + pg_dump (-C)

But…

- No way to apply transaction logs
- And, harder to optimize performance
- Similarly, restores are slow(er)

Output is text (...sometimes)

- pg_dump --format=c (custom) … or d (directory)
- --format=d --jobs=X allows for parallel dumps!
- Both allows for parallel restore
- Either way, pg_restore to read and generate the SQL
- With pg_dump-like flexibility

That’s It

Keeping your data secure:

- Replication (Tune in tomorrow)
- Physical backups
  Base backup + WAL transaction logs
- Logical backups
  pg_dump, pg_dumpall, pg_restore

?