



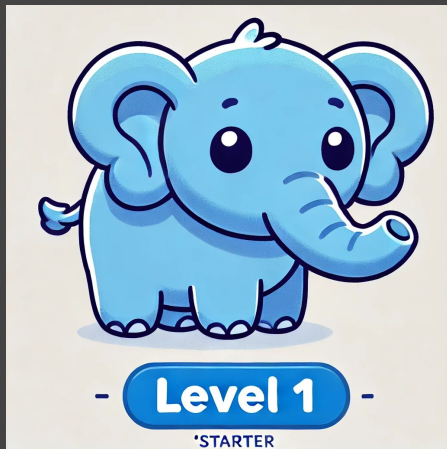
9.2 to 15 and beyond

A case study of a tricky upgrade path

Nick Meyer @ Academia.edu

PGConf NYC 2024

A




9.2 to 15 and beyond

A case study of a tricky upgrade path

Nick Meyer @ Academia.edu

PGConf NYC 2024

A



Prelude: 2008
A database is born

A

DB as an Archaeological Site

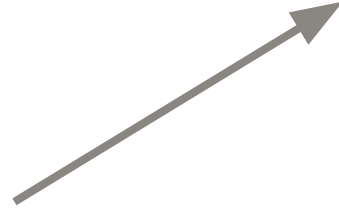
- “Written record” (slack? git?)
- Version: 8.3? Earlier?
- Eventually upgraded to 9.2
 - *At some point before 2018*

Mario Modesto Mata

Creative Commons Attribution-Share Alike 3.0 Unported



(Link to repo with slides)





A

A bit about me (Nick Meyer)

- <https://github.com/aristocrates>
- **Team lead of Platform Engineering**
- **Areas of focus**
 - Developer experience
 - Interface: application and infra
 - **Data layer**
 - **Postgres**





Academia.edu



- <https://www.academia.edu/about>
- **We're hiring!**
- **Our goals**
 1. Ensure that every paper ever written is:
 - ✓ on the internet
 - ✓ available for free
 2. Accelerate the world's research
- **Some stats**
 1. 50 million papers uploaded
 2. 20 million paper recommendations per day



A

What is the point?



What is the point of telling this story?

- Empathy
- Emotion
- Technical details ignorable
 - (but feel free to pay attention)
- Are you:
 - A contributor?
 - Community member?
 - Doing a similar upgrade?

A “Version-splaining”

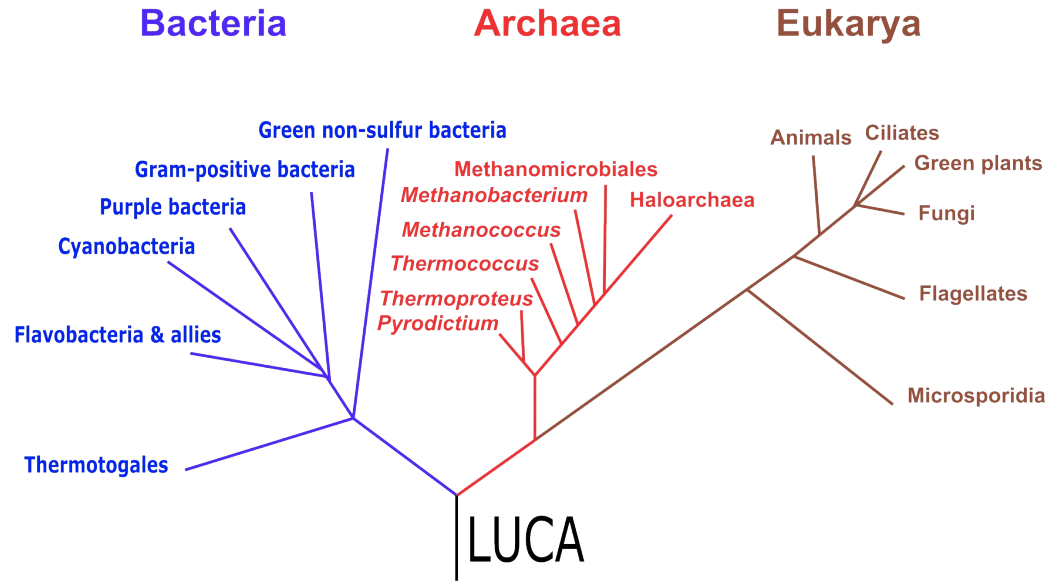
ver·sion·splain (verb)

/ 'vɜr ʒən ,splɛɪn /

To tell someone they should upgrade when they already said they are working on it.

A

“Vertical” sharding

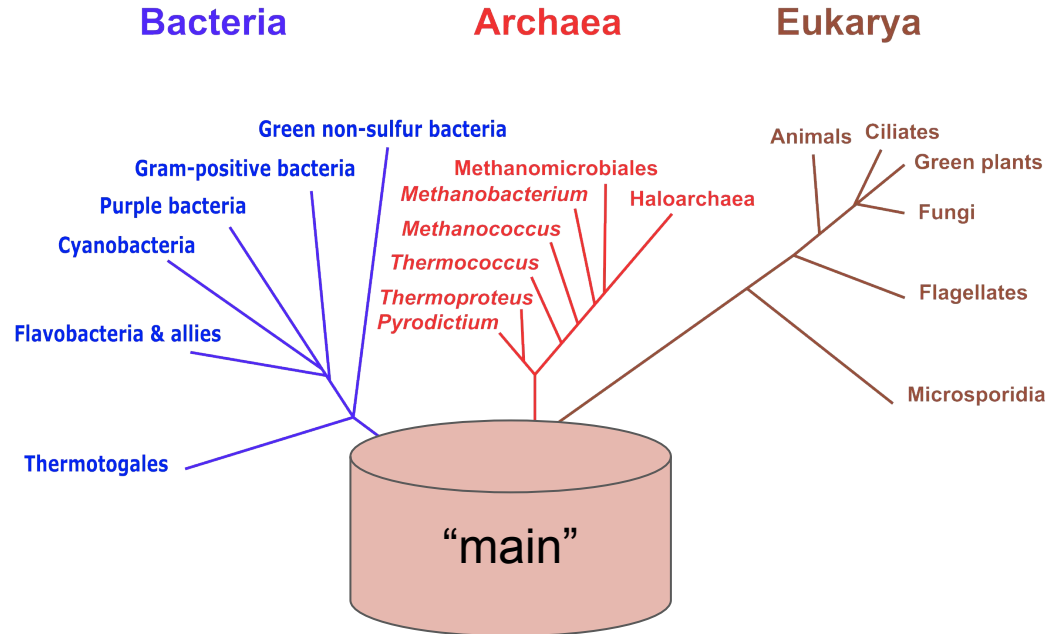


Chiswick Chap

Creative Commons Attribution-Share Alike 4.0 International

A

“Vertical” sharding

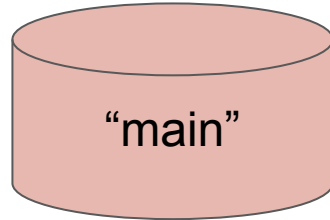


Chiswick Chap

Creative Commons Attribution-Share Alike 4.0 International

A

“main”



“main”

A

What is “main”?

- 600 tables
- 5 TB
- “Creative” use of PL/pgSQL functions
 - and custom types, constraints, etc

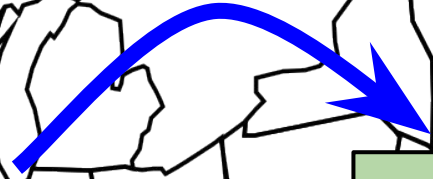
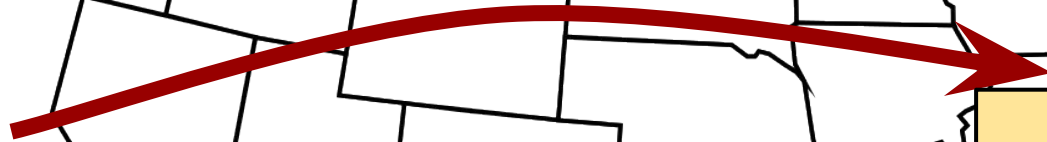
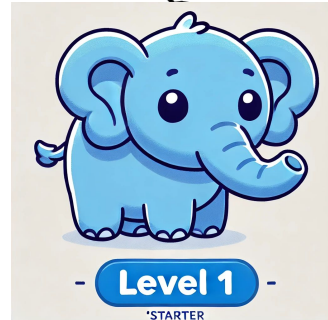
pglogical

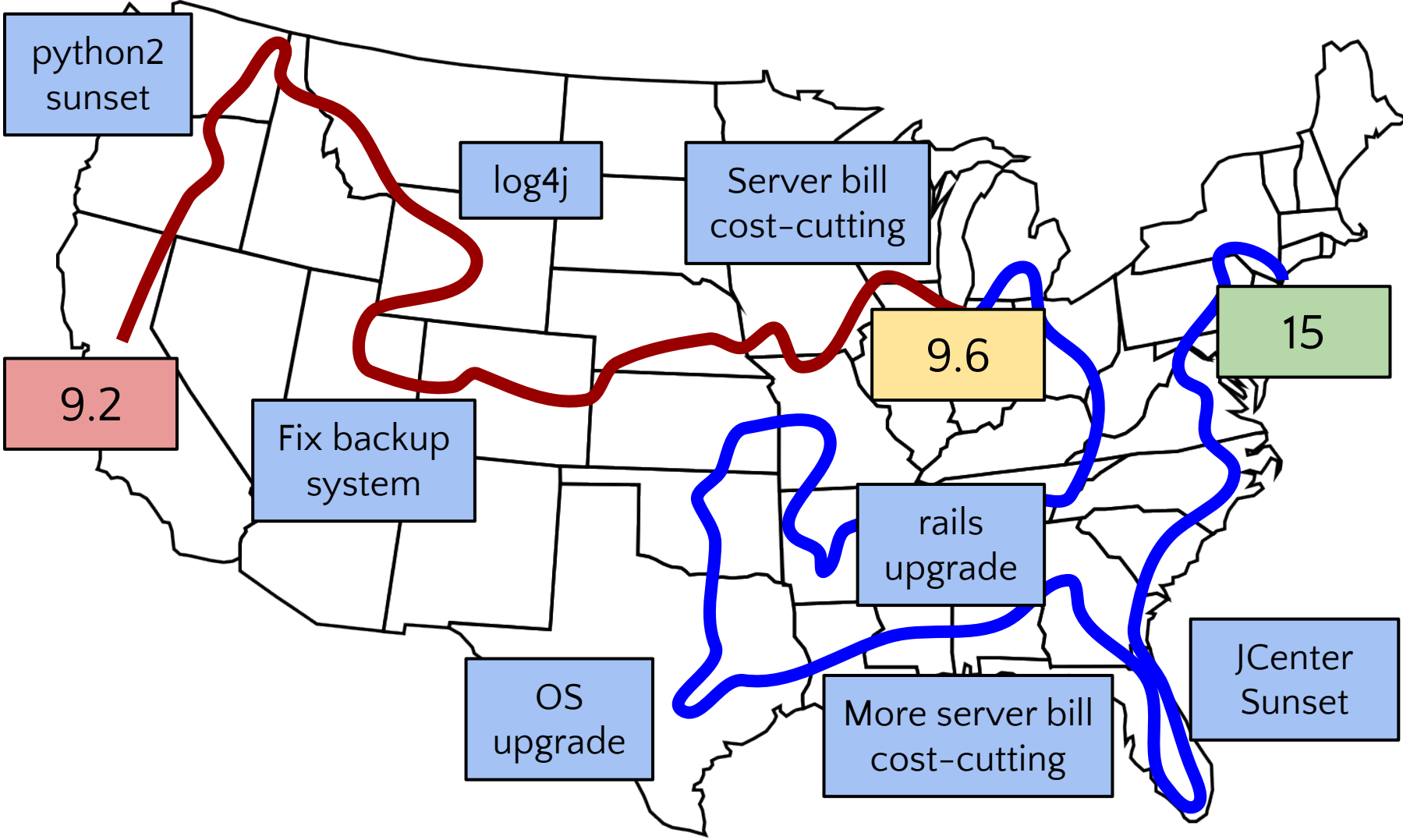
pg_upgrade

9.6

15

9.2





Part 1

9.2 -> 9.6

A



A

Why stop at 9.6?

- <https://www.postgresql.org/docs/release/8.4.0/>
- “Release Date: 2009-07-01”

```
commit 2169e42bef9db7e0bdd1bea00b81f44973ad83c8
Author: Neil Conway <neilc@samurai.com>
Date: Sun Mar 30 04:08:15 2008 +0000
```

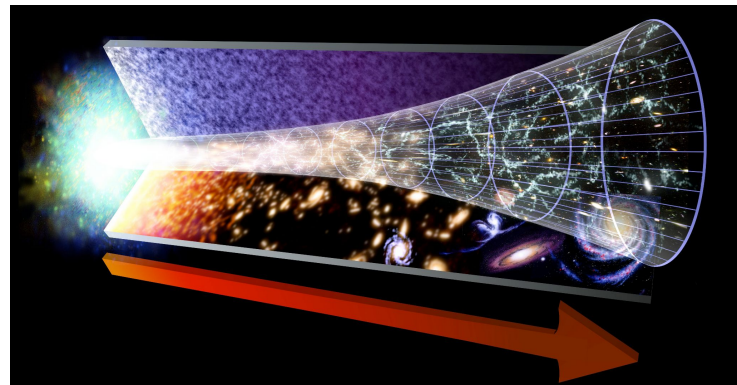
Enable 64-bit integer datetimes by default, per previous discussion.

This requires a working 64-bit integer type. If such a type cannot be found, "--disable-integer-datetimes" can be used to switch back to the previous floating point-based datetime implementation.

A --disable-integer-datetimes

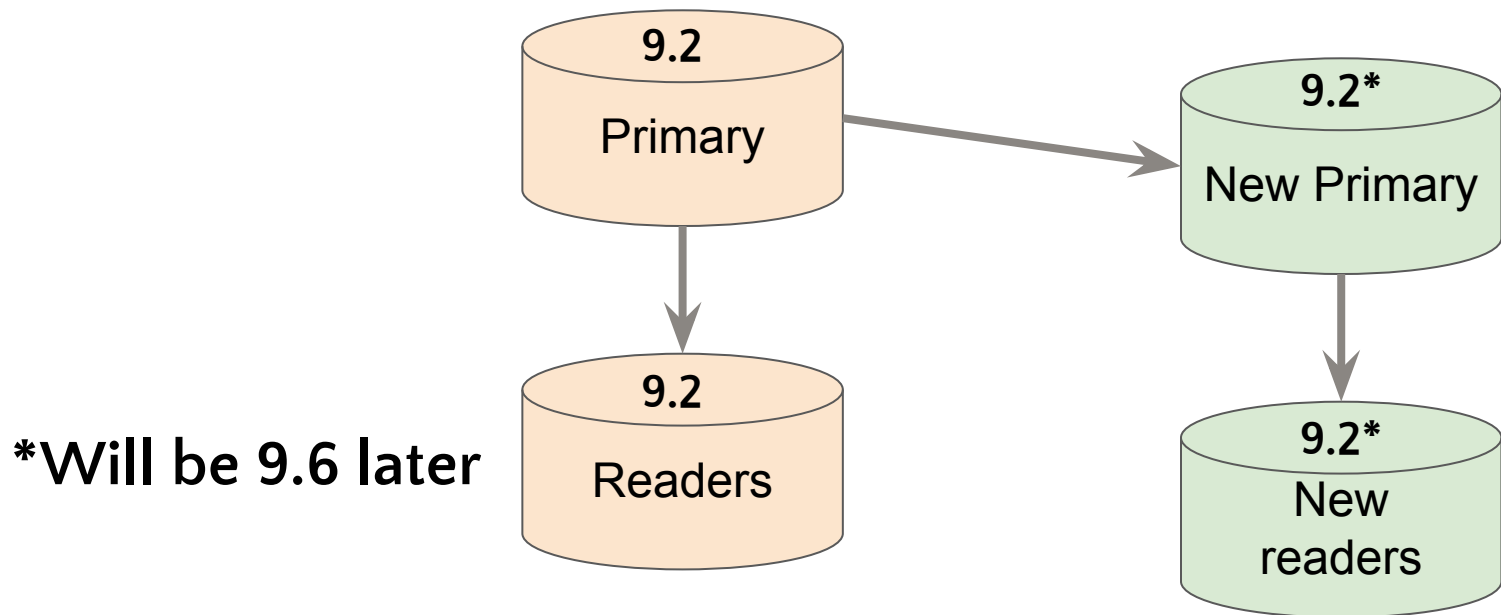
- With data, context endures for years
- `pg_upgrade --check` complains
- => **We built our own postgres binary**
 - And we did that all the way to 9.2

NASA; Dana Berry
https://svs.gsfc.nasa.gov/10128#section_credits



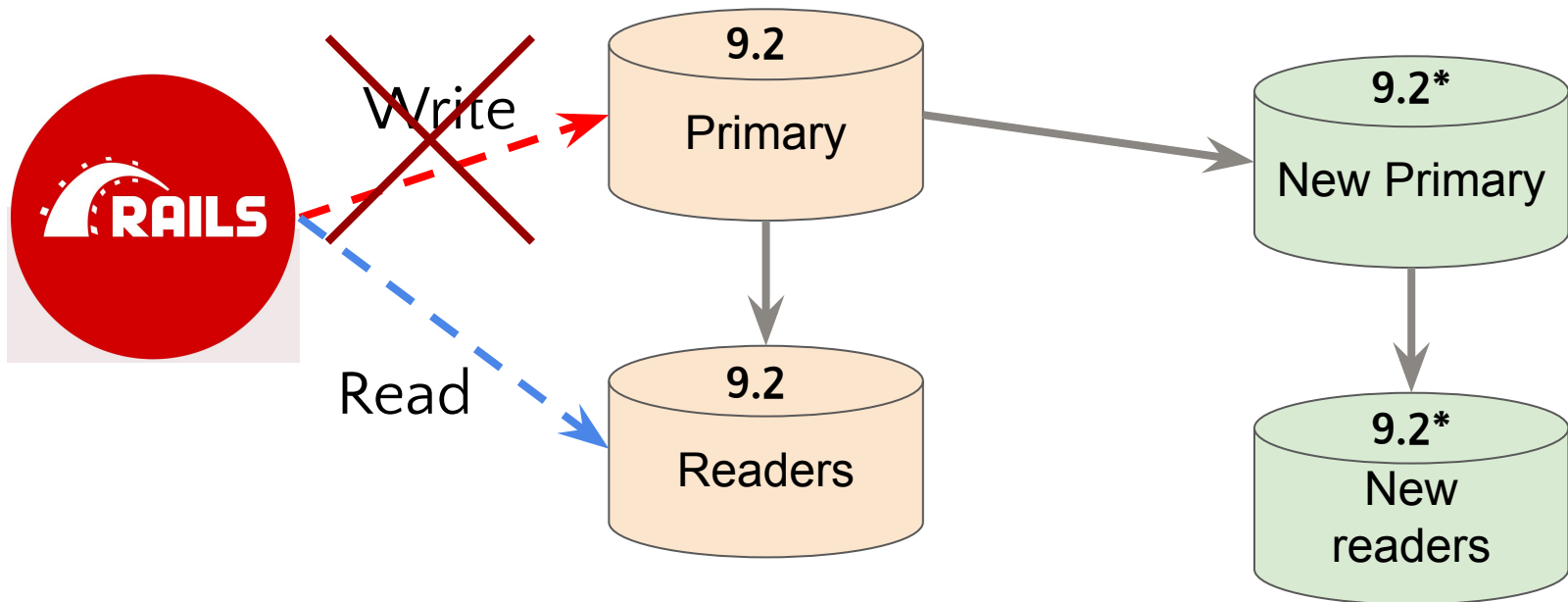
A pg_upgrade with streaming replicas

Step 1: Bring up “new tree” replicas



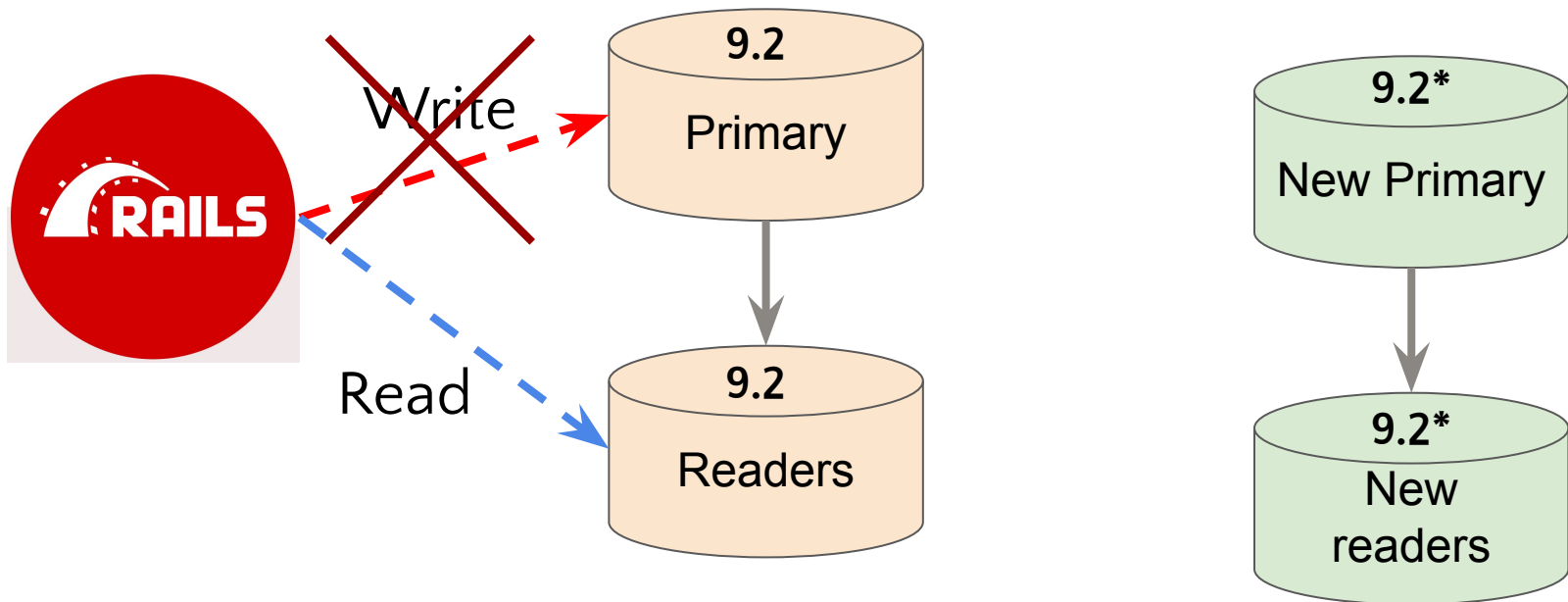
A pg_upgrade with streaming replicas

Step 2: Block writes (and reads)



A pg_upgrade with streaming replicas

Step 3: Promote new primary



A pg_upgrade with streaming replicas

Step 4: Follow the simple 17 step guide

<https://www.postgresql.org/docs/9.6/pgupgrade.html>

(But also reference the latest version of those docs:

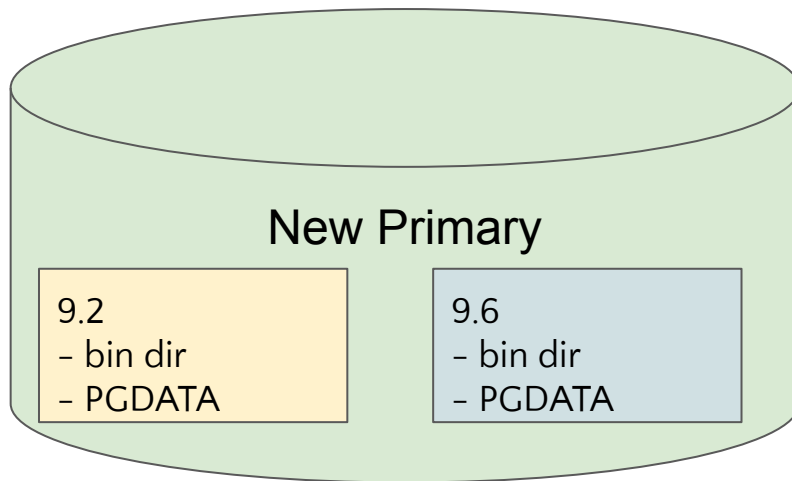
<https://www.postgresql.org/docs/16/pgupgrade.html>)

Diving into the details

A

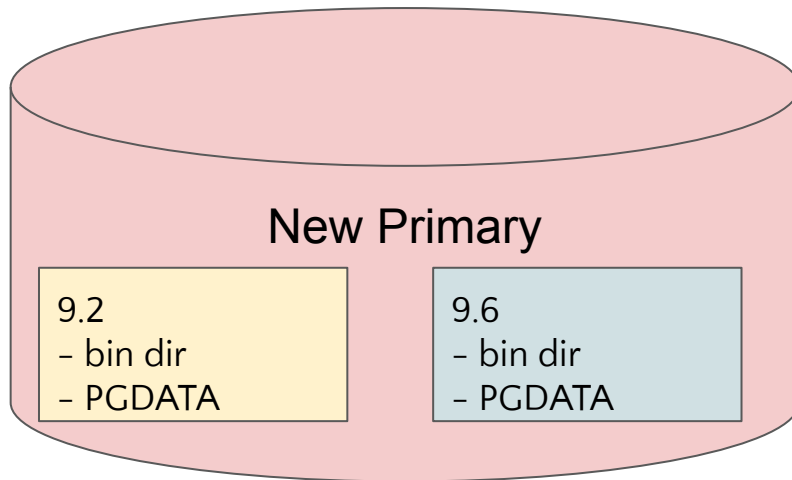
A pg_upgrade with streaming replicas

Step 4a: Install newer version, and initdb



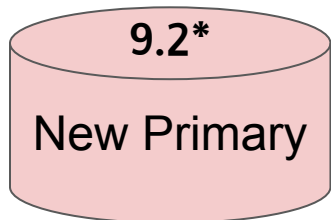
A pg_upgrade with streaming replicas

Step 4b: Stop postgres on new primary
(but keep it running on the replicas)

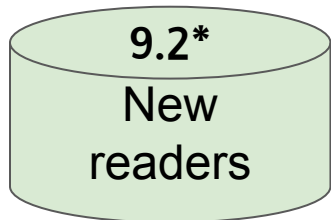


A pg_upgrade with streaming replicas

Step 4c: Check that pg_controldata (checkpoint) matches



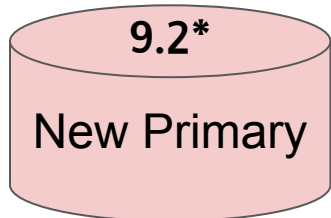
```
$ pg_controldata $PGDATA | grep 'Latest checkpoint location'  
Latest checkpoint location: 0/41933E90
```



```
$ pg_controldata $PGDATA | grep 'Latest checkpoint location'  
Latest checkpoint location: 0/41933E90
```

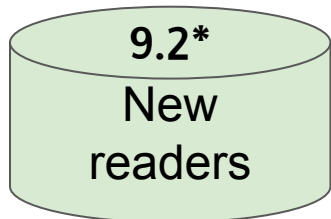
A pg_upgrade with streaming replicas

Step 4c: Check that pg_controldata (checkpoint) matches



```
$ pg_controldata $PGDATA | grep 'Latest checkpoint location'  
Latest checkpoint location: 0/4193BE90
```

Do not proceed unless they match

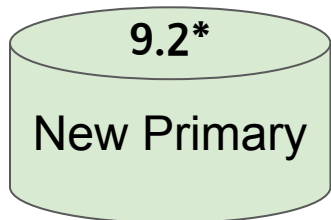


```
$ pg_controldata $PGDATA | grep 'Latest checkpoint location'  
Latest checkpoint location: 0/4193BE90
```


A

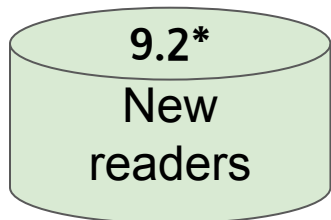
Aside: what we did before step 4b

Before stopping postgres on new primary: checkpoint in loop



```
postgres=# checkpoint;  
$ pg_controldata $PGDATA | grep 'Latest checkpoint location'  
Latest checkpoint location: 0/41933E90
```

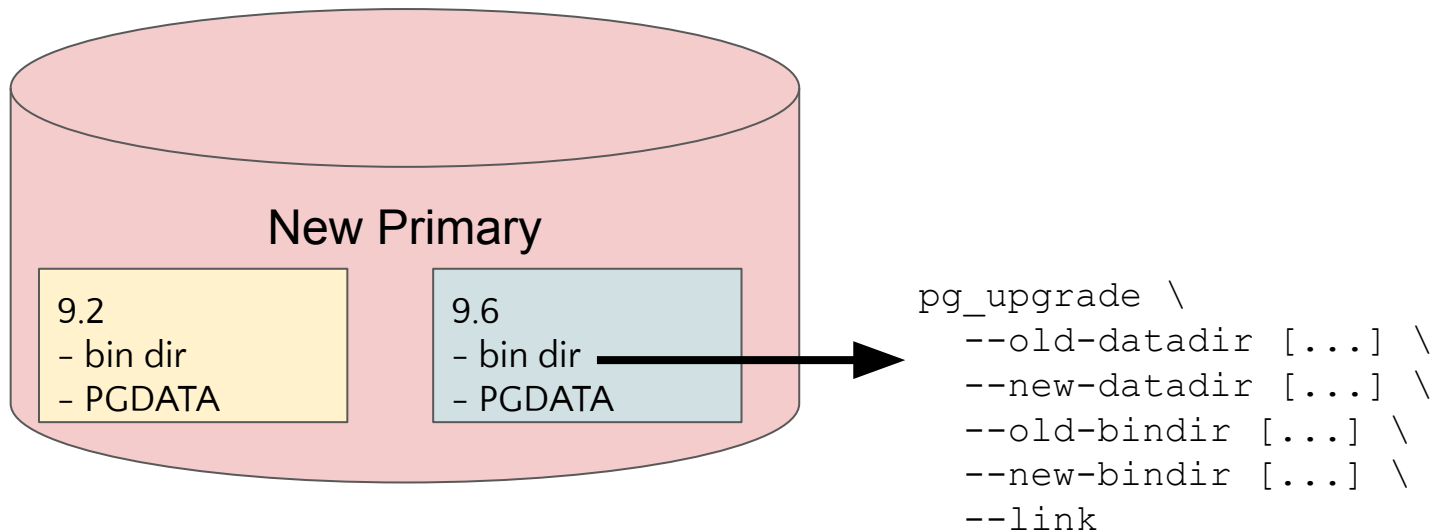
Do not proceed unless they match



```
postgres=# checkpoint;  
$ pg_controldata $PGDATA | grep 'Latest checkpoint location'  
Latest checkpoint location: 0/41933E90
```

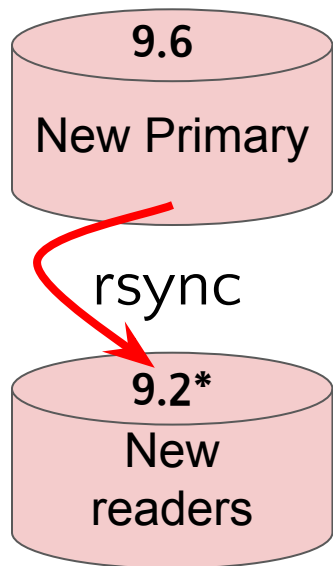
A pg_upgrade with streaming replicas

Step 4d: Run pg_upgrade with --link (ensure no errors)



A pg_upgrade with streaming replicas

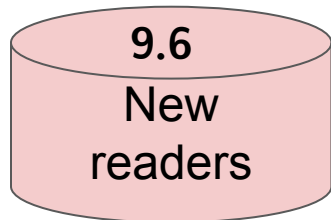
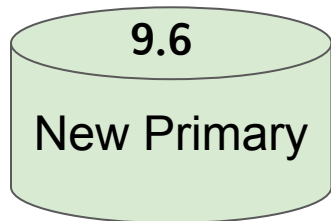
Step 4e: Stop replicas, install new pg and... run rsync (?)



```
rsync \
  --archive \
  --delete \
  --hard-links \
  --size-only \
  --no-inc-recursive \
  /opt/PostgreSQL/9.2 \
  /opt/PostgreSQL/9.6 \
  standby.example.com:/opt/PostgreSQL
```

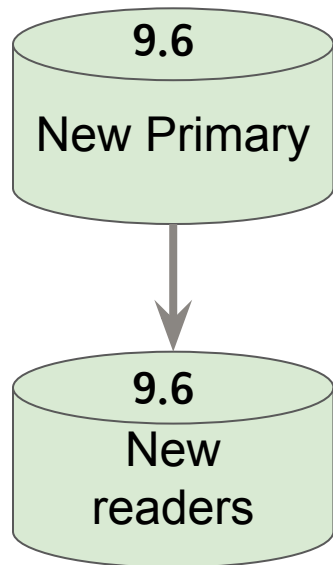
A pg_upgrade with streaming replicas

Step 4f: Start new primary



A pg_upgrade with streaming replicas

Step 4g: Then start streaming replicas



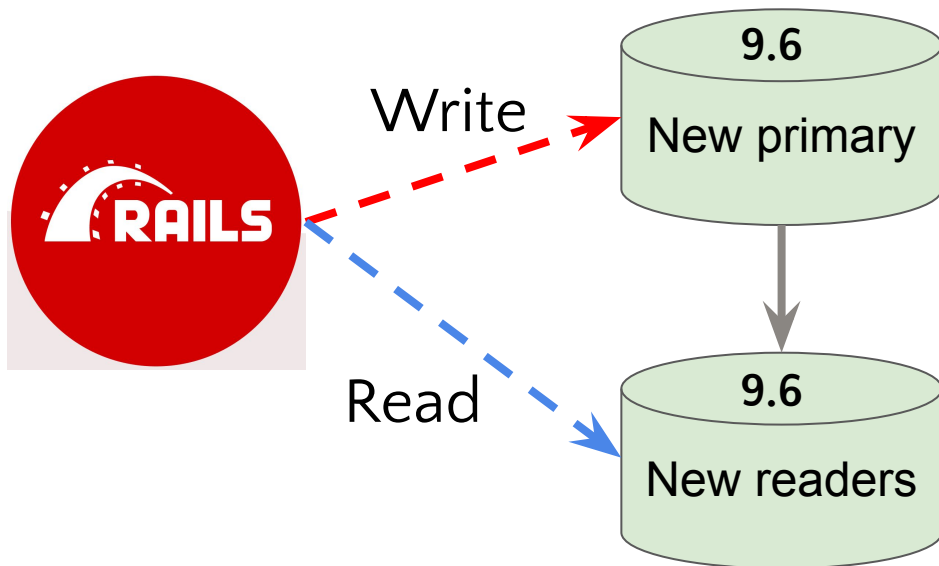
A pg_upgrade with streaming replicas

Step 5: Run ANALYZE on all tables

- Academia-specific: run *before* resuming reads/writes
- We have caught corruption at this stage
- It does extend the required maintenance window

A pg_upgrade with streaming replicas

Step 6: Point application at new nodes, resume writes



A

Should I do this?

No

A

Should I do this?

~~No~~

(it depends)

A

What could go wrong?

- Standby corruption
- Postgres FM | pg_upgrade: the tricky and dangerous parts
- pgsql-hackers: pg_upgrade instructions involving "rsync --size-only" might lead to standby corruption?

A

Why might you want to do this anyway?

- **If you have no other choice**
- **People run this in production, and (some) say it works**
 - This is what matters

Part 2

9.6 -> 15

A



A

How to get from 9.6 to 15

- **✗ pg_dump + pg_restore**
 - Too slow, we can't shut the site down for 2 days


A

How to get from 9.6 to 15

- **✗ pg_dump + pg_restore**
 - Too slow, we can't shut the site down for 2 days
- **✗ pg_upgrade**
 - `--disable-integer-datetime`
 - Postgres 10 removed support for that compile flag

A

How to get from 9.6 to 15

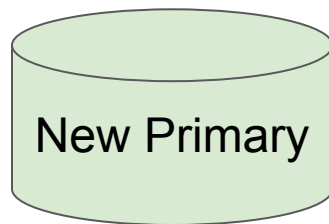
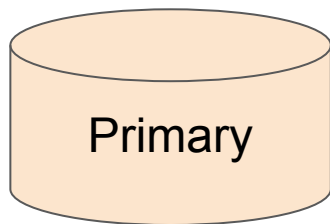
-  **logical replication**
 - “No” downtime
 - Keep logical replica in sync real-time
 - Built-in needs postgres 10 or higher...
 - ...but pglogical extension works on 9.6

A**Recommended reading****PostgreSQL 12 High Availability Cookbook, Shaun Thomas**

- **Chapter 7: PostgreSQL Replication -> pglogical**
 - (if you still need pglogical)
- **Chapter 15: Zero-downtime Upgrades**

A Logical upgrades (high level)

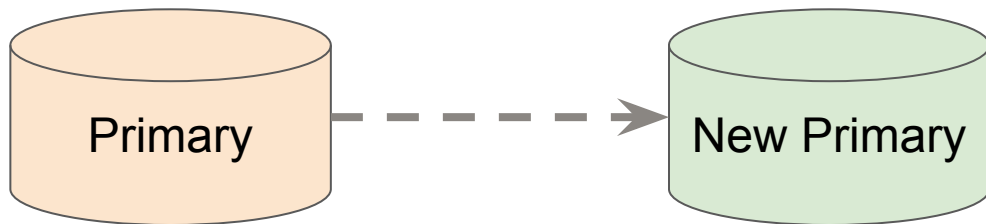
Step 1: Make a new main 15 DB, with the same schema



Read replicas, failovers

A Logical upgrades (high level)

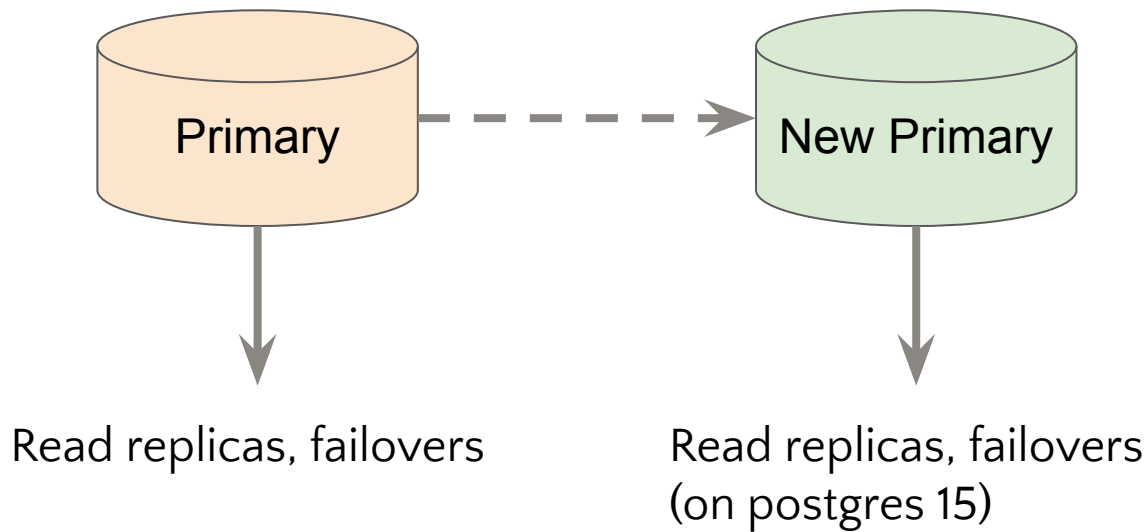
Step 2: Copy data, then stay in sync with changes



Read replicas, failovers

A Logical upgrades (high level)

Step 3: Bring up new tree (streaming replication)

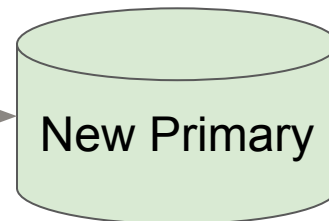
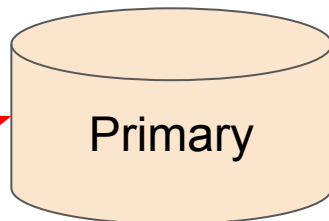


A Logical upgrades (high level)

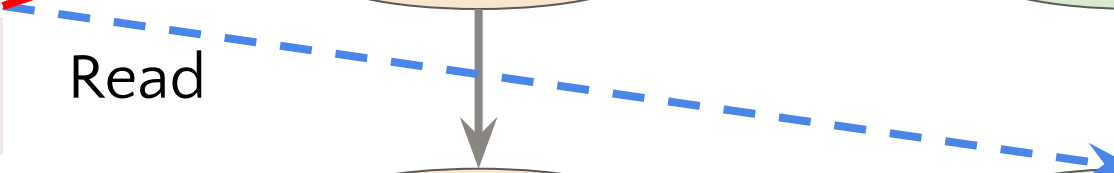
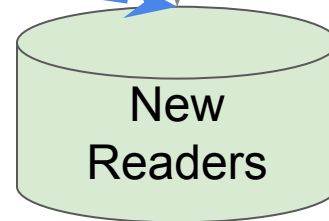
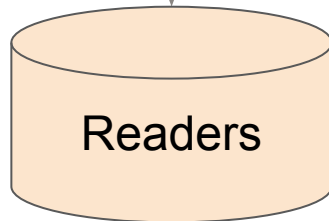
Step 4: Test reads



Write

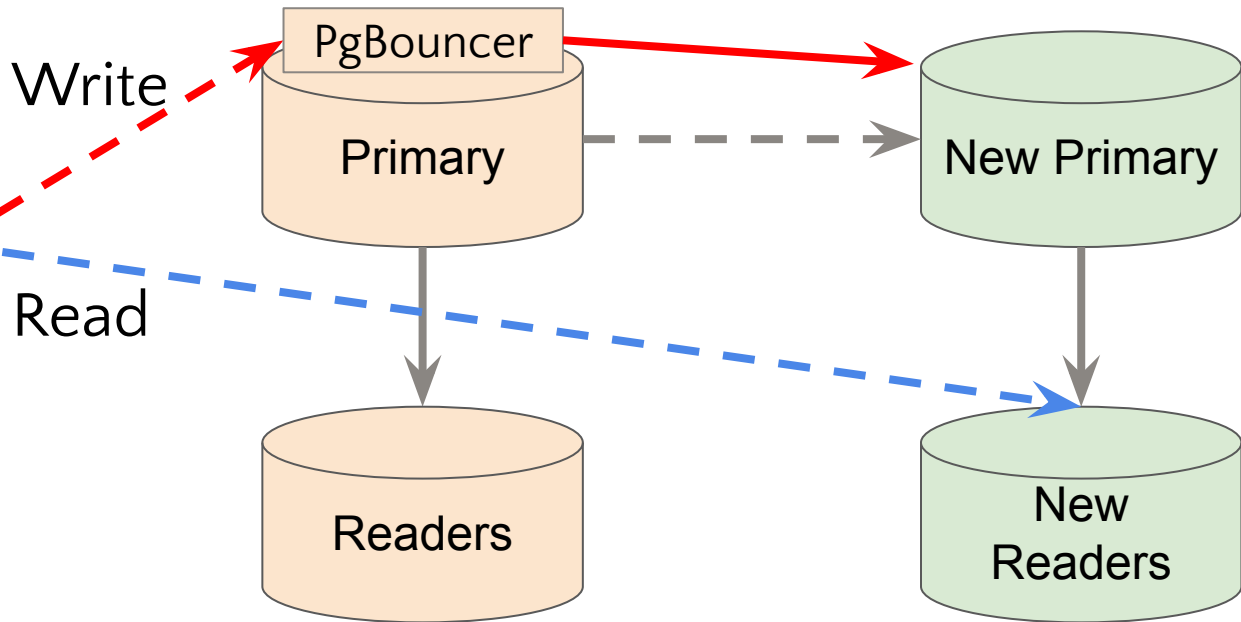


Read



A Logical upgrades (high level)

Step 5: Switch writes



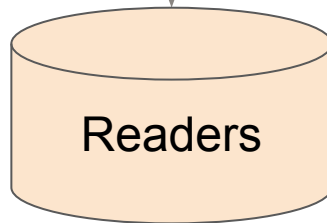
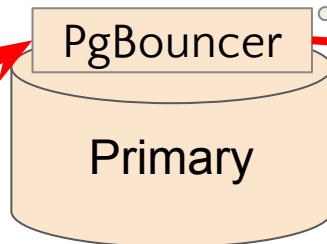
A Logical upgrades (high level)

Step 5: Switch writes

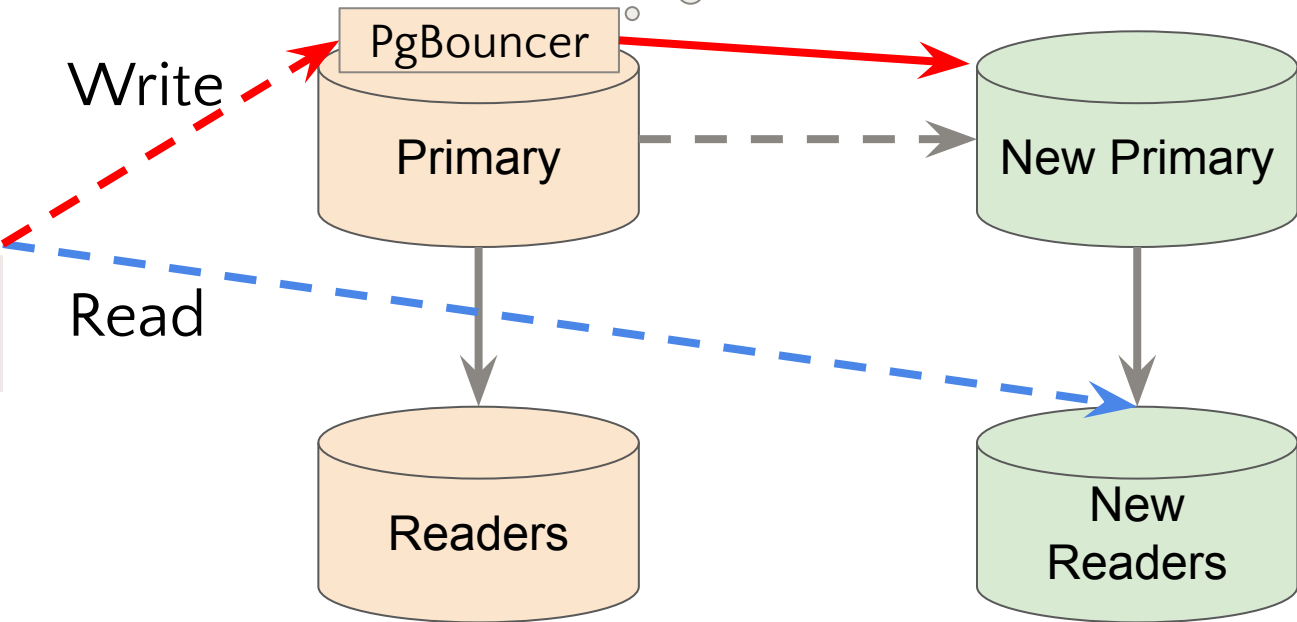
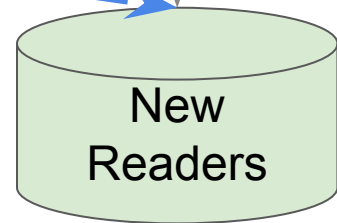
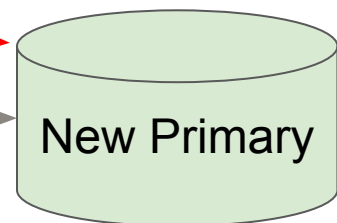


Write

Read

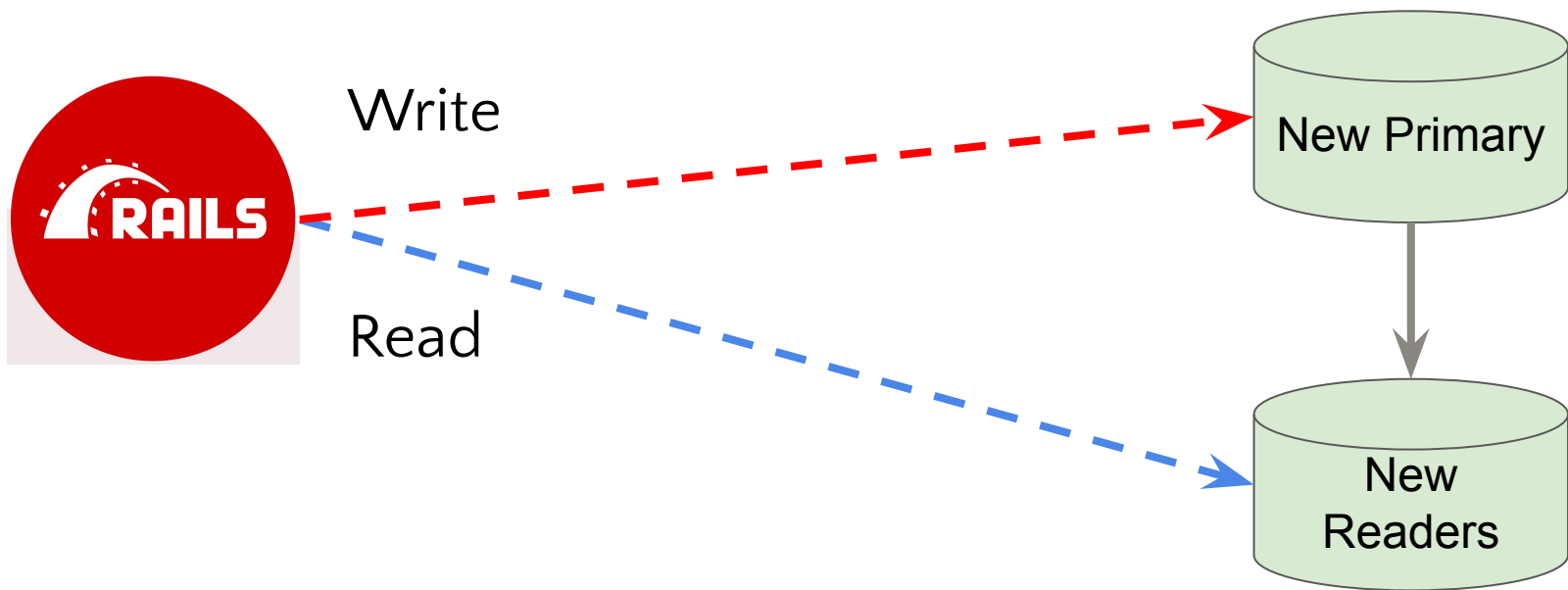


```
PAUSE ;  
RELOAD ;  
RESUME ;
```



A Logical upgrades (high level)

Step 5: Switch writes (after academia-config deploy)



Diving into the details

A

A

“Solved problems”

- **Breaking changes**
 - Parsing pg_dump output
 - Tons of surgical fixes
- **Creating indexes and constraints *after* data load**
- **Monitoring WAL backlog during initial sync**
 - Especially on large tables using TOAST
- **Smooth backup solution transition**

A Reality: complications at every turn (the highlights)

1. Schema changes (migrations)
2. Duplicate strings, in spite of a **UNIQUE** index...
3. pglogical bugs

A

1. Schema changes (migrations)

- 3x-5x per week on average
- Logical replication has no schema change support
- pgl_ddl_deploy extension
 - “Transparent Logical DDL Replication”
- **Magic?**

A

pgl_ddl_deploy: the devil is in the corner cases

- **No CREATE INDEX support (by design)**
 - => manually detect new indexes and create them on 15
 - pg_query ruby gem helped
- **Not all DDL was guaranteed to work**
- **We just didn't trust it to not break**
 - => cumbersome QA + sign-off process

A

2. Duplicate strings, in spite of a UNIQUE index 🤔

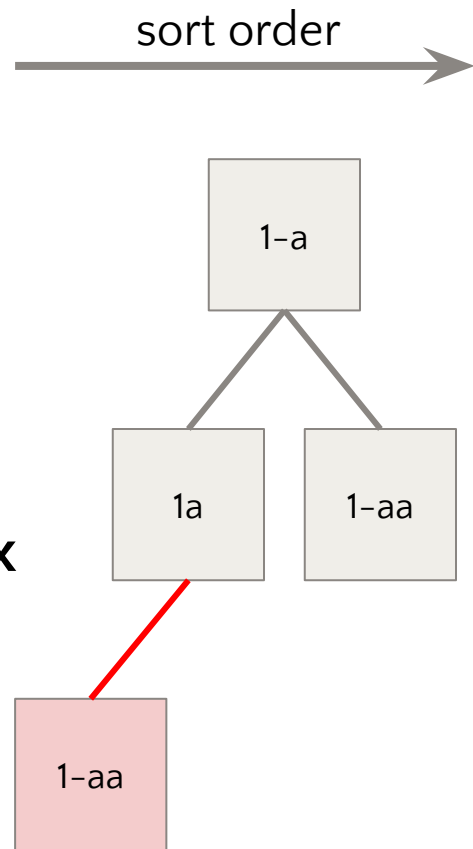
- ```
SELECT
 tbl1.*
FROM
 table_name tbl1
 JOIN table_name tbl2 ON tbl1.name = tbl2.name
WHERE
 tbl1.id = 12345;
```

- There is a unique index on table\_name.name
- But two rows were returned

**A**

## Likely culprit: “in-place” OS upgrade

- Past: ubuntu 14.04 -> 16.04
- Physical replication to a 16.04 replica
- OS upgrade = “glibc” upgrade
  - What postgres uses to sort strings
  - Every version changes its mind about sorting
- String uniqueness enforced by btree index
  - Index persists the old sorting
  - So ever since: index (slightly) wrong



# A

## 3. pglogical bugs

- **Confusing error messages**
- **Sometimes crashes -> data loss**
  - From incorrect replication slot handling
  - Had to monitor for this
- **CREATE INDEX CONCURRENTLY considered harmful?**
  - <https://github.com/2ndQuadrant/pglogical/issues/469>
  - “Simple 60 item checklist” to start over

# Favorite new features





**A**

**Nick Meyer's esoteric list of favorite features (9.3-9.6)**

- **lock\_timeout**
- **pg\_stat\_wal\_receiver**
- **pg\_stat\_progress\_vacuum**
- **VACUUM improvements**

**A**

**Things we probably already know about (10-15)**

- **Built-in logical replication**
- **Declarative partitioning**
- **MERGE**

**A** Nick Meyer's esoteric list of favorite features (10-15)

- `pg_stat_statements_info`

**A**

**Nick Meyer's esoteric list of favorite features (10-15)**

- **pg\_stat\_statements\_info**
- **pg\_stat\_progress\_\***

A

## Nick Meyer's esoteric list of favorite features (10-15)

- `pg_stat_statements_info`
- `pg_stat_progress_*`
  - `pg_stat_progress_create_index`
  - `pg_stat_progress_copy`
  - ~~`pg_stat_progress_basebackup`~~ (`pgbackrest info`)

A

## Nick Meyer's esoteric list of favorite features (10-15)

- **pg\_stat\_statements\_info**
- **pg\_stat\_progress\_\***
  - pg\_stat\_progress\_create\_index
  - pg\_stat\_progress\_copy
  - ~~pg\_stat\_progress\_basebackup~~ (pgbackrest info)
- **pg\_sequences view**

**A**

## Nick Meyer's esoteric list of favorite features (10-15)

- **pg\_stat\_statements\_info**
- **pg\_stat\_progress\_\***
  - pg\_stat\_progress\_create\_index
  - pg\_stat\_progress\_copy
  - ~~pg\_stat\_progress\_basebackup~~ (pgbackrest info)
- **pg\_sequences view**
- **pg\_hba\_file\_rules**

**A**

## Nick Meyer's esoteric list of favorite features (10-15)

- **pg\_stat\_statements\_info**
- **pg\_stat\_progress\_\***
  - pg\_stat\_progress\_create\_index
  - pg\_stat\_progress\_copy
  - ~~pg\_stat\_progress\_basebackup~~ (pgbackrest info)
- **pg\_sequences view**
- **pg\_hba\_file\_rules**
- **psql --csv**



**A**

## Nick Meyer's esoteric list of favorite features (10-15)

- **pg\_stat\_statements\_info**
- **pg\_stat\_progress\_\***
  - pg\_stat\_progress\_create\_index
  - pg\_stat\_progress\_copy
  - ~~pg\_stat\_progress\_basebackup~~ (pgbackrest info)
- **pg\_sequences view**
- **pg\_hba\_file\_rules**
- **psql --csv**
- **max\_slot\_wal\_keep\_size**

# Takeaways



**A**

**1. Your upgrade tools are obsolete**

- Beware of bugs
- Do not expect support
- Don't be afraid to ask for help

**A**

## 2. Be both patient and impatient

- The DB doesn't care about your deadline
- Never “fire and forget” a long running operation
  - Build your own progress bar
  - `pg_stat_progress_copy`
  - `pg_stat_progress_create_index`
  - `df -h`

# A

## 3. Learning vs doing

- **Expect uncertainty in time estimates**
  - But still remember “2. Be both patient and impatient”
- **DEFERRABLE constraints**
- **ALTER TYPE can run in a transaction**
  - But only in postgres 12+
- **glibc**

**A**

## 4. Celebrate with low-hanging fruit

- **strong\_migrations gem**
  - Bump target postgres version
- **Teach someone about pg\_stat\_progress\_copy**
  - (or pg\_stat\_progress\_create\_index)

**A**

5. Let's seek to understand one another

- Solution-splaining
- Can we make upgrades easier for users?



Thanks for listening!

Questions?