The journey towards active-active replication in PostgreSQL

Jonathan Katz
(he/him/his)
Principal Product Manager – Technical
AWS

Agenda

Overview of PostgreSQL replication

Replication deployment models

Evolution of logical replication to support "active-active"

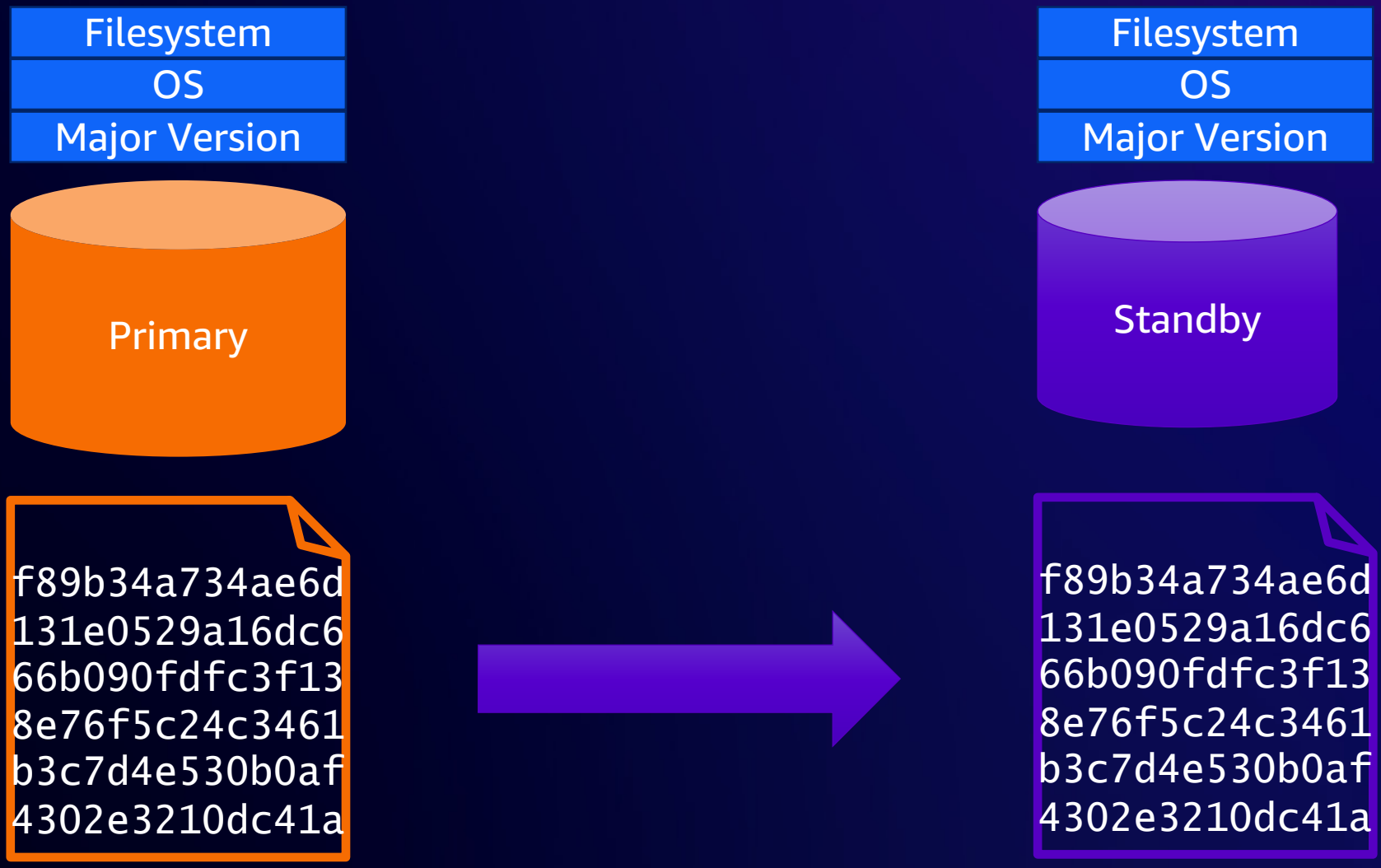
Roadmap for PostgreSQL to support active-active

Overview of PostgreSQL replication

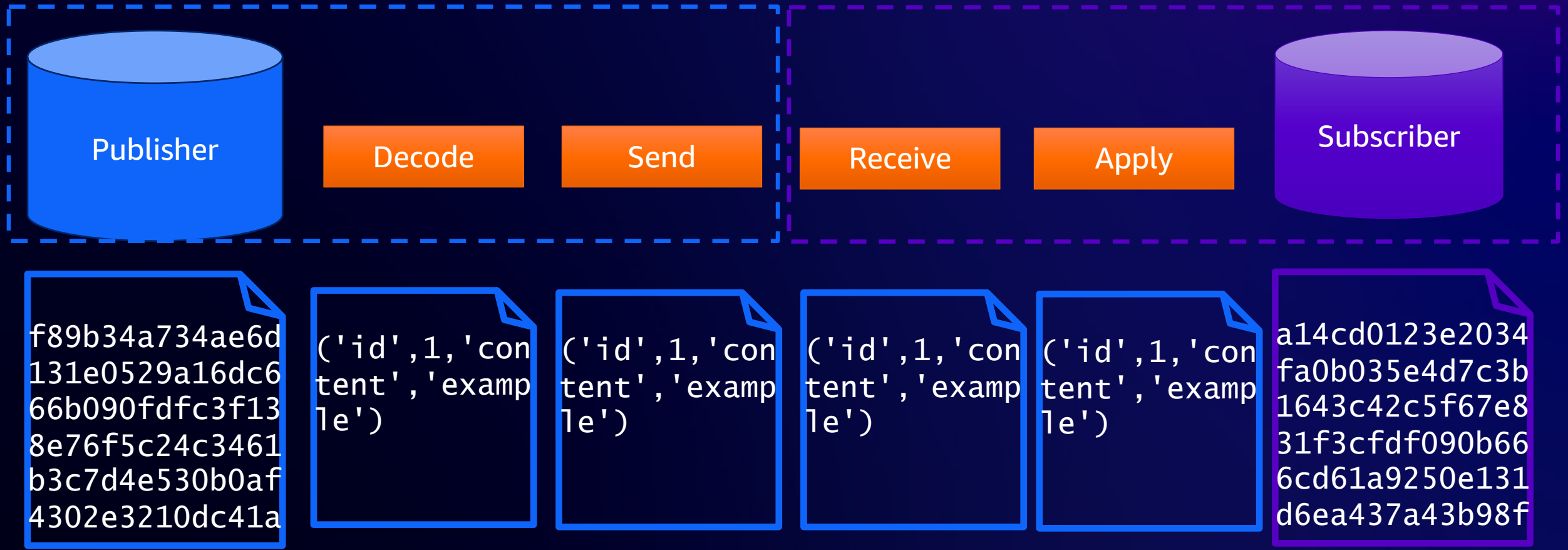
What is replication?

- Copying data between systems
- Physical replication
  - Copies data exactly as it appears on disk
- Logical replication
  - Copies data in a format that can be interpreted by other systems
  - Publisher / subscriber model
  - Can replicate between heterogeneous systems

Physical replication



Logical replication

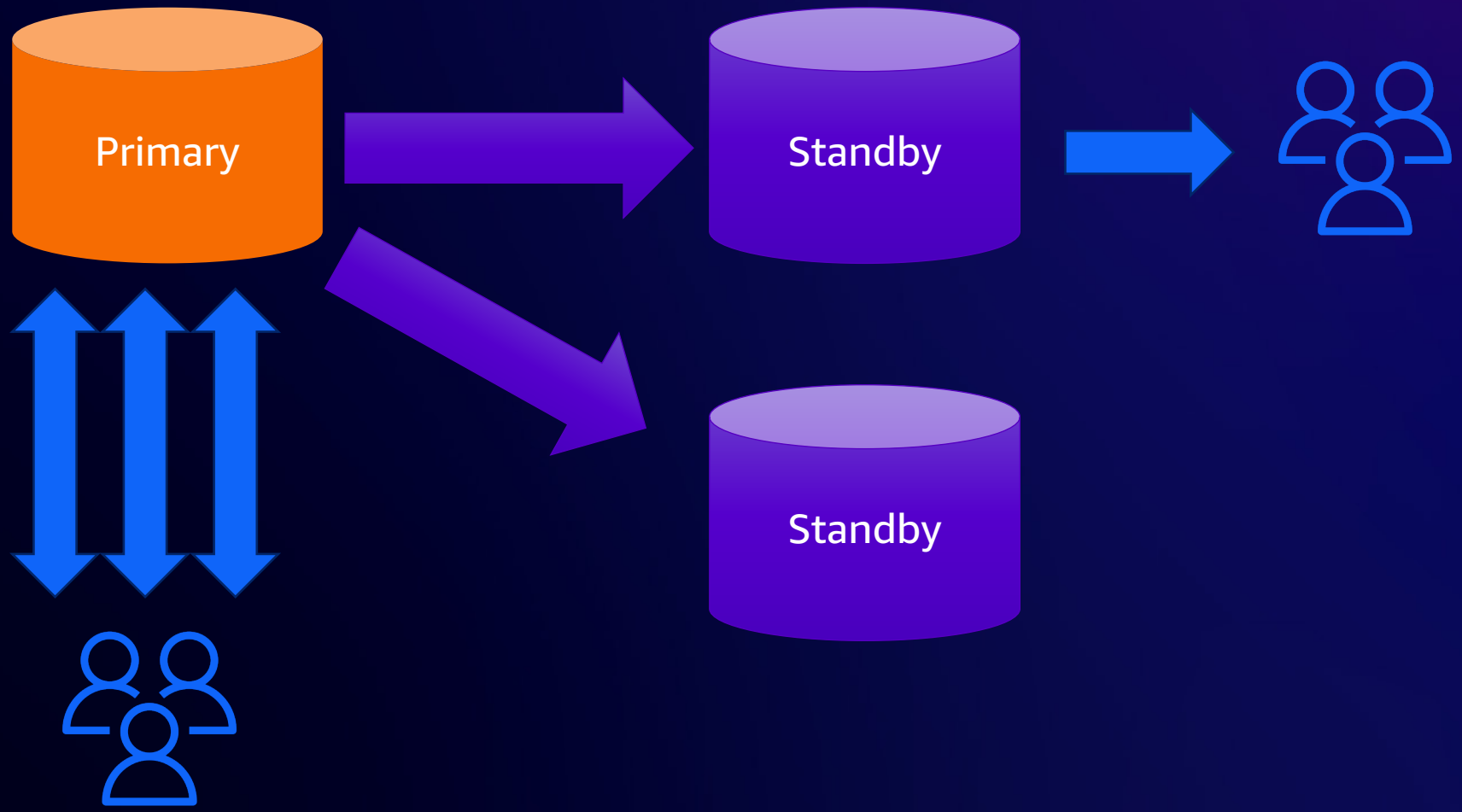


Current use-cases for replication with PostgreSQL

- High availability
- Load balancing read queries
- Change data capture (CDC)
- Extract-transform-load (ETL)
- Data warehousing
- Online major version upgrades
- System migrations
- Data residency (to a degree)

Replication deployment models

Active-standby deployment model



Active-standby deployment model

- One primary (active), one or more replicas (standby)
- Choice of synchronous / quorum commit or asynchronous
- Use-cases
  - High availability
  - Read load balancing
  - Real-time analytics

Active-standby advantage and tradeoffs

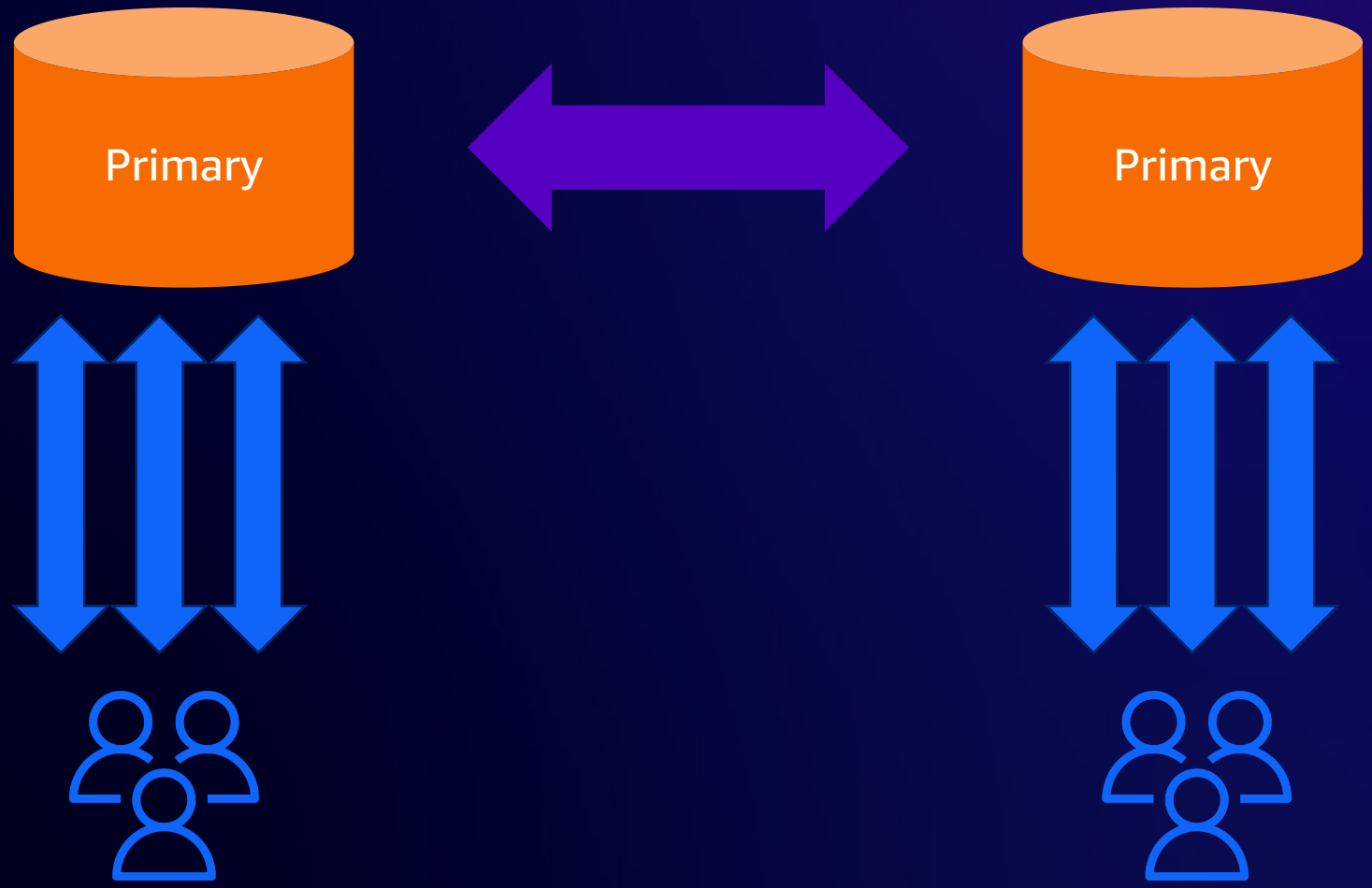
Advantages

- Simple consistency model: one "source of truth"
- Simple for application development

Tradeoffs

- "Extra work" in promoting a standby
  - Heartbeat
  - Determine "best available" standby
  - Write traffic redirection

Active-active deployment model



Active-active deployment model

- One or more primaries (active) that replicate between each other
  - Can also include standbys, but not in "high availability set"
- Use-cases
  - High availability
  - "Blue / green" deployments (upgrades, application changes)
  - System migrations

Active-active advantage and tradeoffs

Advantages

- "No failover" – redirect write traffic

Tradeoffs

- Requires conflict detection / resolution
- Applications need to be designed for active-active

What does PostgreSQL need to support active-active?

- PostgreSQL already supports active-active*
- Logically replicate between partitions across different publishers
- Some extensions / 3rd party tools provide "active-active" support
- PostgreSQL 16 added "bidirectional replication" support

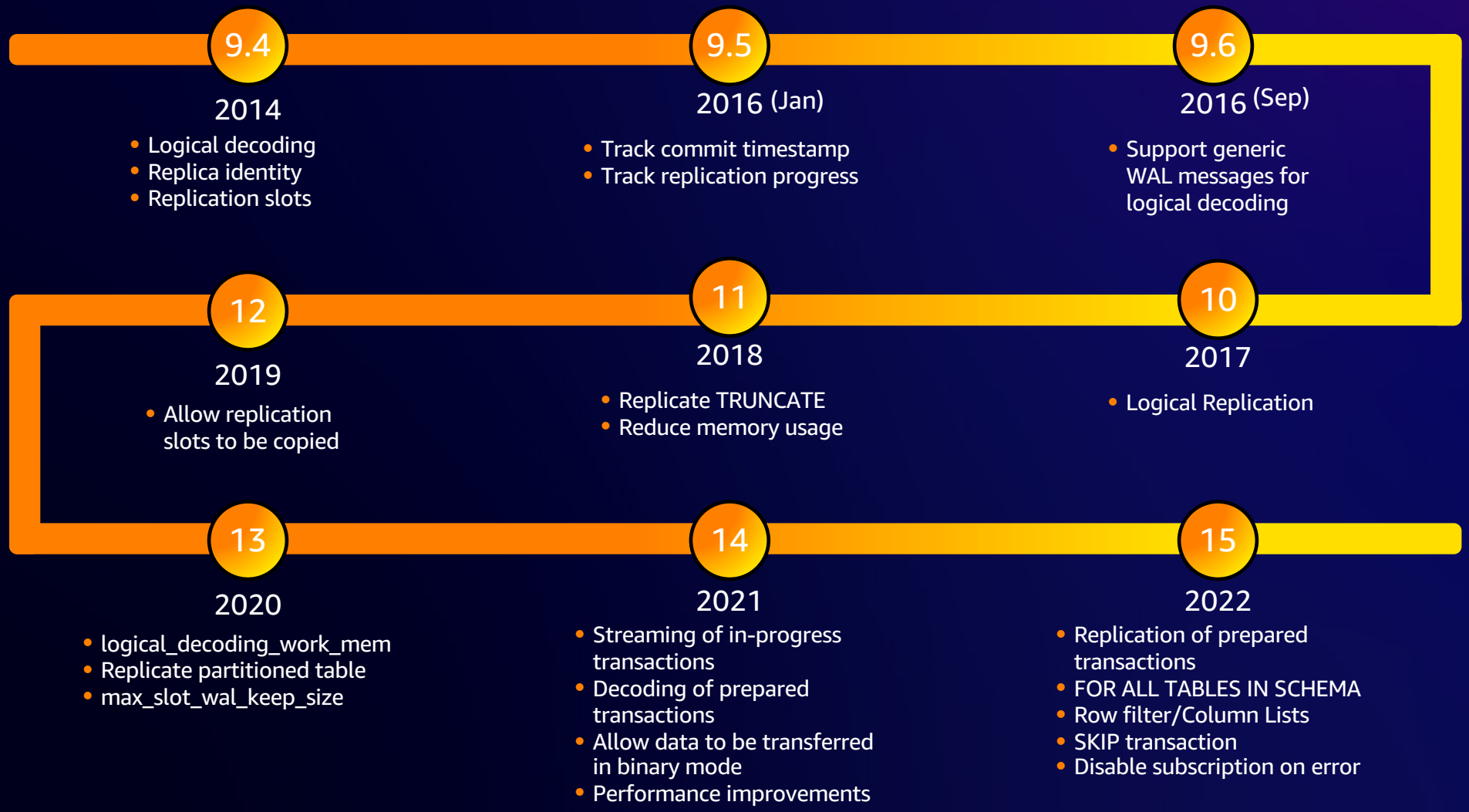
What does PostgreSQL need to *better* support active-active?

Features that allow PostgreSQL to natively support active-active:

- Replication of all/most objects
- Replication of all/most commands
- Improvements to conflict detection
- Conflict resolution / conflict statistics
- Node synchronization
- (Two-phase commit (2PC) transaction manager?)

Evolution of logical replication to support active-active

Evolution of logical replication in PostgreSQL



Logical replication enhancements in PostgreSQL 16

PostgreSQL 16 logical replication highlights

Origin filtering ("bidirectional replication")

Logical replication from standbys

Parallel apply of transactions

Security enhancements

Initialize subscriber using binary format

Use indexes on subscriber for `REPLICA IDENTITY FULL`

What is "origin filtering?"

- "origin" identifies where a change original comes from

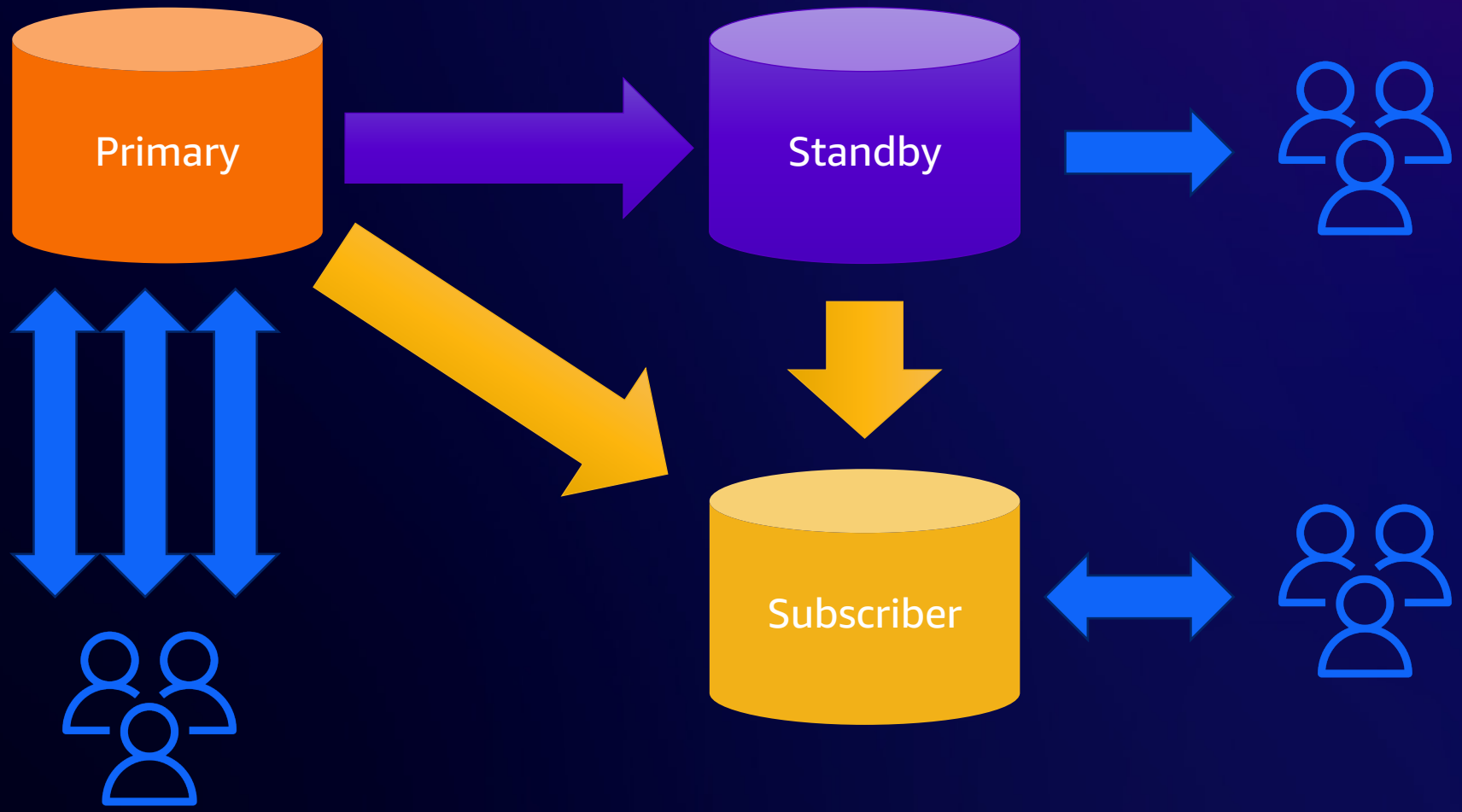
- `origin=any` (default)
  - publisher sends all changes regardless of origin

- `origin=none`
  - subscriber only requests changes that do not have origin set
  - Prevents loopbacks
  - Allows bidirectional / n-way replication

Origin filtering example

```
CREATE PUBLICATION pub_primary1 FOR ALL TABLES; -- on primary1
CREATE PUBLICATION pub_primary2 FOR ALL TABLES; -- on primary2

-- On primary1
CREATE SUBSCRIPTION sub_to_primary2
  CONNECTION <connection_to_primary2>
  PUBLICATION pub_primary2 WITH (origin = none);

-- On primary2
CREATE SUBSCRIPTION sub_to_primary1
  CONNECTION <connection_to_primary1>
  PUBLICATION pub_primary1 WITH (origin = none);
```

Logical replication from standbys



Logical replication replication from standbys

Initial proposal in 2016

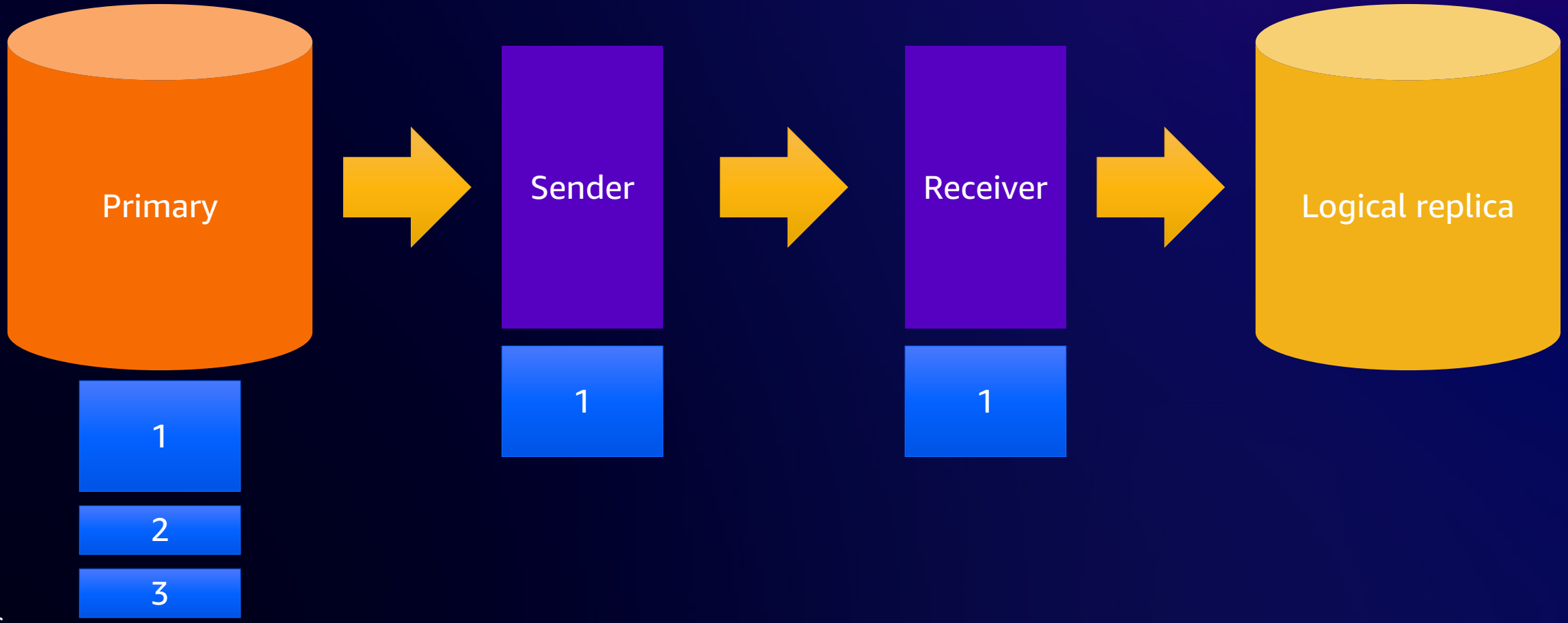
Next major effort in 2018

Efforts continue on and off for years

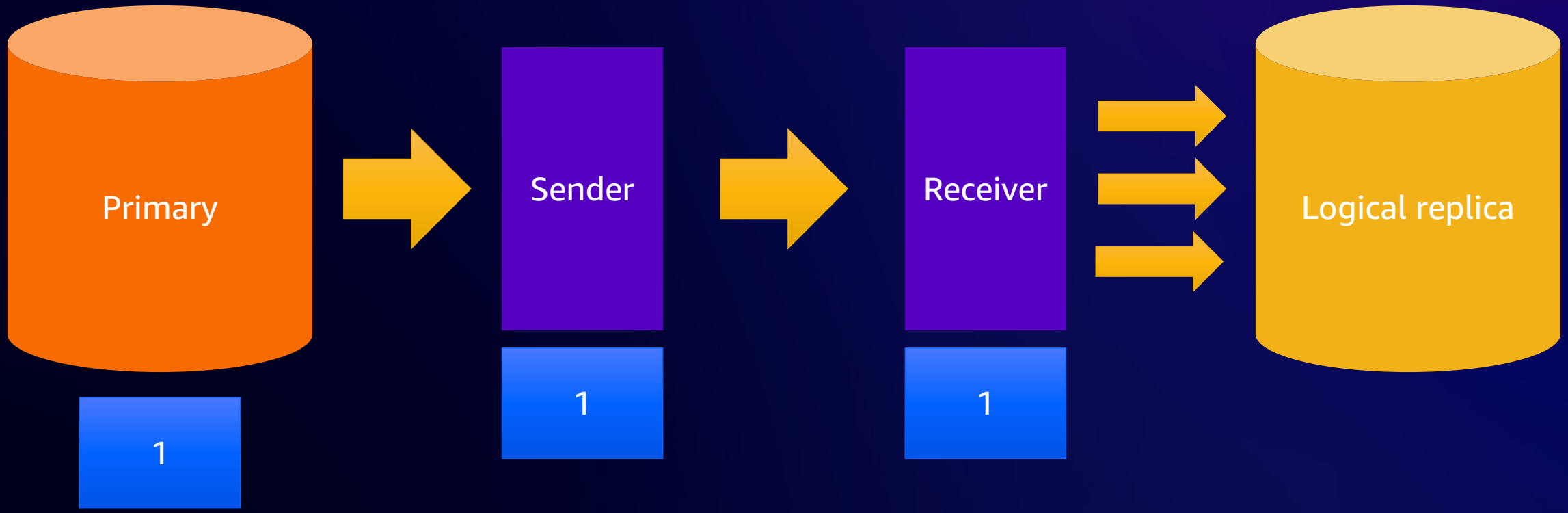
Collaboration between AWS, Microsoft, EDB, Fujitsu

```
Hi,

On 2023-04-07 14:27:09 -0700, Andres Freund wrote:
> I think I'll push these in a few hours. While this needed more changes than
> I'd like shortly before the freeze, I think they're largely not in very
> interesting bits and pieces - and this feature has been in the works for about
> three eternities, and it is blocking a bunch of highly requested features.
>
> If anybody still has energy, I would appreciate a look at 0001, 0002, the new
> pieces I added, to make what's now 0003 and 0004 cleaner.

Pushed. Thanks all!
```

Parallel apply of large transactions



Deep dive on parallel apply of large transactions



Logical replication security enhancements

- Apply transactions as table owner
  - `run_as_owner`
- Allow non-superusers to create subscriptions
  - `pg_create_subscription`

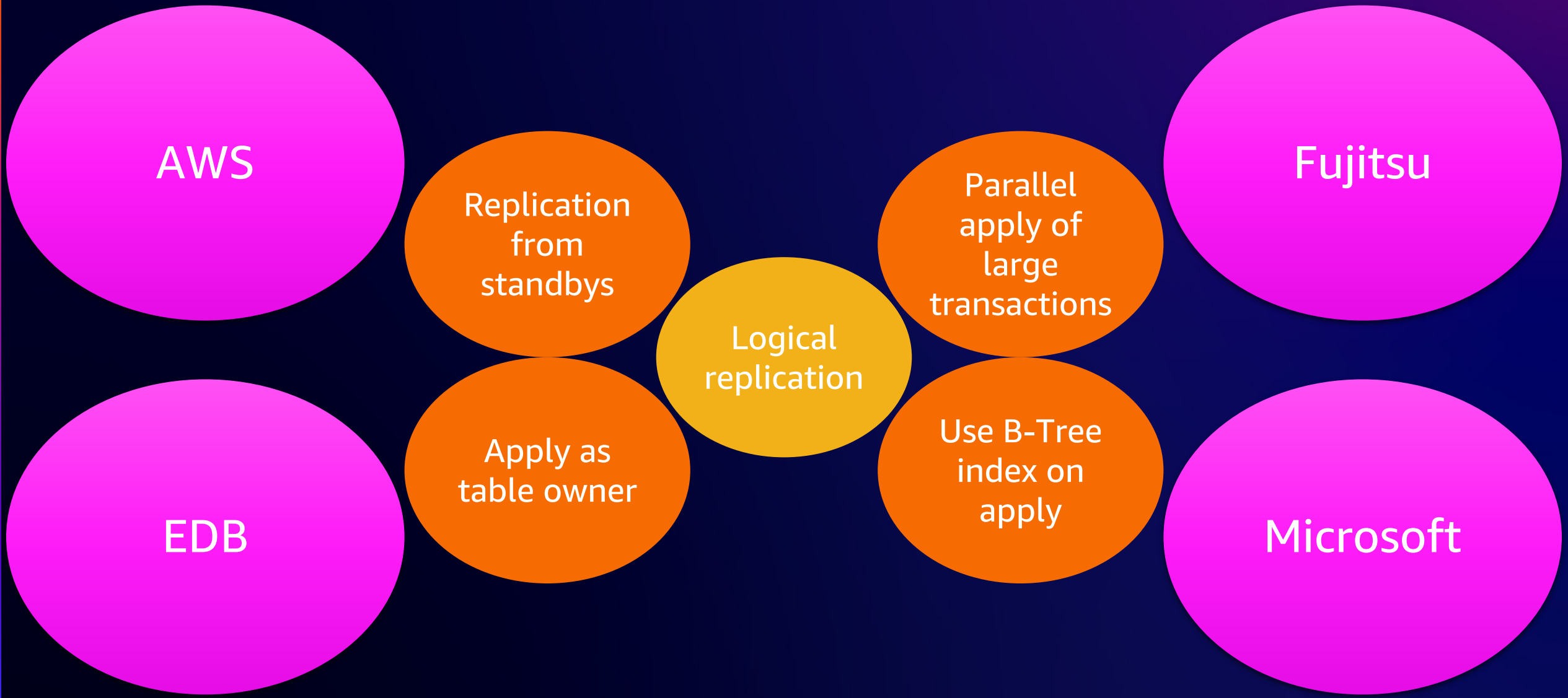
Initialize subscriber using binary format

- Prior to PostgreSQL 16, `binary` only allowed replication in binary format
- Reduces initialization overhead using binary format
- Using binary format restricts where you can replicate changes

```
CREATE SUBSCRIPTION sub1
   CONECTION ..
   PUBLICATION pub1
   WITH (binary = true);
```

Use indexes for `REPLICA IDENTITY FULL`

- `PRIMARY KEY` / `REPLICA IDENTITY` lookups can use an index to find target row
- `REPLICA IDENTITY FULL` requires full sequential scan
- PostgreSQL 16 allows `REPLICA IDENTITY FULL` to use a B-tree index to find target row



Roadmap to support active-active deployments

Logical replication in PostgreSQL 17 and beyond

- DDL replication
  - Deparse the command to pass it in a standard format like JSON
  - Replication of DDL commands
  - Initial sync
- Replication of sequences
- Synchronization of replication slots to allow failover
- Upgrade of logical replication nodes
- Reuse of tablesync workers
- Time-delayed replication

Features to support active-active deployments

- Conflict management
  - Detection
  - Last commits wins resolution
  - Monitoring
- Node initialization, synchronization, resynchronization, pause / resume
- Sequence access methods
- Performance
  - Decoding
  - Apply process
  - Lag recovery

Thank you!

Please complete the session survey in the mobile app

Jonathan Katz

jkatz@amazon.com