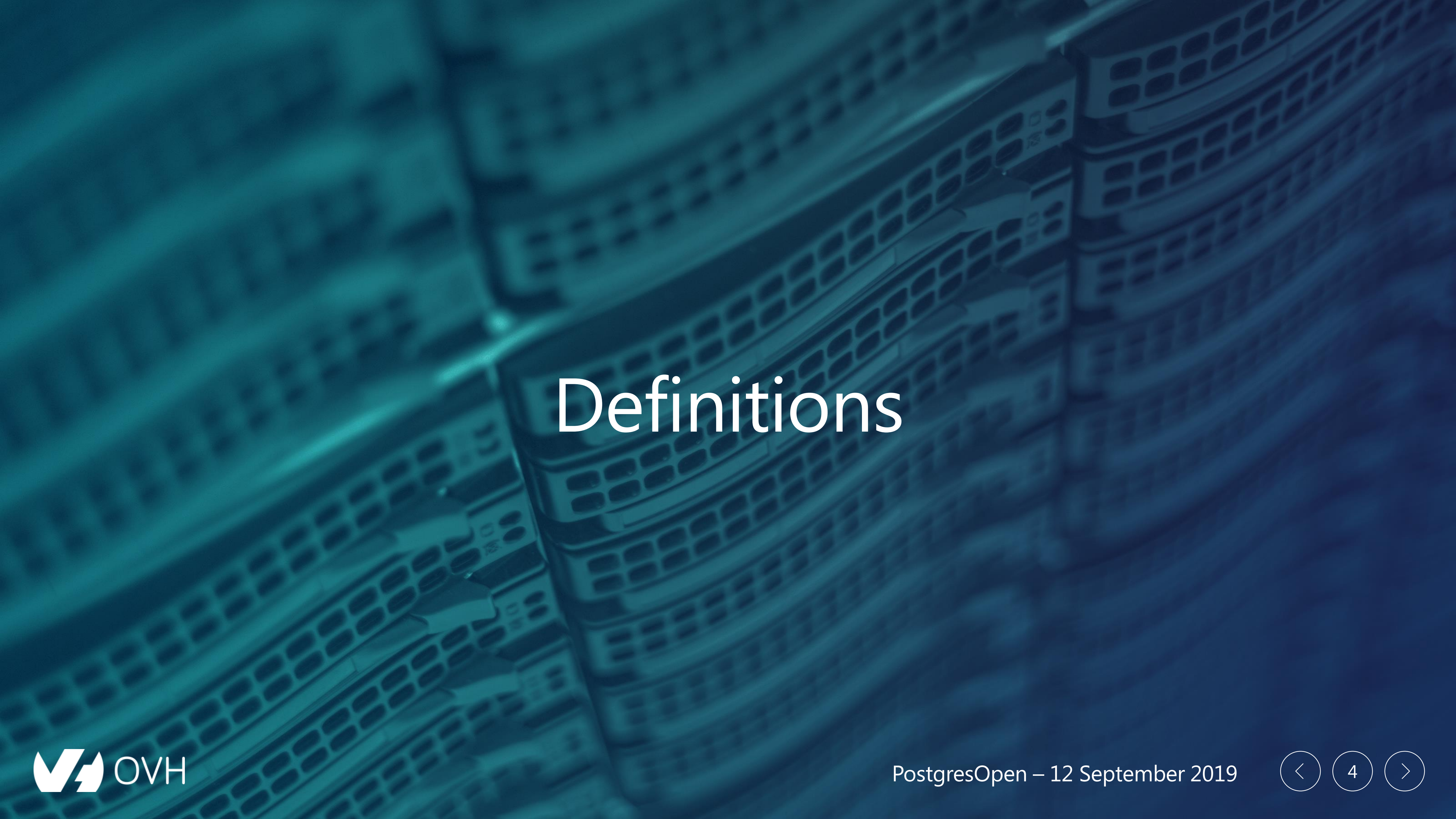
https://knowyourmeme.com/memes/all-the-things

Speaker

- Julien Riou
- DBA since 2012
- Tech lead in the databases team at OVH since 2015
- pgterminate @ github

Overview

- Definitions
- Context
- Updates
- Upgrades
- Conclusion
- What’s next?



Definitions

OVH

Versioning policy

- Starting from version 10

11.4

Major version Minor version

Versioning policy

- Before version 10

9.6.14

Major version Minor version

Versioning policy

- Major versions
  - Released about once a year
  - Includes new features
  - Supported for 5 years

- Minor versions
  - Released at least every 3 months
  - Includes bug and security fixes
  - Critical fixes are released as soon as possible

Definitions

Update

Installing a newer **minor** version of PostgreSQL

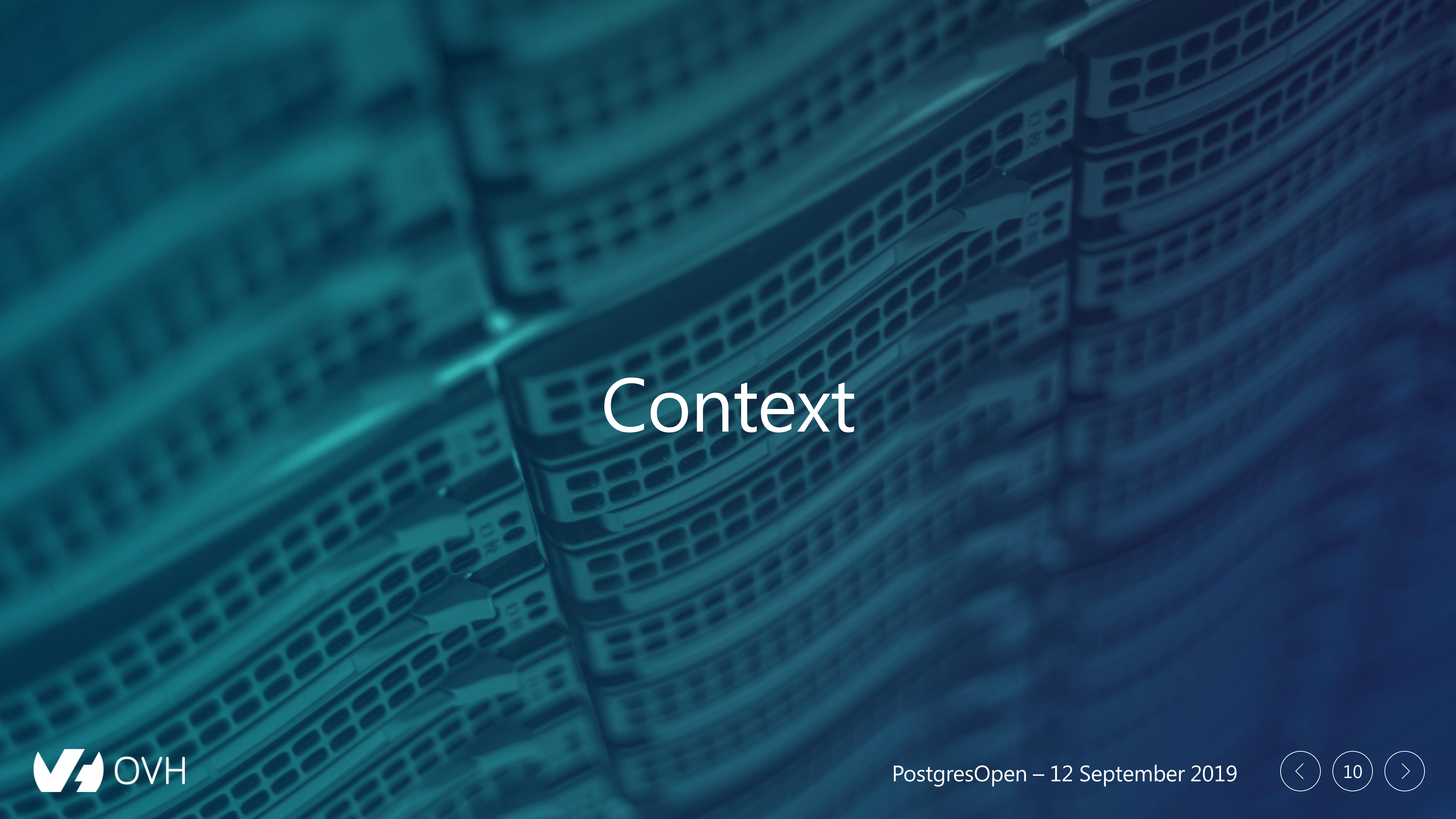
"Minor upgrade" accepted too

Definitions

Upgrade

Installing a newer **major** version of PostgreSQL

"Major upgrade" accepted too



Context

OVH

OVH

Innovation for Freedom

Products

Cloud
Baremetal
VPS
Public cloud
Private cloud
Storage

Platform
Kubernetes
Logs & Metrics Data Platforms
Databases
Big data
AI & Machine Learning

Web hosting
Domain names
Website hosting
E-mail solutions
SSL / CDN
Office & Microsoft solutions

Telecom
Internet offers
Telephony
SMS / Fax
VDI
OverTheBox

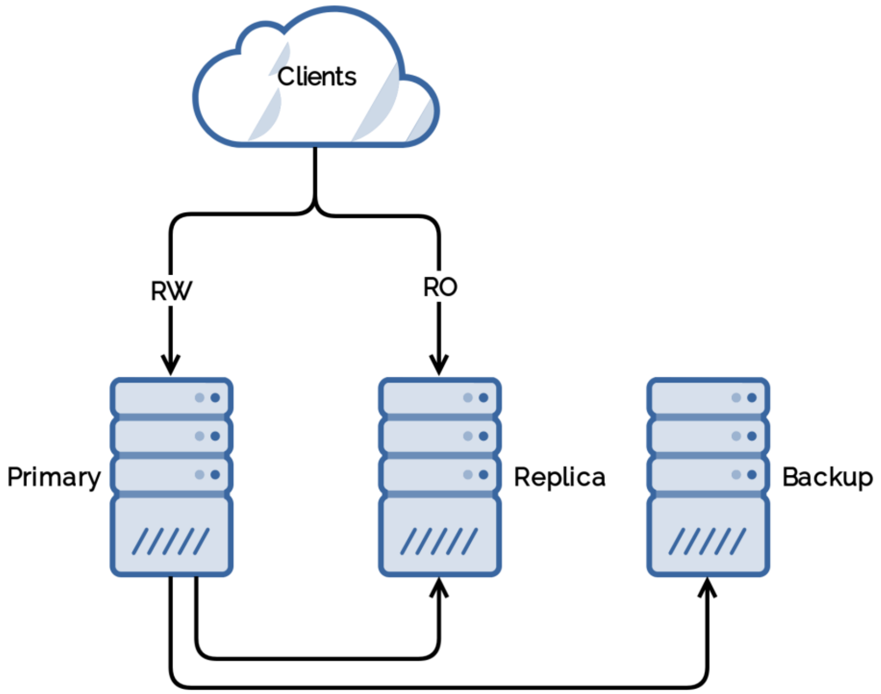
Perimeter

Internal databases

60	3000	700	400
Clusters	Applications	Users	Databases

Cluster example

- MySQL
- PostgreSQL



Updates

Recommendations

- “We always recommend that all users run the latest available minor release for whatever major version is in use.”
- “For minor releases, the community considers not upgrading to be riskier than upgrading.”

https://www.postgresql.org/support/versioning/

https://knowyourmeme.com/memes/all-the-things

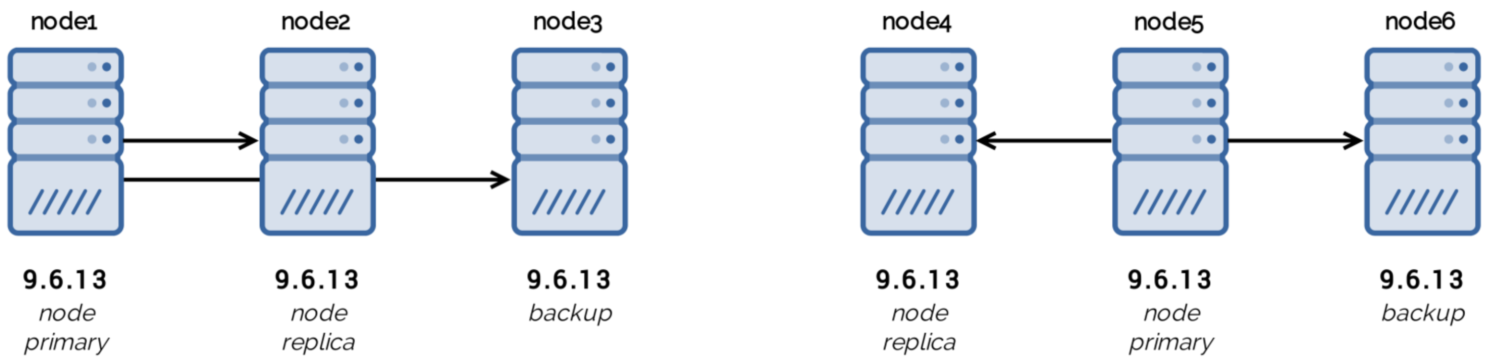
Method

1. Stop the service
2. Install new binaries
3. Start the service

Attention points

- Always read the changelog
- Downtime
  - Can be minimized by using pgbouncer and PAUSE/RESUME commands
- Write intensive clusters
  - Run CHECKPOINT before stopping the service
  - Prepare for a switchover for extreme case
- Patroni
  - Put the cluster on maintenance mode to avoid failovers

Initial state



Clustershell

- “Event-driven open source Python library, designed to run local or distant commands in parallel on server farms or on large Linux clusters”

  http://cea-hpc.github.io/clustershell/

- Binaries
  - clush
  - nodeset

- Python API

Clustershell

- nodeset

```
$ nodeset -ll
@all node[1-6]
@cluster1 node[1-3]
@cluster2 node[4-6]
@node node[1-2,4-5]
@backup node[3,6]
```

Clustershell

- clush

```
$ clush -bw @all
$ clush -bw @cluster1\&@backup
$ clush -bw @cluster1,@cluster2
```

Clustershell

- clush

```
clush> apt-get update
Clush> apt-get upgrade
```

Clustershell

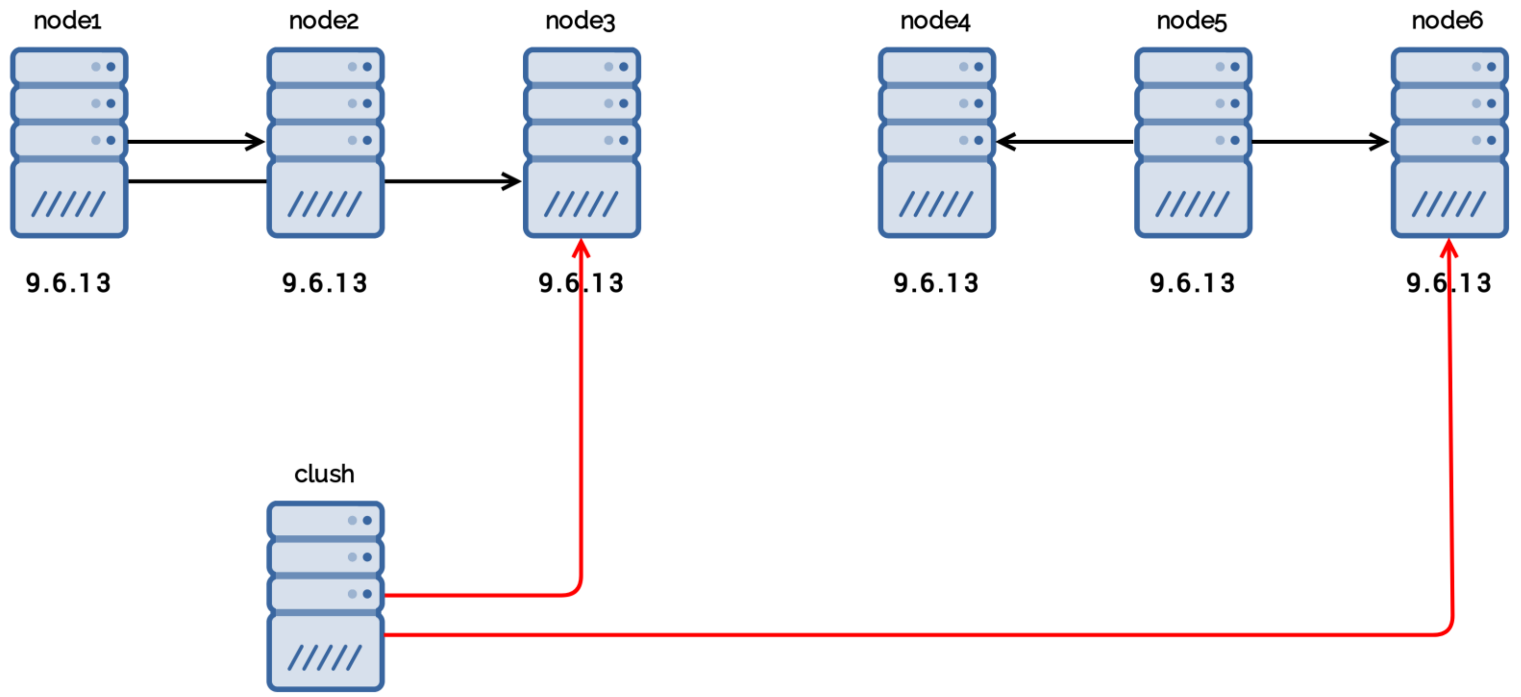
- Backups first

```
$ clush -bw @backup
```

Clustershell

- Backups first

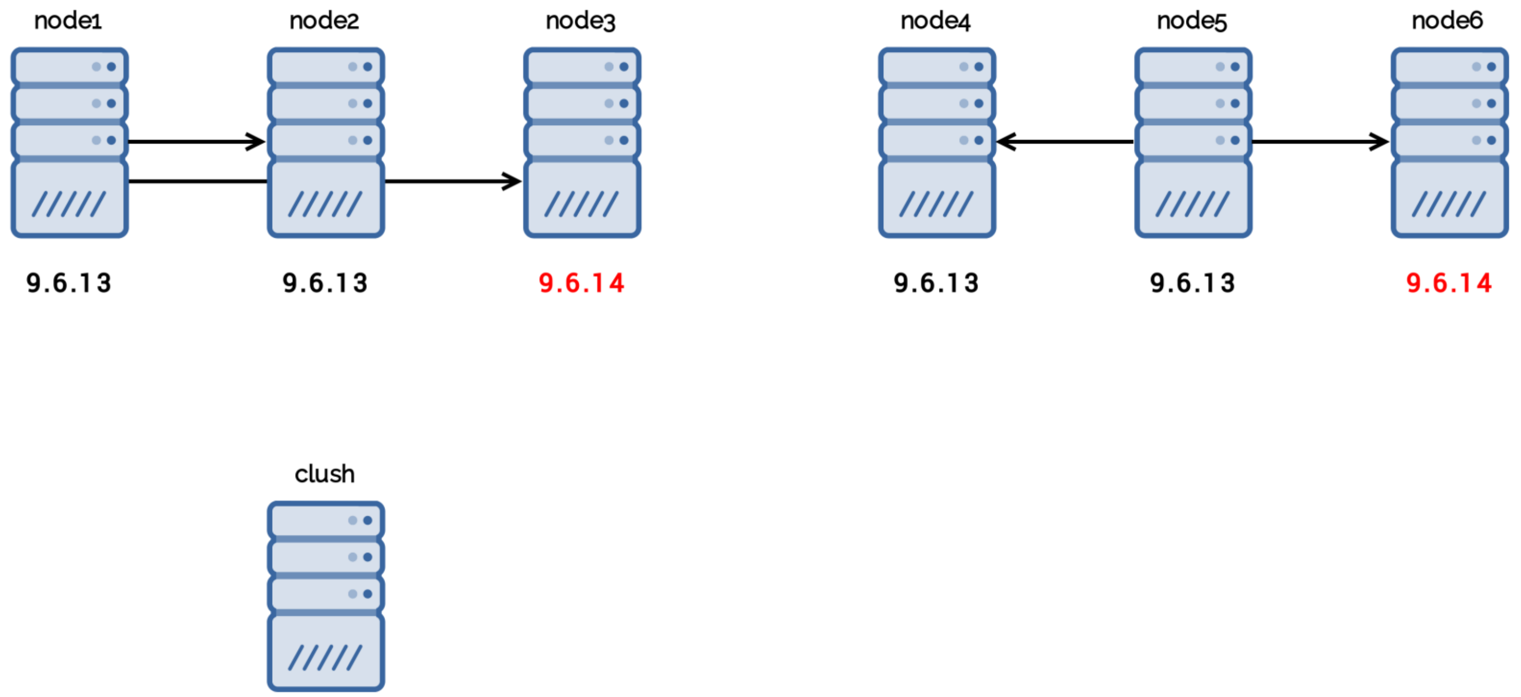
```
$ clush -bw @backup
```



Clustershell

- Backups first

```
$ clush -bw @backup
```



Clustershell

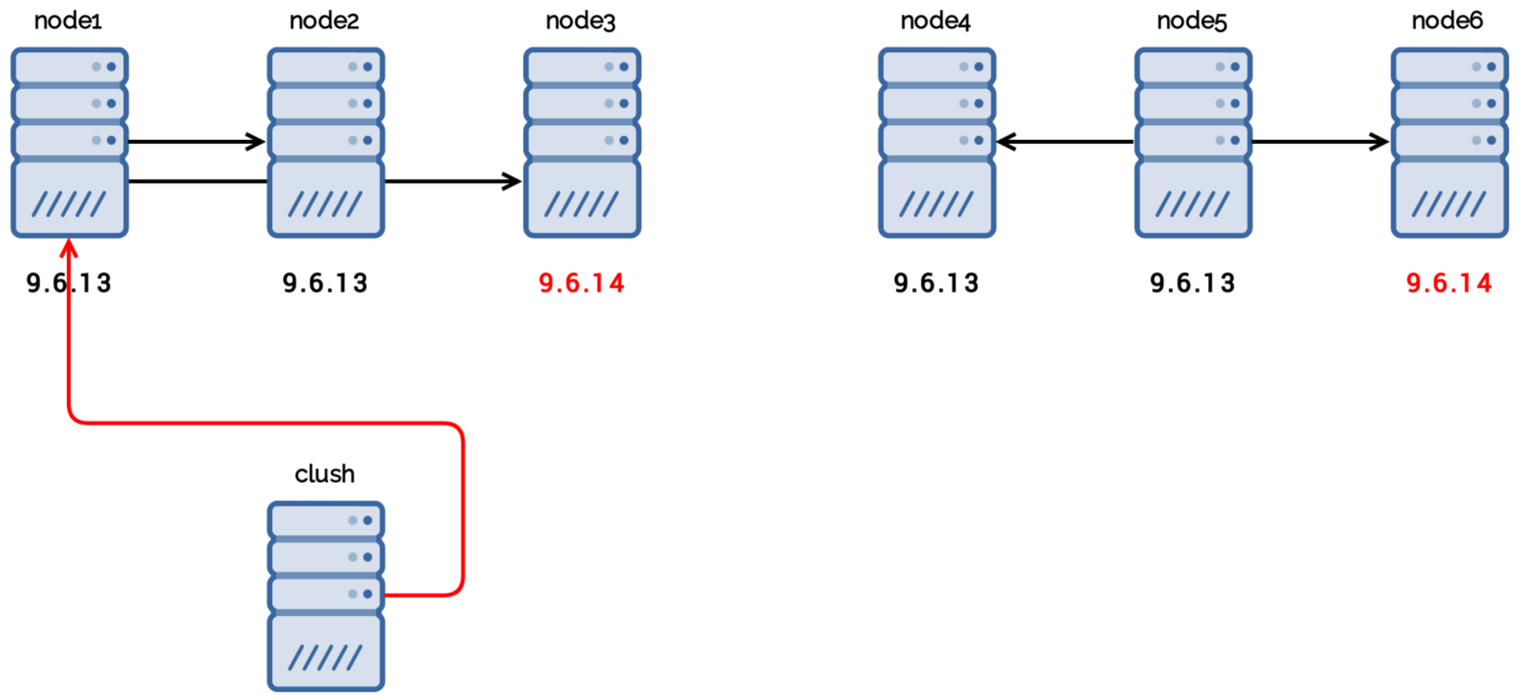
- Then nodes one node at a time (fanout)

```
$ clush -f 1 -bw @node
```

Clustershell

- Then nodes one node at a time (fanout)

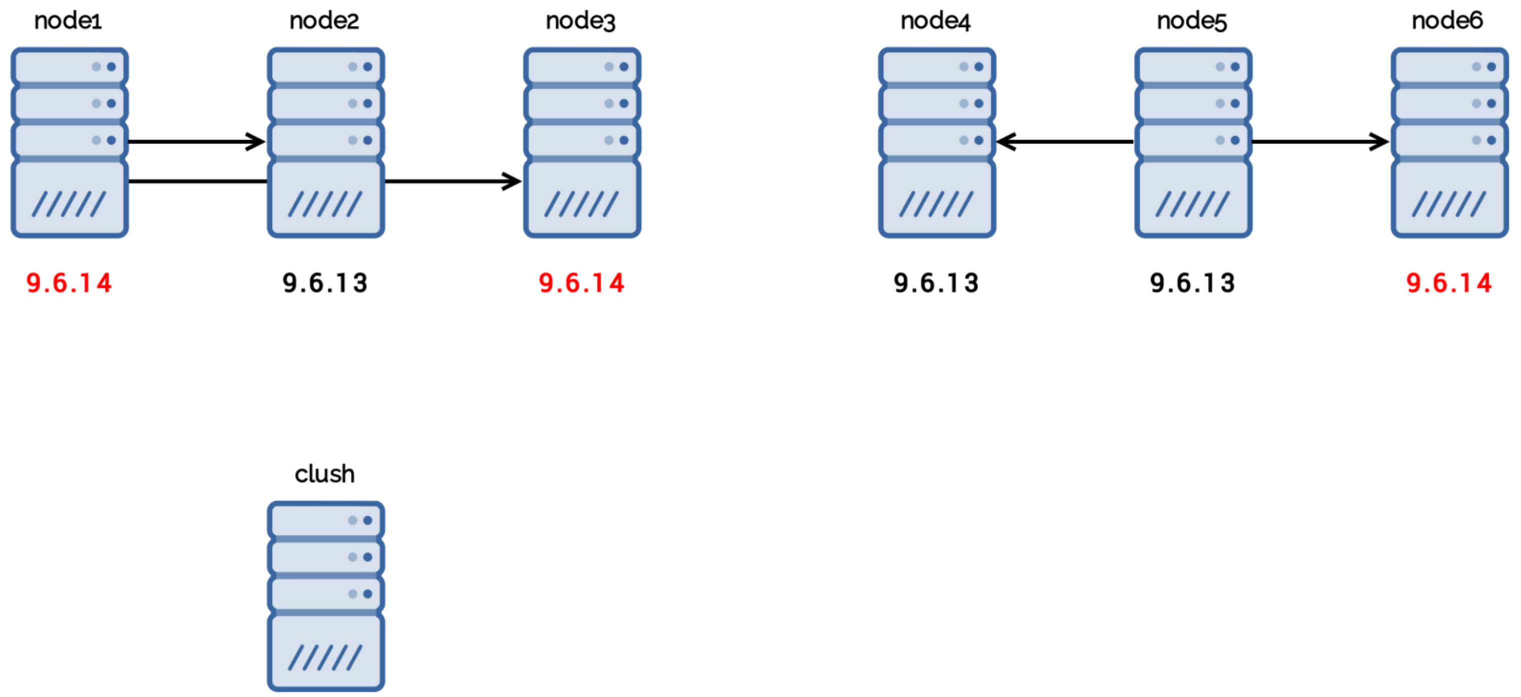
```
$ clush -f 1 -bw @node
```



Clustershell

- Then nodes one node at a time (fanout)

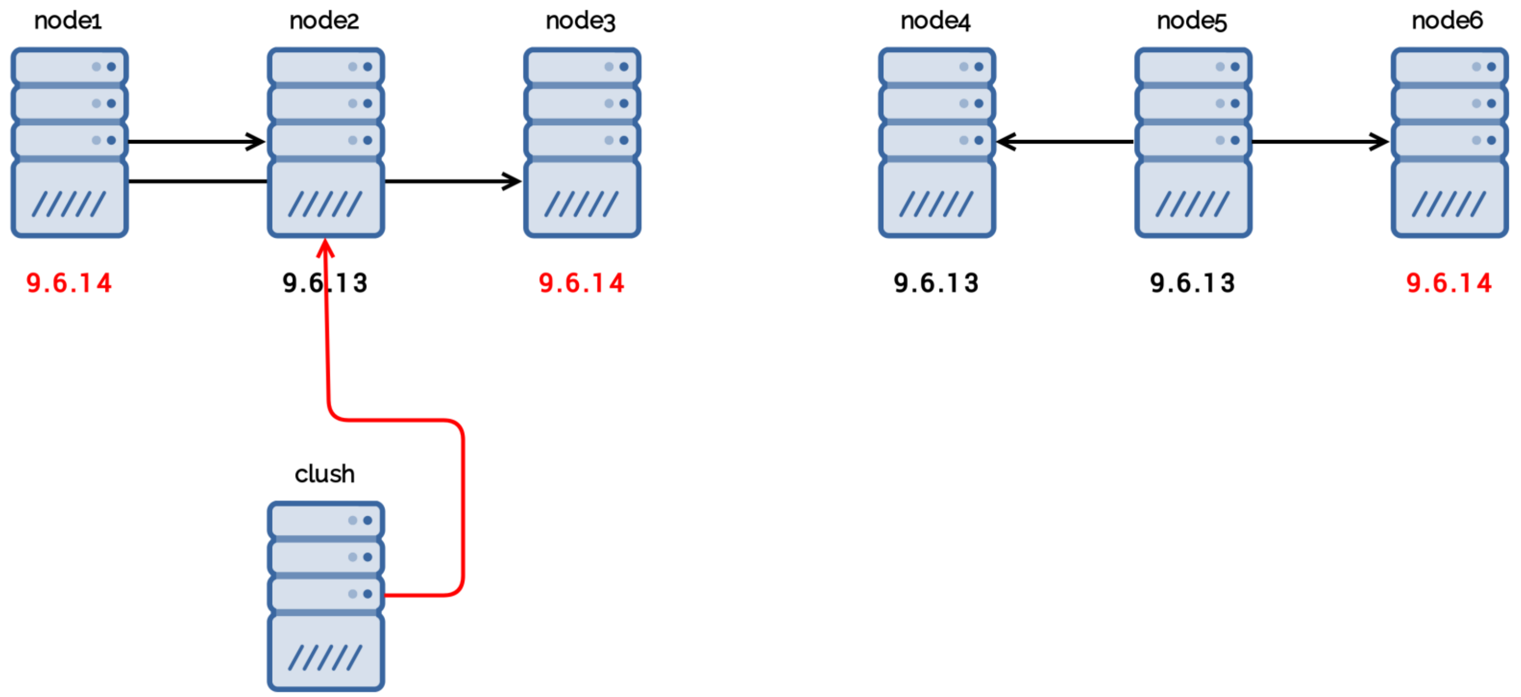
```
$ clush -f 1 -bw @node
```



Clustershell

- Then nodes one node at a time (fanout)

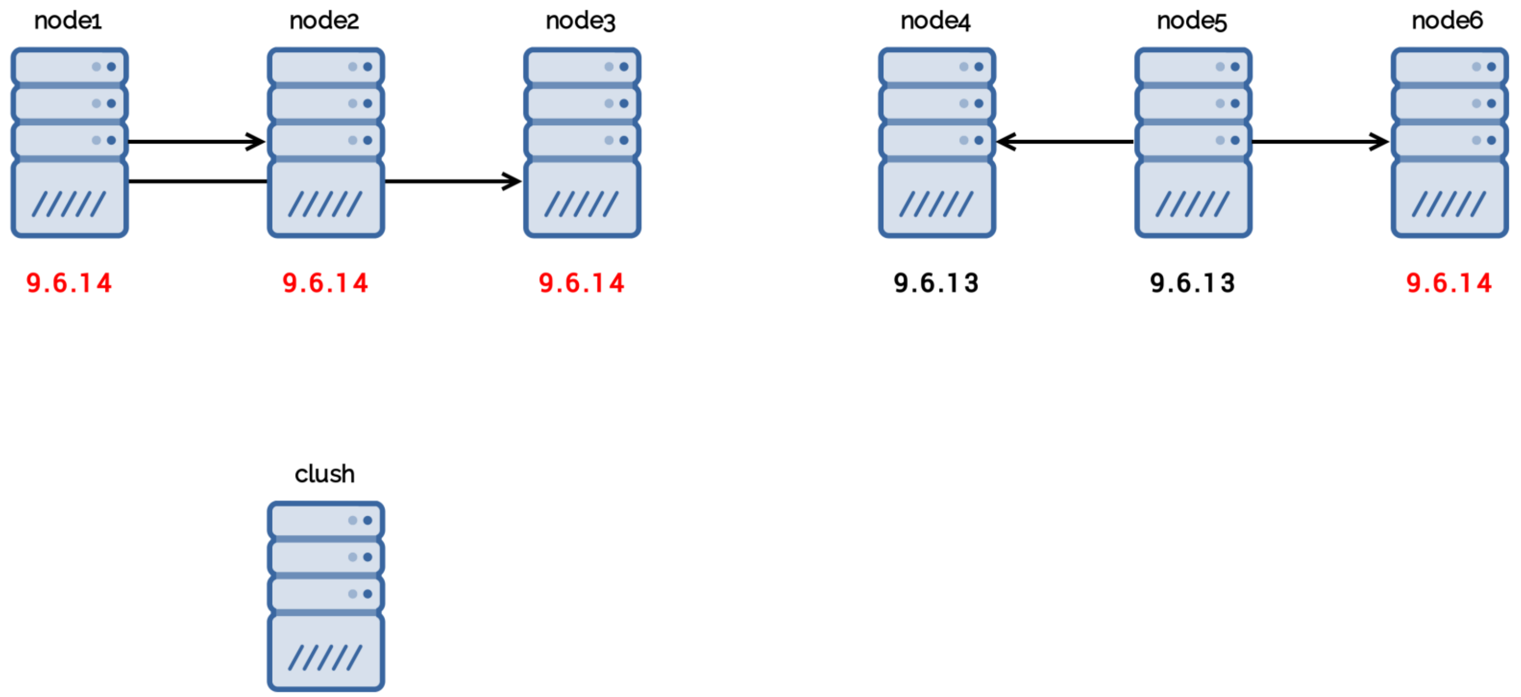
```
$ clush -f 1 -bw @node
```



Clustershell

- Then nodes one node at a time (fanout)

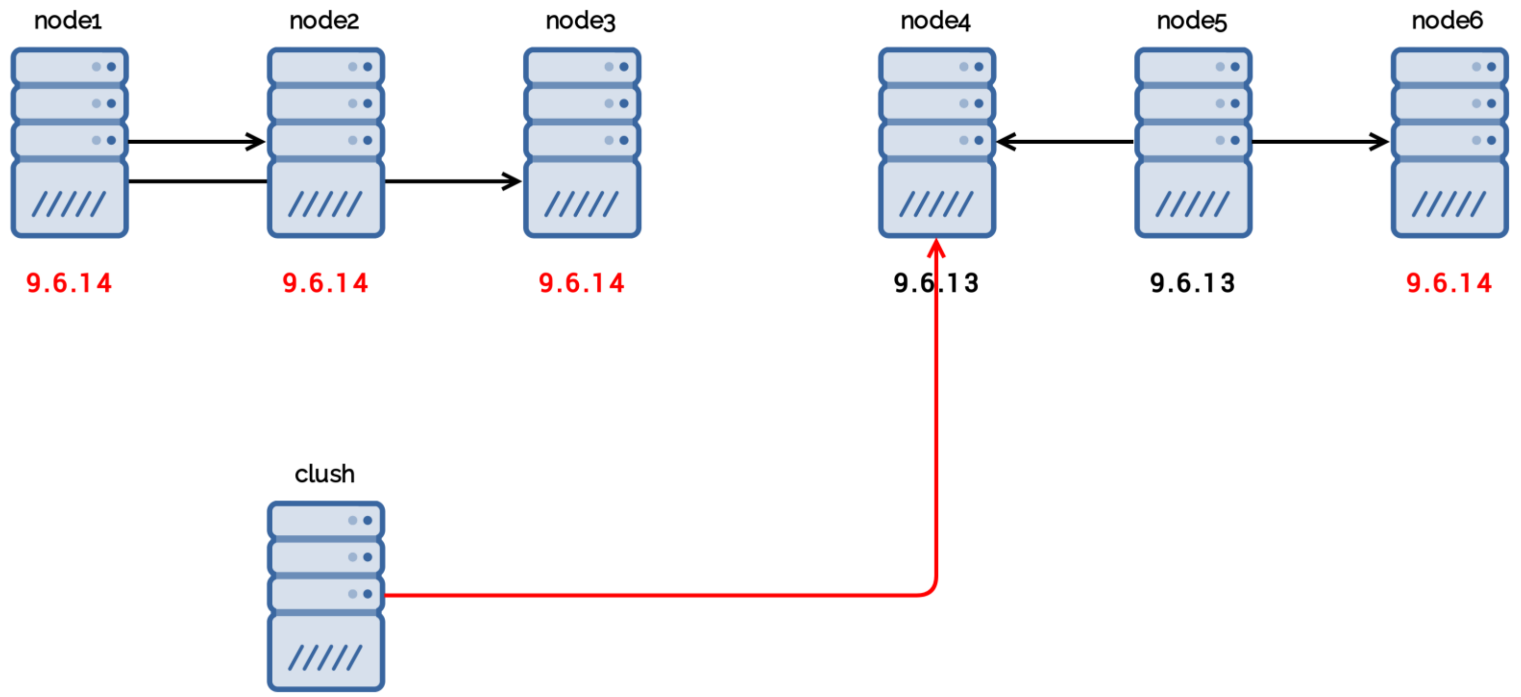
```
$ clush -f 1 -bw @node
```



Clustershell

- Then nodes one node at a time (fanout)

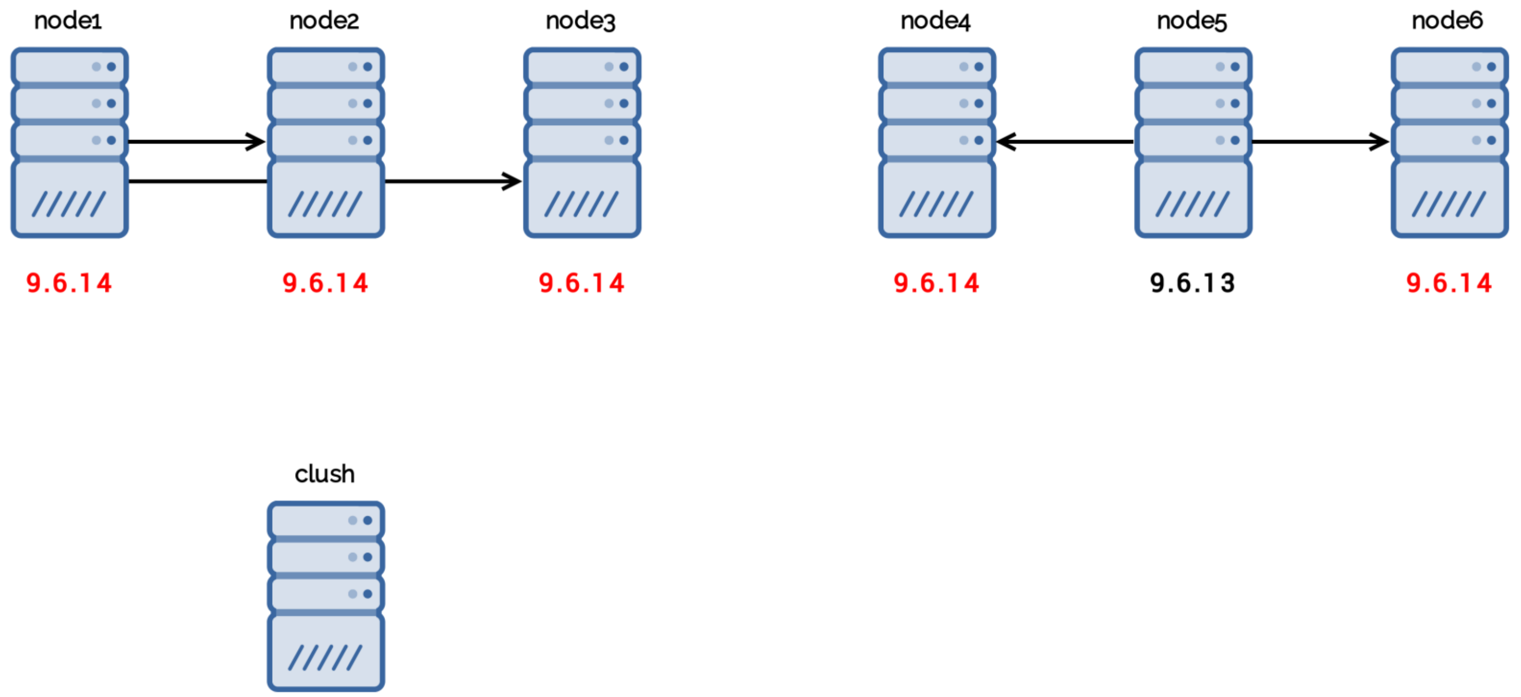
```
$ clush -f 1 -bw @node
```



Clustershell

- Then nodes one node at a time (fanout)

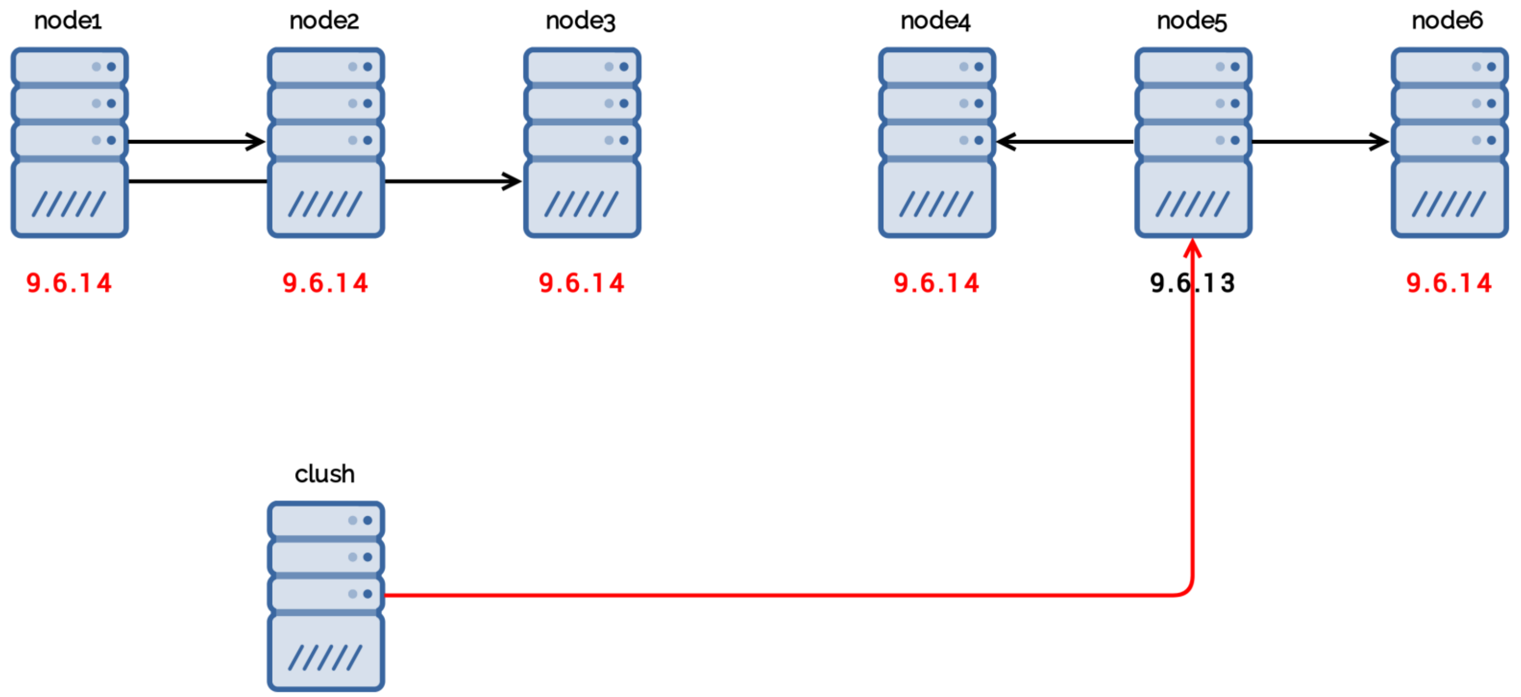
```
$ clush -f 1 -bw @node
```



Clustershell

- Then nodes one node at a time (fanout)

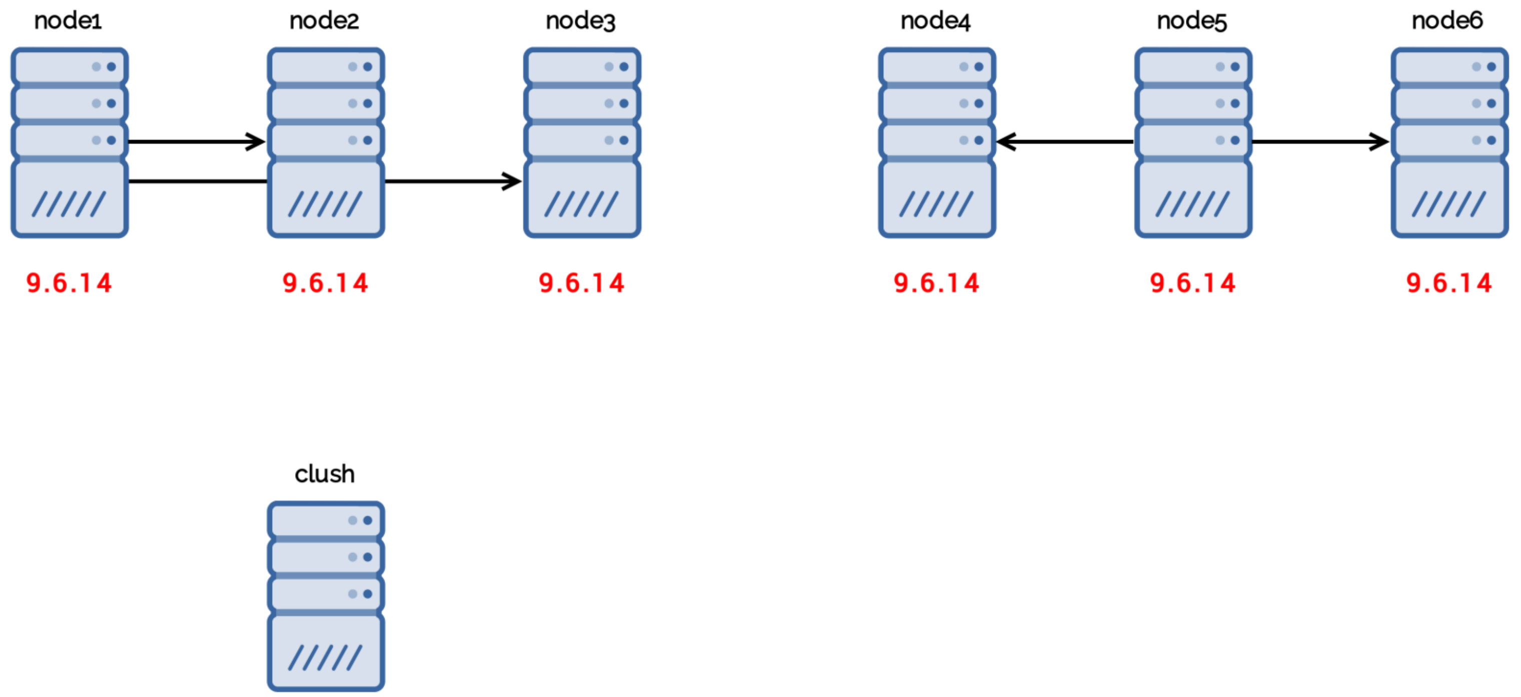
```
$ clush -f 1 -bw @node
```



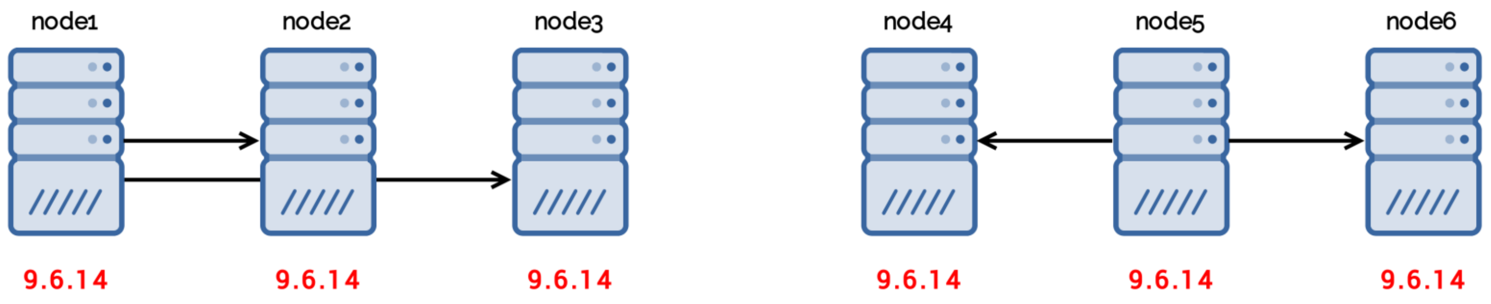
Clustershell

- Then nodes one node at a time (fanout)

```
$ clush -f 1 -bw @node
```



Final state


node1
node2
node3
node4
node5
node6
9.6.14
9.6.14
9.6.14
9.6.14
9.6.14
9.6.14

Limitations

- clush is great for one-shot human simple operations
- Requires development investment to implement complex automation
- At our scale, we use our own automation system
  - Mostly open: PostgreSQL, Flask, Ansible, Celery, …
  - And some internal systems

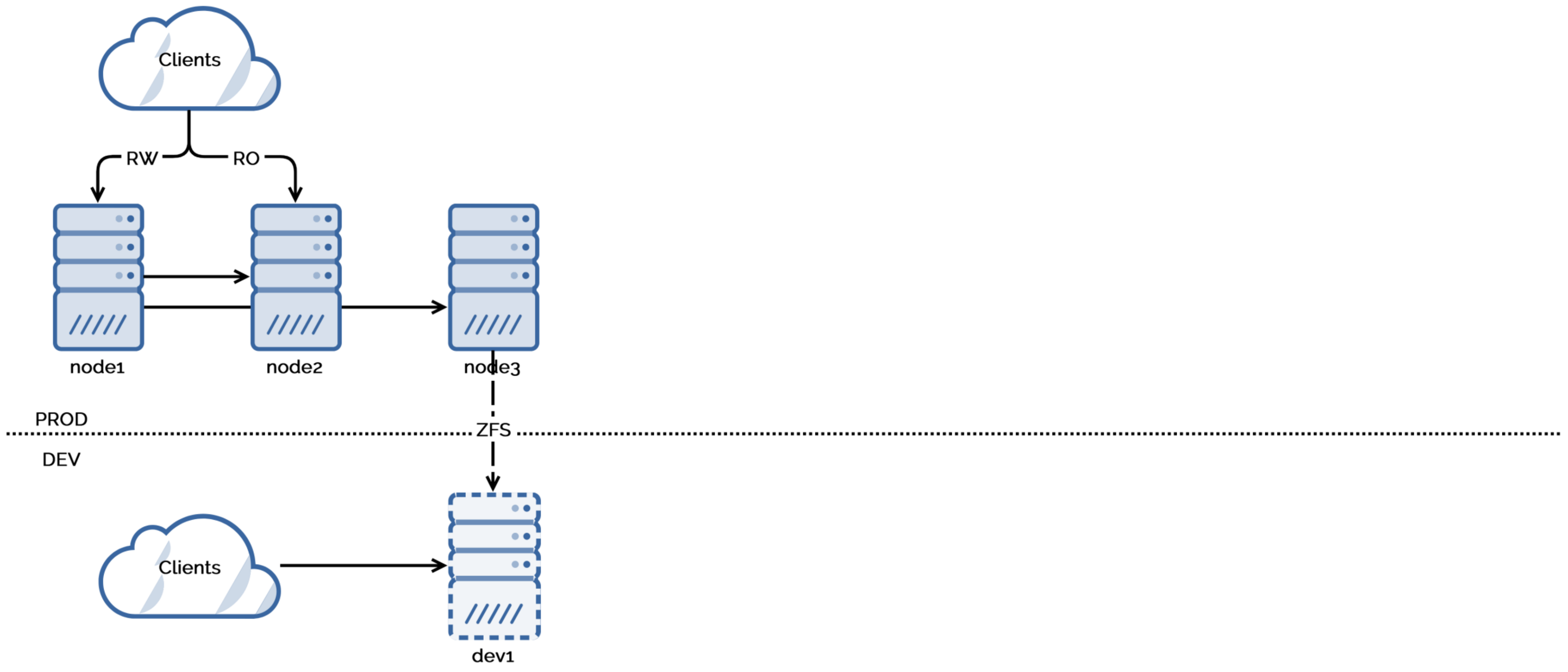
Upgrades

Why?

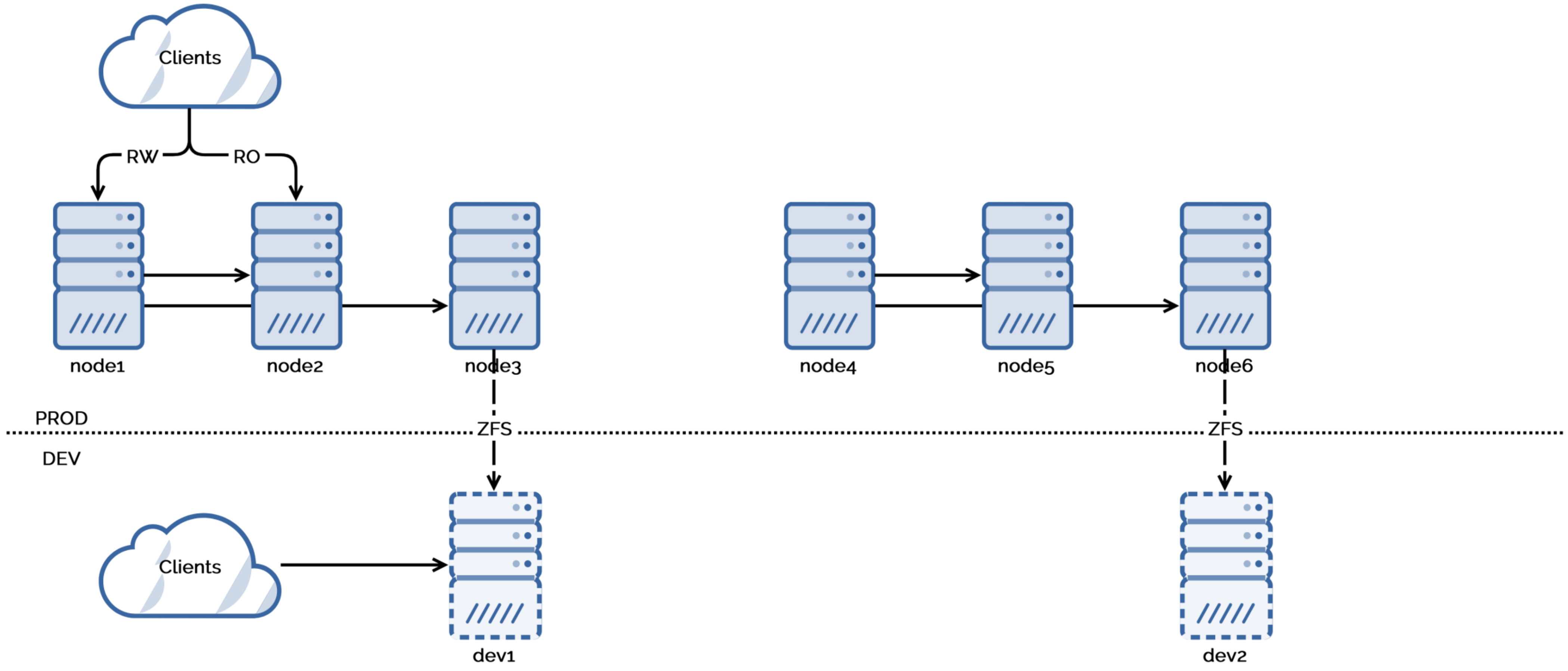
- Support
  - Limited to 5 years
- Better performance
  - Parallelism
  - Optimizations
- New features
  - Materialized views
  - JSON
  - Logical decoding
  - Upsert
  - SCRAM
  - And more…

https://knowyourmeme.com/memes/all-the-things

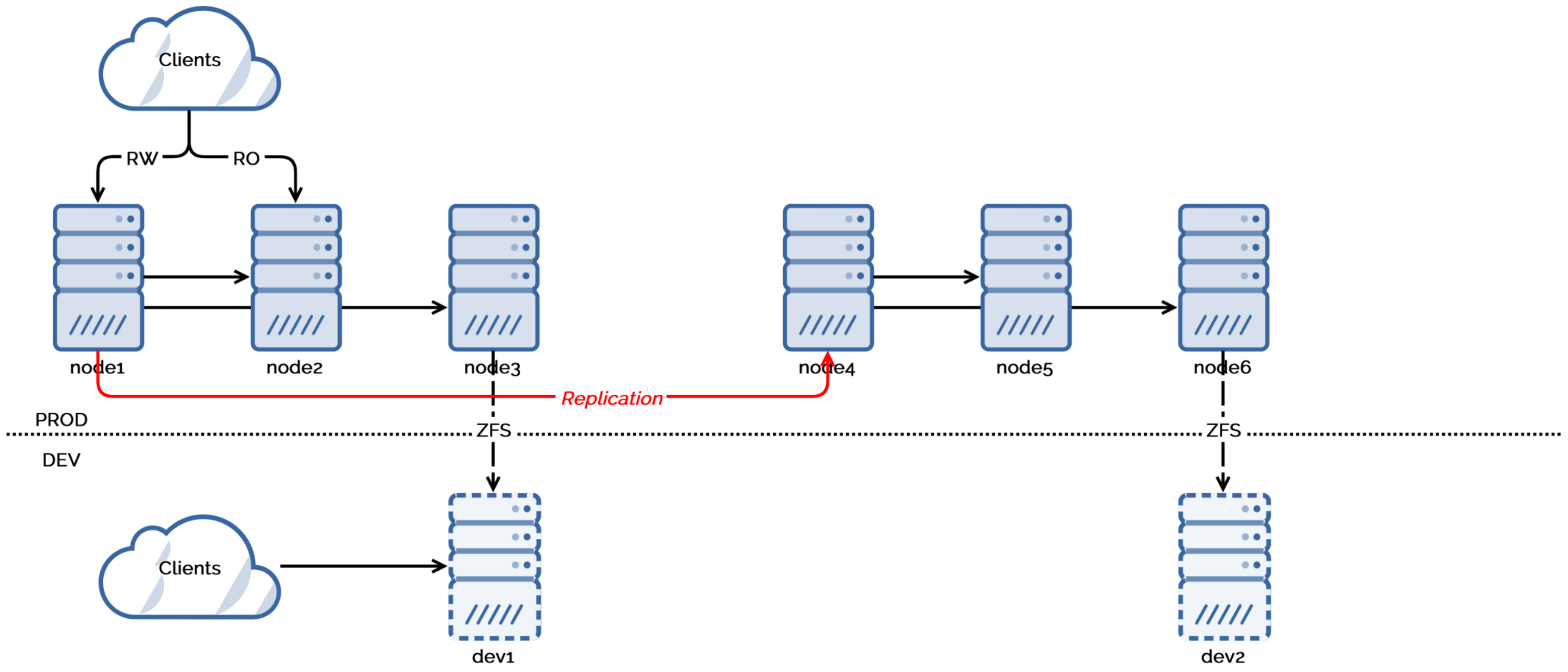
Method



Method

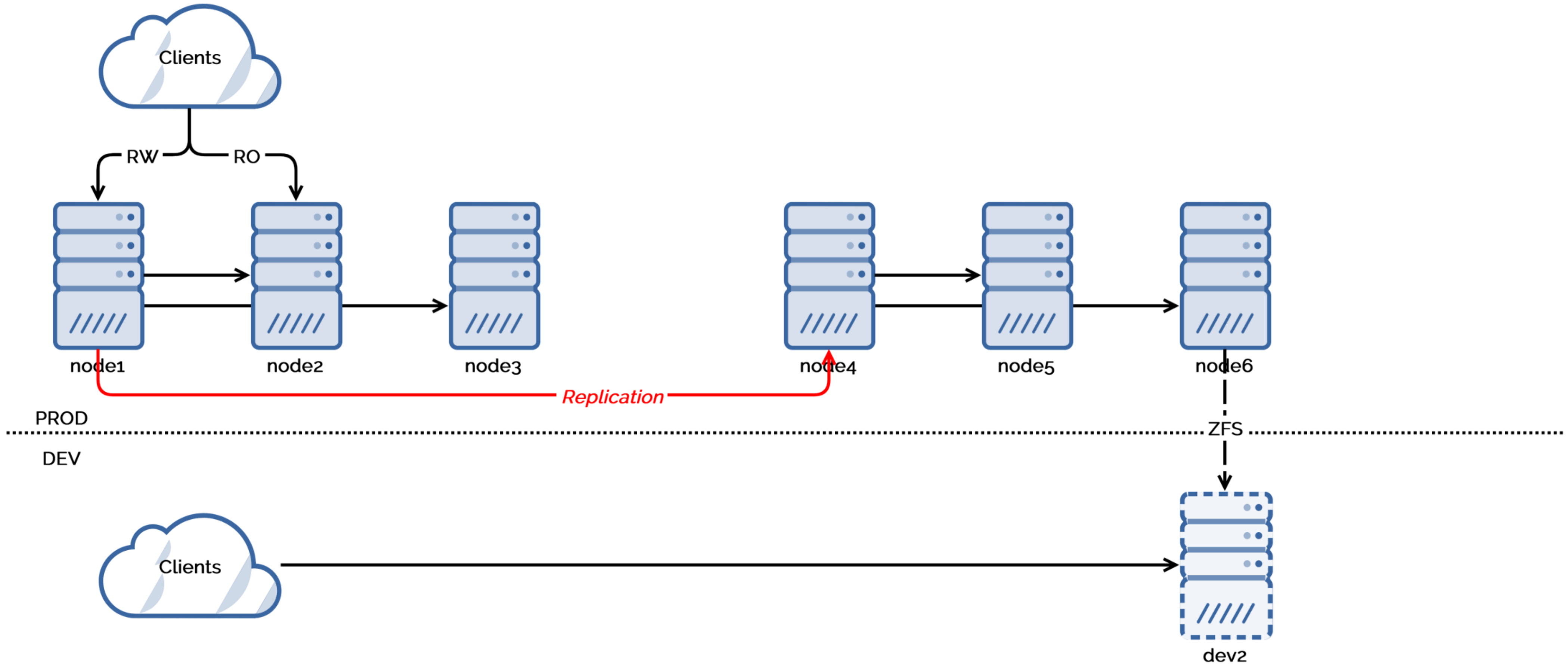

Clients
RW
RO
node1
node2
node3
node4
node5
node6
PROD
ZFS
ZFS
DEV
Clients
dev1
dev2

Method

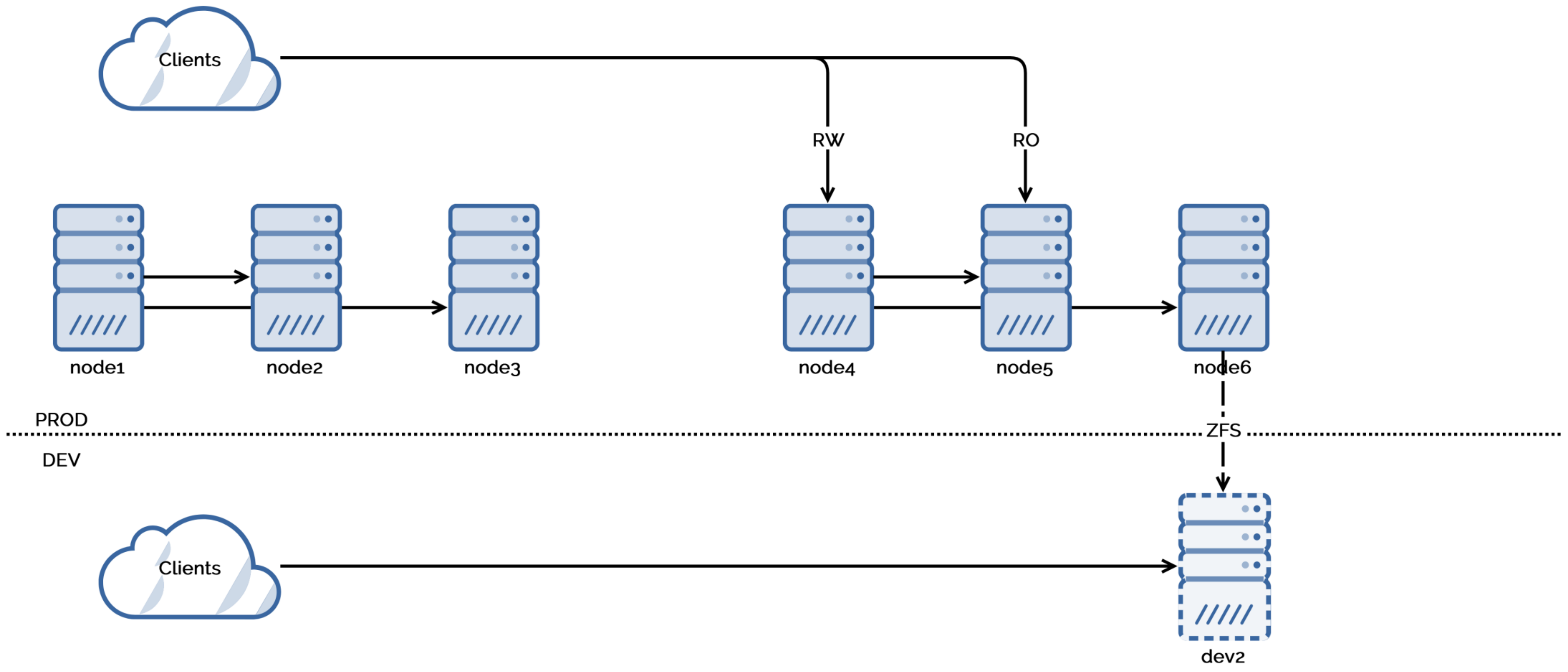


Clients
RW
RO
node1
node2
node3
node4
node5
node6
Replication
PROD
ZFS
ZFS
DEV
Clients
dev1
dev2

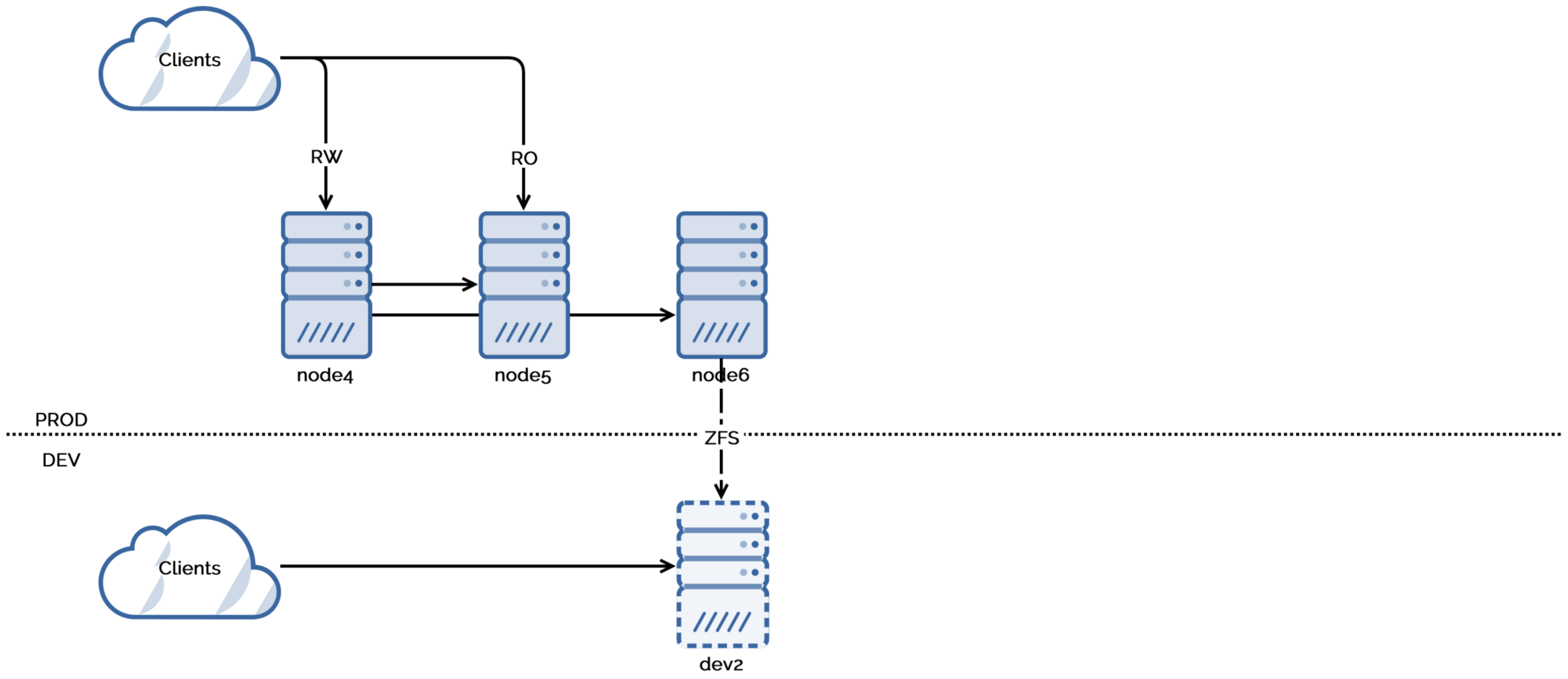
Method


Clients
RW
RO
node1
node2
node3
node4
node5
node6
Replication
PROD
ZFS
DEV
Clients
dev2

Method


Clients
RW
RO
node1
node2
node3
node4
node5
node6
PROD
ZFS
DEV
Clients
dev2

Method



“Replication” methods

- Application
- pg_dump / pg_restore
- pg_upgrade
- Logical replication with pglogical

“Replication” methods

Application

Application

1. Write objects to both clusters
2. Copy old objects to new cluster
3. Switchover

Application

Pros	Cons
Developers are autonomous	Different object management for too much teams
No downtime	Requires a single endpoint or inconsistencies
RDBMS independent	Not a developer priority

Application

- Conclusion

https://github.com/googlefonts/noto-emoji/blob/master/svg/emoji_u1f44e.svg

“Replication” methods

pg_dump / pg_restore

pg_dump / pg_restore

1. Set old cluster to read-only mode
2. Dump old cluster with pg_dump
3. Restore on new cluster with pg_restore
4. Switchover

pg_dump / pg_restore

Pros	Cons
DBA team is autonomous	Extended period of downtime for large databases
Easy to setup	
Wipe table and index bloat	

pg_dump / pg_restore

- Conclusion

https://github.com/googlefonts/noto-emoji/blob/master/svg/emoji_u1f44d.svg

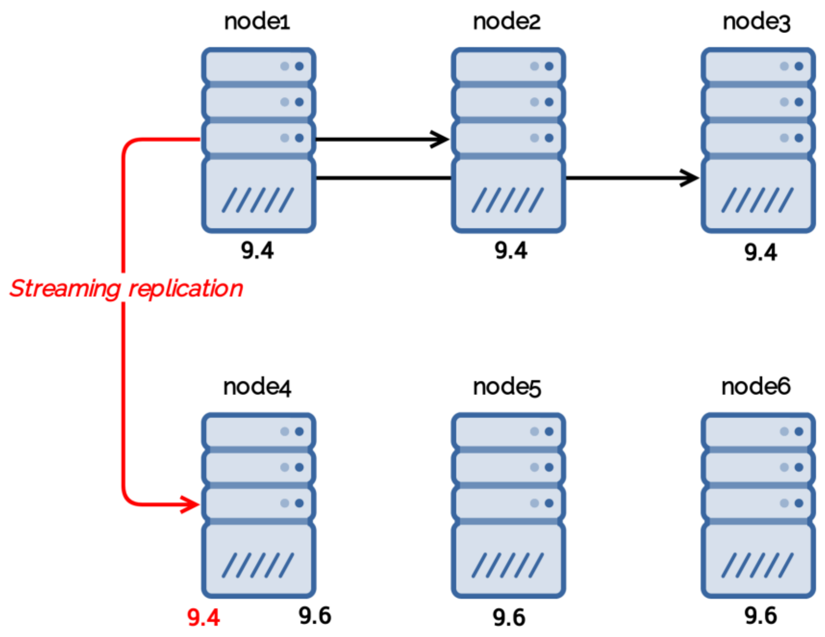
“Replication” methods

pg_upgrade

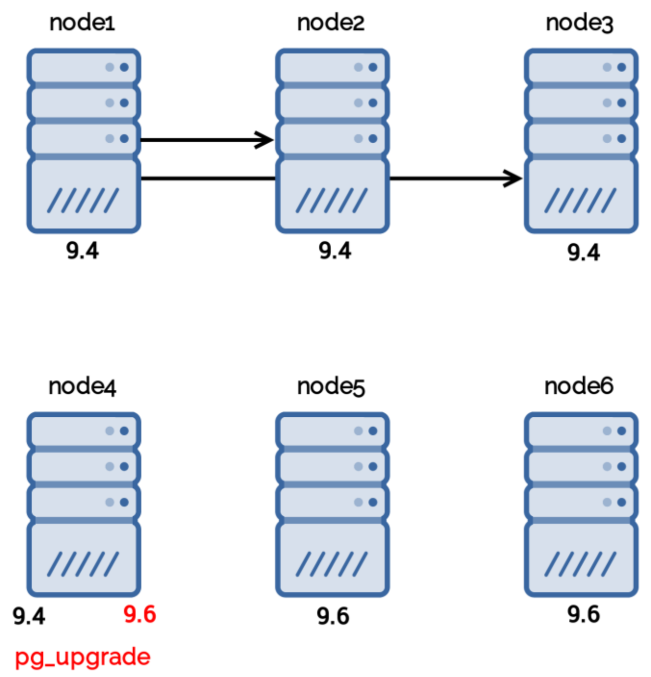
pg_upgrade

1. Install both versions on new cluster
2. Setup streaming replication from old cluster to new cluster
3. Set old cluster to read-only mode
4. Run pg_upgrade on new cluster with hardlinks
5. Update statistics in stage on new cluster
6. Switchover

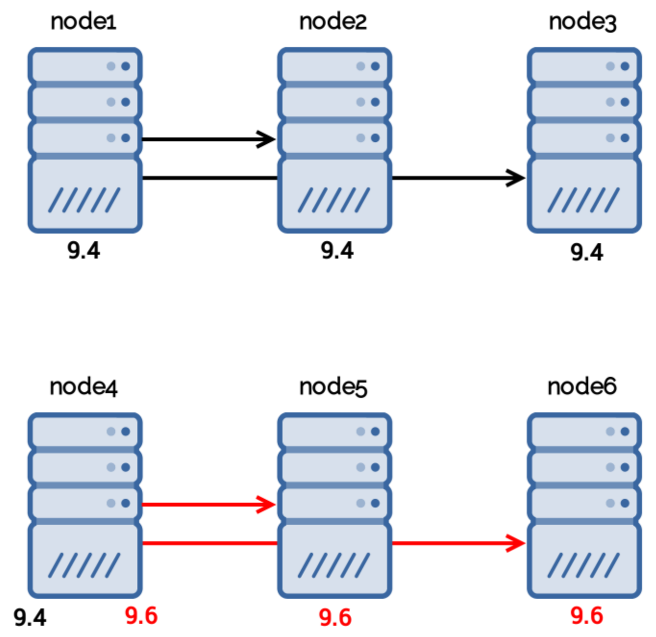
pg_upgrade


node1
node2
node3
9.4
9.4
9.4
Streaming replication
node4
node5
node6
9.4
9.6
9.6
9.6

pg_upgrade


node1
node2
node3
9.4
9.4
9.4
node4
node5
node6
9.4
9.6
pg_upgrade
9.6
9.6

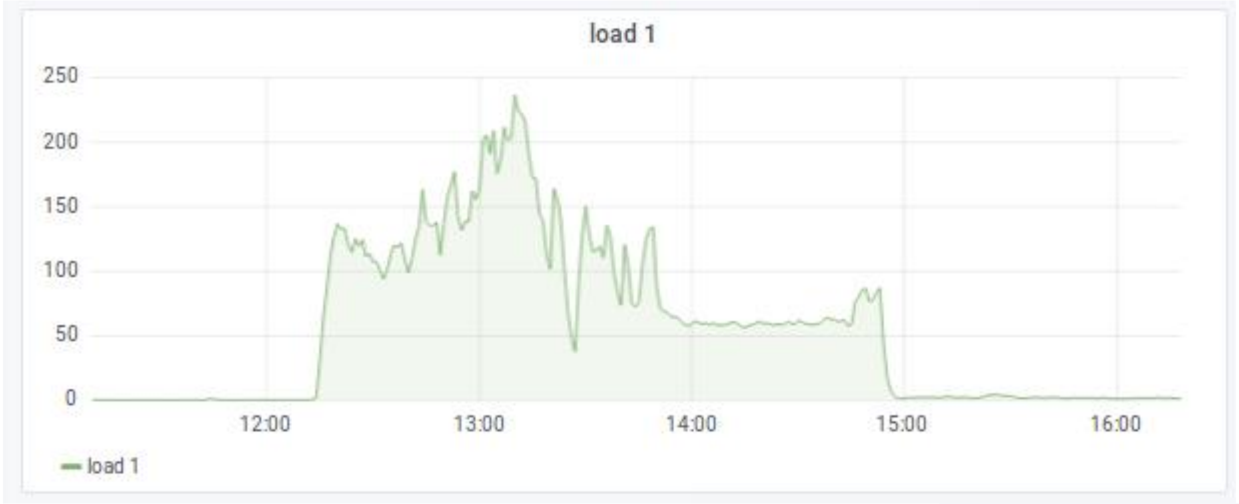
pg_upgrade

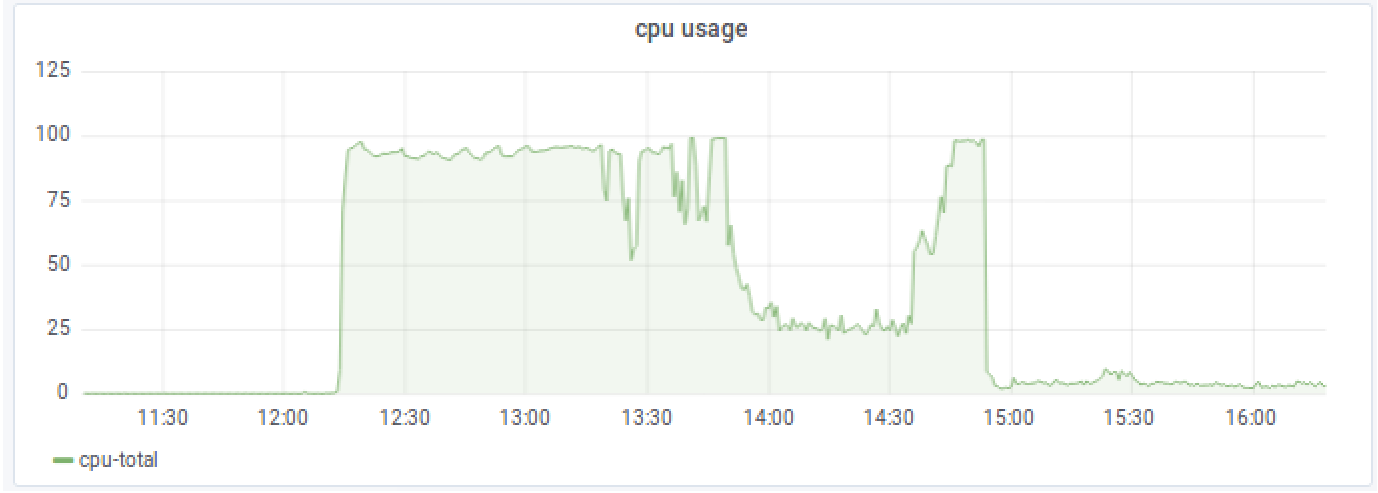


pg_upgrade

Pros	Cons
DBA team is autonomous	Requires multiple versions of binaries on the same host
Very short downtime	Rebuild streaming replication to have up-to-date data
Easy to setup (the first time)	

A word on statistics





A word on statistics

https://knowyourmeme.com/memes/reality-hits-you-hard-bro

A word on statistics

- vacuumdb to the rescue

```
$ vacuumdb --all --analyze-in-stages -j 10
```

pg_upgrade

- Conclusion

https://github.com/googlefonts/noto-emoji/blob/master/svg/emoji_u1f44d.svg

“Replication” methods

Logical replication with pglogical

Logical replication with pglogical

- Requires version 9.4+

- Logical replication
  - Doesn't replicate DDL
  - Doesn't replicate sequences

- pglogical additional functions
  - pglogical.replicate_ddl_command(command text, replication_sets text[])
  - pglogical.synchronize_sequence(relation regclass)

Logical replication with pglogical

1. Setup provider (install extensions, setup node, setup replication set)
2. Dump schema on provider
3. Restore schema on subscriber
4. Setup subscriber (install extensions, setup node, setup subscription)
5. Wait for subscriptions to be in sync
6. Set old cluster to read-only mode
7. Synchronize sequences
8. Switchover

Logical replication with pglogical

Pros	Cons
DBA team is autonomous	Complex setup
Very short downtime	Hard to debug (some logs are too generic)
Database precision	Objects in the database (secrets included)
	High level of locks required
	Encoding must be the same
	Provider can fail and take down production

Logical replication with pglogical

- Deadlocks

```
ERROR:  deadlock detected at character 237
DETAIL:  Process 16477 waits for AccessShareLock on relation 17241 of database 17032;
blocked by process 17333.
         Process 17333 waits for AccessExclusiveLock on relation 4920800 of database
17032; blocked by process 16477.
         Process 16477: <application query>
         Process 17333: SELECT pglogical.replication_set_add_all_tables('default',
ARRAY['public']);
HINT:  See server log for query details.
STATEMENT:  <application query>
```

Logical replication with pglogical

- Sequences

```
ERROR:  duplicate key value violates unique constraint "table_pkey"
```

Logical replication with pglogical

- Encoding must be the same

```
ERROR:  encoding conversion for binary datum not supported yet
DETAIL:  expected_encoding UTF8 must be unset or match server_encoding SQL_ASCII
CONTEXT:  slot "pgl_<slotname>", output plugin "pglogical_output", in the startup
callback
LOG:  could not receive data from client: Connection reset by peer
```

- Crystal clear in the documentation

> ***4.13 Database encoding differences***
> *PGLogical does not support replication between databases with different encoding. We recommend using UTF-8 encoding in all replicated databases.*

https://www.2ndquadrant.com/fr/ressources/pglogical/documentation/

Logical replication with pglogical

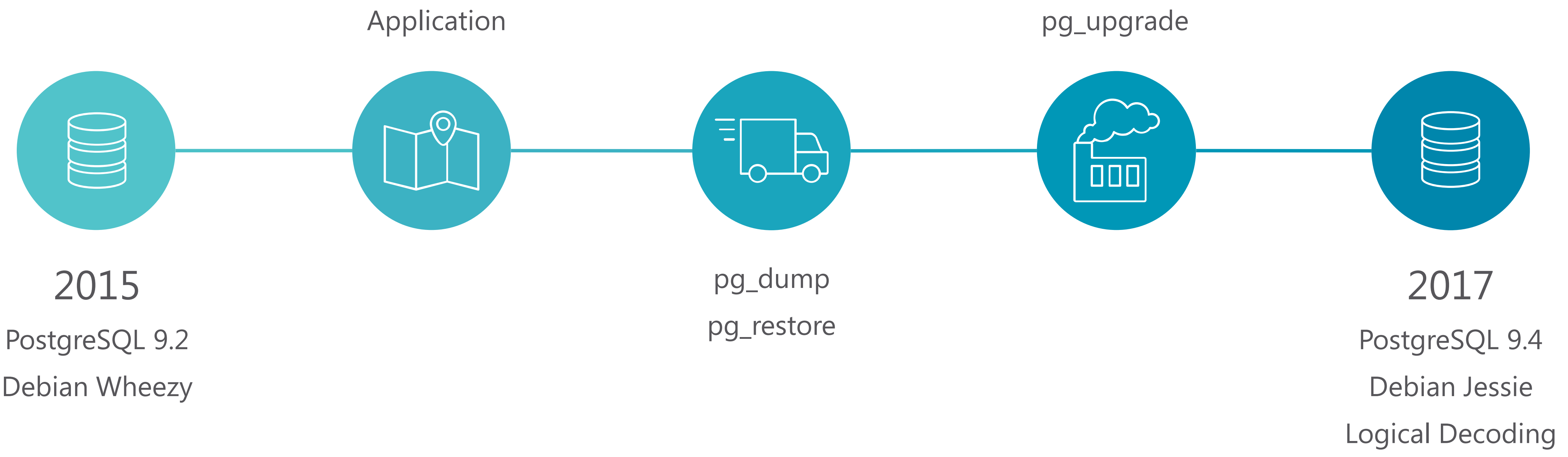
- Avoid explicit locks
- Use UTF-8 encoding
- Use latest pglogical commercial version and support open source
- Or fallback to another solution
  - built-in logical replication
  - pg_upgrade
  - pg_dump and pg_restore

Logical replication with pglogical

- Conclusion

https://github.com/googlefonts/noto-emoji/blob/master/svg/emoji_u1f44d.svg

Timeline


2015
PostgreSQL 9.2
Debian Wheezy
Application
pg_dump
pg_restore
pg_upgrade
2017
PostgreSQL 9.4
Debian Jessie
Logical Decoding

Timeline

pglogical

pg_upgrade

2017

PostgreSQL 9.4

Debian Jessie

Standalone

pg_dump

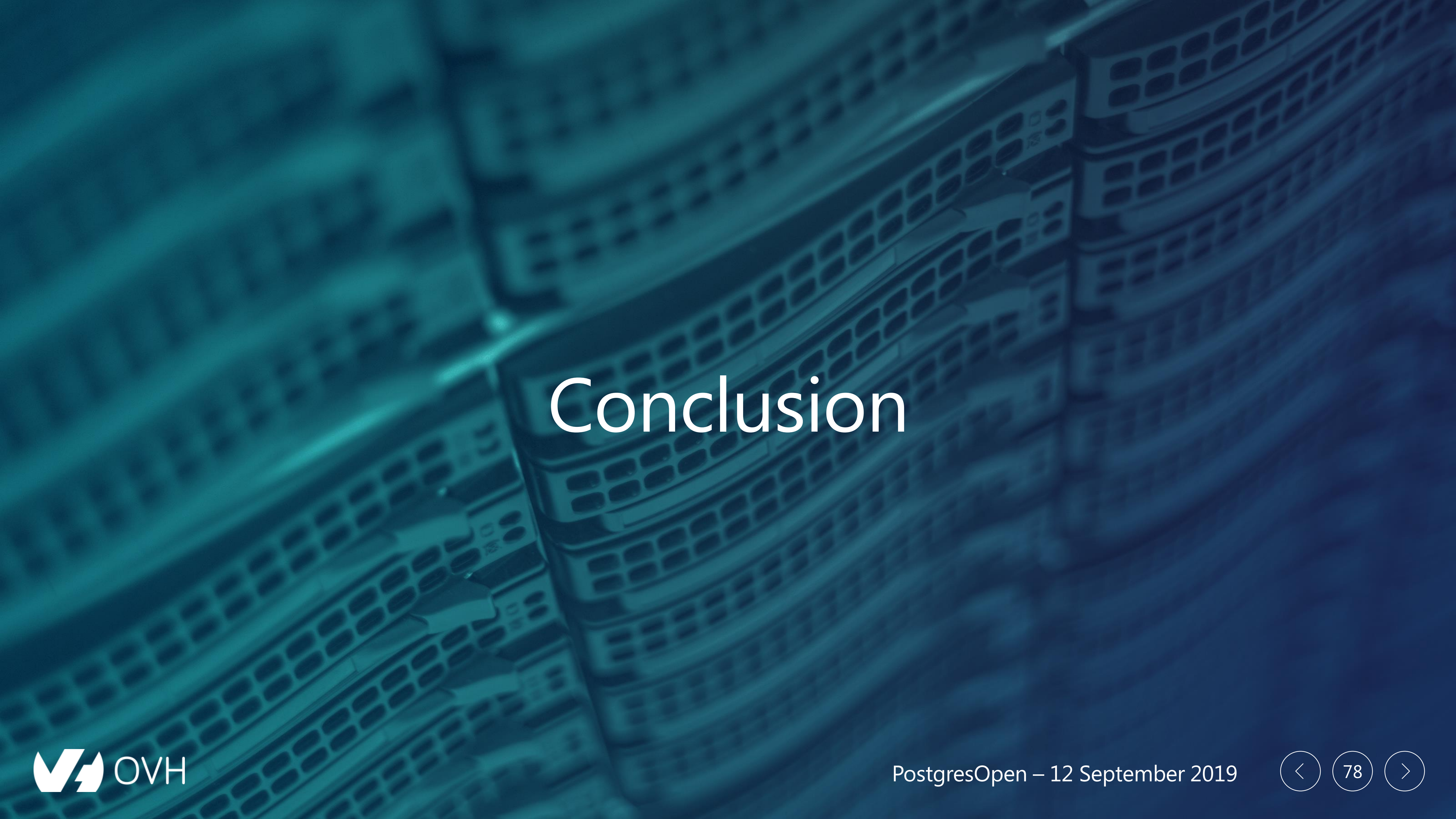
pg_restore

2019

PostgreSQL 9.6

Debian Stretch

Patroni



Conclusion

Conclusion

- New features
- Better stability
- Better performance
- Always secure

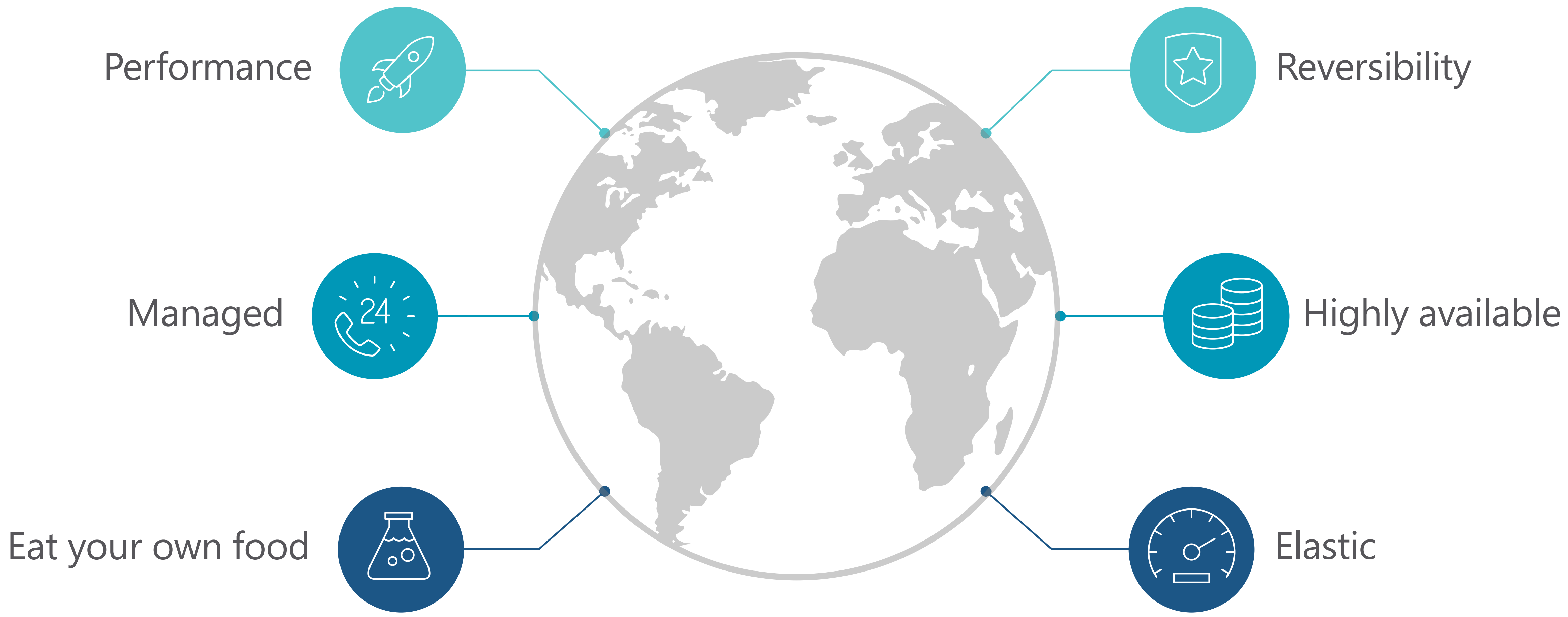
What’s next?

Next

- Upgrade to PostgreSQL 12
- Upgrade to Debian 10
- Migrate from MySQL to PostgreSQL
- Automate, automate, automate!

Extra

Enterprise Cloud Databases



https://labs.ovh.com/ha-database

We are hiring!

- Opensource Database Engineers
- Site Reliability Engineers (Private Cloud, Openstack, DNS, Deploy, Observability)
- Software Engineers (containers, baremetal, web hosting)
- Backend Developers (Python, Go)
- And more

Questions