PostgresPro

Backup and Recovery

PostgreSQL backup and recovery methods

ABOUT ME

Pavel Konotopov

- More than 20+ years in IT;
- Last 5 years working with PostgreSQL;
- Database engineer specialized in PostgreSQL high availability;
- Experience administering 300+ PostgreSQL clusters in a productive environment;
- Last year working in Postgres Professional.

LinkedIn: https://www.linkedin.com/in/pavel-konotopov-262028119
Email: p.konotopov@postgrespro.ru

PostgresPro

TODAY'S AGENDA

- Why we need backups?
- What do we mean by “database backup”?
- What is good for PostgreSQL?
- Overview of PostgreSQL-specific backup tools.
- Advanced backup techniques.
- Backup techniques in PostgreSQL HA clusters.

PostgresPro

WHY DO WE NEED BACKUPS?

- Database restore after disaster (obvious case):
  - Unexpected power outage;
  - Sudden advent of Out of Memory killer;
  - Data corruption;
  - Random cloud instance death;
  - Malicious misrepresentation or deletion of data;
  - And … whatever you cannot imagine ☺

- Fast new replicas creating in Highly Available PostgreSQL installations;
- Creating Sandbox/Dev/Stage/QA/Preprod/UAT environments;
- Point in time database recovery;
- Data archive for future analysis;
- Security standards requirements - HIPAA, PCIDSS, etc;
- Potential response to future challanges.

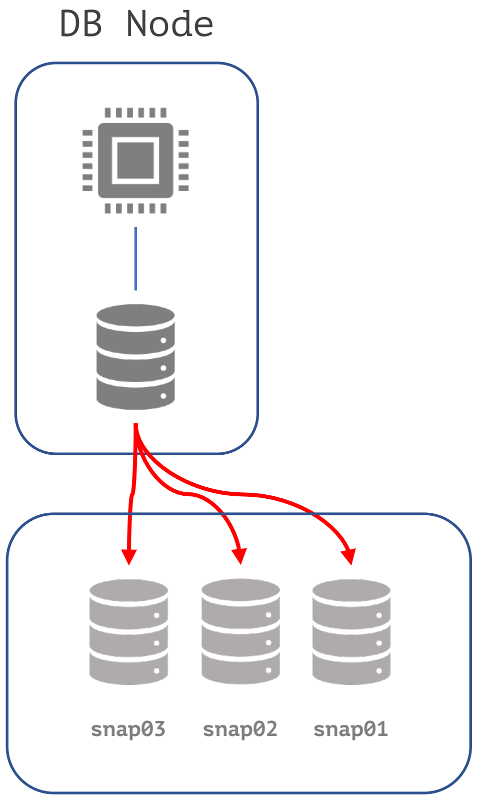
WHAT ARE BACKUP TOOLS?

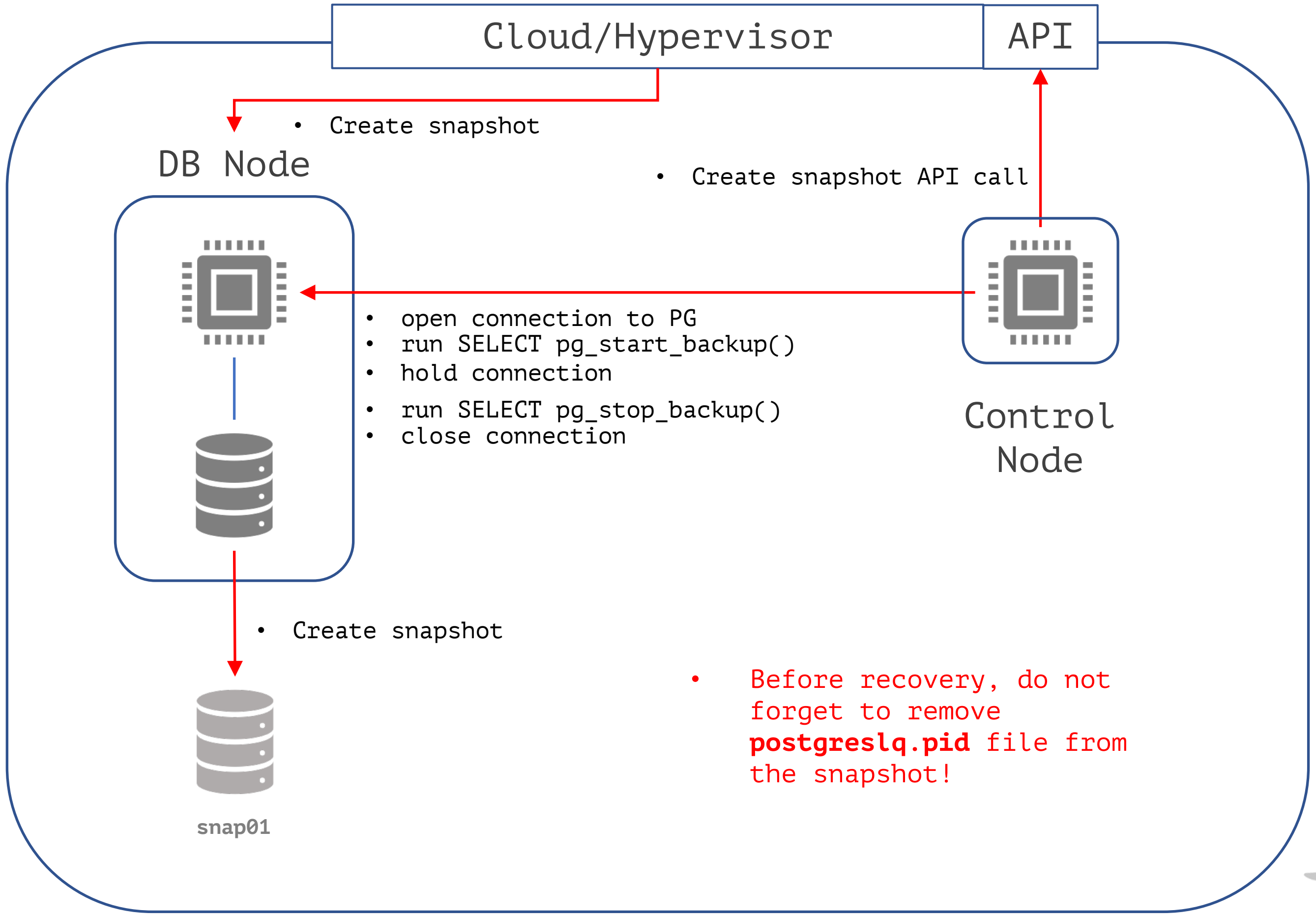
- Non-Database backup tools:
  - Storage or Instance snapshots;
  - Writing your own custom backup scripts;
- Database-related tools:
  - Replicas;
  - Database logical dump
    - pg_dump and pg_restore utils
  - Database related physical backup tools:
    - Enterprise-level backup systems (NetApp, EMC, Microfocus, etc)
    - OpenSource backup tools:
      - pg_basebackup;
      - pgBackRest;
      - WAL-G/WAL-E;
      - pg_probackup;
      - …

But this is not a backup in database case.

NON-DATABASE BACKUP TOOLS

Storage or Instance snapshots



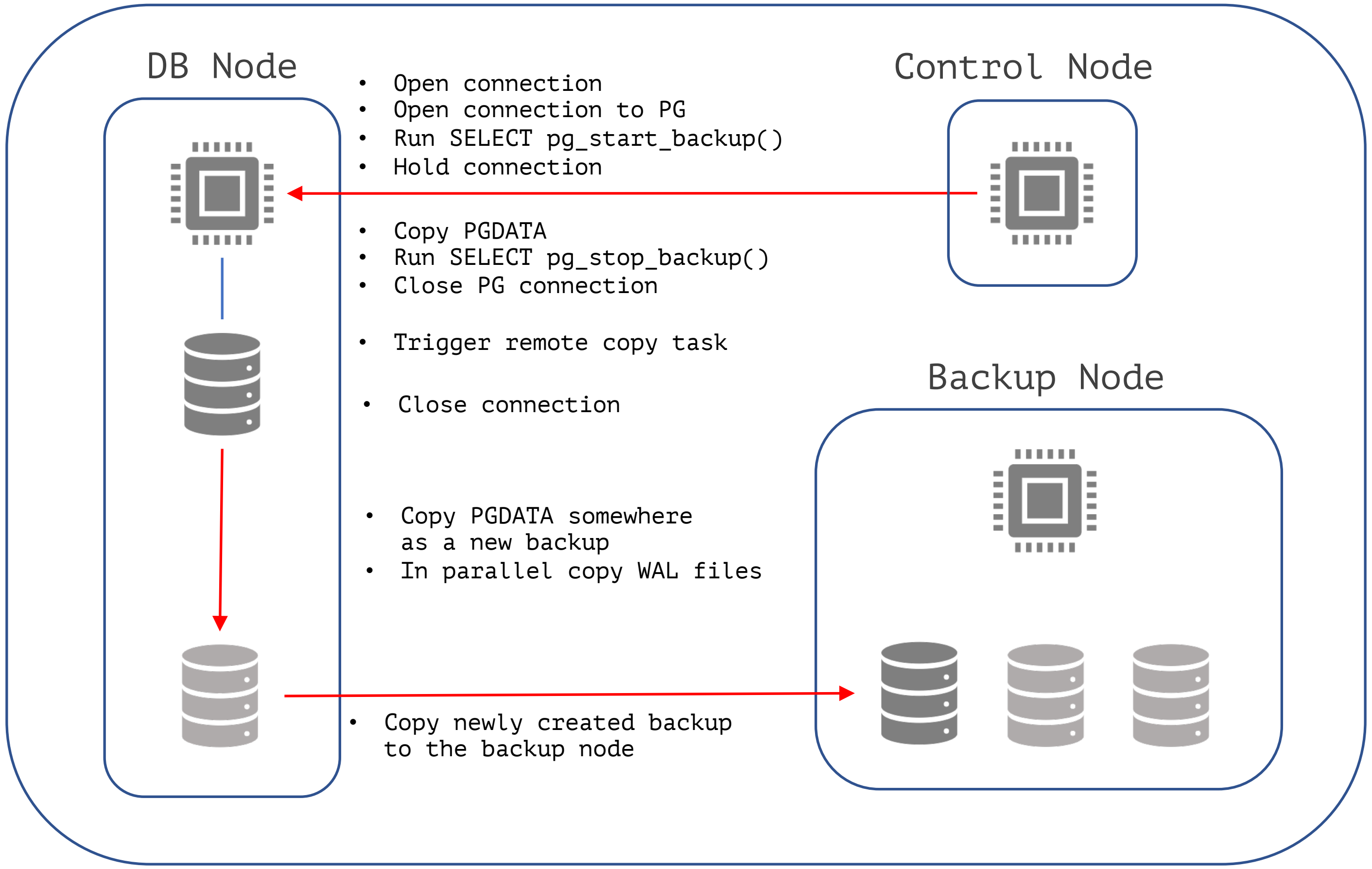


- open connection to PG
- run SELECT pg_start_backup()
- hold connection
- run SELECT pg_stop_backup()
- close connection

- Before recovery, do not forget to remove **postgreslq.pid** file from the snapshot!

NON-DATABASE BACKUP TOOLS

Custom backup scripts



- Open connection
- Open connection to PG
- Run SELECT pg_start_backup()
- Hold connection

- Copy PGDATA
- Run SELECT pg_stop_backup()
- Close PG connection

- Trigger remote copy task

- Close connection

- Copy PGDATA somewhere as a new backup
- In parallel copy WAL files

- Copy newly created backup to the backup node

- The directories need to be copied and traversed in a certain order;
- You must create a backup_label file.

NON-DATABASE BACKUP TOOLS

What's wrong here?

- ➢ Too complicated, many potential points of failure;
- ➢ In both cases we should maintain snapshots or backups ourselves;
- ➢ Taking backup could be too long, we want it to be faster!
- ➢ No well-known implementation of these approaches, in both cases we should make our own scripts;
- ➢ Big database changes - large snapshots, large size backups;
- ➢ No incremental and differential backups are possible;
- ➢ No Point In Time Recover (PITR);
- ➢ We need advanced backup tools!

DATABASE-RELATED BACKUP TOOLS

Database Logical Dump

- pg_dump and pg_restore are main utilities for this;
- Makes a dump as SQL code;
- The dump will be for one particular point in time (PostgreSQL doing snapshot when dump has began).

BUUUT …

- Recovery takes very long time if the Database is large:
  - Data loading;
  - Indexes creation;
  - No statistics!
  - No streaming backups, no point in time recovery possible!

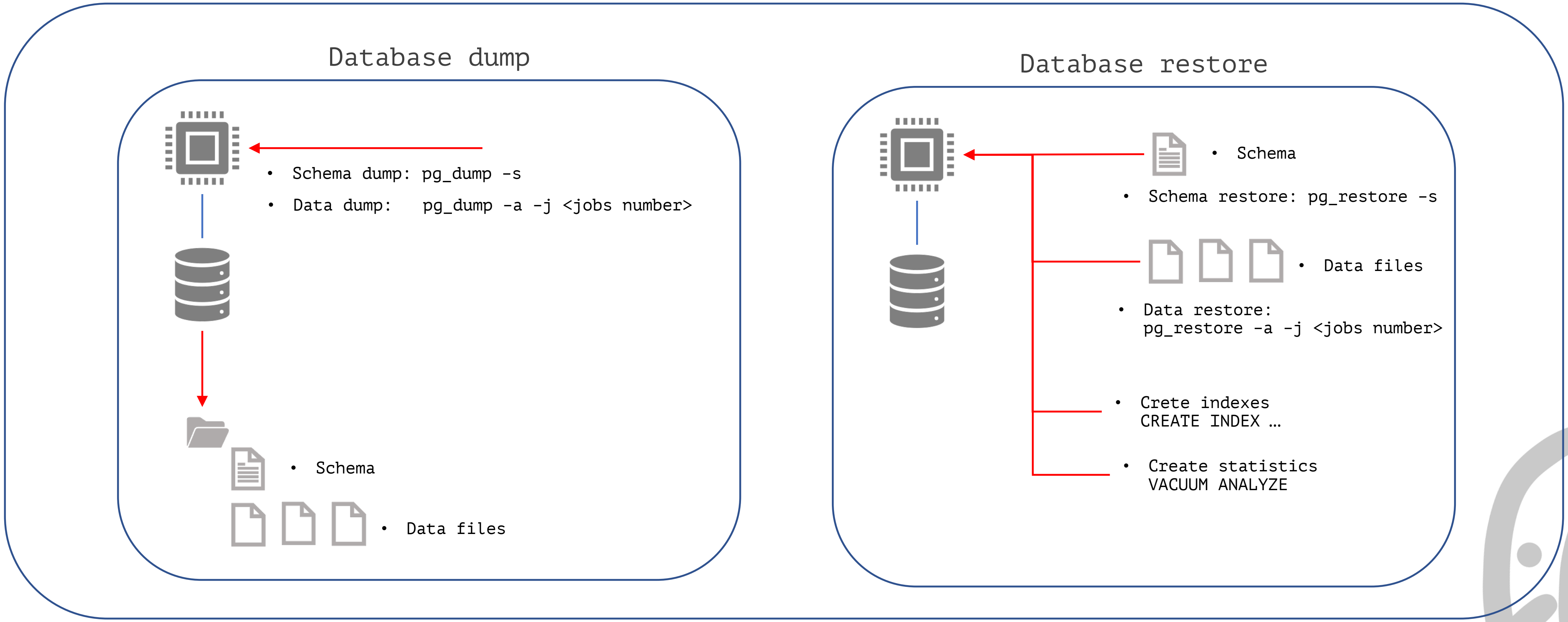
DATABASE-RELATED BACKUP TOOLS

Database Logical Dump Optimizations

- **DUMP**
  - ✓ First it dumps only schema, then data;
  - ✓ pg_dump/pg_restore can parallelize for speedy data dump/restore;
  - ✓ Use COPY command to save/load data into/from separate files/tables;
- **RESTORE**
  - ✓ Load schema and data separately;
    - ✓ Parallel data uploading
    - ✓ Background indexes creation (CONCURRENTLY)
  - ✓ But still no statistics!
    - ✓ Need to run VACUUM ANALYSE.

DATABASE-RELATED BACKUP TOOLS

Database Logical Dump/Restore procedure



Database dump

- Schema dump: pg_dump -s
- Data dump: pg_dump -a -j <jobs number>
- Schema
- Data files

Database restore

- Schema
- Schema restore: pg_restore -s
- Data files
- Data restore: pg_restore -a -j <jobs number>
- Crete indexes CREATE INDEX …
- Create statistics VACUUM ANALYZE

DATABASE-RELATED BACKUP TOOLS

When Logical Dump approach is useful?

- When you are migrating to the major PG version;
- For some reason pg_upgrade is not possible;
- You don't want to drag "garbage" in binary files to the new version;
- You can afford to stop the service for a while;
- The size of the database is relatively small (< 1Tb).

- Useful to validate a newly restored database or test backups (let’s remember this)!

DATABASE-RELATED BACKUP TOOLS

Backup tools for physical backups

- pg_basebackup - is "out of the box" tool;
- Every PostgreSQL installation has it;
- Can take backup locally and remotely by postgres protocol;

BUUUT...

- What if the database is large (>1Tb)?
- What if we have very small maintenance window for database restore?
- What if we have a limited backup storage size?
- What if we are not sure if the backups are valid?
- What if we want to restore DB state to the point in time?
- More "what if"!!!
- We need more advanced backup tool!

- It's cool to have backups, but not cool not to be able to recover in a reasonable amount of time!

DATABASE-RELATED BACKUP TOOLS

Good tools for physical backups

- **Usable**
  - Well documented
  - Automation possibilities
- **Scalable**
  - Parallel execution is possible
  - Implemented compression methods
  - Incremental and differential backups
- **Reliable**
  - WAL archiving
  - Streaming backups
  - Rotation and expiration policies
  - Encryption

DATABASE-RELATED BACKUP TOOLS

Good tools for physical backups

- WAL-G
- WAL-E
- pgBackRest
- pg_probackup
- Barman
- …

DATABASE-RELATED BACKUP TOOLS

Good tools for physical backups

- **WAL-G (Yandex, community)**
  - https://github.com/wal-g/wal-g
  - Docs - https://wal-g.readthedocs.io
  - Apache License, Version 2.0, lzo lib is licensed under GPL 3.0+.
- **WAL-E (community)**
  - https://github.com/wal-e/wal-e
  - Docs - https://github.com/wal-e/wal-e
  - BSD 3-Clause license
- **pgBackRest (Crunchy Data, community)**
  - https://github.com/pgbackrest/pgbackrest
  - Docs - https://pgbackrest.org/user-guide.html
  - MIT license
- **pg_probackup (PostgresPro, community)**
  - https://github.com/postgrespro/pg_probackup
  - Docs - https://postgrespro.com/docs/postgrespro/13/app-pgprobackup
  - PostgreSQL license
- **Barman (EDB, community, requires pg_basebackup)**
  - https://github.com/EnterpriseDB/barman
  - Docs - https://pgbarman.org/documentation/
  - GNU General Public License 3.0

DATABASE-RELATED BACKUP TOOLS

Common features: backup repository

- **WAL-G/WAL-E**
  - Your responsibility (DIY)
- **pgBackRest**
  - --stanza option
  - Common repository for many instances
- **pg_probackup**
  - --instance option
  - Common repository for many instances
- **Barman**
  - <server_name> option
  - Common repository for many instances

DATABASE-RELATED BACKUP TOOLS

Common features: logging

- **WAL-G/WAL-E**
  - WAL-E: WALE_LOG_DESTINATION - syslog, stderr; WALE_SYSLOG_FACILITY - local0-7,user
  - WAL-G: STDOUT/STDERR 2>&1 > logfile
- **pgBackRest**
  - log-level-console, log-level-file, log-level-stderr, log-path
  - OFF, ERROR, WARN, INFO, DETAIL, DEBUG, TRACE
- **pg_probackup**
  - log-level-file, log-filename, log-rotation-size, log-rotation-age
  - VERBOSE, LOG, INFO, NOTICE, WARNING, ERROR, OFF
- **Barman**
  - Global logfile
  - DEBUG, INFO, WARNING, ERROR, CRITICAL

DATABASE-RELATED BACKUP TOOLS

Common features: WAL archiving/restoring

- **WAL-G/WAL-E**
  - archive_command = “wal-g/wal-e wal-push …”
  - restore_command = “wal-g/wal-e wal-fetch …”
- **pgBackRest**
  - archive_command = “pgbackrest archive-push…”
  - restore_command = “pgbackrest archive-get..”
  - Can work in asynchronous mode!
- **pg_probackup**
  - archive_command = “pg_probackup archive-push…”
  - restore_command = “pg_probackup archive-get…”
- **Barman**
  - archive_command = “rsync …”
  - restore_command = ”barman get-wal…”

What is WAL?
Write Ahead Log - the files where all the changes occurring in the DBMS are recorded before their will be applied into DB, to ensure the possibility of restoring. Having WAL, we can replay it from the beginning (usually since the last backup) to a certain point, thereby restoring the state of the DBMS for a certain period of time.

Why do we need WAL archiving?
Ensure that the DBMS can be restored to a point in time - Point In Time Recovery (PITR)

PostgresPro

DATABASE-RELATED BACKUP TOOLS

Retention policies

- **WAL-G/WAL-E**
  - delete
  - retain N – number of backups in place
  - Before <wal-segment>
- **pgBackRest**
  - Full & Differential Backup Retention - number of backups to retain
  - Archive Retention
  - Defined in configuration file
- **pg_probackup**
  - --retention-redundancy
  - --retention-window
  - delete --expired --wal
- **Barman**
  - retention_policy = {REDUNDANCY value RECOVERY WINDOW OF value {DAYS | WEEKS | MONTHS}}

PostgresPro

DATABASE-RELATED BACKUP TOOLS

Remote backups, Object storages support

- **WAL-G/WAL-E**
  - stream
- **pgBackRest**
  - ssh,
  - stream, but with ssh :)
- **pg_probackup**
  - ssh
  - stream
- **Barman**
  - ssh
  - stream

- **WAL-G/WAL-E**
  - AWS, S3 compat, GS, Azure, Swift
- **pgBackRest**
  - AWS, S3 compat, GS, Azure
- **pg_probackup**
  - Not Yet Implemented
- **Barman**
  - AWS S3, Azure
  - barman-cloud-backup script
  - barman-wal-archive script

DATABASE-RELATED BACKUP TOOLS

Parallel backup/restore

- **WAL-G/E**
  - WALG_UPLOAD_CONCURRENCY, WALG_DOWNLOAD_CONCURRENCY
  - WALE_UPLOAD_CONCURRENCY, WALE_DOWNLOAD_CONCURRENCY
- **pgBackRest**
  - --process-max
- **pg_probackup**
  - -j num_threads
- **Barman**
  - parallel_jobs = n (rsync-mode only)

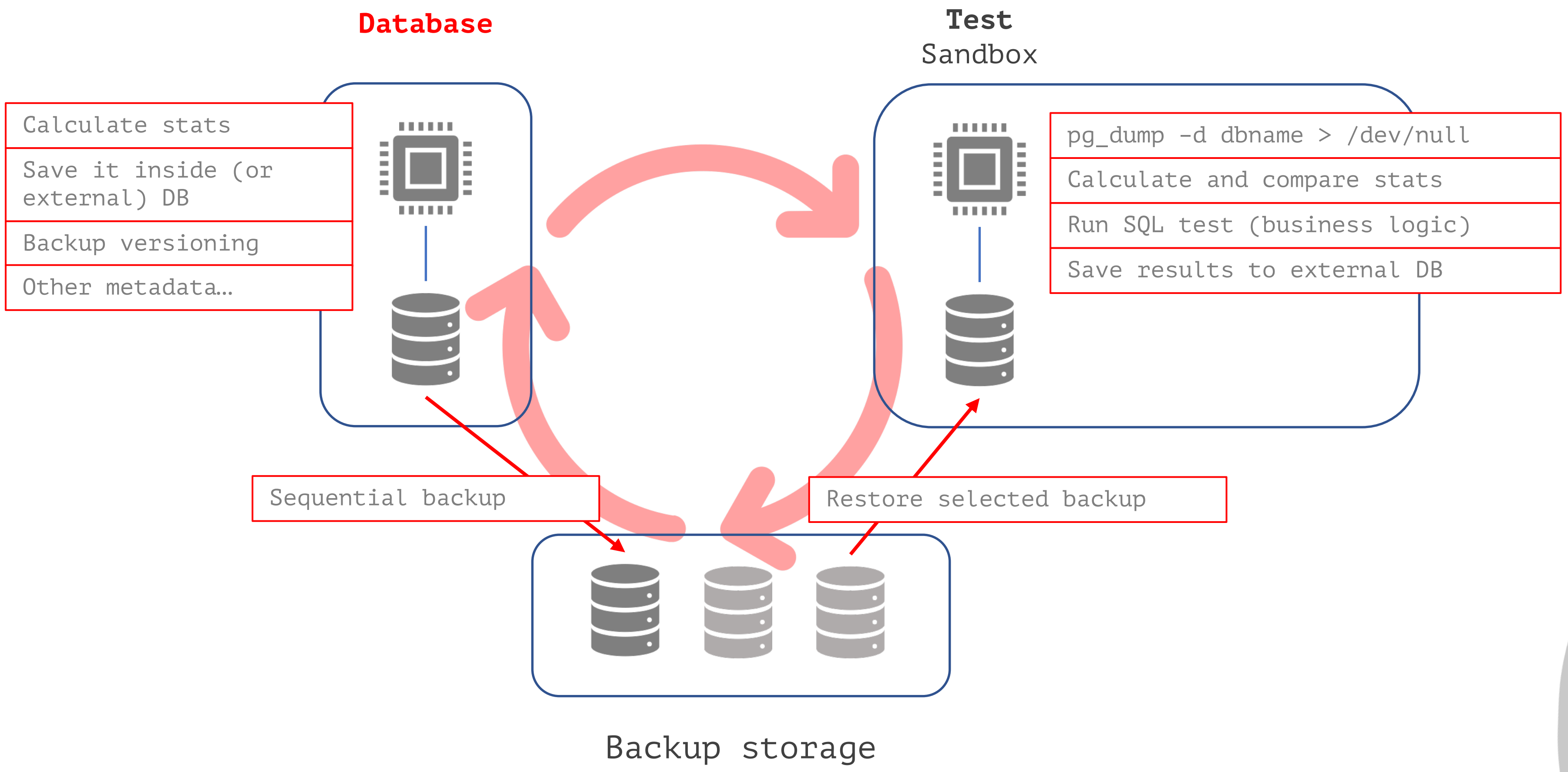
DATABASE-RELATED BACKUP TOOLS

Validation

- **WAL-G**
  - WALG_VERIFY_PAGE_CHECKSUMS
  - wal-verify option
- **pgBackRest**
  - file-level checksums
  - page checksums on backup
- **pg_probackup**
  - file-level checksums
  - page-level checksums
  - validate command (checkdb -amcheck - check indexes)
  - check backup integrity after backup and before restore
- **Barman**
  - custom hooks
- **pg_verifybackup** since PostgreSQL 13!

DATABASE-RELATED BACKUP TOOLS

Validation: how to ensure that backup is valid?


Database
Test
Sandbox
Calculate stats
Save it inside (or external) DB
Backup versioning
Other metadata…
pg_dump -d dbname > /dev/null
Calculate and compare stats
Run SQL test (business logic)
Save results to external DB
Sequential backup
Restore selected backup
Backup storage

DATABASE-RELATED BACKUP TOOLS

Compression

- **WAL-G/WAL-E**
  - LZO
  - WALG_COMPRESSION_METHOD (lz4, lzma, brotli)
- **pgBackRest**
  - --compress (gzip)
  - --compress-level
  - --compress-level-network
- **pg_probackup**
  - --compress-algorithm (zlib, pglz)
  - --compress-level
- **Barman**
  - compression = gzip (basebackup-mode only)
  - network_compression (rsync-mode only)

DATABASE-RELATED BACKUP TOOLS

Encryption

- **WAL-G/WAL-E**
  - WALE_GPG_KEY_ID, gpg
  - WALG_GPG_[KEY,PATH,PASSPHRASE]
  - Yandex Cloud KMS support
  - WALG_LIBSODIUM_[KEY,PATH]
- **pgBackRest**
  - --repo-cipher-type = aes-256-cbc
  - --repo-cipher-pass
- **pg_probackup**
  - Not Yet Implemented
  - There is the problem with Russian laws, we need to obtain a special license to include term “encryption” into.
- **Barman**
  - Not Yet Implemented

DATABASE-RELATED BACKUP TOOLS

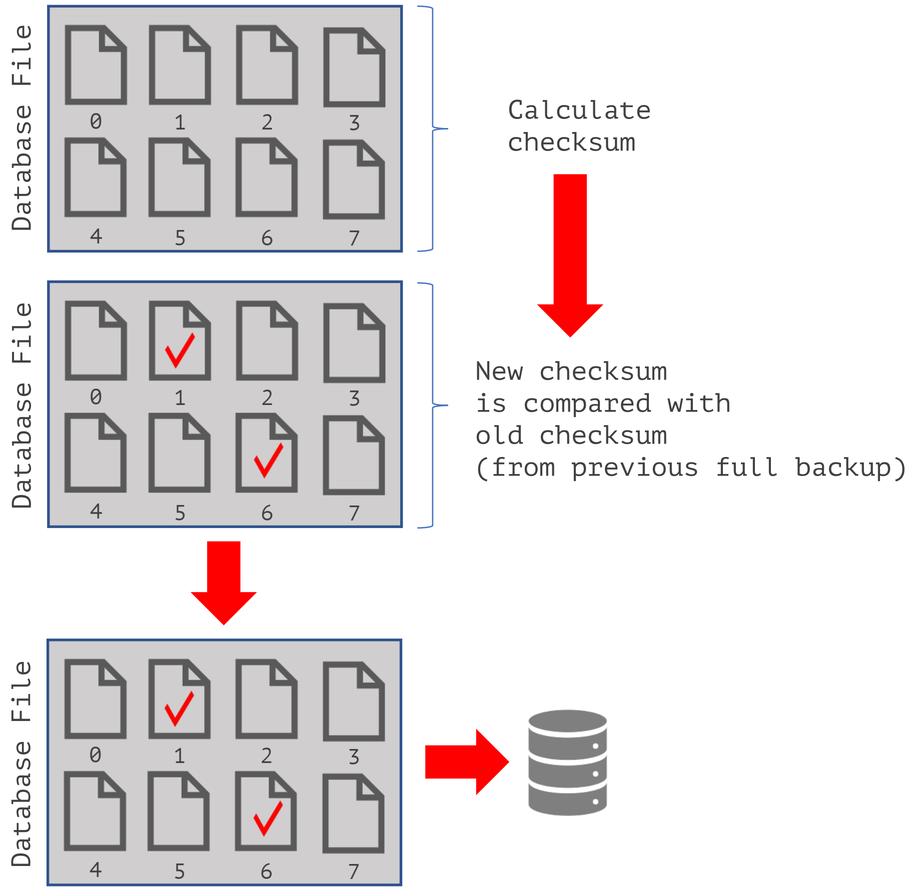
Incremental/Differential

- **WAL-G** (8Kb granularity)
  - page-level incremental DELTA backup
- **pgBackRest**
  - file-level incremental (compare file timestamps)
  - file-level differential
- **pg_probackup** (8Kb granularity)
  - page-level incremental
  - **PTRACK**
    requires PostgreSQL patch:
    - https://github.com/postgrespro/ptrack
  - PAGE (requires WAL archive)
  - DELTA (compare page LSNs)
  - Backup management - MERGE, we can merge the chain of the incremental backups into FULL backup
- **Barman**
  - file-level incremental (rsync-mode only)

ADVANCED BACKUP: INC/DIFF

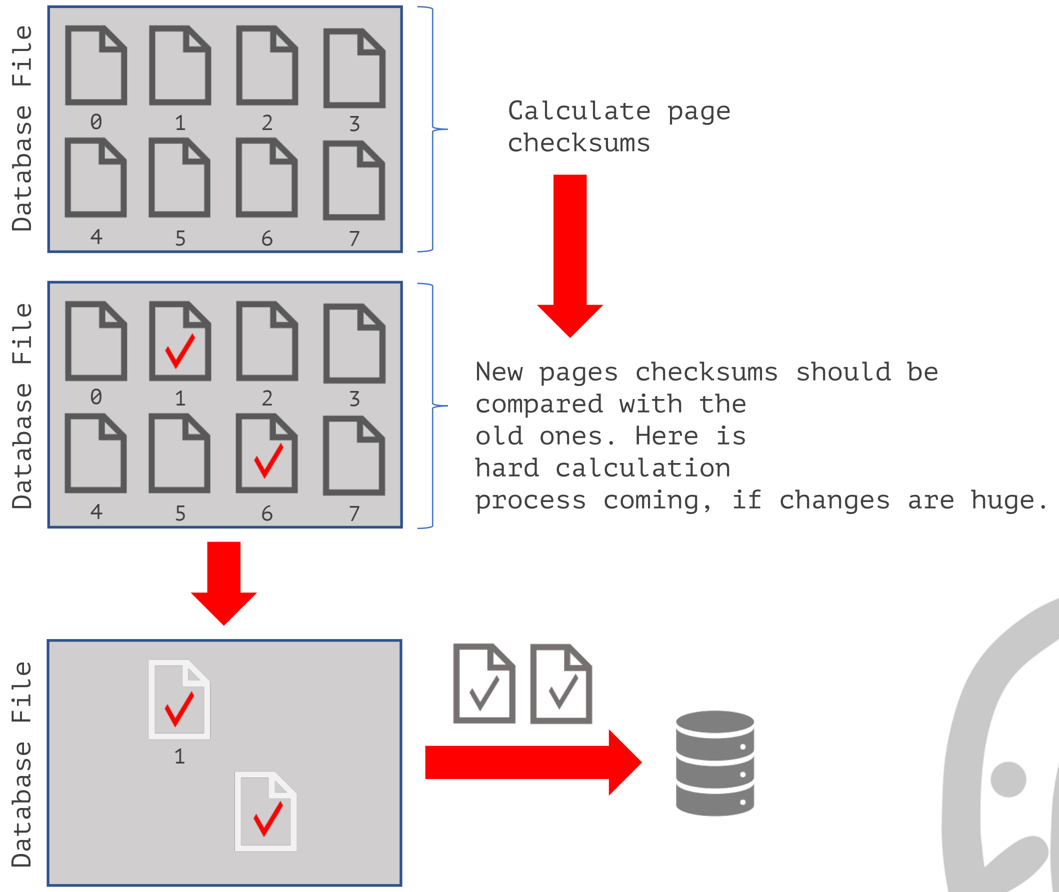
The idea of differential and incremental backups

Backup changed files: Barman, pgBackRest



Backup only
this file

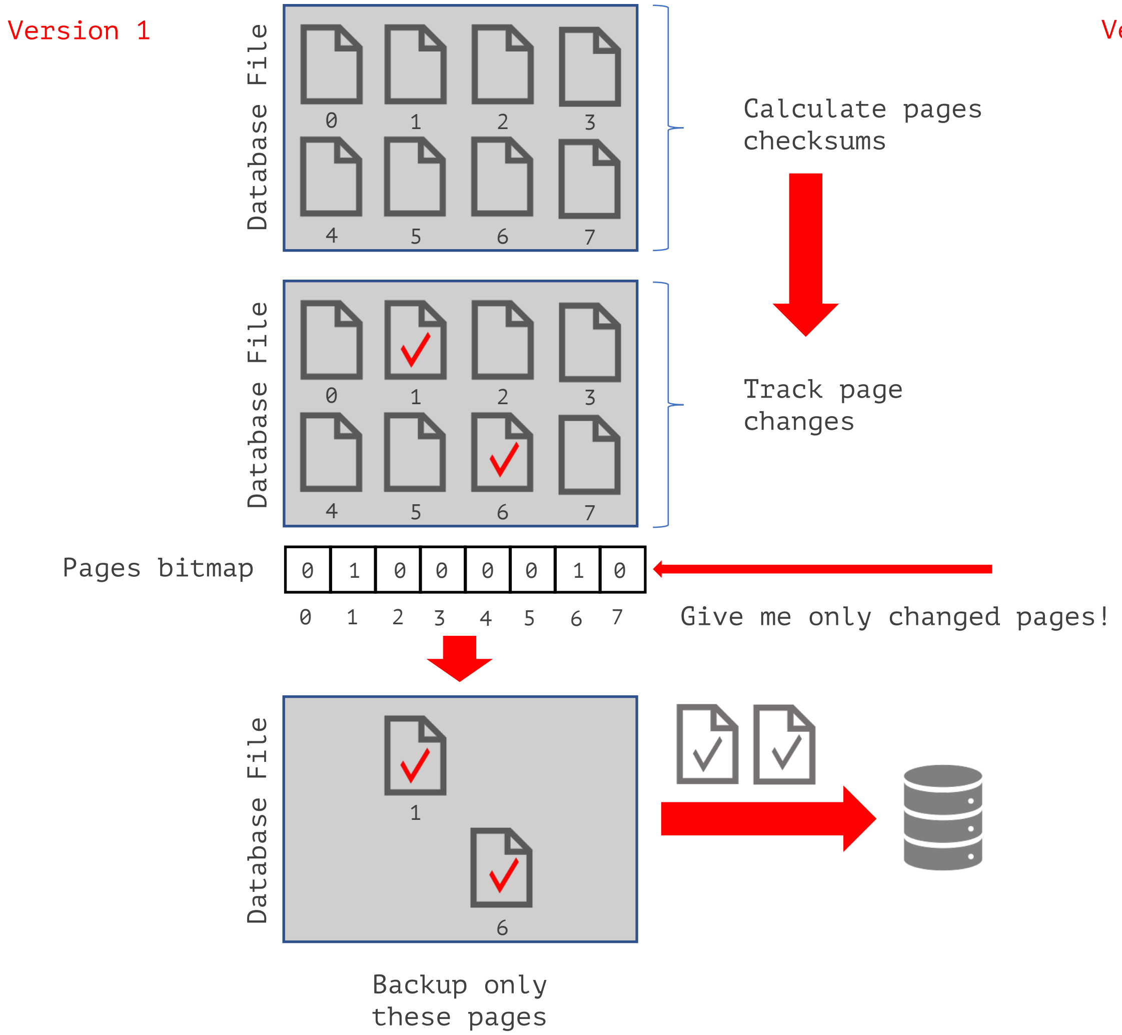
Backup changed pages: WAL-G/E, pg_probackup

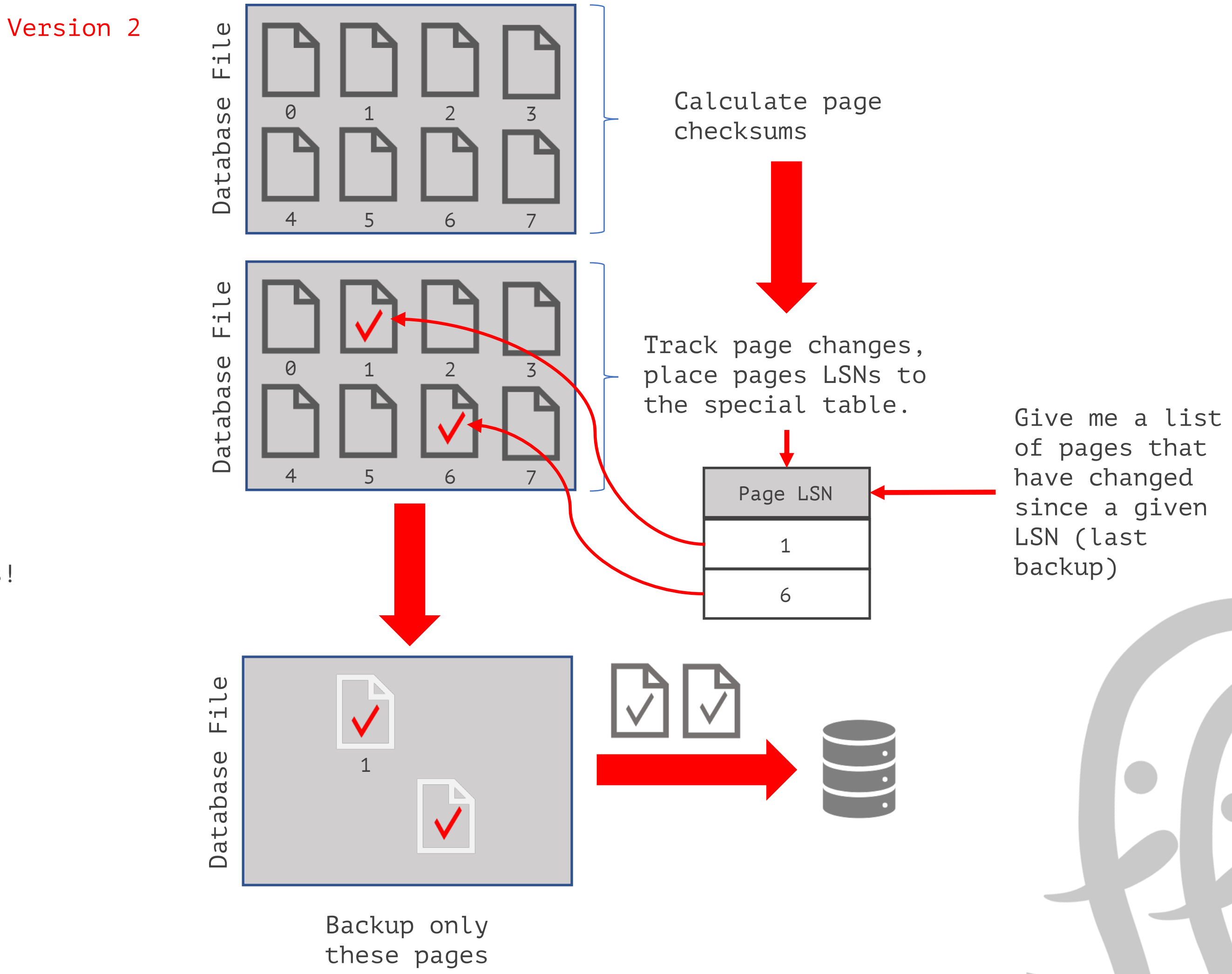


Backup only
these pages

ADVANCED BACKUP: PTRACK

Bitmap and Page LSN: pg_probackup + PTRACK extension





PostgresPro

DATABASE RELATED BACKUP TOOLS

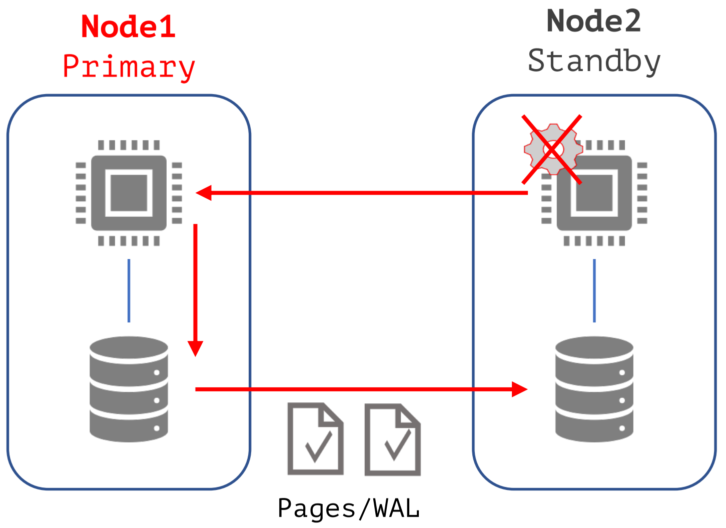
Catchup

- **WAL-G**
  - Creates a copy of a PostgreSQL instance using the backup catalog.
  - wal-g catchup-push /path/to/master/postgres --from-lsn replica_lsn
  - wal-g catchup-fetch /path/to/replica/postgres backup_name

- **pg_probackup**
- Creates a copy of a PostgreSQL instance without using the backup catalog.
  - pg_probackup catchup -b catchup_mode --source-pgdata=path_to_pgdata_on_remote_server --destination-pgdata=path_to_local_dir
- Also we are able to catchup primary by using backup catalog and incremental copies.

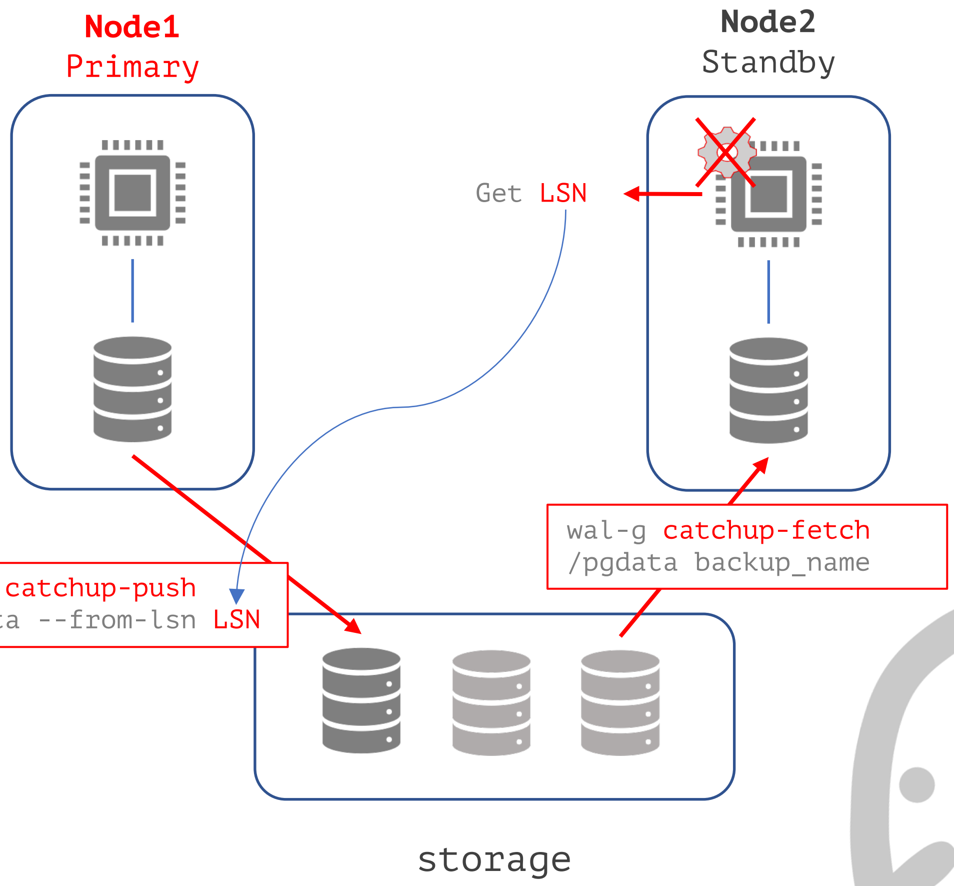
ADVANCED BACKUP: CATCHUP

pg_probackup



```
pg_probackup -b PTRACK
--sourcepgdata=remote_pgdata
--destination-pgdata=local_pgdata
```

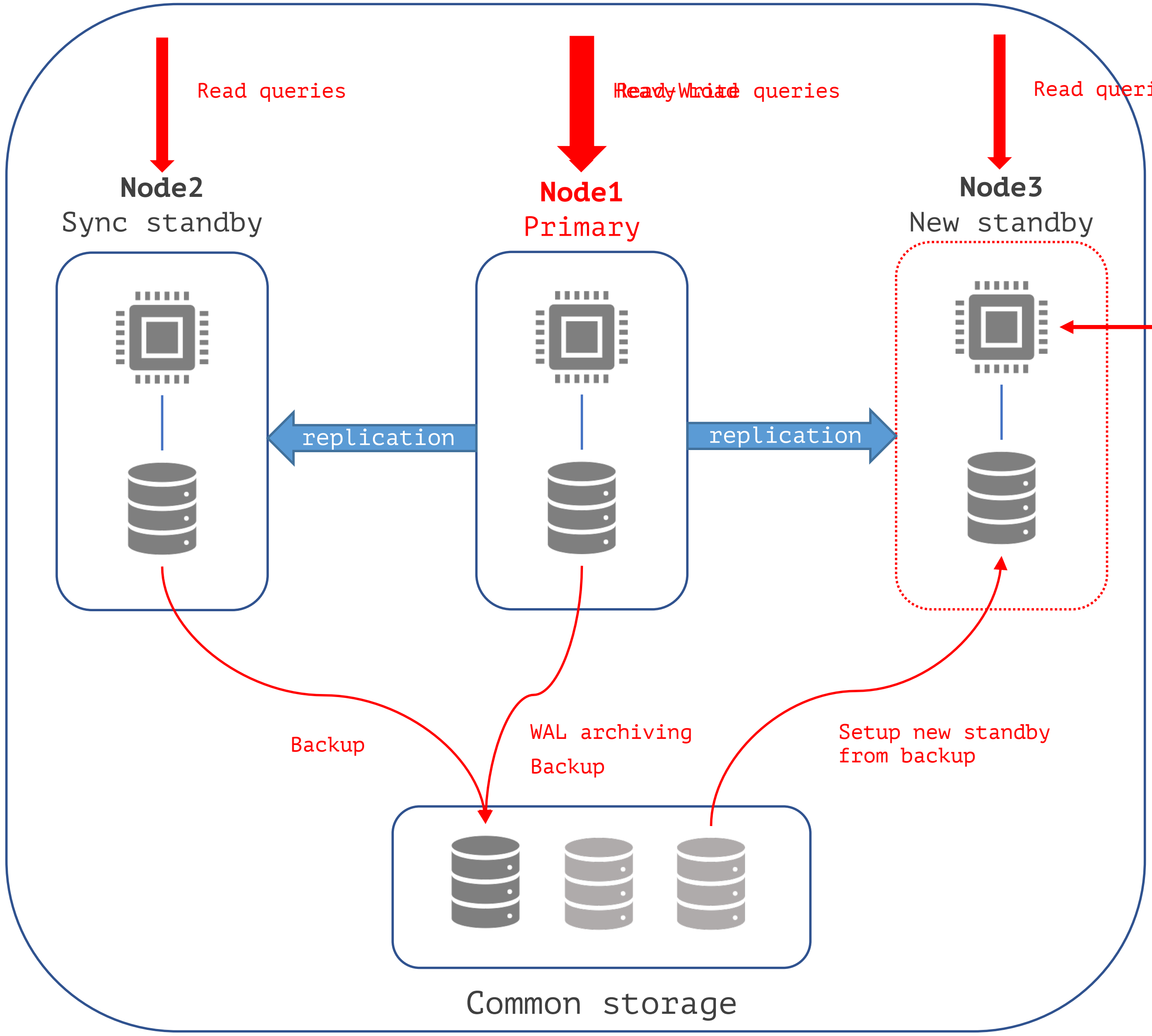
WAL-G



```
wal-g catchup-push
/pgdata --from-lsn LSN
```

```
wal-g catchup-fetch
/pgdata backup_name
```

BACKUP IN HA DEPLOYMENTS



Patroni configuration file:

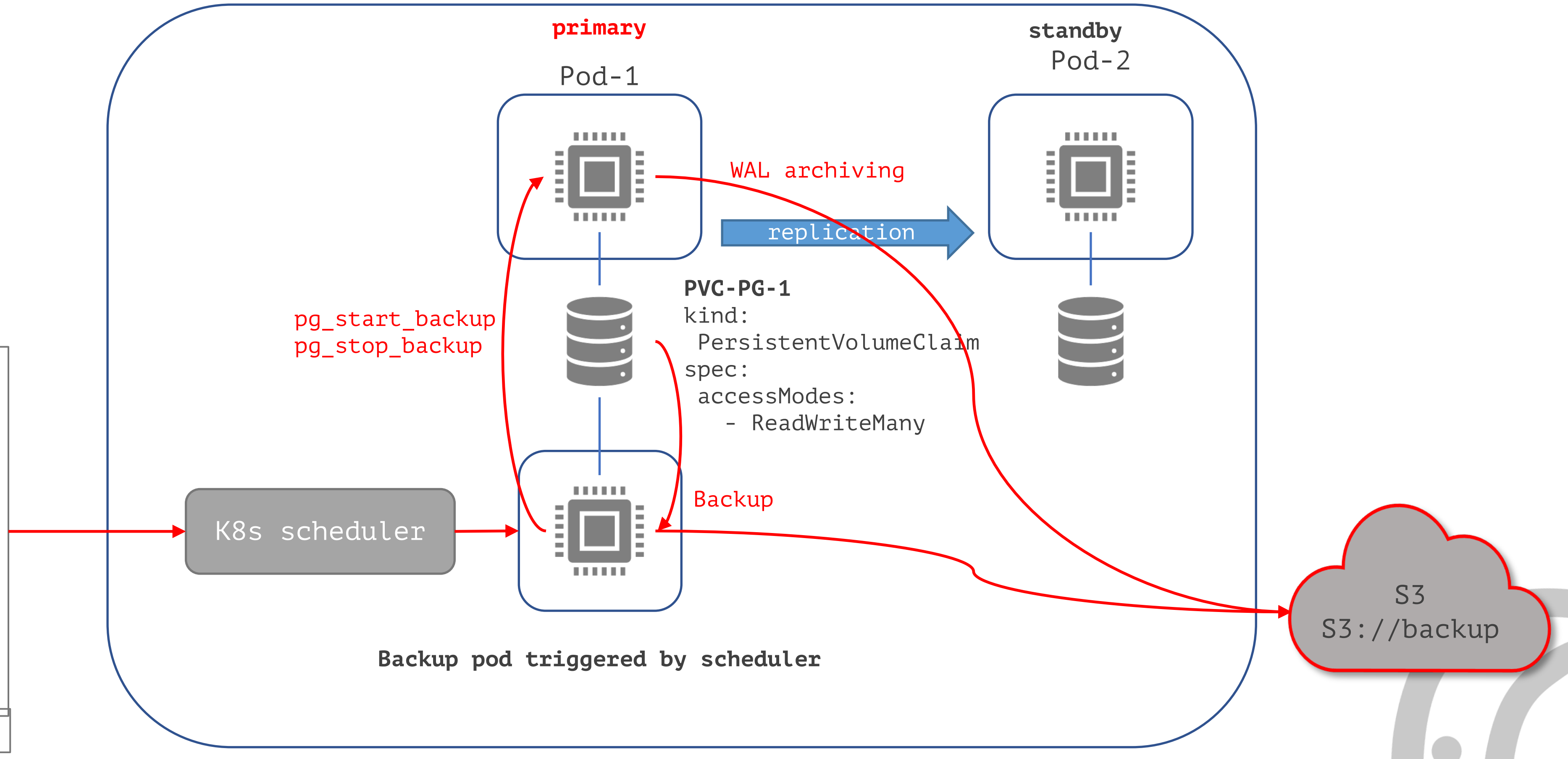
```
postgresql:
 create_replica_methods:
    - probackup
  probackup:
    command:
     - ‘pg_probackup restore --instance cluster -B /backup-dir’
    keep_data: True
    no_params: True
```

PostgresPro

BACKUP IN K8S

```
kind:
 CronJob
spec:
 jobTemplate:
    spec:
      containers:
        - name: backup
          image: backup:v0.1
          volumeMounts:
              - name: pvc-pg-1
                mountPath: /pgdata
                readOnly: true
    schedule: ‘0 1 * * *’
```

Run this job every day at 01:00am



COMPARISON TABLE

Tool	Common repo	Logging	Diff/Inc	Archive	Remote	S3/Cloud	Encryption	Validation	Parallel backup/restore	Compression	Streaming	Catchup
WAL-G	No	No	Yes/page	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
WAL-E	No	Yes	No	Yes	No	Yes	Yes	Yes	Yes	Yes	No	No
pgBackRest	Yes	Yes	Yes/file	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No
pg_probackup	Yes	Yes	Yes/file/ page/PTRACK	Yes	Yes	No	No	Yes	Yes	Yes	Yes	Yes
Barman	Yes	Yes	Yes/file	Yes	Yes	Yes plugin	No	No	No/ rsync	No/ rsync	Yes	No

PostgresPro