

ZERO-DOWNTIME* MIGRATION WITH PERCONA CLUSTERSYNC FOR MONGODB

► Adnan Šupić
PERCONA

▶ **Locked in** is not a topology

Vendor / platform	Atlas, Enterprise, Community → PSMDB
Environment	Cloud, on-premises, provider, or region
Version	Migrate and upgrade in one pass
Capacity	Choose a different node or shard count
Data scope	Move everything or selected namespaces

► Where PCSM stands



Placement: PCSM on a dedicated host

- PCSM runs as a **client process** that connects to both clusters.
- **High-availability feature** uses multiple PCSM instances in an ACTIVE/STANDBY configuration – GA.

► Pick your ground

DEPLOYMENT MODEL	MAIN TRADE-OFF
Dedicated host	Isolates PCSM load from database nodes
Target node	Short target path; shares target resources
Source node	Short source path; shares production resources

- **Demo placement:** PCSM is on the same host with the target cluster.

▶ Version paths

SOURCE MAJOR	SUPPORTED TARGET MAJORS
MongoDB 6	6, 7, 8
MongoDB 7	7, 8
MongoDB 8	8

- Same workload on newer major: migrate and upgrade in one pass.
- Minimum versions: 6.0.17, 7.0.13, 8.0.0.
- Target FCV must be at least the source FCV, downgrades are rejected.

► Topology paths

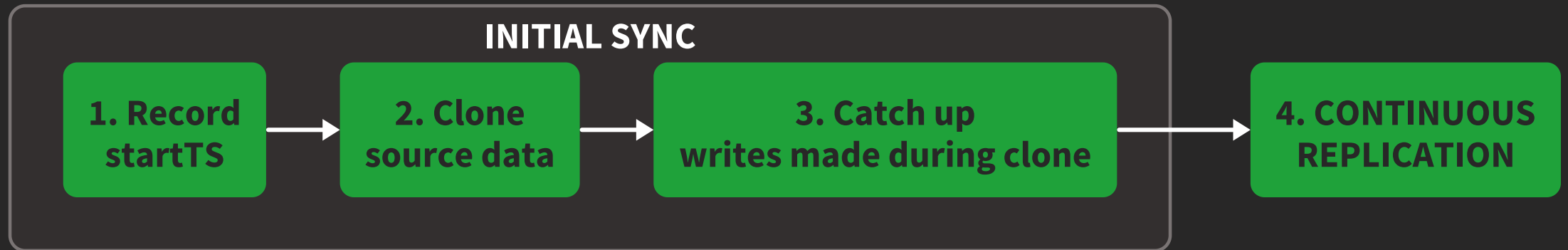
SOURCE \ TARGET	REPLICA SET	SHARDED
Replica set	Supported	Unsupported (clones unsharded)
Sharded	Supported	Supported

► Move first. Cut over when ready.

PCSM | pre-flight › clone source data › catch up › replicate › no new writes › finalize › repoint › verify

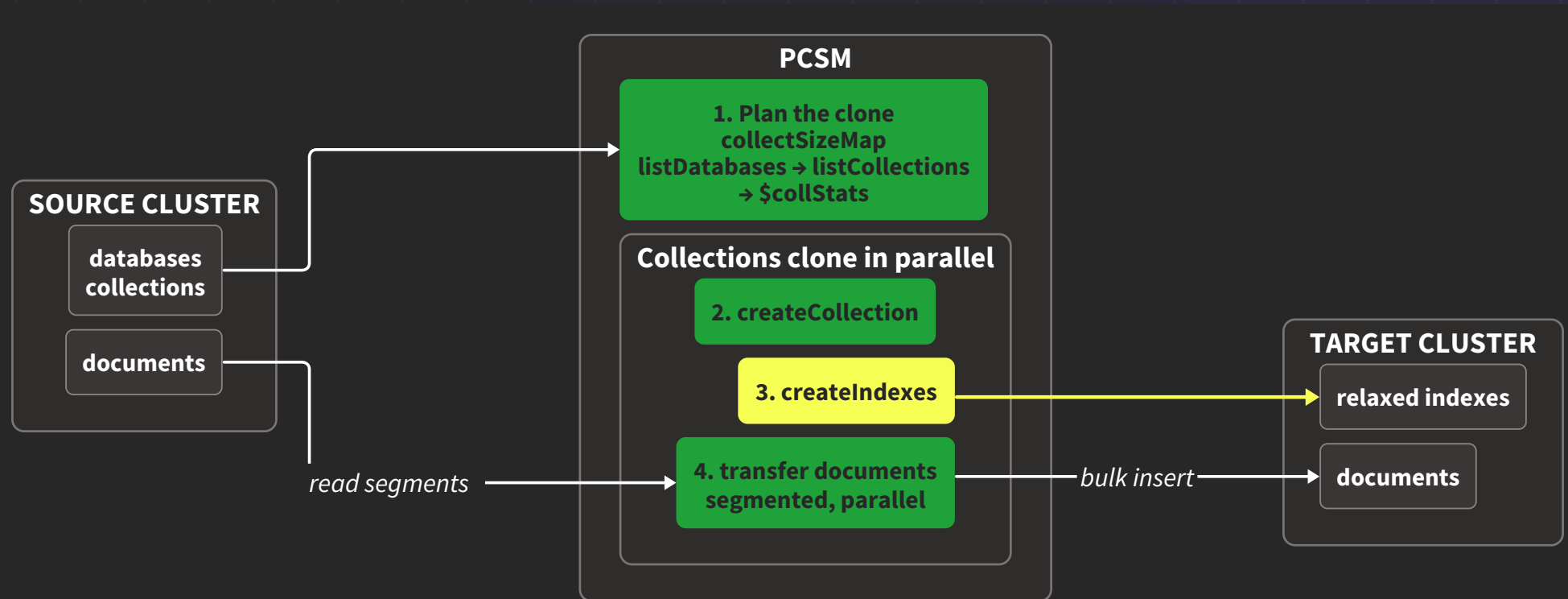
- **Initial sync is clone + catch-up.** PCSM clones source documents, then applies changes that arrived during the clone.
 - Keep source writes running while PCSM clones data and catches up.
- **Replication keeps running** after initial sync until you choose the cutover.
 - Watch lag, then choose the cutover window when appropriate.
- **Cutover keeps the human in the loop:** stop writes, drain lag, finalize, repoint, verify.
- **Lift and shift** in the same migration.

► Initial sync: clone, then catch up



- `appendOpLogNote` pins `startTS` **before the clone** – marks replay start.
- **Clone completion sets** `finishTS` – catch-up target.

► Clone: move source data



- PCSM inventories namespaces, estimates their size, creates relaxed indexes, then transfers documents.
- Collections clone in parallel. Within each collection, readers feed target insert workers.
- The limit is source reads, the network path and target writes.

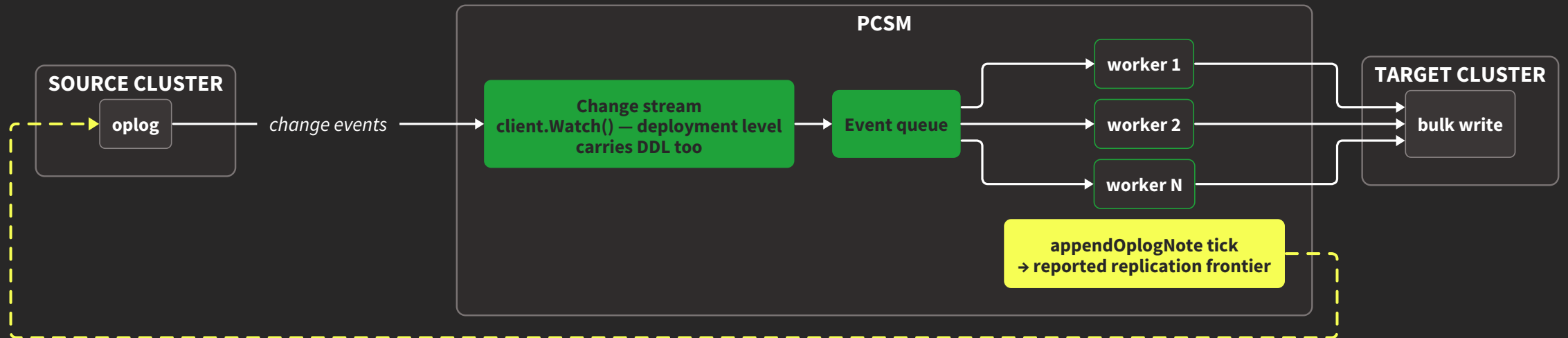
► Relaxed indexes during sync

SOURCE PROPERTY	TARGET DURING MIGRATION
<code>unique: true</code>	non-unique
<code>prepareUnique: true</code>	non-unique
<code>expireAfterSeconds</code> (TTL)	<code>MaxInt32</code> — effectively disabled
<code>hidden: true</code>	unhidden

- Out-of-order inserts cannot satisfy uniqueness until the full dataset exists.
- Active TTL indexes could delete cloned documents before catch-up finishes.
- Finalization restores these properties.

***Zero until cutover. Finalization length depends on index restoration.**

► Replication



- One deployment-level change stream carries document changes and DDL.
- Events stay ordered per document. Capped collections stay on one worker to preserve insertion order.
- The limit is the source change stream, the network path and target write capacity.



PCSM
0.8.0
results

► Read /status

```
curl -s http://localhost:2242/status
```

```
pcsm status
{
  "state": "running",
  "lagTimeSeconds": 2,
  "eventsRead": 184213,
  "eventsApplied": 184201,
  "initialSync": {...}
}
```

- Use `lagTimeSeconds` as the cutover signal.
- After **finalize** is done a new field `finalization` contains more information.

► Pre-flight

1. **Confirm the version and topology path.**
2. **Scope the namespaces** that belong in the migration.
3. **Choose PCSM placement** and confirm target capacity.
4. **Grant source permissions** for namespace reads and `appendOpLogNote`.
5. **Allowlist the PCSM host** and test both cluster connections.
6. **Plan pause, finalize and repoint** to make the cutover.
7. **Define data and metadata checks** to verify the target.

▶ Verify data and metadata separately

VERIFY	WHAT TO CHECK
Data	document counts, checksums or sampled hashes, application reads
Metadata	collection options, index constraints, shard keys and chunk layout

- Data is current when lag time == 0, even prior to finalization.
- Metadata is not byte-for-byte comparable.
 - eg. different chunk owners when shard counts differ
- Reopen writes when you choose, before or after verification.

▷ Cutover

▷ [Finalize is terminal. Plan the write pause around index validation time.]

1. Stop source writes.
2. Wait for `lagTimeSeconds: 0`.
3. **Finalize.** Replication stops first, then PCSM restores `unique` / `prepareUnique`, TTL and hidden index properties.

```
pcsm finalize
```

4. Wait for `finalization.completed`, then inspect `unsuccessfulIndexes[]`.
Validating many or large unique indexes is what stretches this step.
5. Repoint the app and resume writes on the target.

► Rough around the edges

Topology support	Sharded → sharded and SH → RS supported, RS → SH not supported.
Cluster manager on source	<code>appendOpLogNote</code> command requires <code>clusterManager</code> role on the source.
Cluster metadata	PCSM does not move users, roles, zones, database primaries or balancer settings.
Sharding changes	Don't use shard operations during replication.
Recovery	An oplog gap ends the run. Clone failover restarts the clone.
Failback	Finalize is terminal, sync can't be reversed.
Single source per target	One PCSM migration can target a cluster.

- What else? **You tell us** what you'd like to see included in the roadmap!

▶ Track PCSM development

- **Code:** github.com/percona/percona-clustersync-mongodb
- **Docs:** docs.percona.com/percona-clustersync-for-mongodb
- **WIP:** distributed multi-instance shard replication, reverse replication / fail-back, and data verification.
- **Tickets and roadmap:** [PCSM Jira Backlog](#)
- **Contact:** `adnan.supic@percona.com`



Code



Docs



Jira backlog