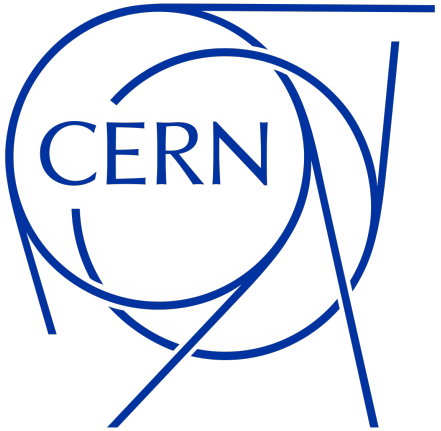
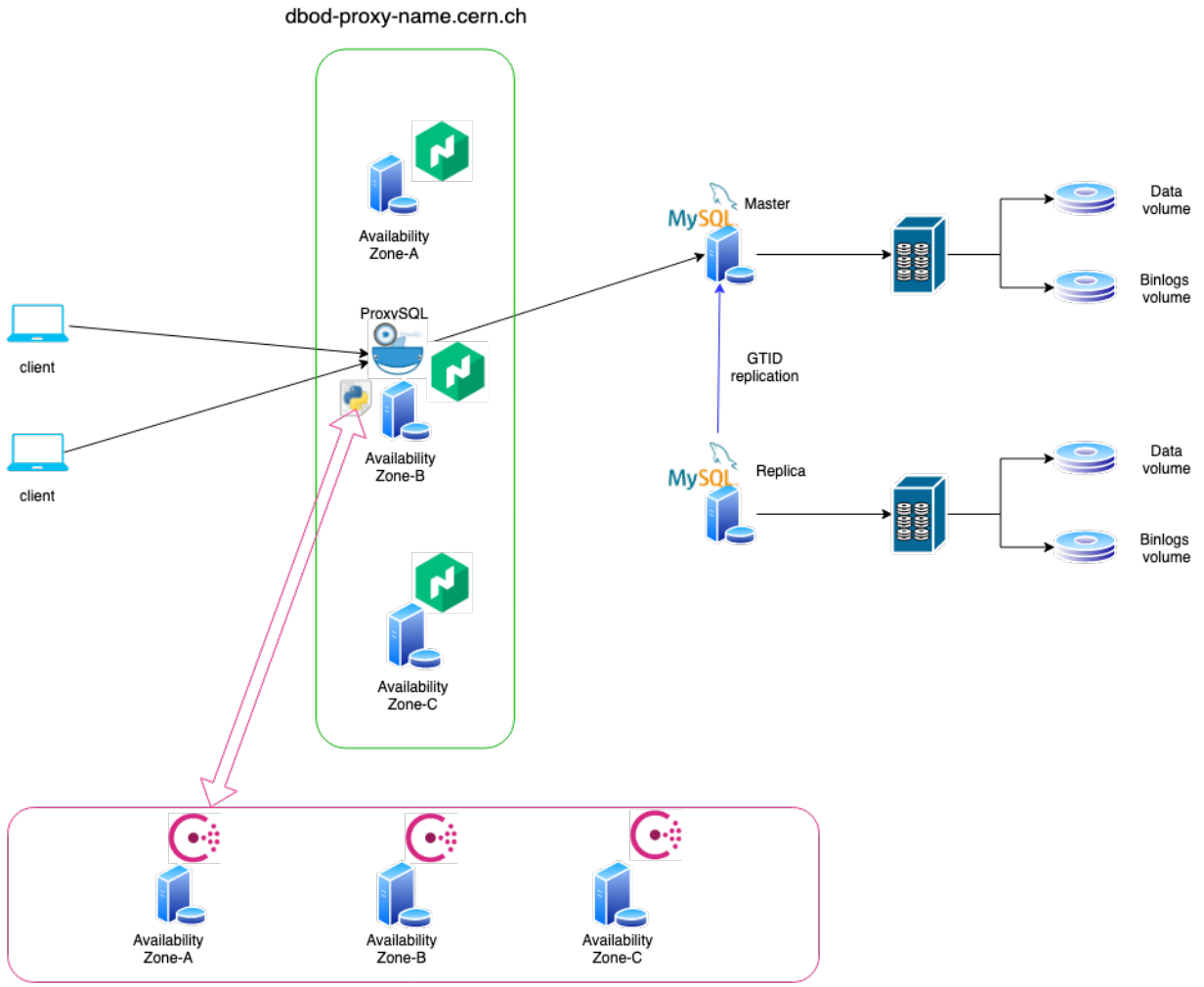


Migration of DBOD ProxySQL MySQL HA Clusters to InnoDB HA clusters

Nikolaos Smyrnioudis on behalf of the DBOD team

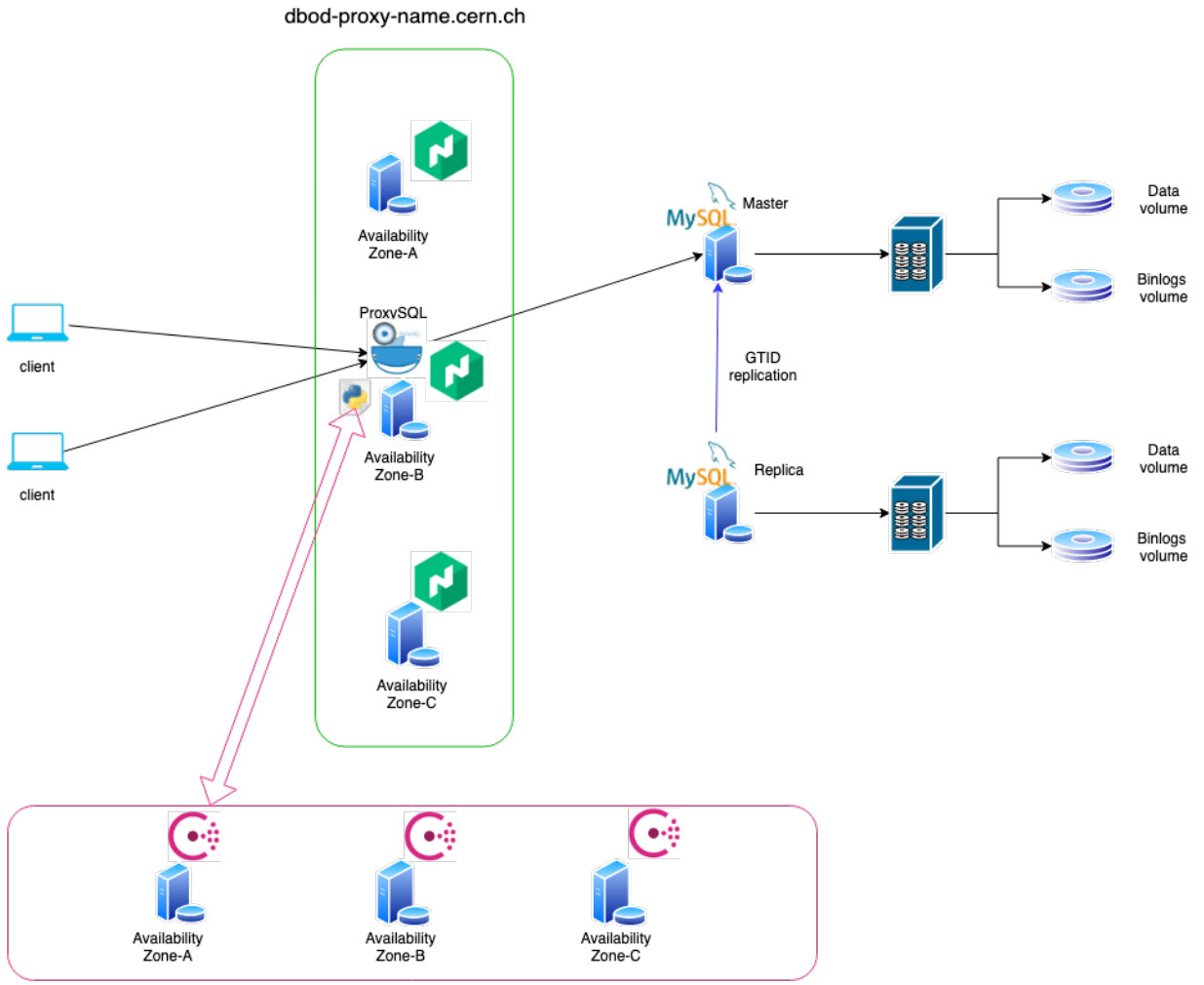
23/9/2024

The current MySQL HA Situation



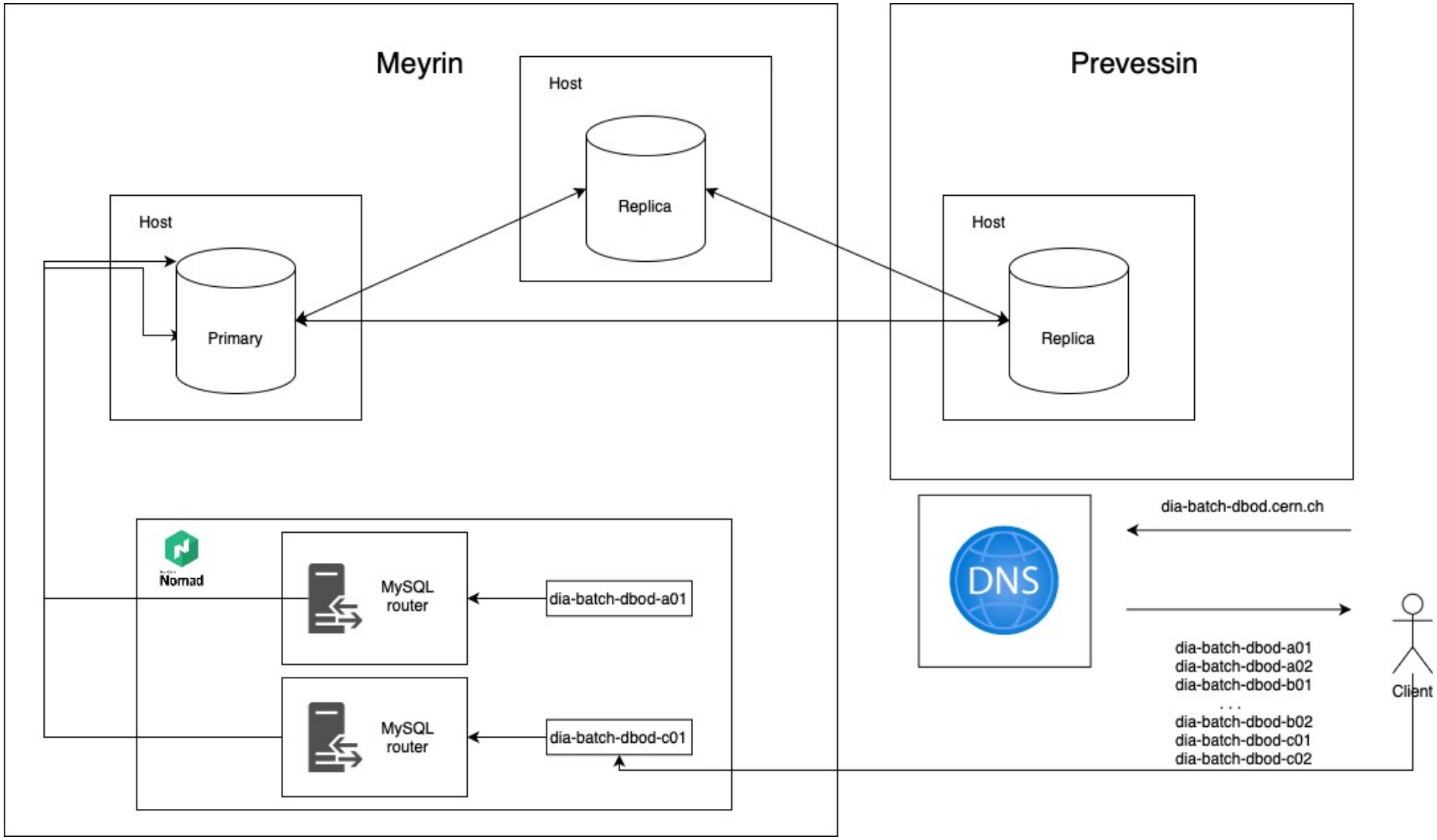
- ProxySQL based
  - Deployed since 2020 in production
- Clients connect to a proxy deployed in a Nomad cluster
- 1 primary and 1 replica (GTID based semi-synchronous replication)

The current MySQL HA Situation - Drawbacks

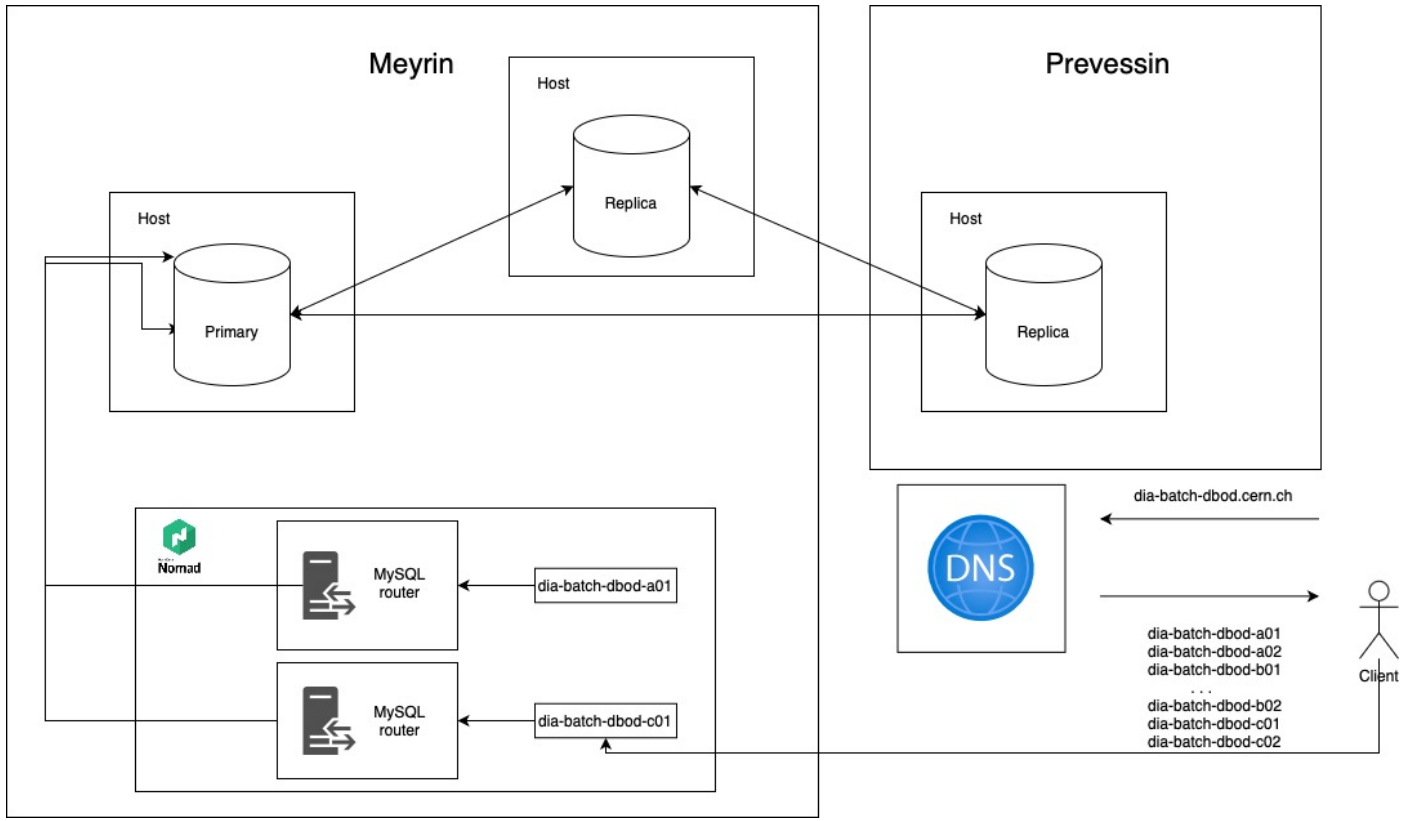


- Code (most importantly failover logic) developed in-house, requiring extensive maintenance effort to keep up with MySQL updates
- Only one proxy at a time supported

The InnoDB Cluster deployment



The InnoDB Cluster deployment - Benefits



- Higher number of redundant components at all levels
- Better automated and easier to use
- Easier to maintain than the existing solution
- Failover and fallback logic provided by the solution (no extra customisation layer required)
- Out of the box solution
- Operations (e.g Switchover, rejoin, remove) can be done with built in mysql-shell with minimal effort in our side

Differences when running with InnoDB Cluster

- Every table must have a primary key
  - We add automatically invisible primary keys for existing tables and new tables
  - If a user tries to add a primary key on a table where we have already added a primary key the operation will fail
- Maximum transaction size
  - The default limit is 150MB, we don’t expect any of our instances will ever reach a limit bigger than this, limit can be increased upon request

User communities

- PANDA/Harvester - 2 clusters
  - harvester
  - harvester_2
- Drupal - 5 clusters
  - drupalk8s_stg (Staging cluster)
  - drupalk8s_crit (Critical cluster)
  - drupalk8s_13
  - drupalk8s_11
  - drupalk8s_10
- Keycloak/SSO - 2 clusters
  - keycloak_qa (Staging/QA cluster)
  - keycloak

- We execute the migration on the available Staging clusters first
- Harvester InnoDB staging cluster set up
- For Drupal we migrate each cluster one day at a time
- We leave production keycloak for last

Migration procedure

- Tested with all three different setups.
- Used exact copies of the instances in our separate staging environment (used recent backups)
- Artificial load with **sysbench** during the tests. Sysbench is a widely used tool that generates synthetic workloads, simulating real loads.
- **~30** seconds of downtime spread out during the intervention, lowered throughput and increased latency spread out throughout the intervention
- No need for clients to change anything in their connection details

Cluster	Data size (GB)	Time (mins)
Keycloak	~35	35
Drupal	~170	190
Harvester	~15	30

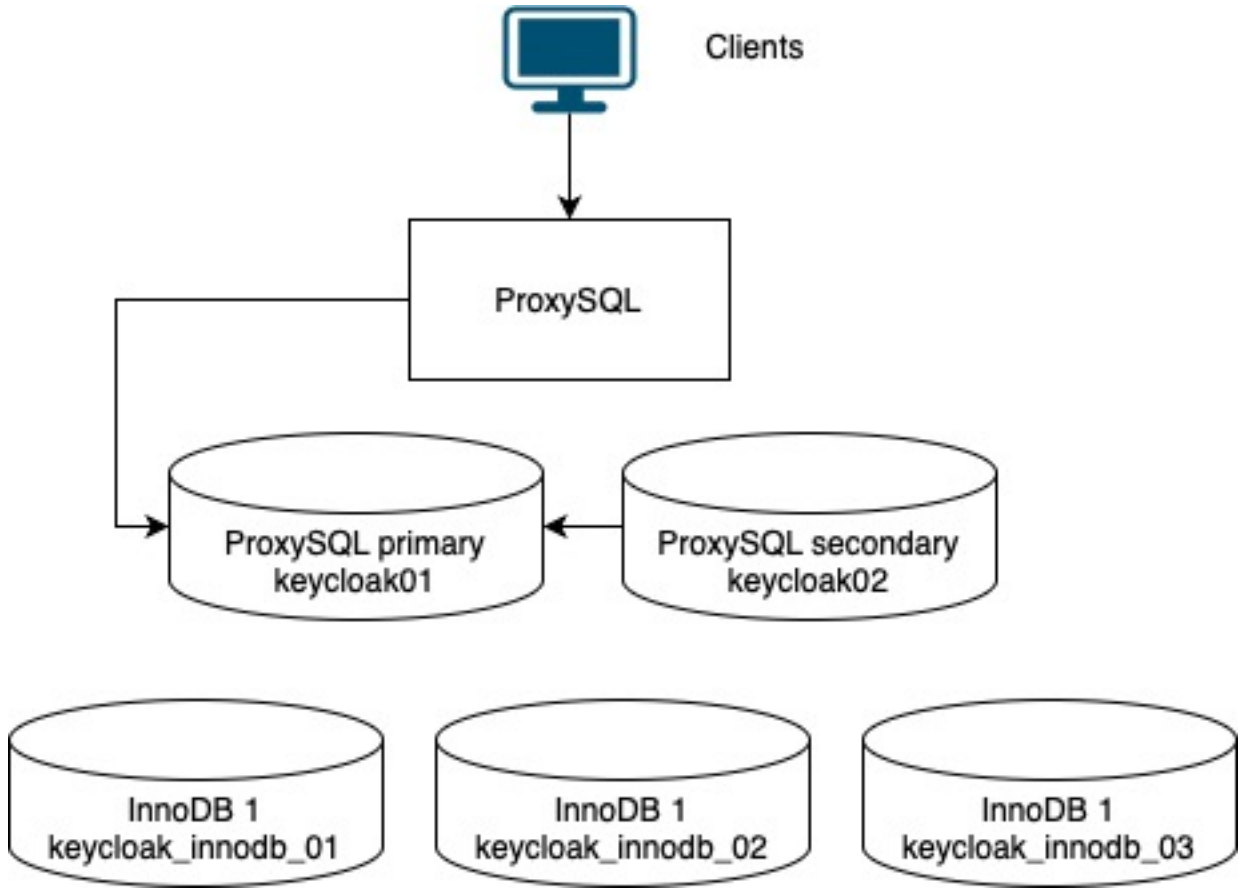
Migration procedure - Impact on the users

- 30 seconds of unavailability throughout the intervention window
- Clients need to reconnect 2 times through the intervention
- Occasional slight drop of throughput and increase in latency
- **Will reevaluate our downtime estimate with the QA cluster interventions**
- During the Migration we plan to be in contact with the affected service managers

Cluster	Data size (GB)	Time (mins)
Keycloak	~35	35
Drupal	~170	190
Harvester	~15	30

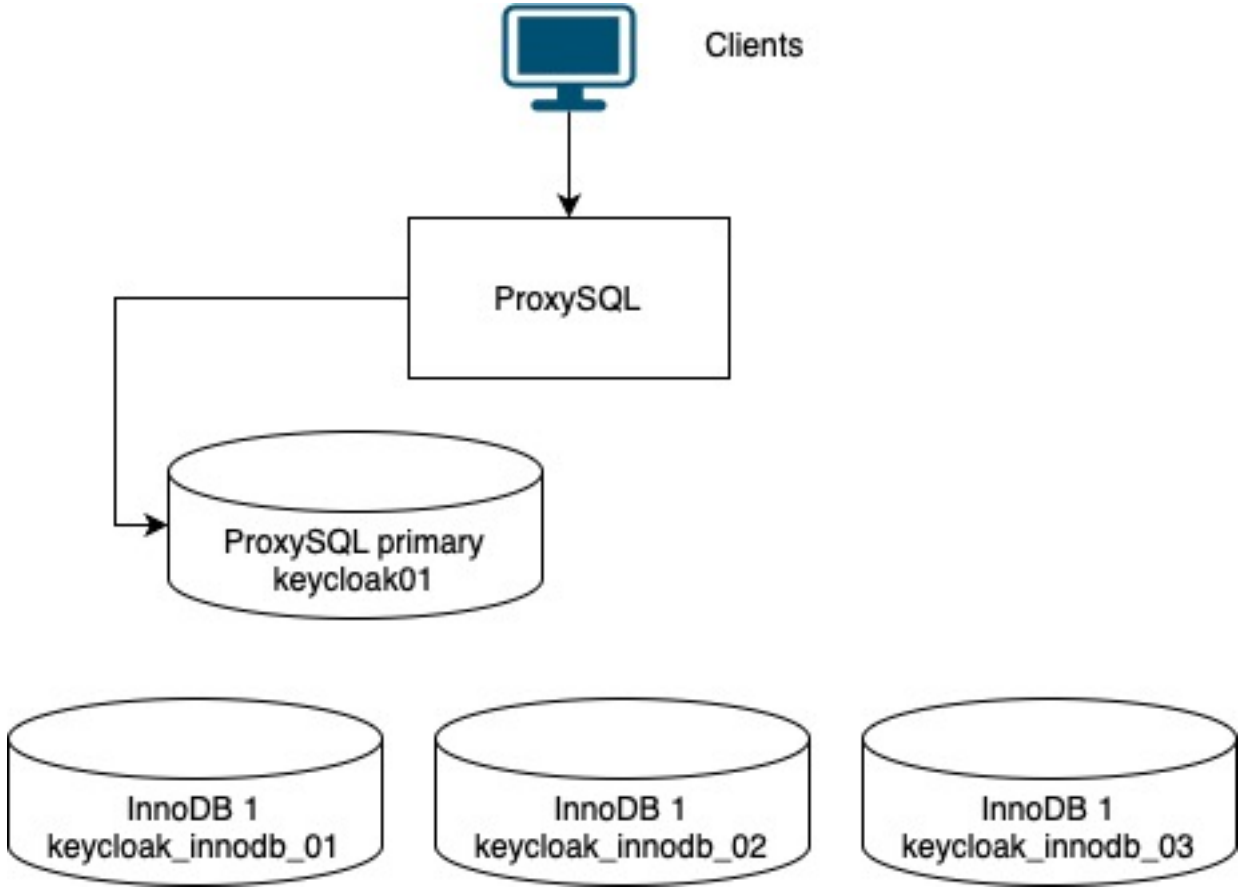
Migration procedure - 1

- Stop replica of ProxySQL cluster
- Bootstrap HA cluster using the primary
- Gradually add new instances
- Deploy MySQL Router and switch clients from ProxySQL to MySQL
- Failover to a fresh InnoDB Cluster instance



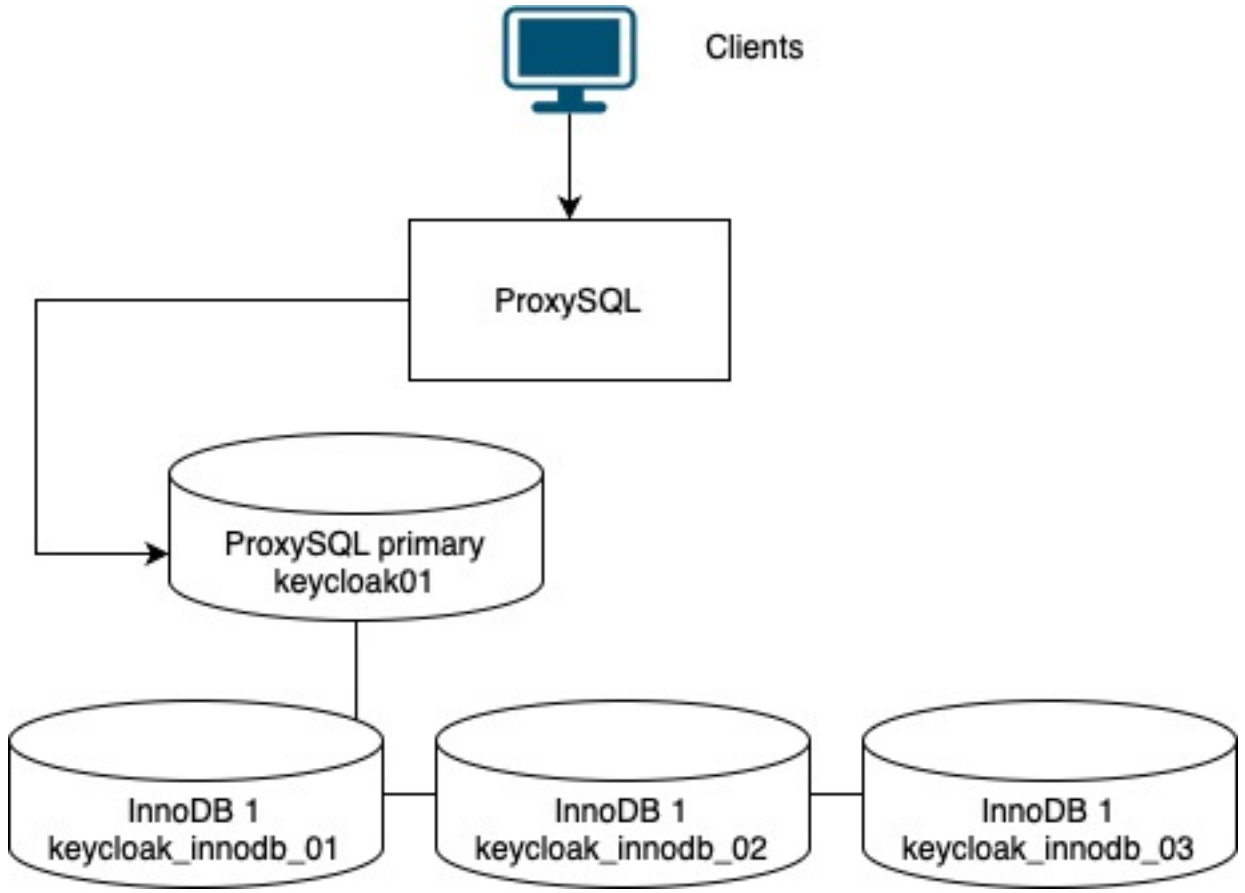
Migration procedure - 2

- Stop replica of ProxySQL cluster
- Bootstrap HA cluster using the primary
- Gradually add new instances
- Deploy MySQL Router and switch clients from ProxySQL to MySQL
- Failover to a fresh InnoDB Cluster instance



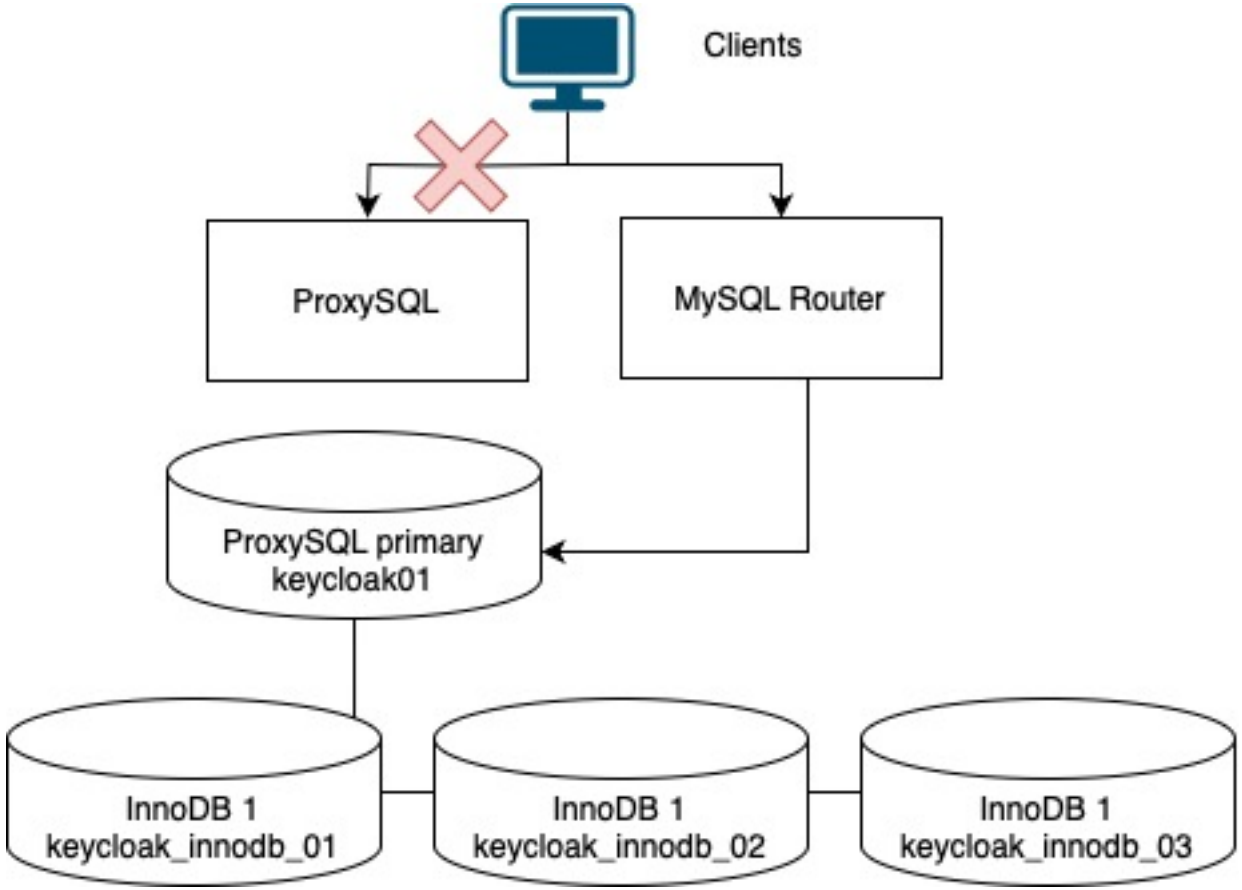
Migration procedure - 3

- Stop replica of ProxySQL cluster
- Bootstrap HA cluster using the primary
- Gradually add new instances
- Deploy MySQL Router and switch clients from ProxySQL to MySQL
- Failover to a fresh InnoDB Cluster instance



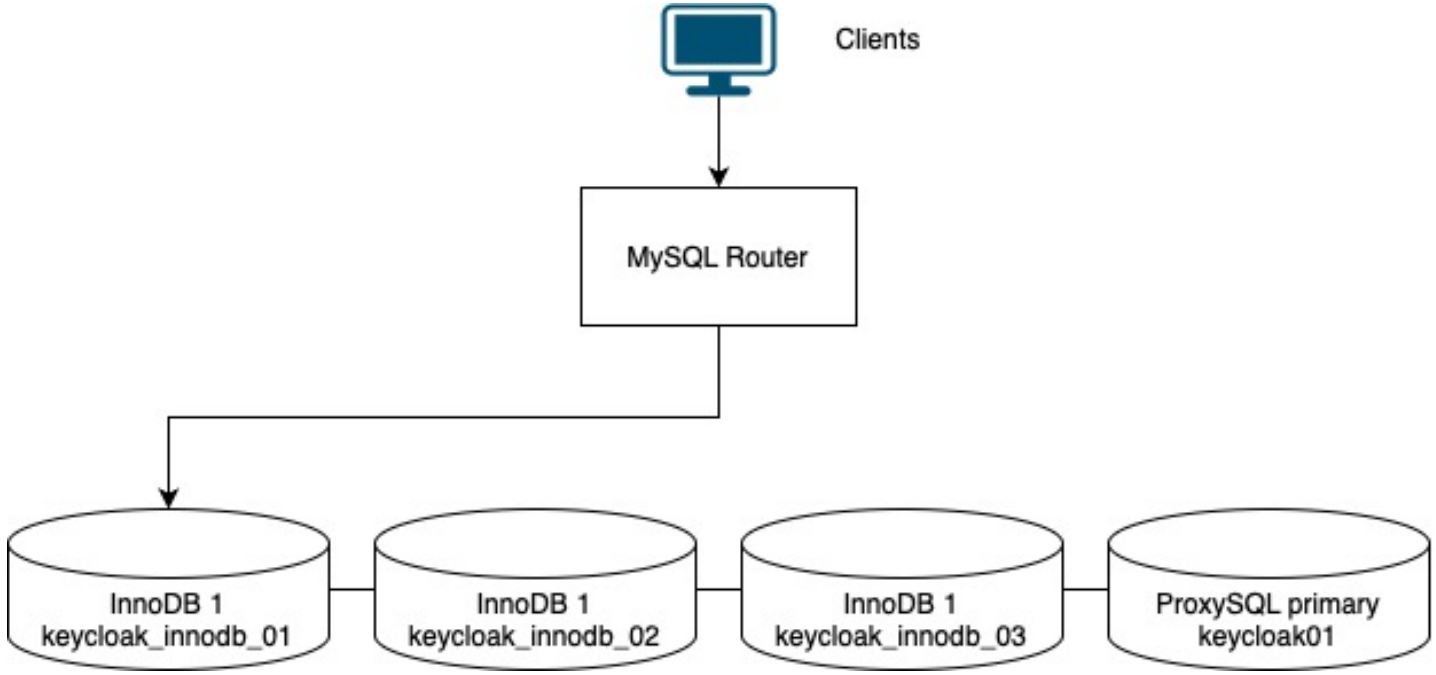
Migration procedure - 4

- Stop replica of ProxySQL cluster
- Bootstrap HA cluster using the primary
- Gradually add new instances
- Deploy MySQL Router and switch clients from ProxySQL to MySQL
- Failover to a fresh InnoDB Cluster instance



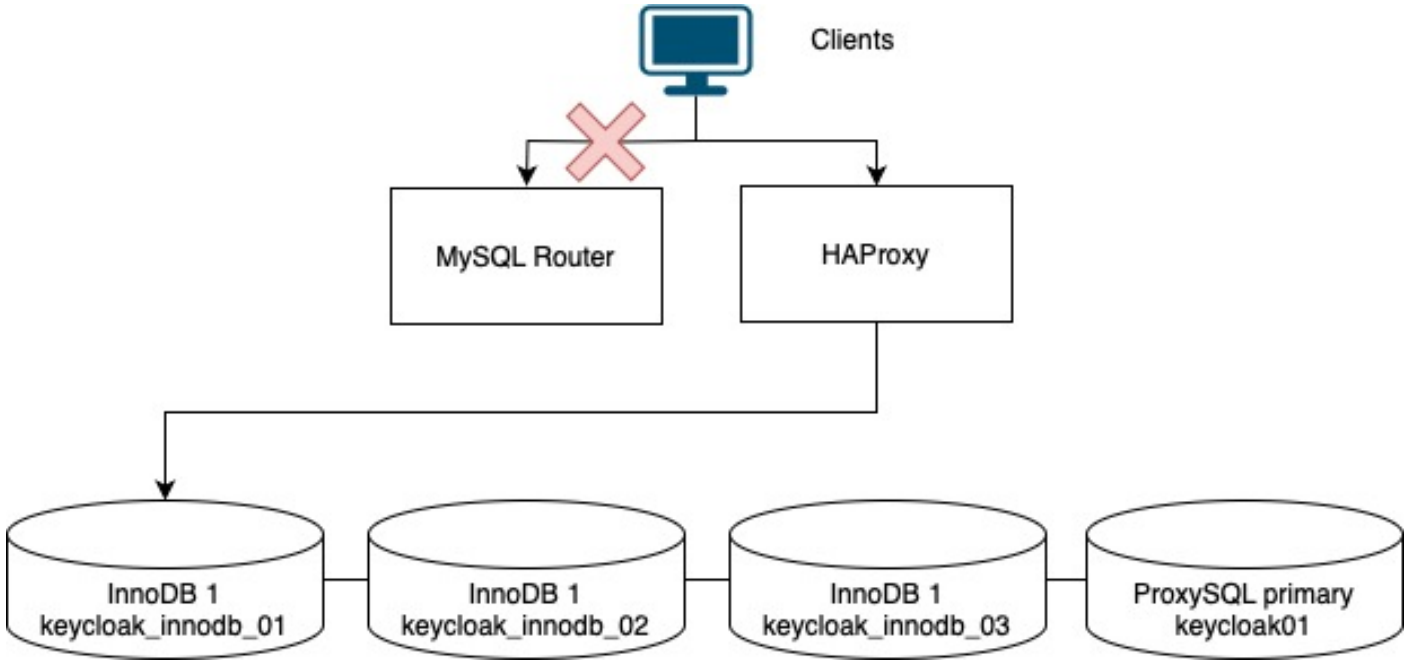
Migration procedure - 5

- Stop replica of ProxySQL cluster
- Bootstrap HA cluster using the primary
- Gradually add new instances
- Deploy MySQL Router and switch clients from ProxySQL to MySQL
- Failover to a fresh InnoDB Cluster instance



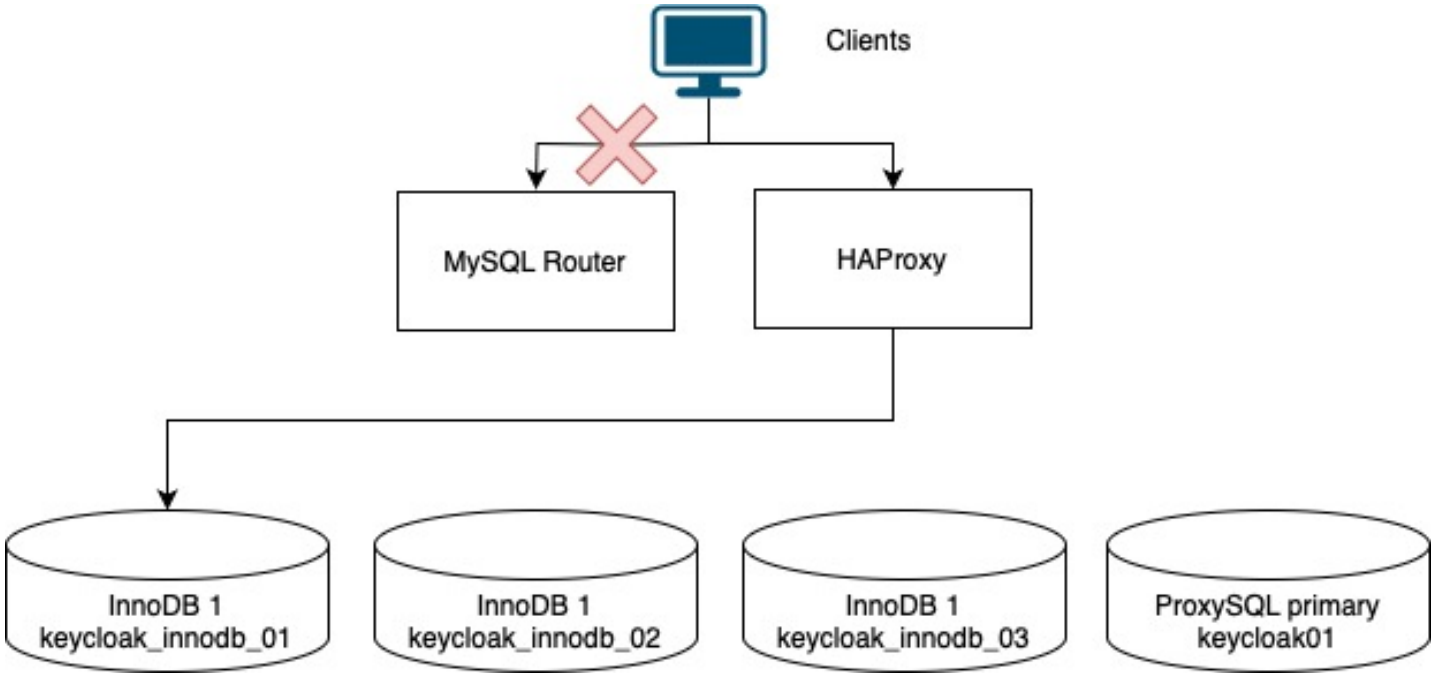
Rollback - 1

- Move clients from MySQL Router to an HAProxy pointing to the primary (no client side change needed)
- Eject InnoDB cluster
- Create a semi-synchronous replica
- Redeploy previous ProxySQL proxy using updated configuration



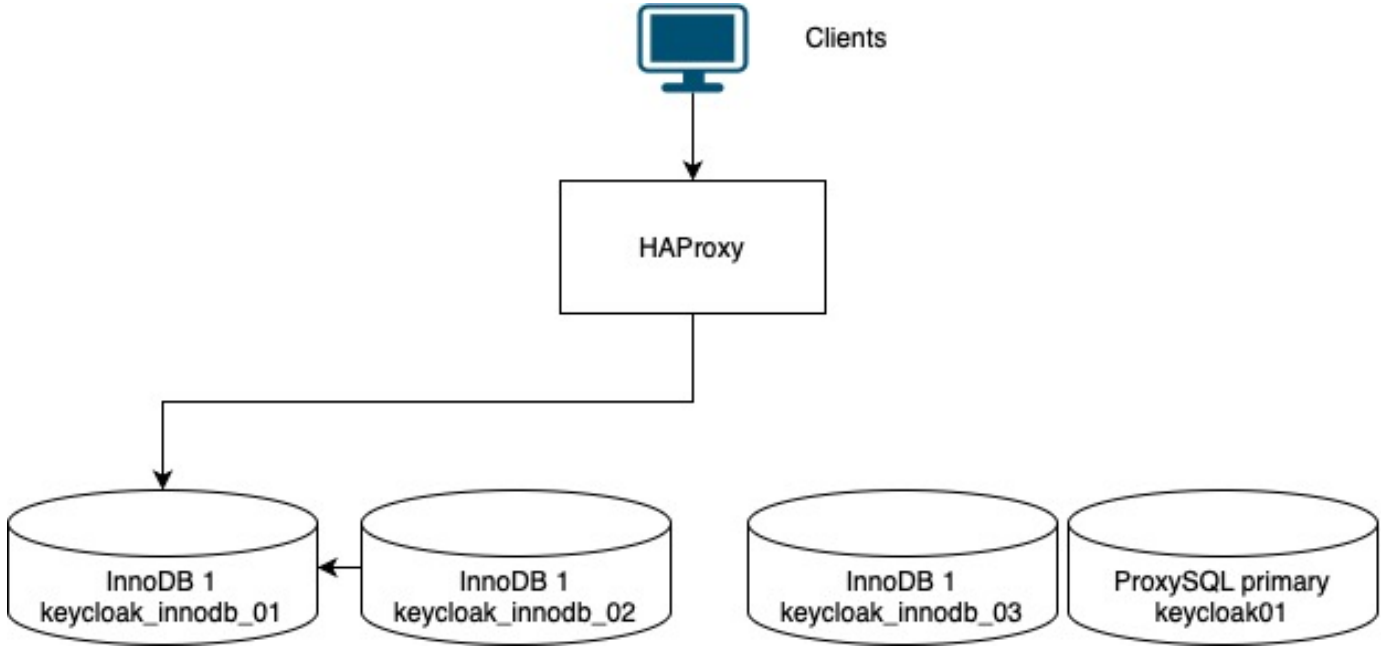
Rollback - 2

- Move clients from MySQL Router to an HAProxy pointing to the primary (no client side change needed)
- Eject InnoDB cluster
- Create a semi-synchronous replica
- Redeploy previous ProxySQL proxy using updated configuration



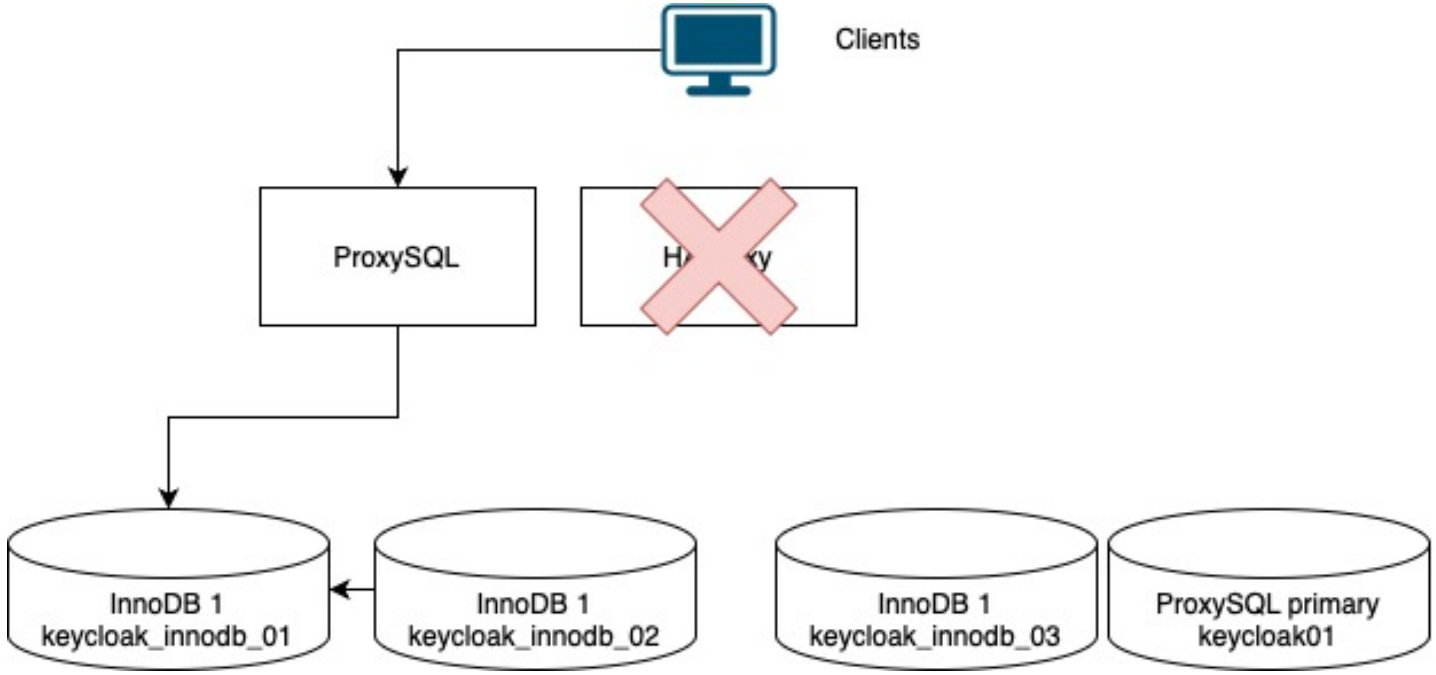
Rollback - 3

- Move clients from MySQL Router to an HAProxy pointing to the primary (no client side change needed)
- Eject InnoDB cluster
- Create a semi-synchronous replica
- Redeploy previous ProxySQL proxy using updated configuration



Rollback - 4

- Move clients from MySQL Router to an HAProxy pointing to the primary (no client side change needed)
- Eject InnoDB cluster
- Create a semi-synchronous replica
- Redeploy previous ProxySQL proxy using updated configuration



Thank you!

Questions?

home.cern