



BRING YOUR OWN CLOUD

Managed ProxySQL[®], in your own AWS account.

How the BYOC model works end to end — what runs where, what crosses the boundary, and what you operate once it is live.

René Cannaò / Alkin Tezuysal

MySQL BR Conf 2026 — A Comunidade Contra-Ataca / 26 September 2026 / São Paulo

 MySQL :6033 /  PostgreSQL :6133 / Admin :6032

SPEAKERS

Who is talking



René Cannaò

Creator of ProxySQL · Founder and CEO

Started ProxySQL in 2013. His work connects database engineering with the operational problems teams face in production.



Alkin Tezuysal

Chief Strategy Officer, ProxySQL

Author, MySQL Rockstar, Oracle ACE Pro (MySQL), and MySQL contributor.

THE TEAM BEHIND IT

What we provide

Support

Help operating business-critical ProxySQL deployments.

Consultancy

Architecture, performance, migration, and reliability work.

Training

Deployment, configuration, and day-to-day operations.

SaaS

ProxySQL.Cloud — fleet management and monitoring, in your account.

| Open source software, with commercial services from the team that builds it.

AGENDA

What we'll cover

01 The proxy tier

What ProxySQL does for a MySQL or PostgreSQL fleet.

03 Standing up a cluster

From account connection to a serving endpoint.

05 Observability

Metrics, query analytics, and the audit trail.

02 The BYOC model

Control plane, data plane, and the line between them.

04 Day-2 operations

Config, credentials, maintenance, scaling, backups.

06 Adoption

What you provide, and how a rollout is sequenced.

01

The proxy tier

What ProxySQL does, for a team that already runs MySQL or PostgreSQL.

THE CORE PRODUCT

What is ProxySQL®?

An open source database proxy, started in 2013. It sits between your applications and your databases.

NATIVE PROTOCOLS

MySQL and PostgreSQL

Each protocol keeps its own session semantics. Apps connect with their existing drivers on :6033 and :6133.

LIVE CONFIGURATION

Change it without a restart

Edit servers, users, and query rules, then
`LOAD ... TO RUNTIME. SAVE ... TO DISK`
persists them.

OPERATIONAL VISIBILITY

See every query shape

Query digests, connection pool state, and Prometheus metrics, from the one place all traffic passes through.

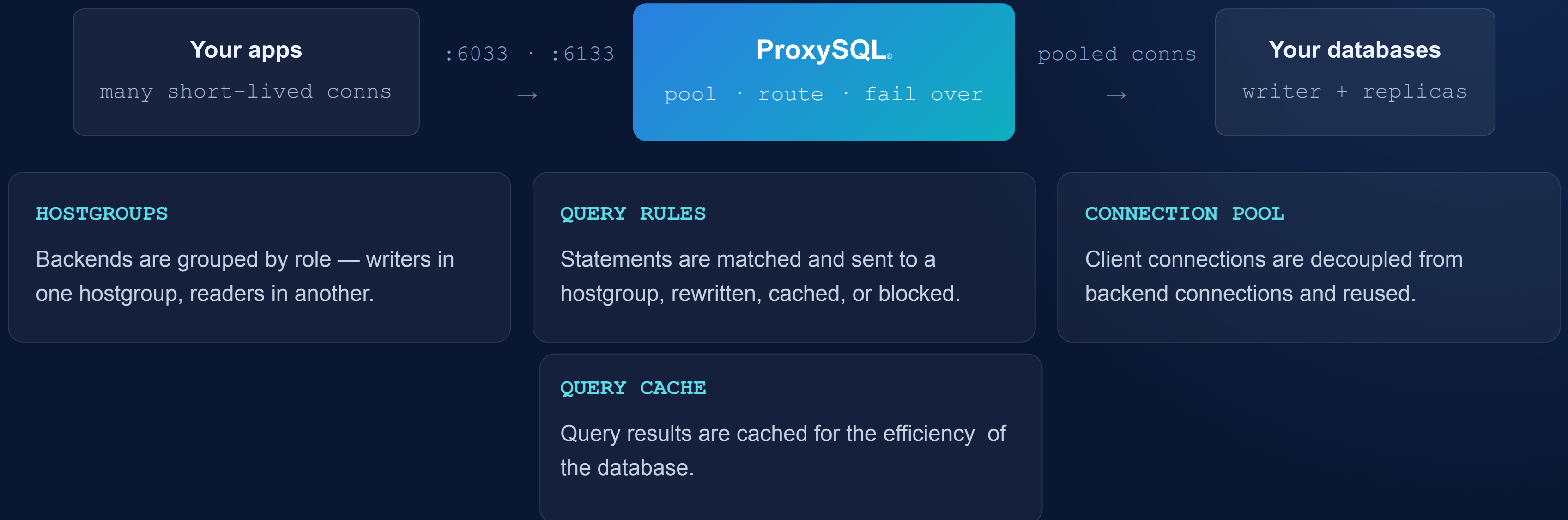
RELEASES

Three cumulative release tiers

3.0	Stable	PostgreSQL and MySQL core functionality.
3.1	Innovative	Stable, plus Fast Forward Traffic Observer, embedded time-series metrics, and newer authentication features.
4.0	AI / MCP	Innovative, plus AI services, MCP access for database agents, and the plugin platform.

TOPOLOGY

Where a proxy sits



FUNDAMENTALS

Connection pooling and multiplexing

A connection is idle for most of its life. ProxySQL holds the client side open and hands a backend connection to whichever session is actually running a transaction.

10,000

client connections at the proxy



200

backend connections in use

The database stops being sized for your application's connection count, and starts being sized for its concurrency.

```
proxysql_client_connections_connected  
proxysql_connpool_conns
```

Illustrative ratio — the real figure depends on transaction length and multiplexing eligibility.

FUNDAMENTALS

Query routing

```
-- read/write split, in two rules
INSERT INTO mysql_query_rules
(rule_id, match_digest, destination_hostgroup)
VALUES
(10, '^SELECT.*FOR UPDATE', 10),
(20, '^SELECT', 20);
-- hostgroup 10 = writer, 20 = readers
```

Ordered rules

Rules are evaluated by `rule_id` . The first match wins, so specific patterns go above general ones.

No application change

The split lives in the proxy, not in your ORM or your connection strings.

More than routing

The same rule table rewrites statements, caches results, sets timeouts, and blocks patterns you never want to see in production.

FUNDAMENTALS

Failover behaviour

The proxy monitors every backend. When one stops answering, it is shunned and traffic moves — your applications keep talking to the same endpoint.

01 · DETECT**Health checks**

Ping, connect, and replication-lag checks run continuously against every backend.

02 · SHUN**Take it out**

The failing host is removed from its hostgroup. Its connections are not reused.

03 · REROUTE**Same endpoint**

Queries go to the remaining backends. Clients see no DNS change and no new connection string.

04 · RETURN**Back in service**

Once checks pass again the host rejoins its hostgroup and starts taking traffic.

02

The BYOC model

We manage the service. You keep the data and the network.

WHY MANAGED

The cost of running it yourself

THE WORK

WHAT IT TURNS INTO

High availability

The proxy must not be the new single point of failure — so you run several, and keep their configuration identical.

Configuration drift

Runtime changes made during an incident outlive the incident. Nobody knows which node is authoritative.

Upgrades

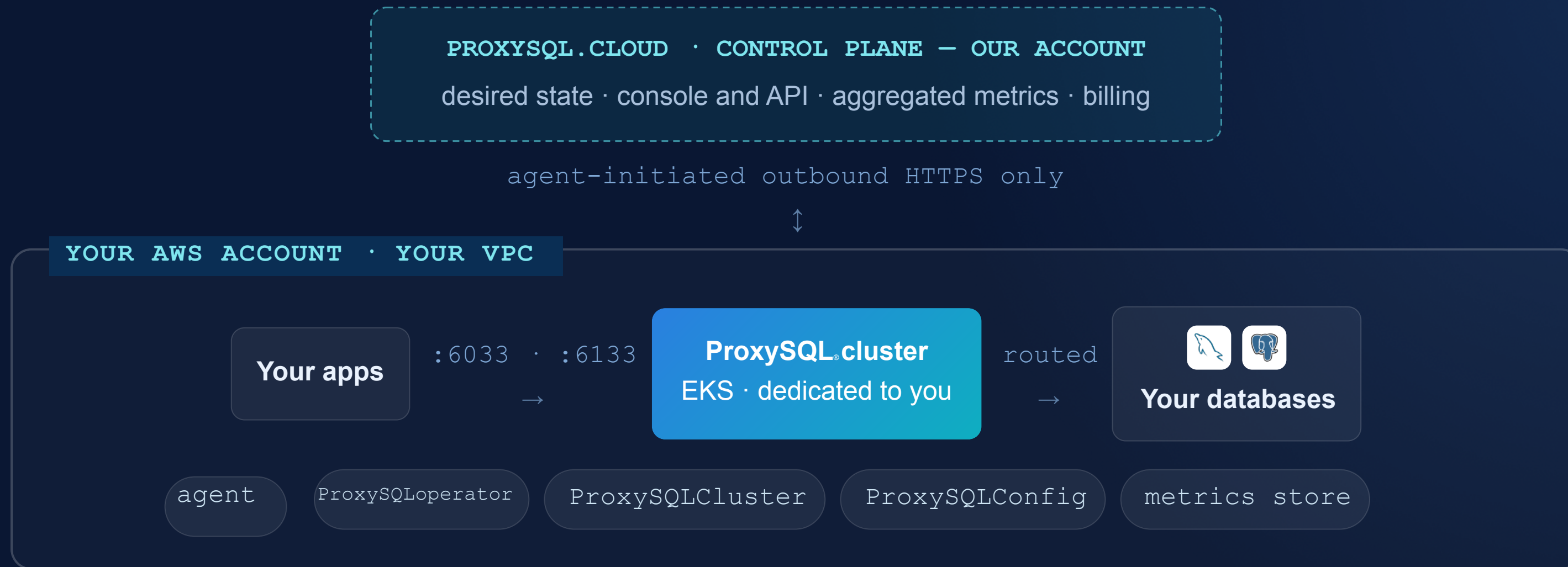
Every version bump is a change to the component every query passes through.

Observability

Exporters, dashboards, and alert rules — built once, then maintained by whoever is on call.

ARCHITECTURE

Two planes



DATA BOUNDARY

What crosses the boundary

LEAVES YOUR VPC

Cluster desired state — replica count, versions, hostgroups, query rules.

Lifecycle status and phase reported by the operator.

Aggregated metrics and query digests — shapes and counters, not values.

Audit events: who changed what, and when.

NEVER LEAVES YOUR VPC

Row data. Every query is served inside your network.

Database credentials and application secrets.

Query text with literals in it.

Backups — they are written to your own S3 bucket.

The agent's network posture

01

Outbound only

The agent opens the connection to us. You do not open a port, a VPN, or a peering link for the control plane.

02

Scoped identity

The agent runs with an IAM role limited to the resources it manages, and a registration token bound to one cluster.

03

Revocable by you

Remove the role or stop the agent and management stops. The proxies keep serving traffic on the state they already have.

The data plane keeps working when the control plane is unreachable. Management pauses; traffic does not.

03

Standing up a cluster

From an empty account to a serving endpoint.

PROVISIONING

Day 0: connecting your account

- 01 You apply a CloudFormation stack that creates a scoped IAM role for the agent.
- 02 You choose the VPC, subnets, and region the cluster will run in.
- 03 The agent installs into your Kubernetes cluster and registers itself.
- 04 You create your first cluster in the console — name, size, engine, backends.

```
# the cluster you asked for, as the operator sees it
kind: ProxySQLCluster
metadata:
  name: customer-acme-01
spec:
  replicas: 3
  engine: mysql
  version: "3.0"
  status:
  phase: Running
```

PROVISIONING

Cluster lifecycle phases

Every cluster reports exactly one phase, projected straight from the Kubernetes operator. The console, the API, and the alerting all use the same word.

● Pending

● Creating

● Running

● Updating

● Degraded

● Drifted

● Failed

● Paused

Degraded is not down

Two of three replicas ready still serves traffic. The phase tells you the fleet is thinner than you asked for.

Drifted is a warning about the future

Something changed the runtime out of band. It works now and will be reverted on the next reconcile.

PROVISIONING

Discovery and autoconfig

You give one reachable backend and a read-only user. Discovery walks the replication topology; autoconfig turns it into a proposed configuration you review before it applies.

01 · TEST**Test the backend**

Connectivity, credentials, and TLS are checked from inside your VPC before anything is saved.

02 · DISCOVER**Read the topology**

Writer and replicas are enumerated, with their roles and replication state.

03 · PROPOSE**Draft the config**

Hostgroups, monitor settings, and a starter read/write split are generated from what was found.

04 · REVIEW**You approve**

Nothing is applied automatically. You see the diff, edit it, then apply.

PROVISIONING

The reconcile loop

Nothing is applied by remote command. You write desired state; the loop closes it.



Config applied. 3/3 replicas synced, 0 drifted.

04

Day-2 operations

Config, credentials, maintenance, scaling, and backups.

DAY - 2

Configuration lifecycle

IMPORT

Bring your existing setup

Load the config from a self-managed ProxySQL and see it as a diff against the managed cluster before you commit.

EXPORT

Take it out again

The full configuration is exportable at any time. It is your config, in a portable form — no lock-in at the data layer.

CAPTURE

Read the live runtime

Capture what a pod is actually running and compare it with desired state. This is how drift is found rather than guessed.

HISTORY

Every version, kept

Each apply creates a numbered `spec_version` with an author and a timestamp. Roll back to any of them.

DAY - 2

Managed app credentials

An application login has two halves — the credential your app presents to the proxy, and the credential the proxy presents to the database. Managed credentials provision, rotate, and revoke both together.

PROXY SIDE

What your app uses

A user in the proxy's own table, scoped to a default hostgroup and a transaction persistence policy.

BACKEND SIDE

What the proxy uses

The matching database grant, so the pooled connection can actually authenticate downstream.

Both halves live in your account. The control plane stores which credentials exist and who revealed them — never the secret itself.

DAY - 2

Drain, pause, windows, and upgrades

01

Drain

A backend is set `OFFLINE_SOFT` .
Existing transactions finish; no new ones are sent. Then it is safe to touch.

02

Pause and resume

A cluster can be paused — reconciliation stops and the phase says so, which is what you want during a planned database migration.

03

Maintenance windows

Recurring windows in your timezone, in which a drain is executed and the backend is returned to service afterwards.

04

Windowed upgrade

A ProxySQL version bump is scheduled into a window and rolled one replica at a time. Health checks gate each step; a failed check halts the roll and leaves the remaining replicas on the old version.

```
Window closed. hostgroup 20 · host db-r2 back ONLINE.
```

DAY-2 OPERATIONS

ProxySQL upgrades, one replica at a time

3.0.10 → 3.0.11 · Stable tier · 3 replicas

01

Pick the version

Choose the target release per cluster, within the tier you run: Stable, Innovative, or AI / MCP.

02

Schedule a window

The upgrade waits for a maintenance window in your timezone, or starts now if you choose.

03

Drain and replace

One replica is drained, replaced on the new version, and returned to service. Traffic moves to its peers meanwhile.

04

Health gates the next

Each replica must pass health checks before the next one starts. A failed check halts the roll.

A bad version reaches one replica, not the fleet. The rest stay on the version they were running.

DAY - 2

Autoscaling

A scale policy watches six signals and adjusts replica count within bounds you set.

client connections

queries per second

connection pool utilization

cpu

memory

query latency

Bounded

Minimum and maximum replica counts are yours.
The policy never leaves that range.

Damped

A cooldown between decisions keeps a spiky
workload from flapping the fleet.

Legible

Every decision is recorded with the signal that
triggered it, so a scale-up is explainable after the
fact.

DAY - 2

Backup and restore

Scheduled, into your bucket

A policy sets frequency and retention. Dumps are written to S3 in your account, under your encryption keys.

Restore to a point in the history

Restoring is applying an earlier configuration version through the same reconcile loop as any other change.

Drills, not assumptions

A restore drill exercises the path without touching the live cluster, so recovery is something you have done before you need it.

WHAT IS IN SCOPE

Proxy configuration: users, hostgroups, query rules, variables, and the version history behind them.

WHAT IS NOT

Your database contents. Those stay with your existing database backup strategy — we do not copy rows.

05

Observability

Every query passes through one place. That is a good place to measure.

OBSERVABILITY

Metrics on the fleet

CLIENT CONNECTIONS

8,412

connected across 3 replicas

BACKEND CONNS

196

in the pool, 44 free

MULTIPLEXING

43×

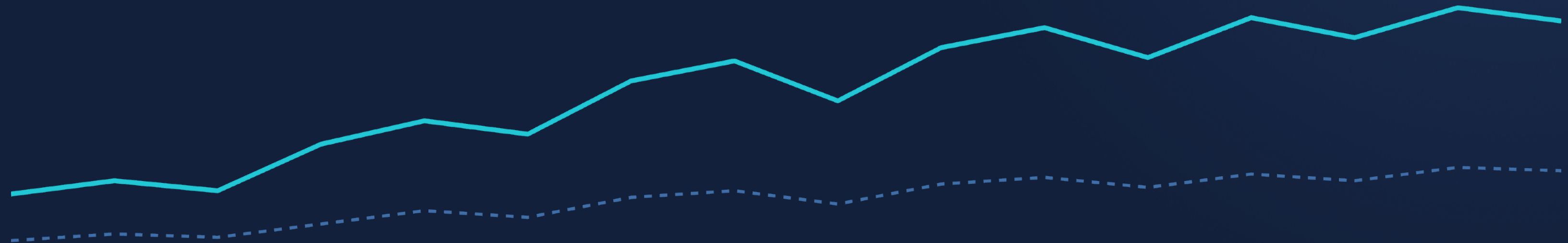
clients per backend conn

P99 LATENCY

3.1_{ms}

added at the proxy hop

QUERIES PER SECOND · LAST 6 HOURS



Sample values from a demo cluster. Solid line: total queries. Dashed: writes.

OBSERVABILITY

Traffic analytics

The proxy already groups traffic by query digest — the statement shape with literals removed. That is what we collect, so the values never leave your VPC.

DIGEST	CALLS	P95	HOSTGROUP
SELECT * FROM orders WHERE user_id = ?	4.2M	1.8 ms	20
SELECT COUNT(*) FROM events WHERE ts > ?	118K	412 ms	20
UPDATE sessions SET last_seen = ? WHERE id = ?	2.9M	2.4 ms	10
SELECT * FROM users WHERE email = ?	861K	0.9 ms	20

Sample values from a demo cluster.

OBSERVABILITY

Activity and audit trail

Cluster logs, on request

Ask for a pod's logs from the console. The agent collects them inside your account and returns the window you asked for.

Activity, always recorded

Applies, scale decisions, credential changes, drains, restores — each with an actor and a timestamp.

Same events, your tooling

Metrics are Prometheus-shaped and dashboards are Grafana-shaped, so this fits the observability stack you already run.

```
14:02:11 config applied spec_version 41
14:02:14 replica sync 3/3
15:31:02 scale up 3 → 4 conn util 82%
18:00:00 window opened db-r2 OFFLINE_SOFT
18:14:37 window closed db-r2 ONLINE
21:45:09 credential rotated app_orders
```

06

Adoption

What it takes to get from this talk to traffic.

ADOPTION

What you provide, what we run

YOU PROVIDE

An AWS account, a VPC, and a Kubernetes cluster to host the data plane.

A scoped IAM role for the agent, created by our CloudFormation stack.

Network reachability from that cluster to your databases.

A read-only user for discovery, and your routing intent.

WE RUN

Provisioning, reconciliation, and version upgrades of the proxy fleet.

The console, API, configuration history, and audit trail.

Metrics collection, dashboards, and alert evaluation.

On-call for the platform, and the upgrade path for ProxySQL itself.

ADOPTION

Rollout in three steps

01

Connect and discover

Apply the stack, install the agent, point discovery at one backend. Nothing is serving traffic yet.

02

Shadow one service

Move a single read-heavy service to the proxy endpoint and compare its latency and error rate against the direct path.

03

Cut over, service by service

Repeat per service, then enable pooling limits, routing rules, and autoscaling once traffic is settled.

At every step the rollback is a connection string. That is what makes this adoptable in a regulated environment.

Health, throughput, and cost on one screen

Health, throughput, backend pool, top queries, and 24-hour cost for a single cluster.

Dashboard

Clusters

Deployments

API Keys

Profile

ORGANIZATION

Teams

Members

Security log

Audit log

Settings

CLUSTER

mysql-bra-26

Running

Account: Managed in Your Account

Updated 10s ago

Refresh

Pause

Upgrade

Discover backends

Overview

Activity

Configuration

Credentials

Backups

Analytics

Alerts

Autoscaling

Settings

HEALTH

healthy

4/4 backends up · 0 alerts firing

THROUGHPUT

0 q/s

PROXYSQL VALUE

RW split 0/100

Multiplexing 0.0:1

Queries routed 0

TOP QUERIES

Slow rate 0.0%

SELECT c FROM sbtest1 WHERE id=? · 57,22...

COMMIT · 5.723x

BACKEND POOL

hg10: 1/1 up · worst 0.9ms

hg20: 3/3 up · worst 1.0ms

COST / USAGE

50 instance-hours (24h)

Est. 0.00 USD

Every phase change, recorded

Phase transitions and agent-reported updates, timestamped, in the order they happened.

☰

ProxySQL.Cloud

Managed in Your Account

☀

Logout

Dashboard

Clusters

Deployments

API Keys

Profile

ORGANIZATION

Teams

Members

Security log

Audit log

Settings

CLUSTER

mysql-bra-26

● Running

Account: Managed in Your Account

Updated 30s ago

Refresh

Pause

Upgrade

Discover backends

Overview

Activity

Configuration

Credentials

Backups

Analytics

Alerts

Autoscaling

Settings

9/22/2026, 1:52:46 PM

updated

via agent

Reached expected number of succeeded pods

9/22/2026, 1:46:46 PM

updated

via agent

load running

9/22/2026, 1:41:48 PM

● Creating

→

● Running

via agent

no rollout in progress

9/22/2026, 1:40:48 PM

● Pending

→

● Creating

via agent

9/22/2026, 1:31:45 PM

updated

via agent

Variables, topology, and query rules in one editor

Override any variable from the catalog. Running and default values sit beside the override; Validate dry-runs before Apply.

☰

ProxySQL.Cloud

Managed in Your Account

☀

Logout

Dashboard

Clusters

Deployments

API Keys

Profile

ORGANIZATION

Teams

Members

Security log

Audit log

Settings

CLUSTER

mysql-bra-26

● Running

Account: Managed in Your Account

Updated 25s ago

Refresh

Pause

Upgrade

Discover backends

Overview

Activity

Configuration

Credentials

Backups

Analytics

Alerts

Autoscaling

Settings

Configuration

Import config

Restore ▾

Export config

Edit ProxySQL variables from the catalog. Use Topology for backends and Query rules for routing; YAML provides the complete document. Validate runs a dry-run; Apply persists and reloads.

► Configuration reference and examples

Variables

Topology

Query rules

YAML

Search variables...

All families ▾

☐ Modified only

59 of 59

Scheduled and on-demand backups

Restore in place or to a new cluster. Credentials are re-provisioned on restore.

Dashboard

Clusters

Deployments

API Keys

Profile

ORGANIZATION

Teams

Members

Security log

Audit log

Settings

CLUSTER

mysql-bra-26

Running

Account: Managed in Your Account

Updated 10s ago

Refresh

Pause

Upgrade

Discover backends

Overview

Activity

Configuration

Credentials

Backups

Analytics

Alerts

Autoscaling

Settings

Backups

On-demand backups of this cluster's configuration and shape. Credentials are re-provisioned on restore.

Back up now

Name	Created	Kind	Size	Actions		
scheduled-2026-09-23T11:04:53Z	9/23/2026, 2:04:53 PM	scheduled	1.6 KiB	Restore	Restore to new cluster	Delete
scheduled-2026-09-22T11:04:53Z	9/22/2026, 2:04:53 PM	scheduled	1.6 KiB	Restore	Restore to new cluster	Delete
manual-cls_8db2fc88-37ae-4f5b-971d-76575d39f418	9/22/2026, 1:29:23 PM	manual	1.6 KiB	Restore	Restore to new cluster	Delete

Proxy-level metrics, refreshed live

Questions, slow queries, transactions, and connection counts, with ranges from 1 hour to 14 days.

☰

ProxySQL.Cloud

Managed in Your Account



Logout

Dashboard

Clusters

Deployments

API Keys

Profile

ORGANIZATION

Teams

Members

Security log

Audit log

Settings

CLUSTER

mysql-bra-26

Running

Account: Managed in Your Account

Updated 1m ago

Refresh

Pause

Upgrade

Discover backends

Overview

Activity

Configuration

Credentials

Backups

Analytics

Alerts

Autoscaling

Settings

Auto-refresh

10s

Refresh

QUESTIONS

114,464

CLIENT CONNECTIONS

SLOW QUERIES

1

SERVER CONNECTIONS

ACTIVE TRANSACTIONS

0

ABORTED CLIENT CONNS

WHERE TO START

One endpoint for your apps. Everything else stays in your account.

READ

The BYOC deployment guide

IAM scope, network requirements, and the agent's install manifest.

TRY

A discovery run

Point it at a staging replica and read the proposed configuration.

ASK

For a security review pack

The boundary diagram and data-flow inventory, ready for your reviewers.



ProxySQL.Cloud® Questions.



Sign in to your account

Email

alkin@proxysql.com

Password

.....

[Forgot password?](#)

Sign in

[Don't have an account? Sign up](#)



Ask us anything about your data tier.

We will run a discovery pass against a staging replica and show you the configuration it proposes.

Managed · BYOC · BYOK



proxysql.com/saas