Database reliability engineering for MySQL

Matthias Crauwels

ConFoo 2022 - Online

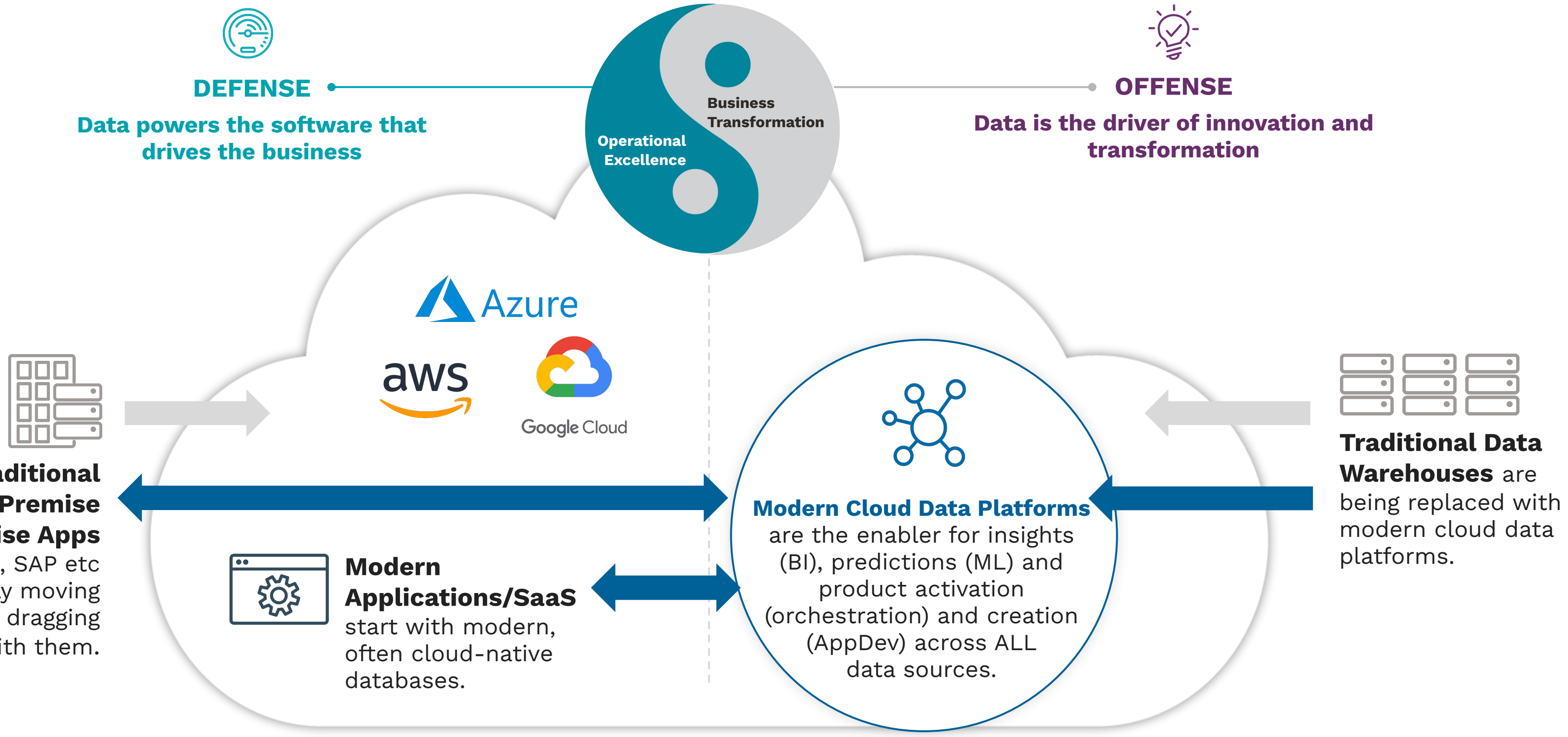
Wed Feb 23rd 2022

Pythian

Speaker

Matthias Crauwels
Principal Consultant
Pythian - OSDB

How the data estate is evolving



DEFENSE

Data powers the software that drives the business

OFFENSE

Data is the driver of innovation and transformation

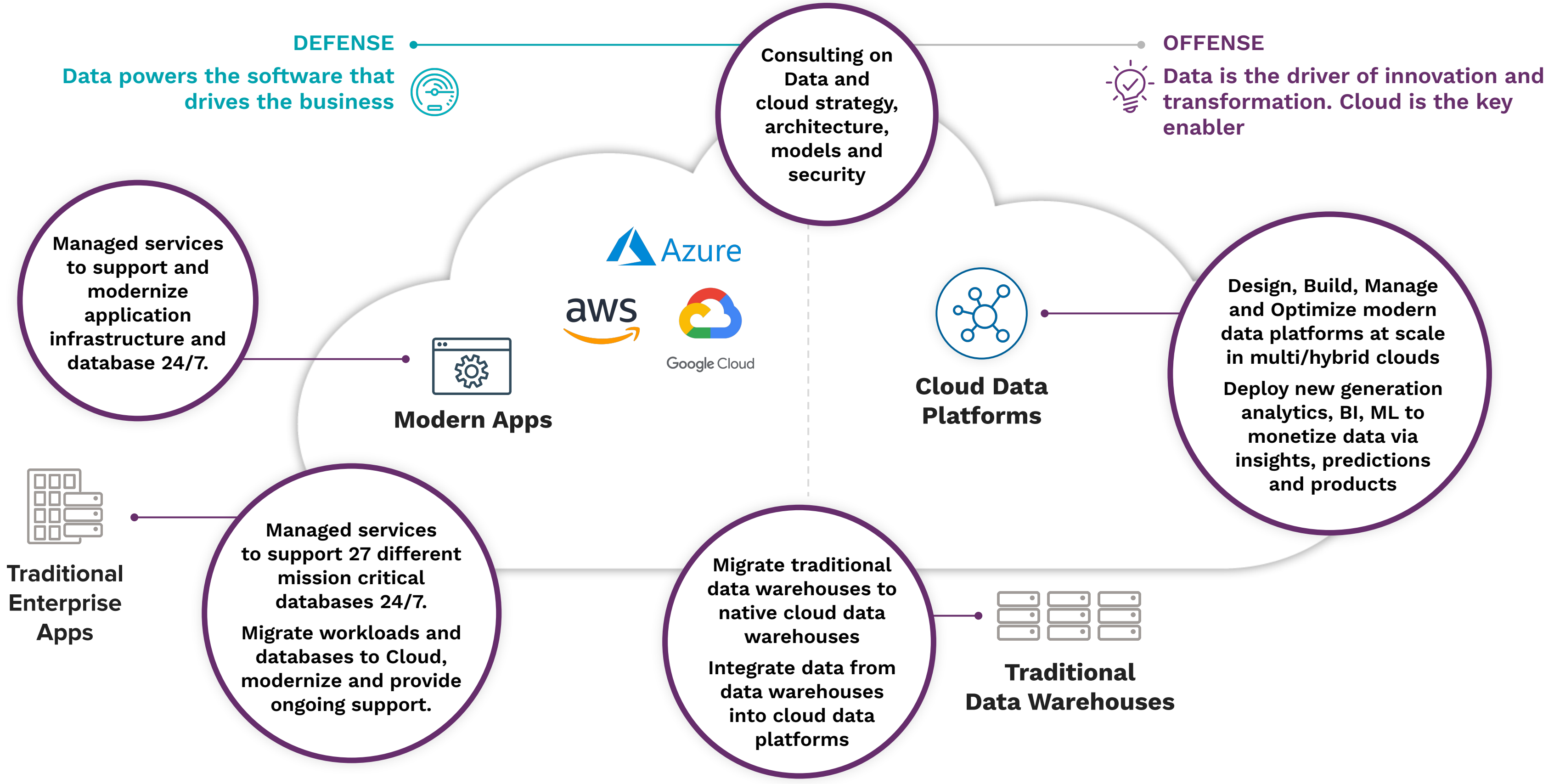
Traditional On Premise Enterprise Apps i.e. Oracle, SAP etc slowly moving to Cloud, dragging data with them.

Modern Applications/SaaS start with modern, often cloud-native databases.

Modern Cloud Data Platforms are the enabler for insights (BI), predictions (ML) and product activation (orchestration) and creation (AppDev) across ALL data sources.

Traditional Data Warehouses are being replaced with modern cloud data platforms.

Pythian's Services Across the Data Estate


DEFENSE
Data powers the software that drives the business
OFFENSE
Data is the driver of innovation and transformation. Cloud is the key enabler
Consulting on Data and cloud strategy, architecture, models and security
Managed services to support and modernize application infrastructure and database 24/7.
Azure
aws
Google Cloud
Modern Apps
Cloud Data Platforms
Design, Build, Manage and Optimize modern data platforms at scale in multi/hybrid clouds
Deploy new generation analytics, BI, ML to monetize data via insights, predictions and products
Traditional Enterprise Apps
Managed services to support 27 different mission critical databases 24/7.
Migrate workloads and databases to Cloud, modernize and provide ongoing support.
Migrate traditional data warehouses to native cloud data warehouses
Integrate data from data warehouses into cloud data platforms
Traditional Data Warehouses

Other presentation

- Tomorrow at 10:30 I have another presentation

  Getting started with InnoDB Cluster in MySQL 8

  Since MySQL 5.7 InnoDB Cluster saw the light. Back then it was not as popular because it was considered too new technology. As with any new product, there were numerous bug reports. With MySQL 8.0 the solution has matured a lot, many of the problems have been addressed and so it has became a full scale high availability solution for MySQL.

AGENDA

- (High) Availability
- Service Discovery
- Observability
- Disaster Recovery

(High) Availability

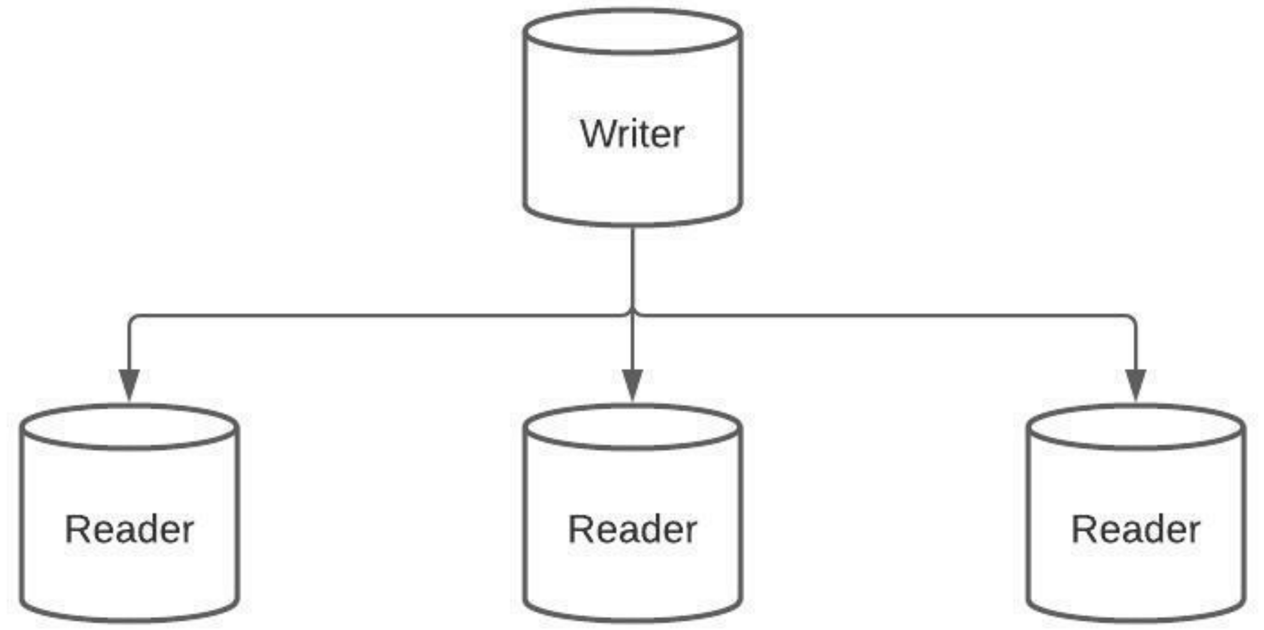
Availability for MySQL

- **When do we consider MySQL available?**
- **We need:**
  - **Just one server to write transactions to**
  - **At least one server to read transactions from**

Availability vs High Availability

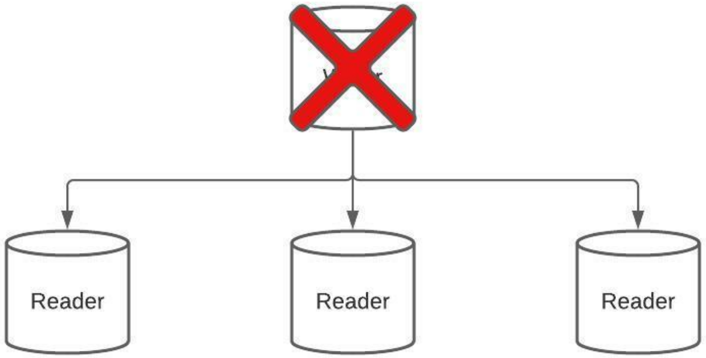
Availability vs High Availability

High Availability for MySQL



MySQL - Single point of failure

- The "writer" is the single-point-of-failure in this topology
- No more writes can happen
- Application will likely be down or at least degraded
- We shall introduce a topology management tool to handle these failures.



Orchestrator

Orchestrator is a High Availability and replication management tool.

It can be used for:

- Discovery of a topology
- Visualisation of a topology
- Refactoring of a topology
- Recovery of a topology

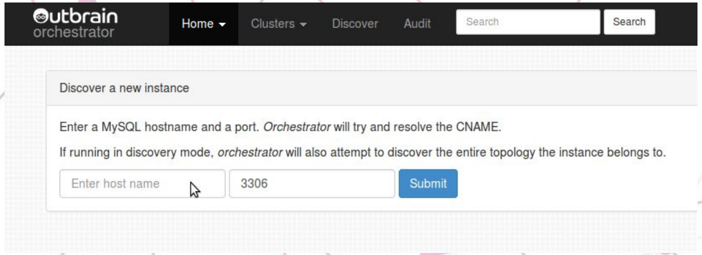
orchestrator

Orchestrator: Discovery

Orchestrator can (and will) discover your entire replication topology as soon as you connect it to a single server in the topology.

It will use regular DBA commands such as: `SHOW SLAVE HOSTS`, `SHOW PROCESSLIST`, `SHOW SLAVE STATUS` to try and connect to the other servers in the topology.

Requirement: the `orchestrator_topology_user` needs to be created on every server in the cluster so it can connect.



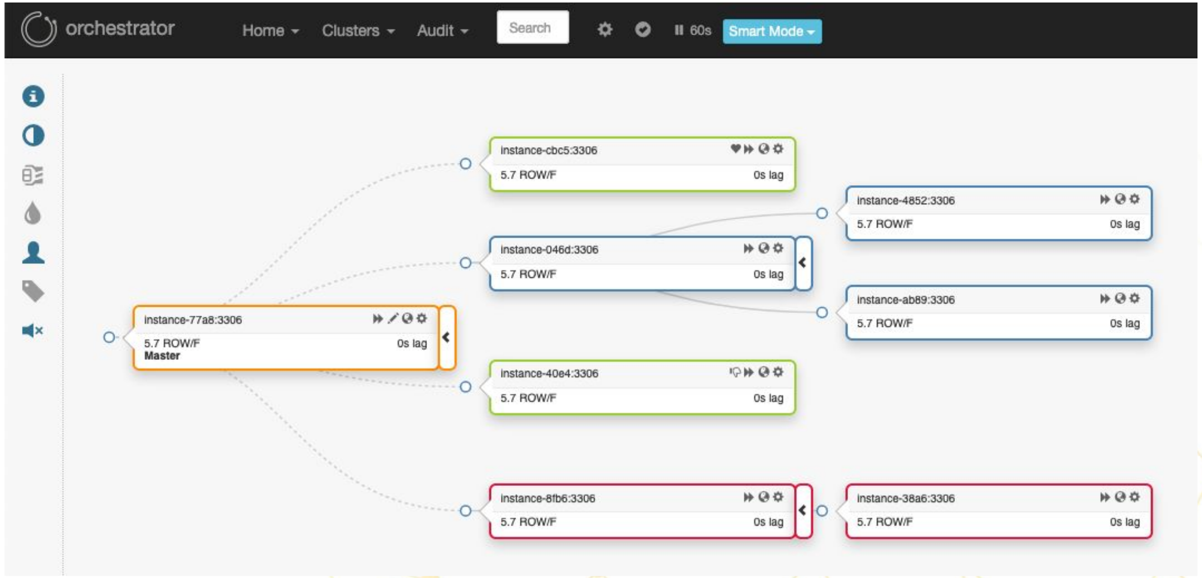
Discover a new instance

Enter a MySQL hostname and a port. *Orchestrator* will try and resolve the CNAME.

If running in discovery mode, *orchestrator* will also attempt to discover the entire topology the instance belongs to.

Orchestrator: Visualization

Orchestrator comes with a web interface that visualizes the servers in the topology.



Orchestrator: Refactoring

Orchestrator can be used to refactor the topology.

This can be done from the **command line tool**, via the **API** or even via the web interface by **dragging and dropping**.

You can do things like

- Repoint a replica to a new master
- Promote a server to a (co-)master
- Start / Stop replica
- ...

Orchestrator: Recovery

All of these features are nice, but they still require a human to execute them. This doesn’t help you much when your master goes down at 3AM and you get paged to resolve this.

Orchestrator can be configured to automatically recover your topology from an outage.

Orchestrator: How recovery works?

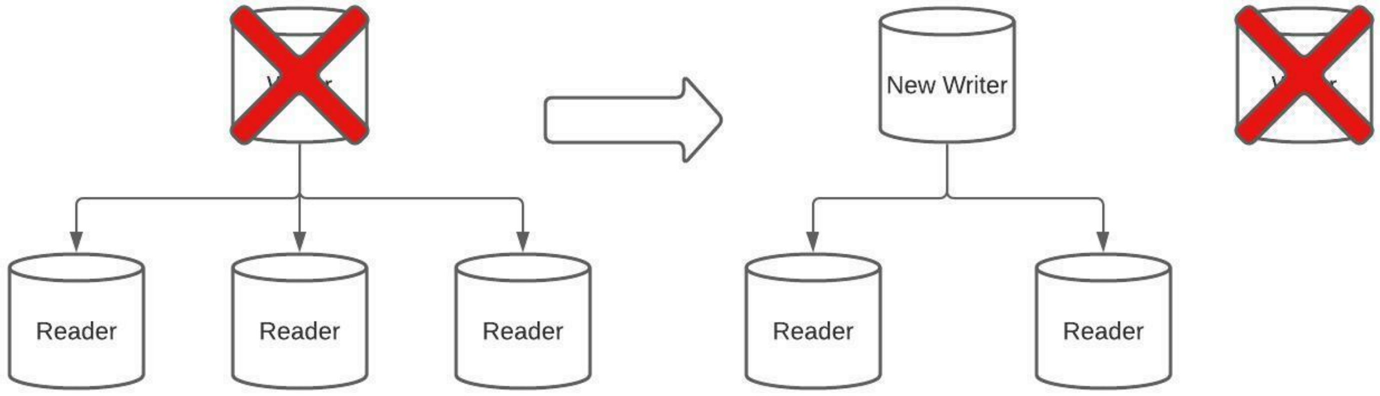
To be able to perform a recovery, Orchestrator first needs to detect a failure.

Typical monitoring tools (think nagios) will probe the master, but what to do on failure? Take immediate action? Retry? How many retries?

As indicated before Orchestrator connects to every server in the topology and gathers information from each of the instances.

Orchestrator uses this information to make decisions on the best action to take. They call this the holistic approach.

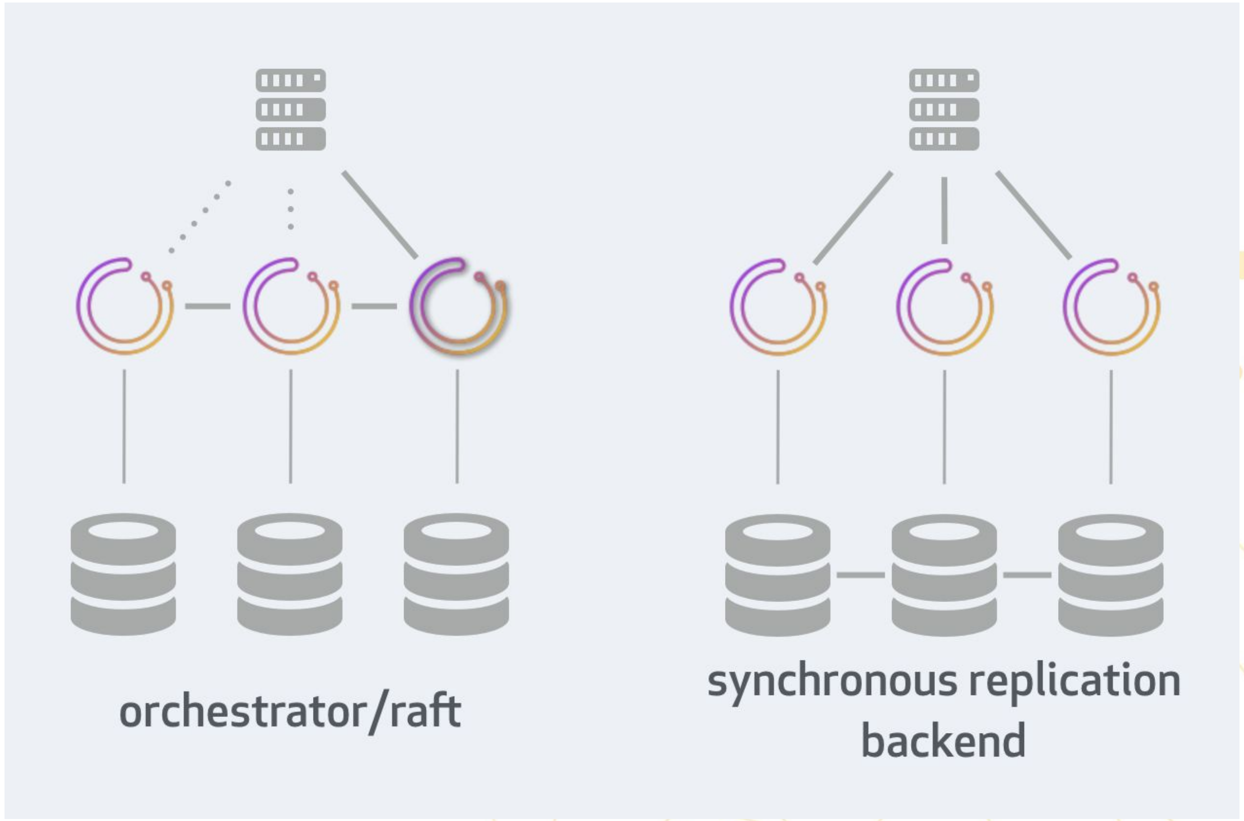
MySQL HA - back to our example



Orchestrator/raft

- Since Orchestrator 3.x
- Orchestrator backend db becomes standalone
  - requirement for MySQL as backend db was dropped (but still available)
  - introduced sqlite as possible backend db
- Implements RAFT consensus protocol, responsible for
  - leader election
  - distribution of data
- Minimal 3 nodes for a HA setup (50% + 1 node quorum)

Orchestrator shared db vs raft



Service Discovery

What is service discovery?

- Know where the writer server is
- Know where the reader server is
- Several options
  - Application deploy?
  - DNS change?
  - Floating Virtual IP?
  - Proxy-server?

ProxySQL: What?

ProxySQL is a high performance layer 7 proxy application for MySQL.

- It provides ‘intelligent’ load balancing of application requests onto multiple databases
- It understands the MySQL traffic that passes through it, and can split reads from writes.
- It understands the underlying database topology, whether the instances are up or down
- It shields applications from the complexity of the underlying database topology, as well as any changes to it
- ...

ProxySQL: Terminology

- Hostgroup

  All backend MySQL servers are grouped into hostgroups. These “hostgroups” will be used for query routing.

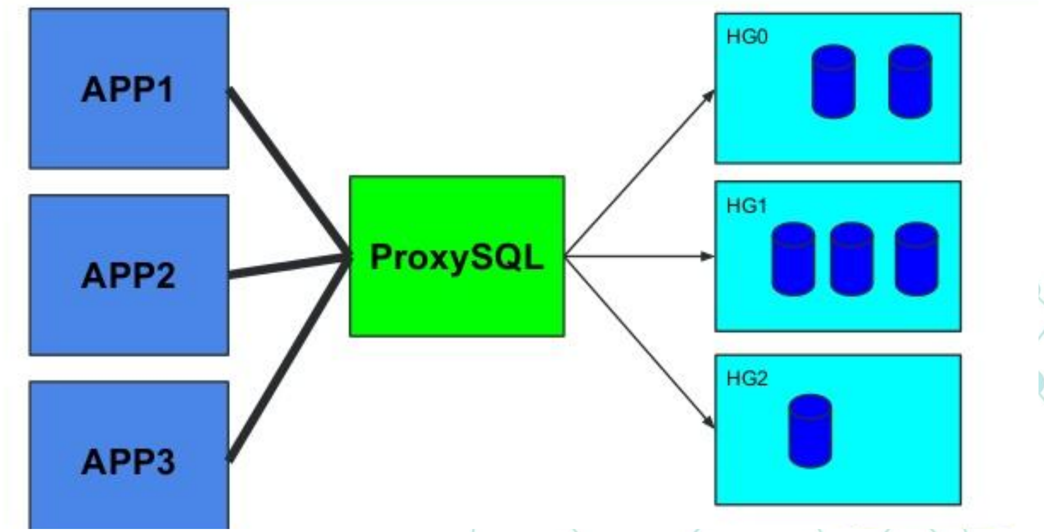
- Query rules

  Query rules are used for routing, mirroring, rewriting or blocking queries. They are at the heart of ProxySQL’s functionalities

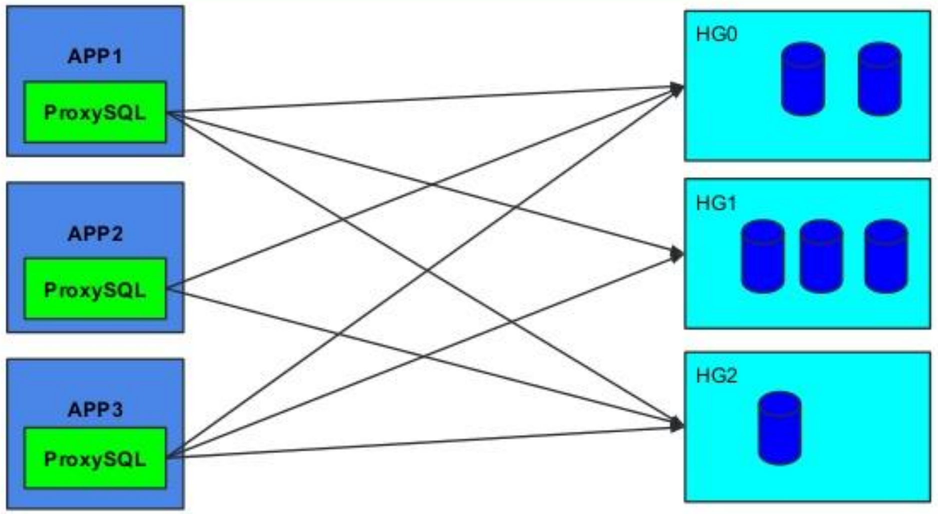
- MySQL users and servers

  These are configuration items which the proxy uses to operate

ProxySQL: Basic design (1)



ProxySQL: Basic design (2)



Observability

What is observability

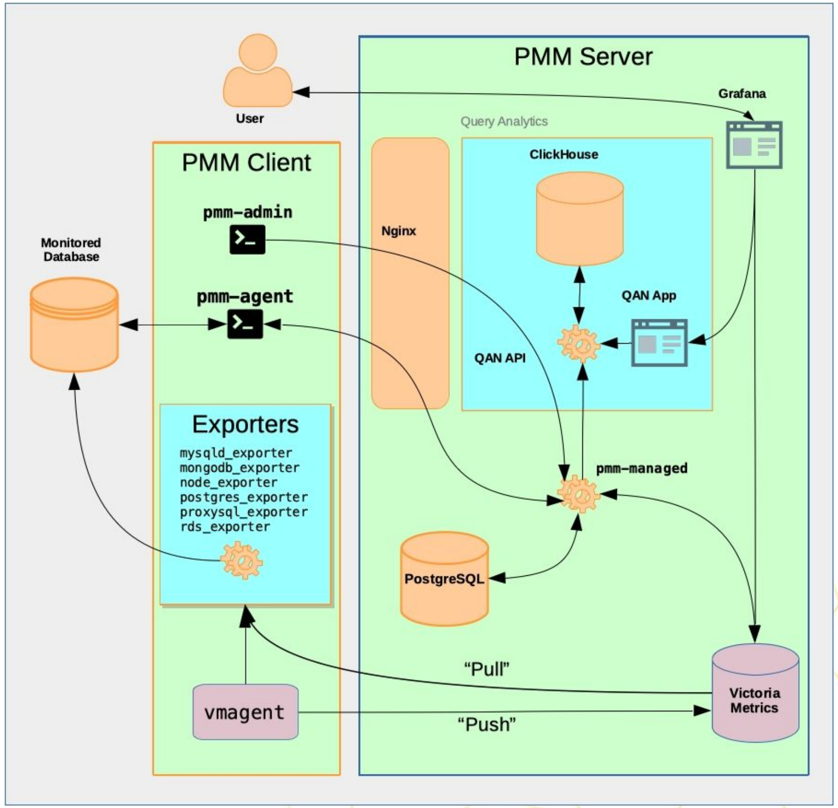
- Know if your systems have an issue (alerting)
  - Pager
  - IM
  - Email
- Troubleshoot performance and/or diagnose issue (graphs)
  - dashboards
  - comparison to a past situation

Percona Monitoring and Management (PMM)

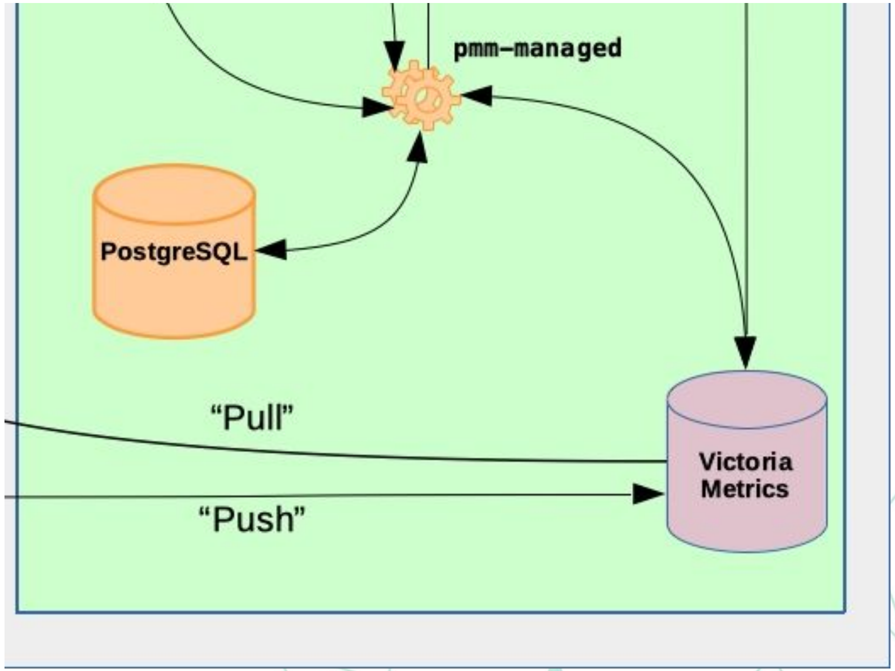
What is PMM?

- All-in-one solution for Observability, focussed on databases
- Free and open source
- Out-of-the-box solutions for
  - MySQL
  - PostgreSQL
  - MongoDB
  - ProxySQL
- Available as
  - docker container
  - OVF (VM image)
  - AMI (AWS)

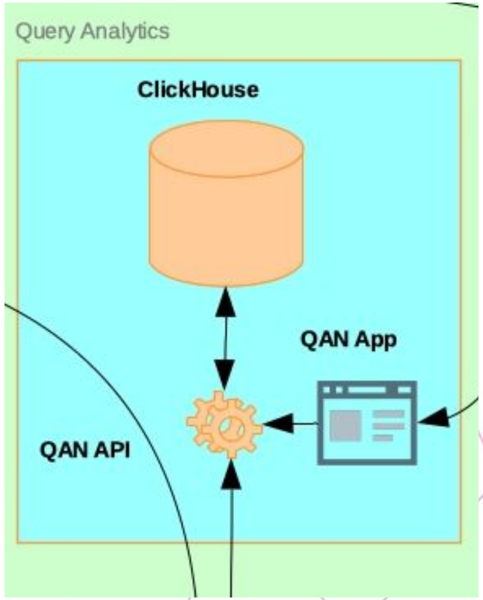
PMM: Components



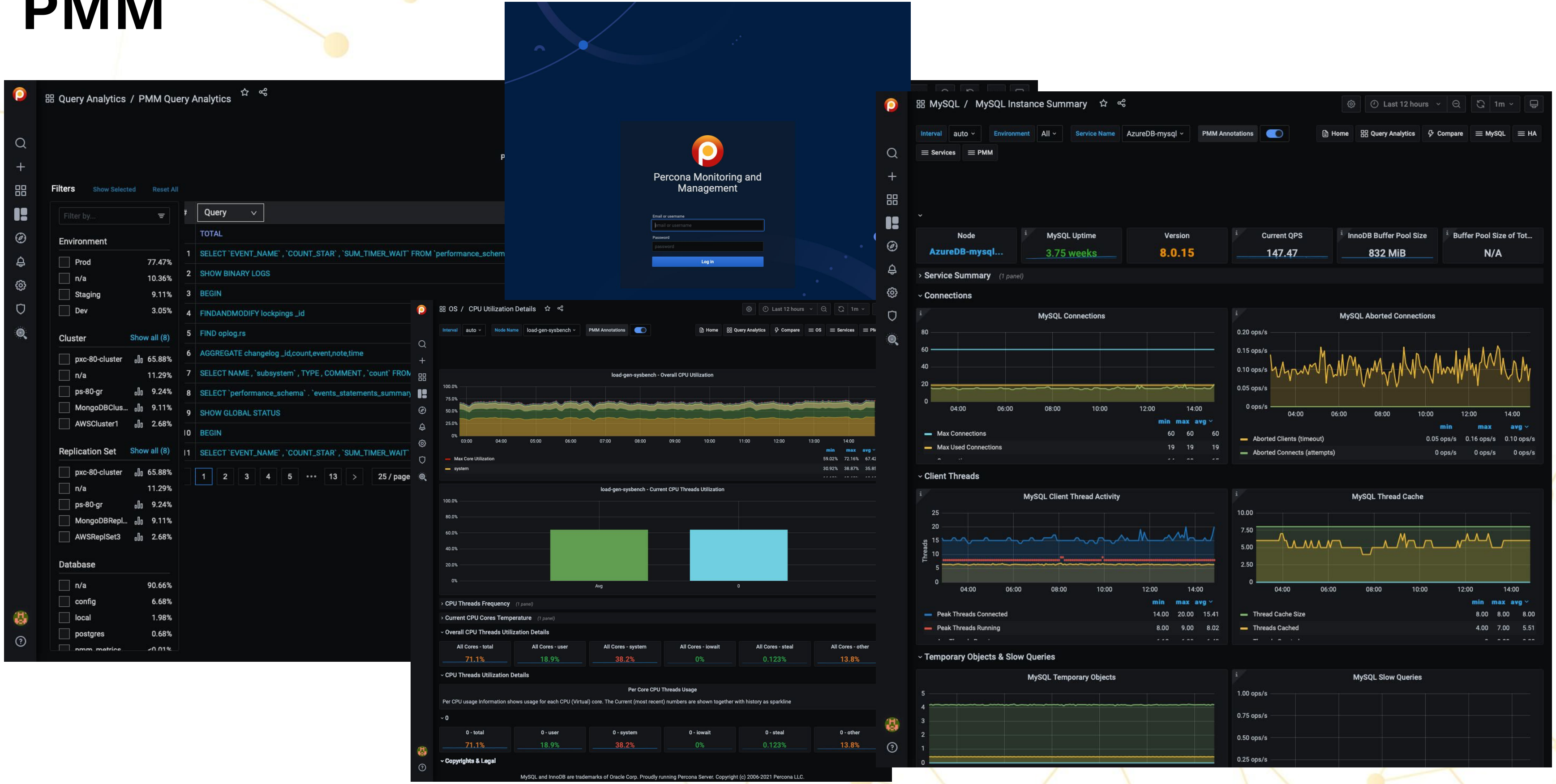
PMM: Components



PMM: Components



PMM



Query Analytics / PMM Query Analytics

Filters · Show Selected · Reset All

Filter by...

Environment

Prod	77.47%
n/a	10.36%
Staging	9.11%
Dev	3.05%

Cluster · Show all (8)

pxc-80-cluster	65.88%
n/a	11.29%
ps-80-gr	9.24%
MongoDBClus...	9.11%
AWSCluster1	2.68%

Replication Set · Show all (8)

pxc-80-cluster	65.88%
n/a	11.29%
ps-80-gr	9.24%
MongoDBRepl...	9.11%
AWSReplSet3	2.68%

Database

n/a	90.66%
config	6.68%
local	1.98%
postgres	0.68%

Query

#	Query
	TOTAL
1	SELECT `EVENT_NAME` , `COUNT_STAR` , `SUM_TIMER_WAIT` FROM `performance_schem
2	SHOW BINARY LOGS
3	BEGIN
4	FINDANDMODIFY lockpings _id
5	FIND oplog.rs
6	AGGREGATE changelog _id,count,event,note,time
7	SELECT NAME , `subsystem` , TYPE , COMMENT , `count` FROM
8	SELECT `performance_schema` . `events_statements_summary
9	SHOW GLOBAL STATUS
10	BEGIN
11	SELECT `EVENT_NAME` , `COUNT_STAR` , `SUM_TIMER_WAIT`

1 2 3 4 5 ••• 13 > 25 / page

Percona Monitoring and Management

Email or username

Password

Log in

OS / CPU Utilization Details

Last 12 hours · 1m

Interval: auto · Node Name: load-gen-sysbench · PMM Annotations

Home · Query Analytics · Compare · OS · Services

load-gen-sysbench - Overall CPU Utilization

	min	max	avg
Max Core Utilization	59.02%	72.16%	67.42
system	30.92%	38.87%	35.85

load-gen-sysbench - Current CPU Threads Utilization

› CPU Threads Frequency (1 panel)

› Current CPU Cores Temperature (1 panel)

⌄ Overall CPU Threads Utilization Details

All Cores - total	All Cores - user	All Cores - system	All Cores - iowait	All Cores - steal	All Cores - other
71.1%	18.9%	38.2%	0%	0.123%	13.8%

⌄ CPU Threads Utilization Details

Per Core CPU Threads Usage

Per CPU usage Information shows usage for each CPU (Virtual) core. The Current (most recent) numbers are shown together with history as sparkline

⌄ 0

0 - total	0 - user	0 - system	0 - iowait	0 - steal	0 - other
71.1%	18.9%	38.2%	0%	0.123%	13.8%

⌄ Copyrights & Legal

MySQL and InnoDB are trademarks of Oracle Corp. Proudly running Percona Server. Copyright (c) 2006-2021 Percona LLC.

MySQL / MySQL Instance Summary

Last 12 hours · 1m

Interval: auto · Environment: All · Service Name: AzureDB-mysql · PMM Annotations

Home · Query Analytics · Compare · MySQL · HA · Services · PMM

Node	MySQL Uptime	Version	Current QPS	InnoDB Buffer Pool Size	Buffer Pool Size of Tot...
AzureDB-mysql...	3.75 weeks	8.0.15	147.47	832 MiB	N/A

› Service Summary (1 panel)

⌄ Connections

MySQL Connections

	min	max	avg
Max Connections	60	60	60
Max Used Connections	19	19	19

MySQL Aborted Connections

	min	max	avg
Aborted Clients (timeout)	0.05 ops/s	0.16 ops/s	0.10 ops/s
Aborted Connects (attempts)	0 ops/s	0 ops/s	0 ops/s

⌄ Client Threads

MySQL Client Thread Activity

	min	max	avg
Peak Threads Connected	14.00	20.00	15.41
Peak Threads Running	8.00	9.00	8.02

MySQL Thread Cache

	min	max	avg
Thread Cache Size	8.00	8.00	8.00
Threads Cached	4.00	7.00	5.51

⌄ Temporary Objects & Slow Queries

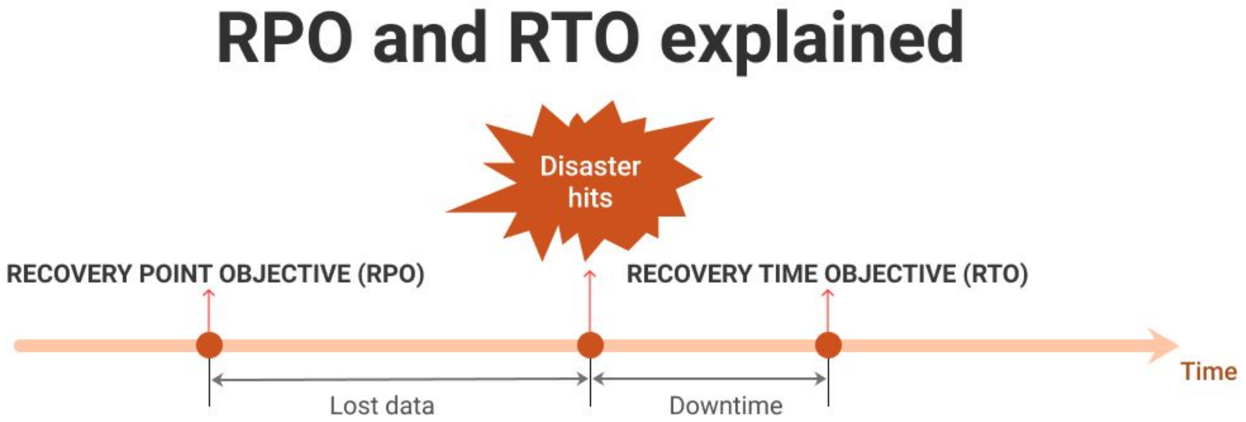
MySQL Temporary Objects

MySQL Slow Queries

Disaster Recovery

What is Disaster Recovery (DR)

- The ability to recover from a catastrophic event or a human error
- RTO vs RPO



Backups

- Fastest RTO is usually achieved with disk snapshots
- How to make consistent snapshot?
    - either stop MySQL on a replica cleanly (`innodb_fast_shutdown = 0`)
    - run a `FLUSH TABLES WITH READ LOCK`+ freeze the filesystem
- Both options block normal operations, so they should be ran on a designated replica.

Backups - Percona Xtrabackup

- supports online, non blocking backups for InnoDB
- full backups
- incremental backups
- physical backup
- make sure to match the Xtrabackup version that corresponds with your MySQL version.

Point-in-time recovery

- use the MySQL binary logs to recover your environment to a specific point-in-time
- if done right this can reduce your RPO to 0
- stream your binary logs to a backup (DR) location
- doing PITR will increase your RTO

Conclusion

Conclusion

- MySQL replication is required for High Availability
- Orchestrator can automate your writer-availability
- ProxySQL will help with Service Discovery
- PMM will expose the environment metrics for observability
- Backups are important for Disaster Recovery

Questions?

Thank you!

crauwels@pythian.com
@mcrauwel