Building DIY MySQL as Service
Who Am I

Sachin Manpathak

- Engineering Lead @Platform9
- Focus on Kubernetes as Platform.
- Alumni: VMware, UT Austin
About Platform9

Mission
Accelerate enterprise hybrid clouds with a SaaS cloud platform that just works

Key Metrics
- 300 cloud regions managed globally
- 500,000 cores of compute under management

Investors
- Menlo
- Canvas Ventures
- Redpoint Ventures
- Hewlett Pack Enterprise

Key Customers
- Autodesk
- Snapfish
- EBSCO
- Rodan + Fields
- Providence Health Care
- Cadence
- Aruba
- Veritas
Agenda

1. Context
2. Desired feature-set
3. Why Kubernetes?
4. Solution
5. Challenges
6. TODOs
7. References
Context

- Platform9 move from Public cloud to on-prem
- Dependency on managed SQL service
Desired Feature Set

**MUST HAVE**
- Drop in Replacement
- Automation
- Availability
- Disaster Recovery

**GOOD IDEA**
- Portable
- Open Source
- Cloud Native
- Observability
Why Kubernetes for MySQL Service?

- **Agility**: Promises instant access to functioning DB.
- **Declarative API**: YAML/JSON manifests to encapsulate MySQL config.
- **Reconciliation**:
  - Updates like config and resource changes reconciled by application controller
  - Kubernetes ensures underlying objects are healthy
- **Support**:
  - Stateful applications a first class construct
  - All major DB vendors support Kubernetes
  - Available on all major public clouds, many private clouds
- **Portability**:
  - Images: Standard format across platforms.
  - Higher level constructs: replicas, storage volumes, service, etc.
Why not Kubernetes?

- **Learning Curve:**
  - Requires developers and devops to fully commit to Kubernetes.
  - It is **not** as easy as consuming DB service from public cloud.
  - If not planned properly (e.g. capacity and availability), can fail in weird ways.

- **Support for External Services:**
  - Storage
  - Load balancing
  - Secret store

- **Cattle model:**
  - Kubernetes scheduler can move workloads around any time.
An application controller (operator) to manage MySQL clusters on Kubernetes.

✓ Self-healing clusters
✓ Highly available reads
✓ Virtually highly available writes
✓ Replication lag detection and mitigation
✓ Resource control
✓ Automated backups and restores
- **Control Plane**
  - MySQL Operator
  - Orchestrator
- **Data Plane**
  - MySQL Pods (StatefulSet)
  - Volumes
  - Services
  - Secrets
- **Monitoring:**
  - MySQL exporter
  - Prometheus
MySQL Cluster

Master

Replica-1

Replica-1

Replica-N

Master Service

Healthy Nodes Service
MySQL Nodes

- Init Container
  - Initialize database credentials, config
- Percona MySQL container
  - Runs standard Percona MySQL image
- Sidecar:
  - Monitors replication lag
  - Backups
● Open Source tool for managing MySQL failovers
● Each MySQL cluster configured with Orchestrator credentials
● Features:
  ○ Automated discovery
  ○ Alerts
  ○ Master failover
Challenges

On-Prem Kubernetes is hard

- Running on On-Prem cloud like Openstack/VMware == More complexity
  - Software based networking
  - Software based storage
  - Misc: Auto scaling limitations
- Running on Bare metal
  - No “cloud” load balancer: MetalLB/NodePort are the only option
  - May get stuck on local-storage as the only storage option
- Must have:
  - Multi-master clusters
  - Fault domains
  - Backups
  - Monitoring
Challenges

- MySQL Operator
  - (Fixed) PVC cleanup
  - Small downtime on failures.
  - MySQL upgrades are disruptive.
  - Backups not as frequent as we would like.
- Orchestrator
  - (Fixed) Unable to recover on node failure.
TODOs

- Always on DB: Multi-Master full sync mode, Proxy SQL support
- Snapshot support
- More backup options
- Seamless upgrades (minor versions)
References

- MySQL Operator
  - [https://github.com/presslabs/mysql-operator](https://github.com/presslabs/mysql-operator)
- MySQL Orchestrator
  - [https://github.com/github/orchestrator](https://github.com/github/orchestrator)
- Kubebuilder
  - [https://github.com/kubernetes-sigs/kubebuilder](https://github.com/kubernetes-sigs/kubebuilder)
Thank you!