KubeCon | CloudNativeCon

North America 2017

on Kubernetes

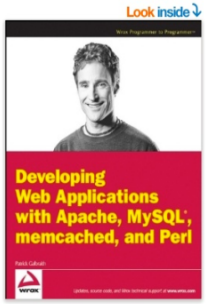
Patrick Galbraith, Principal Engineer, *Oracle Dyn*

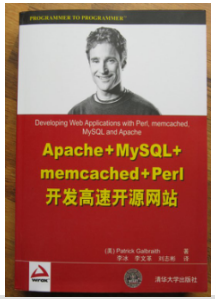
About

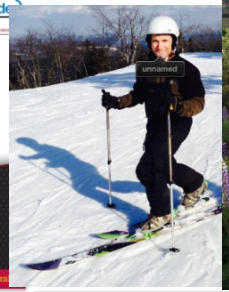


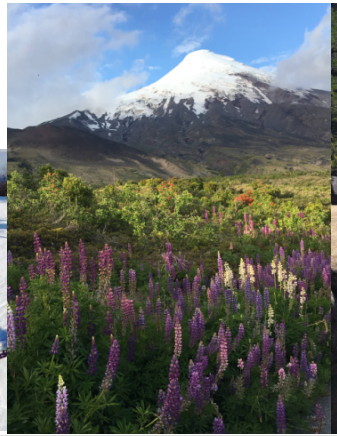

Oracle + Dyn Platform Services

- Oracle Dyn, HP, Blue Gecko, MySQL AB, Classmates, Slashdot, Cobalt Group, US Navy
- MySQL projects: memcached UDFs, DBD::mysql, federated storage engine, Galera on Kubernetes example, Percona helm chart, etc
- Learning, Languages, Family, Outdoors
- Chile
- Not a designer, as you can see :)









About my team

ORACLE + Dyn

- DNS -- https://dyn.com
  - Dyn provides managed DNS for the world's largest and most admired web properties
  - One of the largest, fastest, and most resilient DNS networks in the world
  - Over 3,500 enterprises, including preeminent digital brands like Netflix, Twitter, Linkedin and CNBC
- Email
  - Dyn Email Delivery has helped those with large sending needs for over a decade
- Oracle Cloud (OCI) https://cloud.oracle.com/home
  - Dyn providing Edge Services for Oracle Cloud Infrastructure
  - We're hiring! https://dyn.com/careers/

What is this presentation about?

- Databases in the cloud and how Kubernetes makes it easier to run databases in the cloud
- Stateful applications
- StatefulSets - Vitess
- Operators - MySQL Operator
- Time permitting…

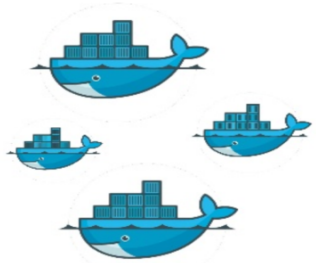
A tale of two open source projects

MySQL and Kubernetes

- MySQL (MySQL, MariaDB, Percona, etc)
  - World's most popular open source database
  - Ubiquitous
  - 20+ years
- Kubernetes
  - Fastest growing open source project
  - Application deployment done right
  - Community-driven
  - Moving target, rapidly developing

Databases: Pets vs. Cattle

- Database containers tend to be more pets than cattle
- A pod of multiple database instances or one?
- *“Kubernetes Pods are mortal.”*
- Stateful Applications
- Consistent access: applications get the same database
- Safety: don’t scale if unhealthy (obvious)
- Persistent storage



Kubernetes ingredients for stateful applications

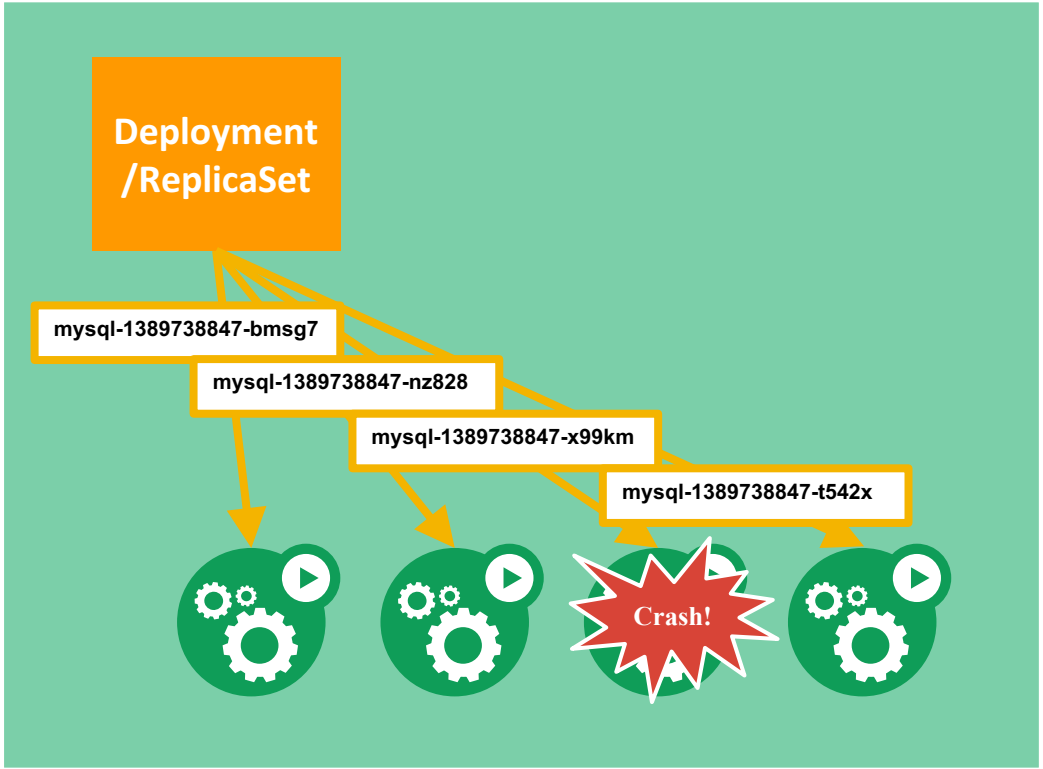
- Services
- PV/PVC, StorageClass
- StatefulSets
  - Init containers
  - VolumeClaimTemplates
- Operators
  - CustomResourceDefinition
- Side-car containers
- nodeSelector

StatefulSets

- Provide guarantees about ordering and uniqueness of pods and pod resources
  - Maintains persistent, sticky identity for each of their pods, across rescheduling
  - Ensures ordinality - Deterministic initialization order.
  - At most one pod per index
  - Pods are not interchangeable and won't attempt to run a pod with the same name
- Stable network identity
  - Consistent name of pod and headless service to set service "domain".
  - Headless service providing stable DNS name - eg "mysql", nodes name "node-N.mysql"
  - Pod name set as subdomain within domain of headless service
  - DNS name is stable across restarts
- Stable Storage.
  - Pods re-mount same persistent storage across restarts, stateful resources
- Safety
  - Scaling is blocked if system is unhealthy

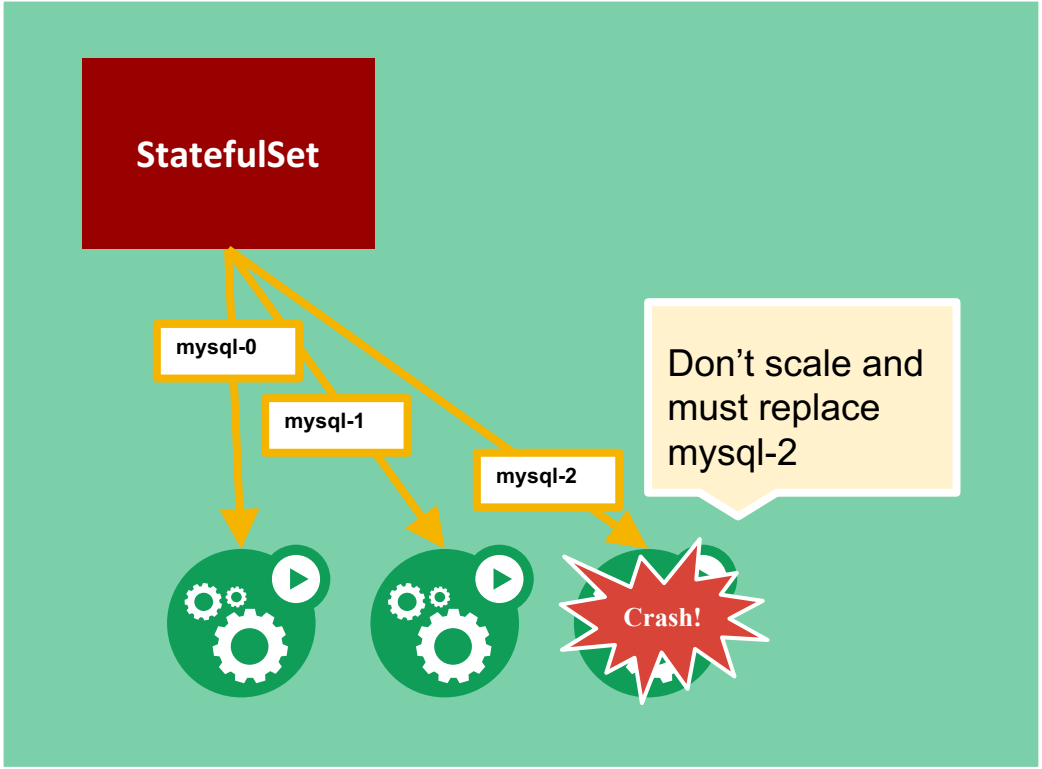
Deployment/ReplicaSet

- Started in no specific order
- name-<depid>-<rand alpha>
- Will scale if crash and replace with another non-unique name



StatefulSet

- Started in order
- Names unique and distinct per node-ordinal
- Must replace failed node by name
- Won’t scale when cluster is unhealthy



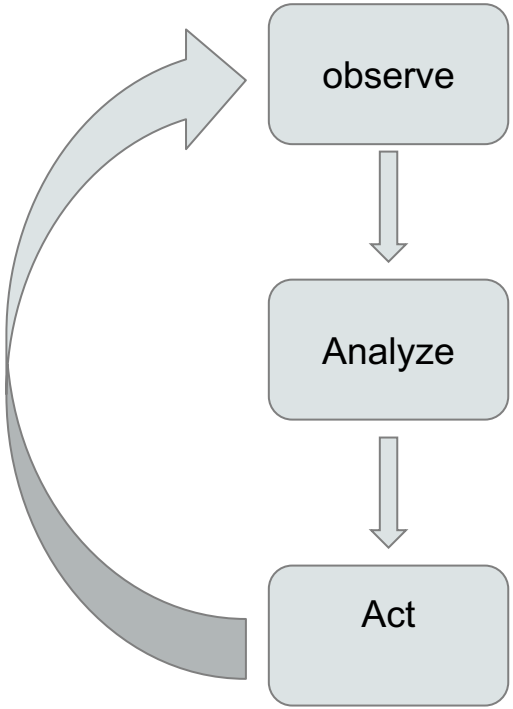
Operators

- Application-specific domain controller
- Encodes application-specific domain knowledge through extending Kubernetes API using custom resource definitions
- Enables users to create, configure, and manage stateful applications
- Build upon resources and controllers
- Leverage Kubernetes primitives like ReplicaSets, StatefulSets, and Services
- Operator executes common application tasks
- Installed as a deployment
- Application run using custom resource definition types for operator

https://coreos.com/blog/introducing-the-etcd-operator.html

Operators

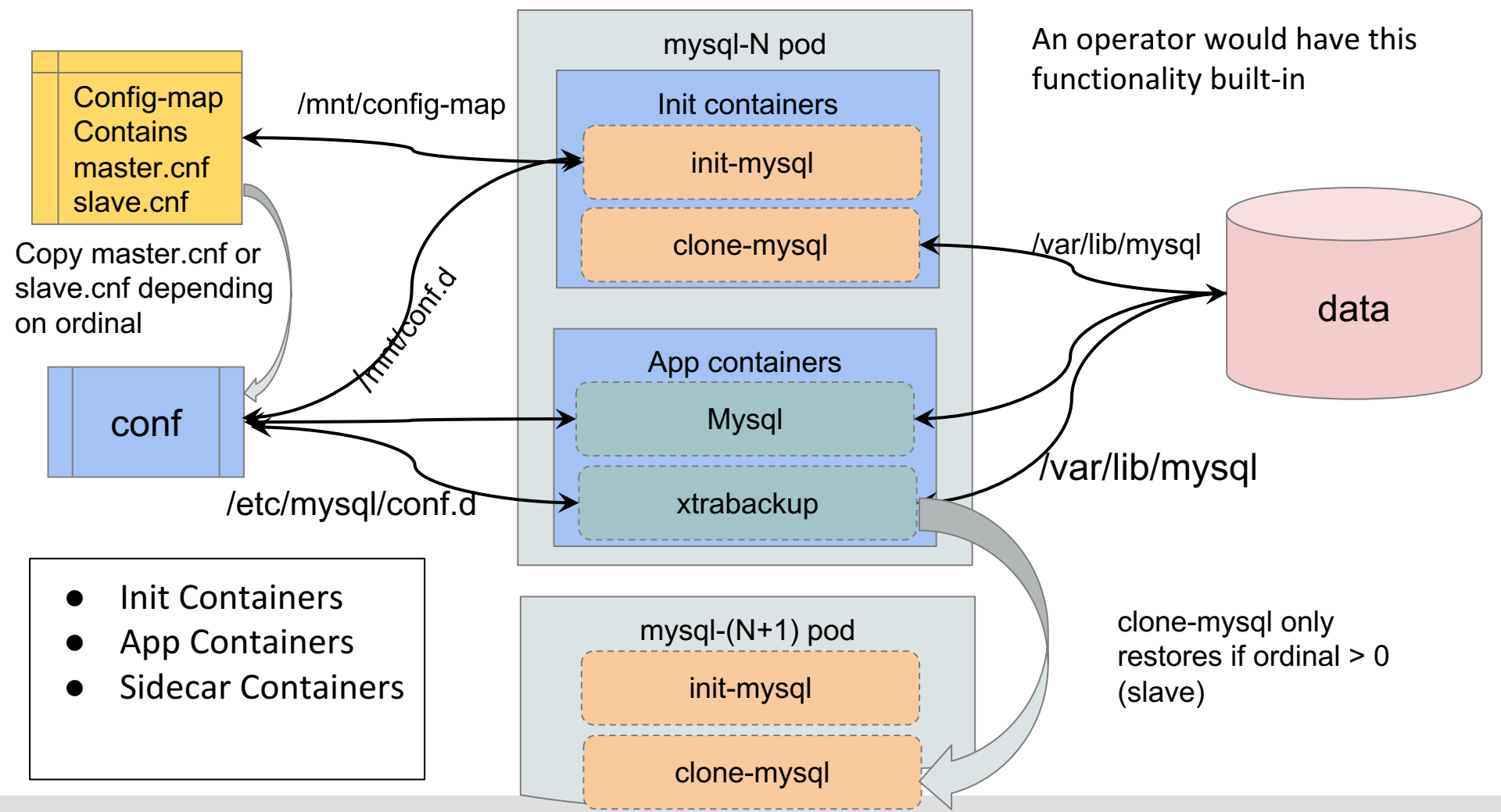
- Create/Destroy: No need for specifications of each member. Just size of cluster.
- Resize: Only specify size. Operator will deploy, destroy, or reconfigure as needed
- Backup: built-in, only need to specify backup policy
- Upgrade: operator will ensure members are correct version avoiding headaches
- Etcd operator https://youtu.be/tPSys734iNk



"etcdX" cluster has 2 running pods
- etcd-0, version 3.1.1
- etcd-1, version 3.2.10

Differences from desired config
- Should be version 3.2.10
- Should have 3 members

How get desired config
- Recover 1 member
- Upgrade etcd-0 to 3.2.10



MySQL Operator

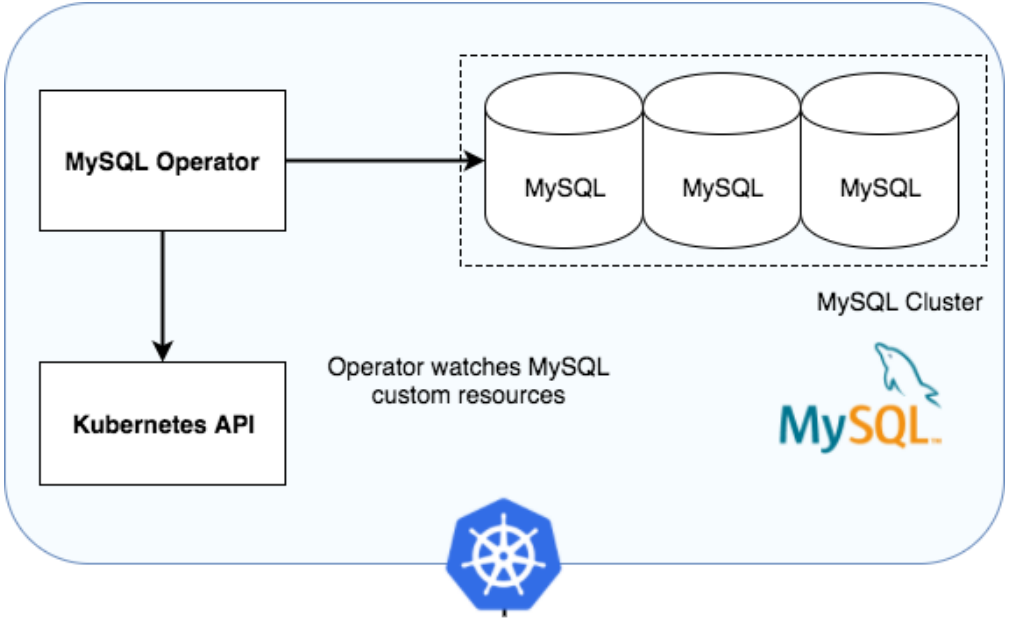
- New Open source project from Oracle -- will be releasing early in 2018
- Deploy a highly-available clustered MySQL instance into Kubernetes with a single command
- Watches the API server for Custom Resource Definitions related to MySQL and acts on those resources
- Backup/Restore made simple with MySQLBackup resource
- PV/PVC Block Storage
- Utilizes Group replication
- Automated backup/restore to/from object storage

MySQL Operator

Features:

Create/Scale/Destroy: Easily create, scale and delete self healing MySQL clusters inside a Kubernetes namespace

- **Simple Configuration**: Provide configuration for your MySQL cluster with simple Kubernetes objects (Secrets and ConfigMaps)
- **Backups**: Defined scheduled backup policies as part of your cluster to make sure your data is always protected or, alternatively, create on-demand snapshot backups with a single command.



Examples: simple cluster and backup

Create a cluster:

```
kubectl create -f mysql-test-cluster.yaml
```

Create a backup:

```
kubectl create -f backup.yaml
```

List backups:

```
kubectl get mysqlbackups
```

Fetch an individual backup:

```
kubectl get mysqlbackup primary/123
```

```
---
apiVersion: "mysql.oracle.com/v1"
kind: MySQLCluster
metadata:
  name: mysql-test-cluster

---
apiVersion: "mysql.oracle.com/v1"
kind: MySQLBackup
metadata:
  name: mysql-test-cluster-example-snapshot-backup
  namespace: default
spec:
  cluster:
    name: mysql-test-cluster
  # These credentials are used to
  # upload to object storage
  secretRef:
    name: oci-credentials
```

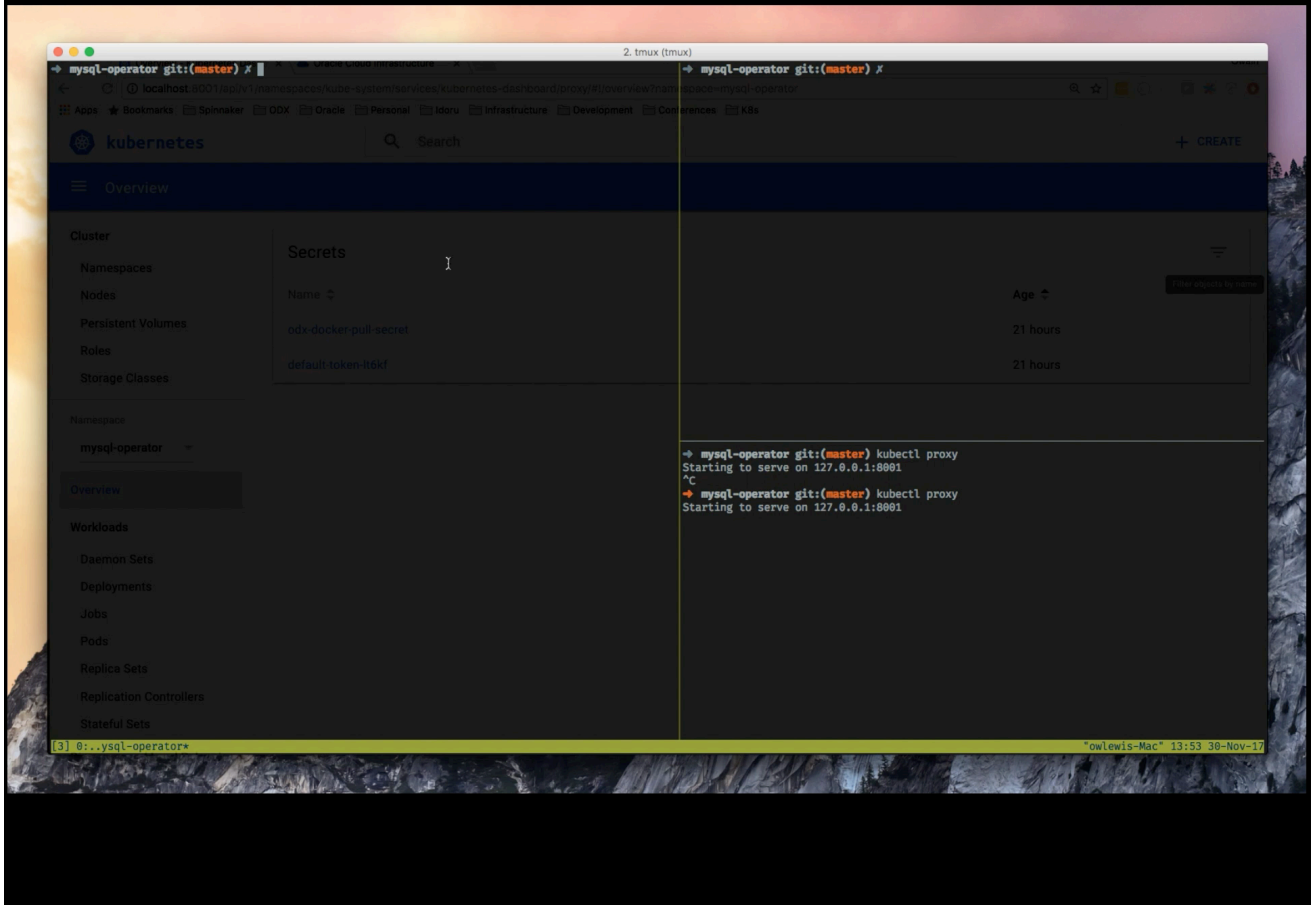
Examples : create a cluster with PV

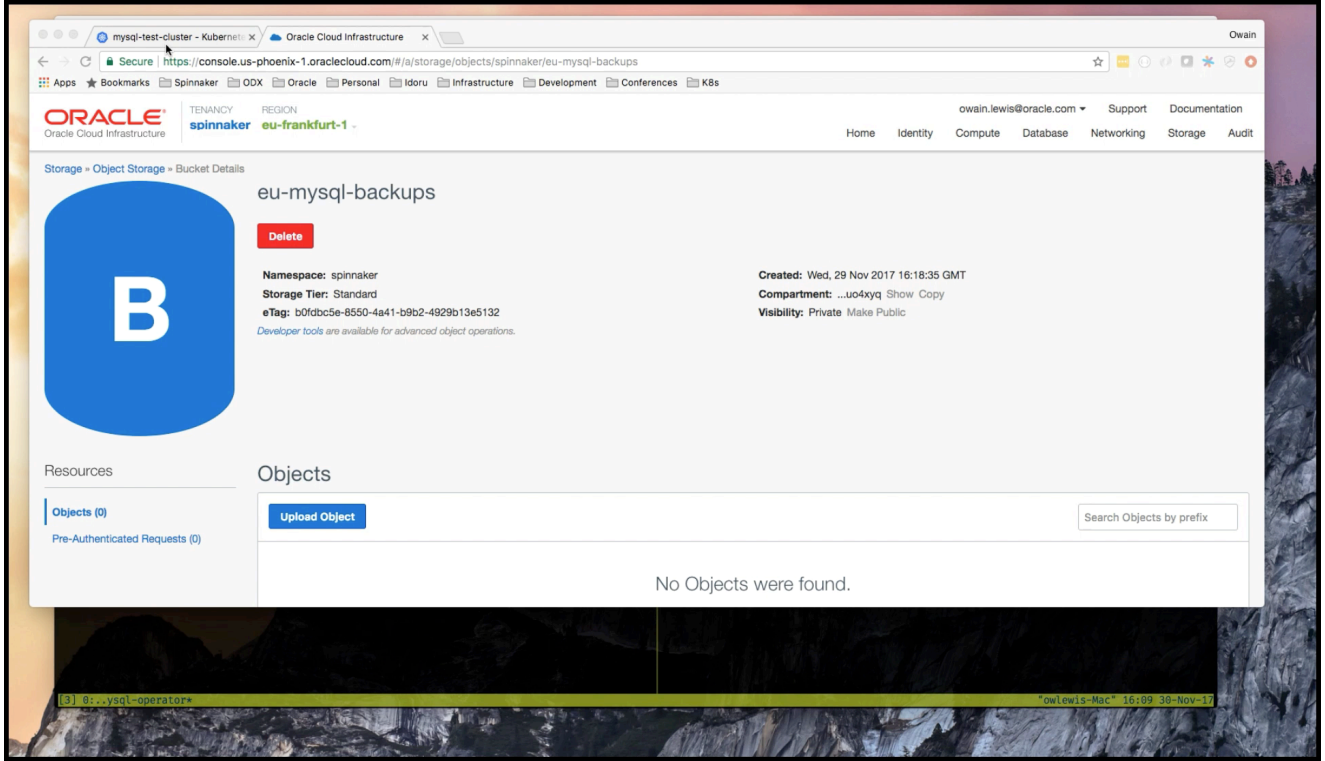
```
---
apiVersion: v1
kind: PersistentVolume
metadata:
  labels:
    type: local
  name: mysql-local-volume
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 10Gi
  hostPath:
    path: /tmp/data
  persistentVolumeReclaimPolicy: Recycle
  storageClassName: manual
```

```
---
apiVersion: "mysql.oracle.com/v1"
kind: MySQLCluster
metadata:
  name: example-mysql-cluster-with-volume
spec:
  replicas: 3
  volumeClaimTemplate:
    metadata:
      name: data
    spec:
      storageClassName: manual
      accessModes:
        - ReadWriteMany
      resources:
        requests:
          storage: 1Gi
```

MySQL Operator Demos

- Create a 3-node MySQL cluster!
  https://youtu.be/SeD5eSh3lL8
- Backups made easy! On-demand backup of an existing cluster
  https://youtu.be/h_W5xtdce0k

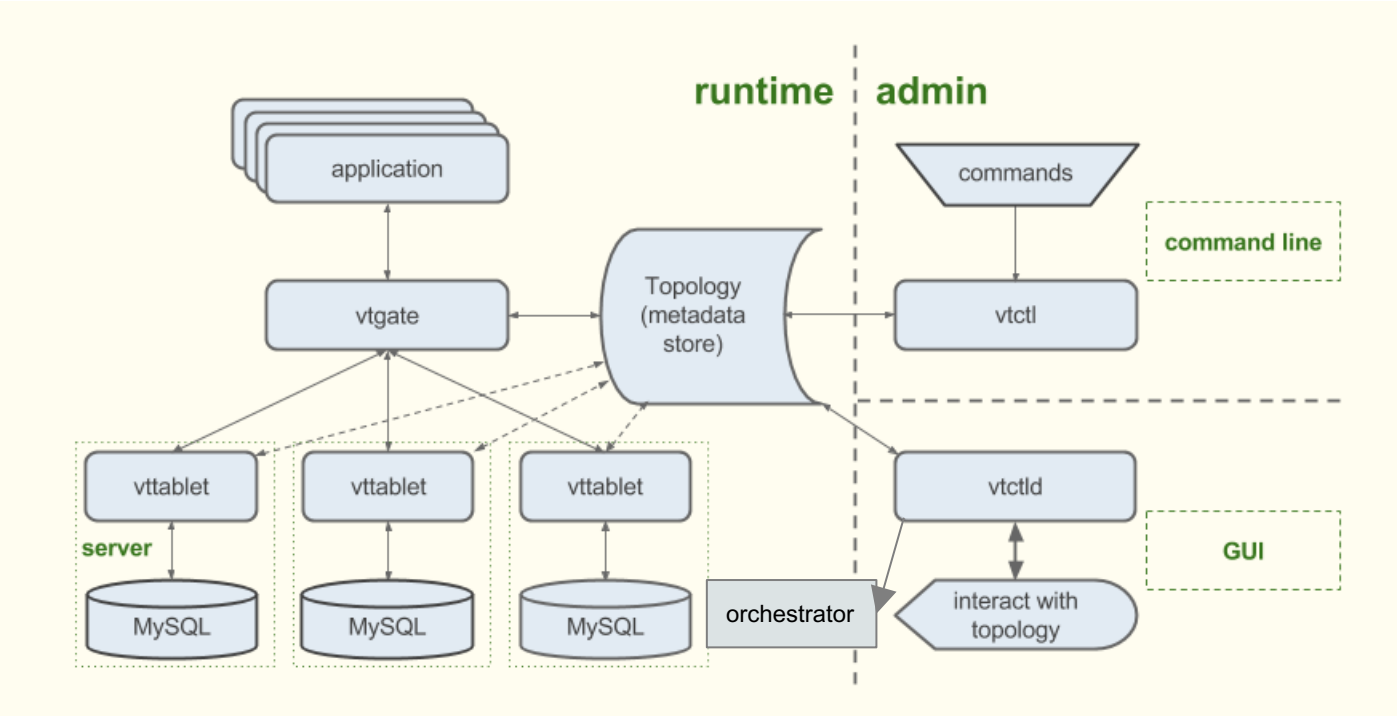




Vitess: sharding on a silver platter (http://vitess.io)

- Cloud ready (The cloud is coming!)
- If you want to migrate your MySQL application to the cloud, Vitess is an excellent vehicle to use
- YouTube database store since 2011
- Helps you scale with transparent sharding and ability to continue sharding out (or in)
- Cluster management - backups and schema tools
- Looks like MySQL database in usage, but is sharded underneath
- Connection pooling
- Protects MySQL with query - de-duping, re-writing, sanitization, and black-listing, putting limits on unbound queries
- Monitoring tools - built into all Vitess binaries.

Vitess



Sharding

- A shard consists of a single master that handles writes, multiple slaves that handle reads
- Start with no sharding with a single keyspace, then move tables to different keyspaces
  - Horizontal - split or merge shards in a sharded keyspace
  - Vertical - moving tables from an unsharded keyspace to a different keyspace
- Re-sharding
  - split, merge, add new cells and replicas
  - Filtered replication key ingredient ensures source tablet data is transferred to proper destination tablet, GTID used
- Vschema - Vitess Schema - ties together all databases managed by Vitess and helps to decide where queries should go
- Vindex - cross-shard index provides a way to map a column value to a keyspace ID

Schema and Vschema

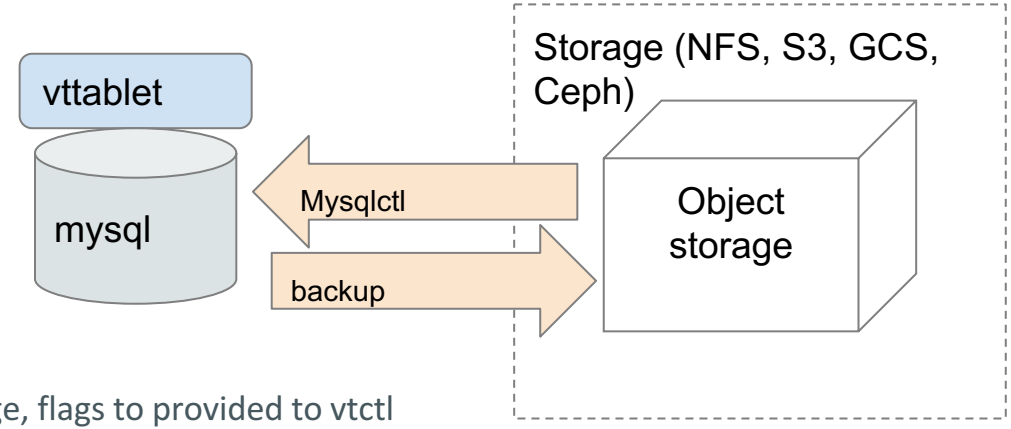
```
# schema - user keyspace
create table user(
  user_id bigint,
  name varchar(128),
  primary key(user_id)
);
```

```
// vschema - user keyspace
{
  "sharded": true,
  "vindexes": {
    "hash": {
      "type": "hash"
    },
  "tables": {
    "user": {
      "column_vindexes": [
        {
          "column": "user_id",
          "name": "hash" # NOTE: vindex name
        }
      ]
    }
  }
}
```

Reparenting

- Reparenting
  - Changing shard's master tablet to another host
  - Changing slave tablet to have a different master
- GTID
  - GTID is used to initialize replication
  - GTID stream used in parenting
- Types of reparenting
  - PlannedReparentShard - healthy master tablet to new master
  - EmergencyReparentShard - forces reparent to a new master when master unavailable. Data connect be retrieved from current master
- Re-sharding
  - split, merge, add new cells and replicas
  - Filtered replication key ingredient ensures source tablet data is transferred to proper destination tablet

backups

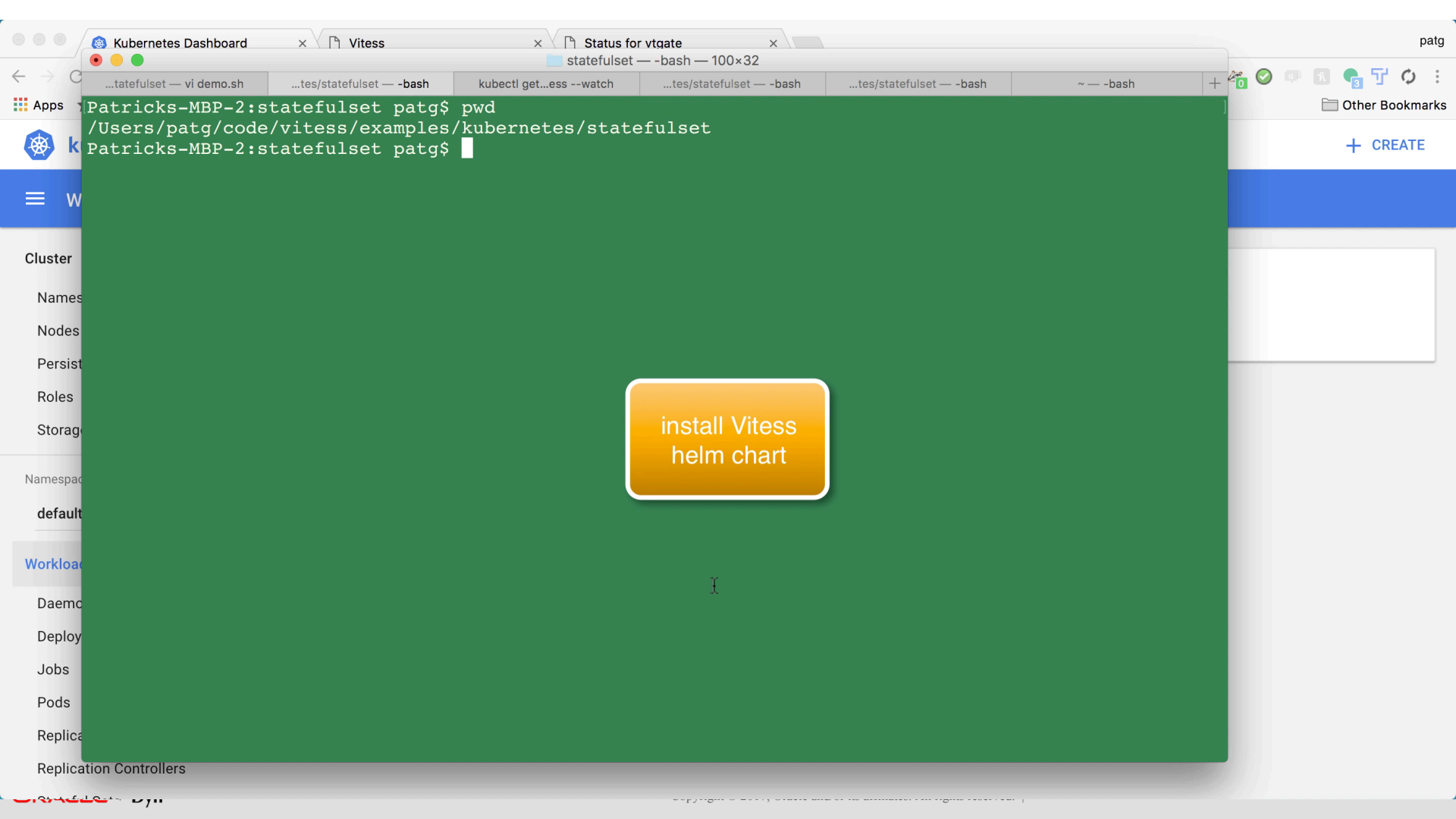


- Backup
  - Plugins for NFS, GCS, S3, Ceph
  - VTtablet must have access to where storage, flags to provided to vtctl
  - `vtctl backup <tablet-alias>`
- Restore
  - For restore, VTtablet started with arguments specifying whether to restore from a backup, the storage implementation and backup storage root (path)
- Commands
  - `vtctl backup <tablet-alias>`
  - `vtctl ListBackups <keyspace/shard>`
  - `vtctl RemoveBackup <keyspace/shard>`

Vitess StatefulSet Demo

https://youtu.be/9in5HenJ9xY

- Create two shards
- Load database schema and vschema
- Scale one shard from 2 replicas to 3
- Delete StatefulSet, delete master pod, reparent
- Anthony Yeh, https://github.com/enisoc/vitess.git, statefulset-demo branch


Kubernetes Dashboard
Vitess
Status for vtgate
patg
statefulset — -bash — 100×32
...tatefulset — vi demo.sh
...tes/statefulset — -bash
kubectl get...ess --watch
...tes/statefulset — -bash
...tes/statefulset — -bash
~ — -bash
Apps
Other Bookmarks
CREATE
Cluster
Names
Nodes
Persist
Roles
Storag
Namespac
default
Workloa
Daemo
Deploy
Jobs
Pods
Replica
Replication Controllers
Patricks-MBP-2:statefulset patg$ pwd
/Users/patg/code/vitess/examples/kubernetes/statefulset
Patricks-MBP-2:statefulset patg$
install Vitess
helm chart

Questions Preguntas Fragen सवाल...

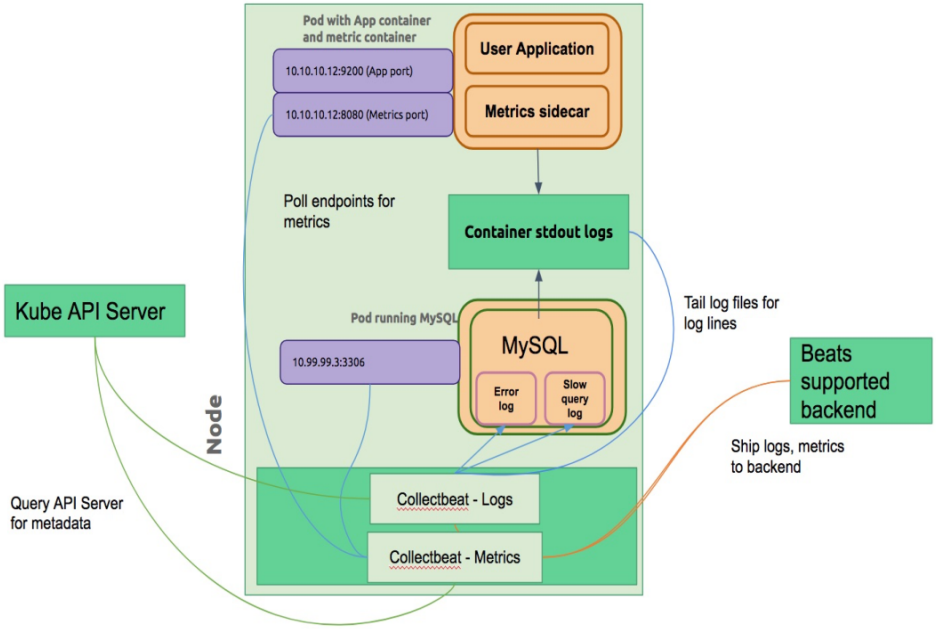
Thank you!

Special thanks to: Platform Team Oracle Dyn, Anthony Yeh, Sugu Sougoumarane, Owain Lewis, Steve Lerner, SeveralNines, et al!

Extra slides - resources

Collectbeats

- Open Source effort within Ebay
- Inspired by Metricbeat module “drop in” concept.
- Find all pods exposing metrics and start polling
- Ability to
  - Collect metrics from any pod that exposes them
  - Collect logs from STDOUT or logs on container
  - Append pod metadata to logs and metrics
  - Allow pods to push metrics through Graphite protocol and parse them uniquely
  - Stitch stack-traces for application logs
- http://bit.ly/2zT4VNZ



Other MySQL installation examples

- **`helm install --name kubeconf stable/percona`**
  - https://www.youtube.com/watch?v=kZOk0w6YKMA
- Simple replication with a stateful set https://kubernetes.io/docs/tasks/run-application/run-replicated-stateful-application/
  - https://www.youtube.com/watch?v=M7EWFL83Vdc
- Galera StatefulSet by SeveralNines https://severalnines.com/blog/mysql-docker-running-galera-cluster-kubernetes
  - https://youtu.be/3vn7Jd9z4kc

NodeSelector

- Can be used to achieve affinity to a specific node
- Label assigned to node

```
kubectl label nodes ip-172.xxx.ec2.internal boxtype=c3.medium
```

```
spec:
  containers:
    - image: mysql:5.6
      name: mysql
      ...<snip>...
   nodeSelector:
     boxtype: c3.medium
```

Node and Pod Affinity/AntiAffinity

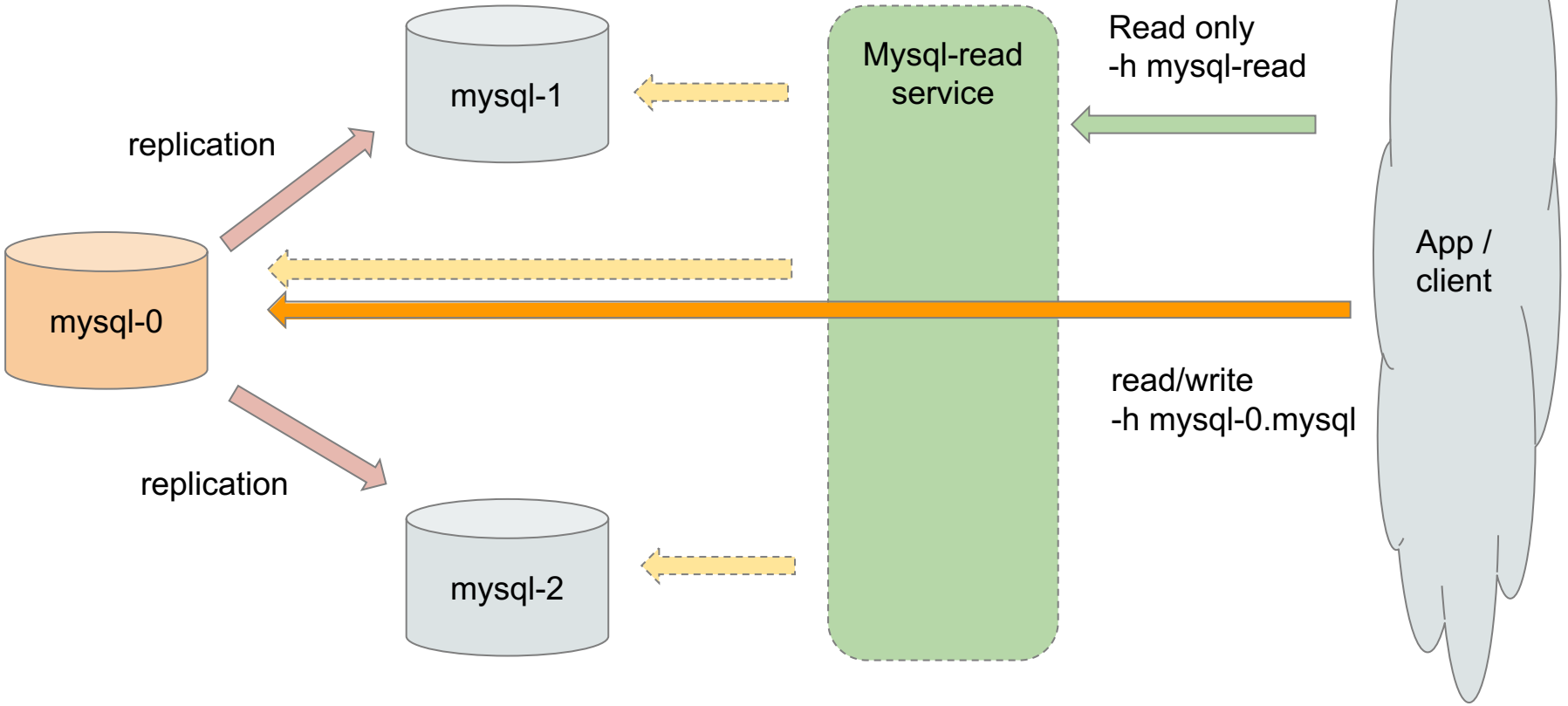
- Based on labels
- nodeAffinity/nodeAntiAffinity uses labels on node
- podAffinity/podAntiAffinity labels of pods already running on node

```
metadata:
  name: with-node-affinity (or with-pod-affinity)
spec:
  affinity:
    nodeAffinity:
```

- Hard - “must” requiredDuringSchedulingIgnoredDuringExecution
- Soft - “preferred” preferredDuringSchedulingIgnoredDuringExecution

MySQL simple replication StatefulSet

https://kubernetes.io/docs/tasks/run-application/run-replicated-stateful-application/



severalnines

MariaDB Galera StatefulSet example

- https://severalnines.com/blog/mysql-docker-running-galera-cluster-kubernetes
- Provides both Deployment (ReplicaSet) and StatefulSet
- StatefulSet much more dynamic and ordinality gives more control
- Manually-provisioned PV/PVC using labels
- Utilizes etcd cluster for donor discovery in entry-point script
- No need for init containers as SST innate with Galera
- https://youtu.be/3vn7Jd9z4kc

Headless Service

- Service with `clusterIP` of `None`
  - Makes it so all nodes have an A-record of `service-<ord>.<service name>`
  - This service is not used for application or clients
- For read-only, create another service - `mysql-read` which can be of any type (`nodePort, clusterIP, loadBalancer`)

```yaml
apiVersion: v1
kind: Service
metadata:
  name: mysql
  labels:
    app: mysql
spec:
  ports:
  - name: mysql
    port: 3306
  clusterIP: None
  selector:
    app: mysql
```

```yaml
apiVersion: apps/v1beta1
kind: StatefulSet
metadata:
  name: mysql
spec:
  selector:
    matchLabels:
      app: mysql
  serviceName: mysql
  replicas: 3
  template:
    metadata:
      labels:
        app: mysql
```

VolumeClaimTemplate

- Define a ServiceType
- Specify type in VolumeClaimTemplate

```
volumeClaimTemplates:
- metadata:
    name: data
  spec:
    accessModes: ["ReadWriteOnce"]
    storageClassName: slow
    resources:
      requests:
        storage: 10Gi
```

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: slow
provisioner: kubernetes.io/aws-ebs
parameters:
  type: io1
  iopsPerGB: "10"
```

Simple single instance with the Percona chart

- Now in Helm charts for MySQL, MariaDB, Percona
  - `helm install --name kubeconf stable/percona (or mysql, mariadb)`
  - `kubectl exec -it kubeconf-percona-xxx -- mysql -u root -p$(kubectl get secret --namespace default kubeconf-percona -o jsonpath="{.data.mysql-root-password}" | base64 --decode; echo) -h kubeconf-percona`
  - Set password with secret name and key in app manifest
- Fronted with simple service for application
- Easy to modify to add volumes
- Even better to run as a StatefulSet

https://www.youtube.com/watch?v=kZOk0w6YKMA