Fernado Camargos
Senior Architect
Percona

PERCONA LIVE 2023

Deploying MySQL on Kubernetes with the PerconaOperator

Monday, May 22nd

01:30 PM MST

PERCONA

Deploying MySQL on Kubernetes with the Percona Operator

Fernando Laudares Camargos, Senior Architect
Chetan Shivashankar, Kubernetes Technical Lead

Table of Contents

PERCONA

Installation and customization

Installation and customization

- Deploying on GKE
- Anti-affinity and tolerations
- Changing MySQL options
- Custom ressources options
- Application and system users

PERCONA

Deploying on GKE

https://docs.percona.com/percona-operator-for-mysql/pxc/gke.html

```
git clone -b v1.12.0 https://github.com/percona/percona-xtradb-cluster-operator
cd percona-xtradb-cluster-operator
```

```
gcloud auth login
gcloud container clusters create my-cluster-1 --project <project name> --zone us-central1-a --cluster-version 1.23 --machine-type n1-standard-4 --num-nodes=3
gcloud container clusters get-credentials my-cluster-1 --zone us-central1-a --project <project name>
```

```
kubectl create clusterrolebinding cluster-admin-binding --clusterrole cluster-admin --user $(gcloud config get-value core/account)
kubectl create namespace pxc
kubectl config set-context $(kubectl config current-context) --namespace=pxc
```

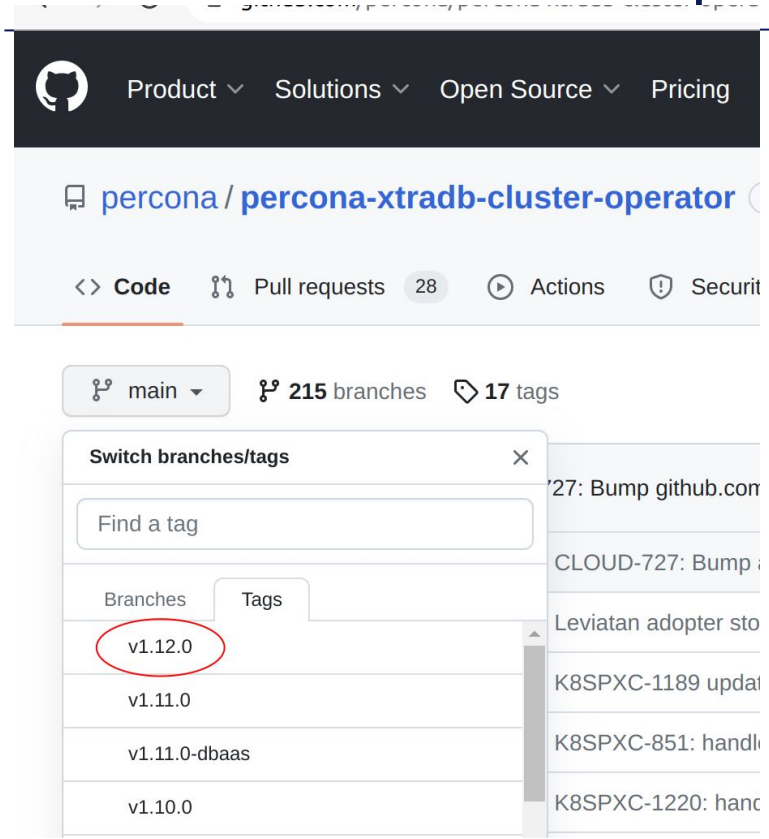
```
kubectl apply -f deploy/crd.yaml
kubectl apply -f deploy/rbac.yaml
kubectl apply -f deploy/operator.yaml
kubectl create -f deploy/secrets.yaml
```

```
kubectl apply -f deploy/bundle.yaml
```

```
kubectl apply -f deploy/cr.yaml
```


PERCONA

Deploying on GKE



Generic	EKS	GKE
		gcloud auth login
	eksctl create cluster -f ~/cluster.yaml	gcloud container clusters create m
		gcloud container clusters get-crede
		kubectl create clusterrolebinding cl
git clone -b v1.12.0 https://github.com/percona/percona-xtradb-cluster-operator		
cd percona-xtradb-cluster-operator		
kubectl create namespace pxc		
kubectl config set-context $(kubectl config current-context) --namespace=pxc		
kubectl apply -f deploy/crd.yaml	kubectl apply -f deploy/bundle.yaml	
kubectl apply -f deploy/rbac.yaml		
kubectl apply -f deploy/operator.yaml		
kubectl create -f deploy/secrets.yaml		
kubectl apply -f deploy/cr.yaml		

PERCONA

Deploying on GKE

https://docs.percona.com/percona-operator-for-mysql/pxc/gke.html

```
git clone -b v1.12.0 https://github.com/percona/percona-xtradb-cluster-operator

cd percona-xtradb-cluster-operator

gcloud auth login

gcloud container clusters create nando-1 --project pl2023-k8s-tutorial --zone us-central1-a --cluster-version 1.23
--machine-type n1-standard-4 --num-nodes=3

gcloud container clusters get-credentials nando-1 --zone us-central1-a --project pl2023-k8s-tutorial

kubectl create clusterrolebinding cluster-admin-binding --clusterrole cluster-admin --user $(gcloud config
get-value core/account)

kubectl create namespace pxc

kubectl config set-context $(kubectl config current-context) --namespace=pxc

kubectl apply -f deploy/bundle.yaml

kubectl apply -f deploy/cr.yaml
```

* Remember to adjust: **cluster name**, **project name**, and possibly **zone**

```
kubectl get pods -w
```


PERCONA

Accessing test VMs

https://bit.ly/45jtWwX

```
chmod +x 600 pl2023

ssh -i pl2023 ubuntu@<IP>
```


PERCONA

Deploying on GKE

```
$ gcloud container clusters list
NAME     LOCATION       MASTER_VERSION   MASTER_IP      MACHINE_TYPE   NODE_VERSION     NUM_NODES  STATUS
nando-1  us-central1-a  1.23.17-gke.300  34.122.11.227  n1-standard-4  1.23.17-gke.300      3      RUNNING
```

N1 standard | N1 high-memory | N1 high-cpu | N1 shared-core

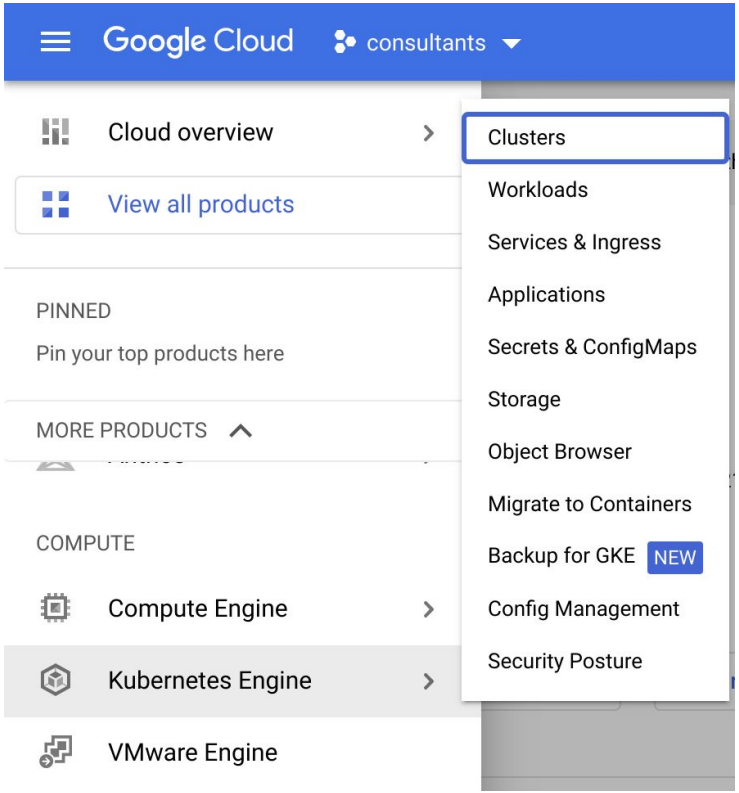
N1 standard machine types have 3.75 GB of system memory per vCPU.

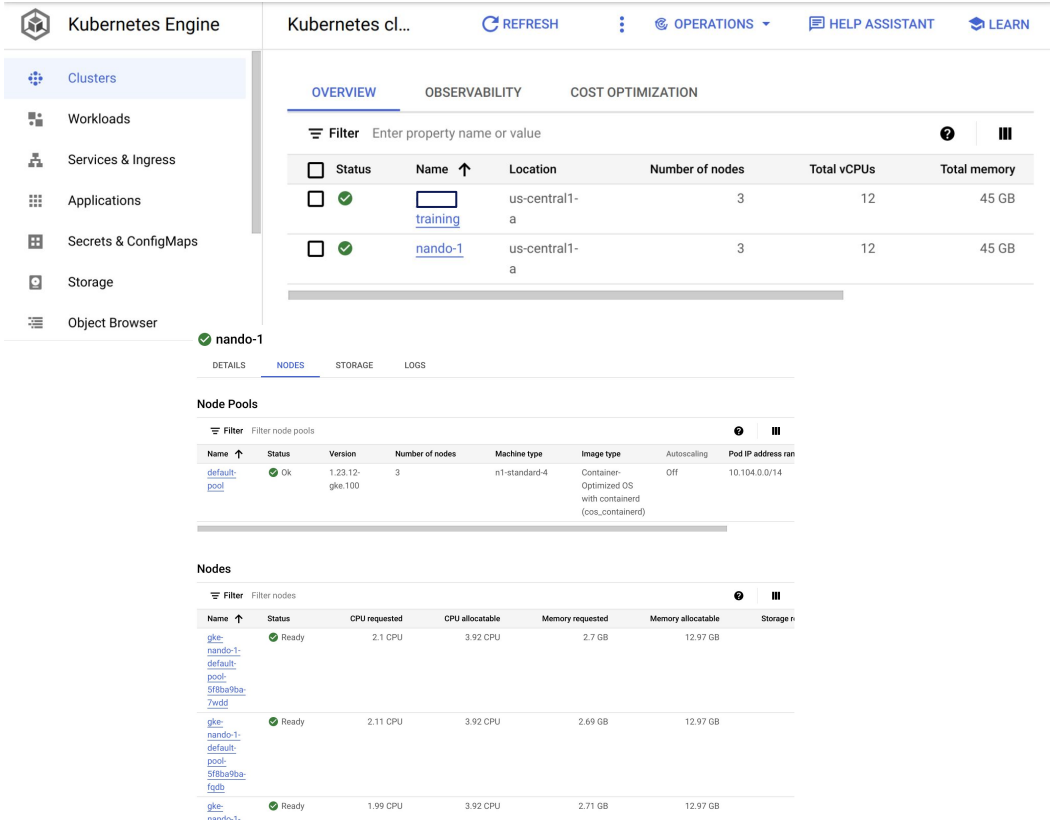
Machine types	vCPUs*	Memory (GB)	Max number of persistent disks (PDs)†	Max total PD size (TB)	Local SSD	Maximum egress bandwidth (Gbps)‡	Tier 1 egress bandwidth (Gbps)
n1-standard-1	1	3.75	128	257	Yes	2	N/A
n1-standard-2	2	7.50	128	257	Yes	10	N/A
n1-standard-4	4	15	128	257	Yes	10	N/A

PERCONA

Deploying on GKE

https://console.cloud.google.com





PERCONA

Deploying on GKE

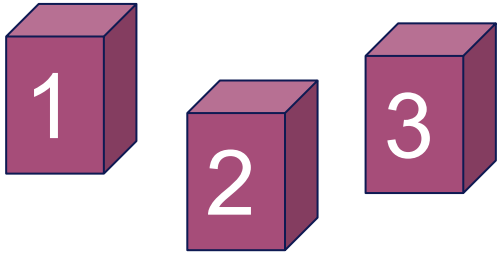
Status	Name	Location	Number of nodes	Total vCPUs	Total memory
✓	nando-1	us-central1-a	3	12	45 GB

Nodes

```
$ kubectl get nodes
```

Filter Filter nodes

Name
gke-nando-1-default-pool-81437ff3-6j1f
gke-nando-1-default-pool-81437ff3-7w1x
gke-nando-1-default-pool-81437ff3-nsw1



PERCONA

Deploying on GKE

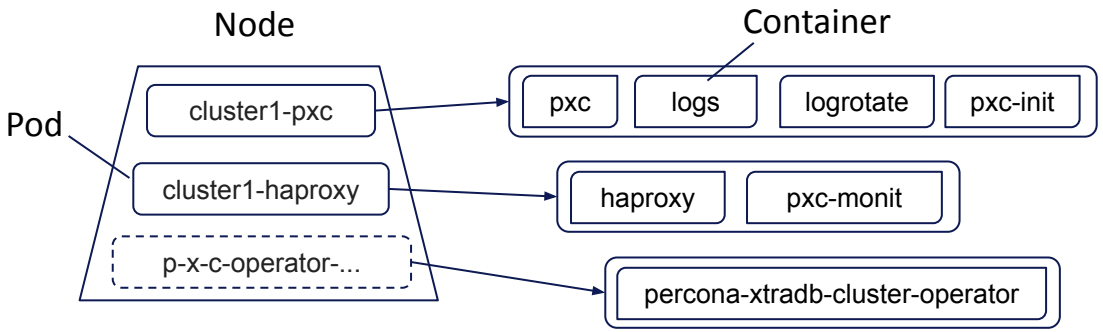
```
$ kubectl get pxc

NAME        ENDPOINT                STATUS   PXC   PROXYSQL   HAPROXY   AGE
cluster1    cluster1-haproxy.pxc    ready    3                3         7m
```

```
$ kubectl get pods
NAME                                          READY   STATUS    RESTARTS   AGE
cluster1-haproxy-0                            2/2     Running   0          6m1s
cluster1-haproxy-1                            2/2     Running   0          4m44s
cluster1-haproxy-2                            2/2     Running   0          4m18s
cluster1-pxc-0                                3/3     Running   0          6m1s
cluster1-pxc-1                                3/3     Running   0          4m48s
cluster1-pxc-2                                3/3     Running   0          3m36s
percona-xtradb-cluster-operator-5dbc998f8b-mm7wx   1/1     Running   0          6m36s
```


PERCONA

Deploying on GKE

```
$ kubectl get pods -o wide
NAME                                                  READY   STATUS    RESTARTS   AGE     IP          NODE                                          NOMINATED NODE   READINESS GATES
cluster1-haproxy-0                                    2/2     Running   0          9m1s    10.104.1.7  gke-nando-1-default-pool-81437ff3-6j1f        <none>           <none>
cluster1-haproxy-1                                    2/2     Running   0          7m44s   10.104.0.7  gke-nando-1-default-pool-81437ff3-7w1x        <none>           <none>
cluster1-haproxy-2                                    2/2     Running   0          7m18s   10.104.2.5  gke-nando-1-default-pool-81437ff3-nsw1        <none>           <none>
cluster1-pxc-0                                        3/3     Running   0          9m1s    10.104.2.4  gke-nando-1-default-pool-81437ff3-nsw1        <none>           <none>
cluster1-pxc-1                                        3/3     Running   0          7m48s   10.104.1.8  gke-nando-1-default-pool-81437ff3-6j1f        <none>           <none>
cluster1-pxc-2                                        3/3     Running   0          6m36s   10.104.0.8  gke-nando-1-default-pool-81437ff3-7w1x        <none>           <none>
percona-xtradb-cluster-operator-5dbc998f8b-mm7wx      1/1     Running   0          9m36s   10.104.1.6  gke-nando-1-default-pool-81437ff3-6j1f        <none>           <none>
```



PERCONA

Affinity

https://docs.percona.com/percona-operator-for-mysql/pxc/constraints.html

The *mysql* section of the *cr.yaml* file allows for the customization of node affinity:

- **affinity** rules make a pod eligible to run on a node hosting other pods labelled a certain way
- **anti-affinity** is the opposite: define the criteria that renders a pod ineligible to run on a given node
  - the standard employed in Percona operators

```
pxc:
(...)
  affinity:
    antiAffinityTopologyKey: "kubernetes.io/hostname"
```

pods will avoid residing within the same **host**

* It is also possible to use standard Kubernetes constraints to define affinity rules, a more advanced approach.

PERCONA

K9s (1)

k9s is a terminal client tool to ease the management of K8s:

https://github.com/derailed/k9s

```
Context: gke_consultants-206215_us-central1-a_nando-1     <0> all       <a>      Attach
Cluster: gke_consultants-206215_us-central1-a_nando-1     <1> pxc       <ctrl-d> Delete
User:    gke_consultants-206215_us-central1-a_nando-1     <2> default   <d>      Describe
K9s Rev: v0.26.3 ⚡v0.26.7                                              <e>      Edit
K8s Rev: v1.23.13-gke.900                                               <?>      Help
CPU:     4%                                                             <ctrl-k> Kill
MEM:     9%
```

```
Pods(pxc)[7]
NAME↑                                          PF READY RESTARTS STATUS   CPU MEM %CPU/R %CPU/L %MEM/R %MEM/L IP
cluster1-haproxy-0                             ● 2/2          0 Running   55   8      9    n/a      0    n/a 10.1
cluster1-haproxy-1                             ● 2/2          0 Running   57   8      9    n/a      0    n/a 10.1
cluster1-haproxy-2                             ● 2/2          0 Running   50   8      8    n/a      0    n/a 10.1
cluster1-pxc-0                                 ● 3/3          0 Running   21 447      2    n/a     39    n/a 10.1
cluster1-pxc-1                                 ● 3/3          0 Running   21 437      2    n/a     38    n/a 10.1
cluster1-pxc-2                                 ● 3/3          0 Running   18 442      1    n/a     38    n/a 10.1
percona-xtradb-cluster-operator-5dbc998f8b-mm7wx ● 1/1        0 Running   11  22     11      5    113      4 10.1

<pod>
```


PERCONA

K9s (2)

Containers in a pod:

```
Context: gke_consultants-206215_us-central1-a_nando-1      <a>       Attach          <f> Show…
Cluster: gke_consultants-206215_us-central1-a_nando-1      <?>       Help
User:    gke_consultants-206215_us-central1-a_nando-1      <l>       Logs
K9s Rev: v0.26.3 ⚡v0.26.7                                  <p>       Logs Previous
K8s Rev: v1.23.13-gke.900                                  <shift-f> PortForward
CPU:     5%                                                <s>       Shell
MEM:     9%

                              Containers(pxc/cluster1-pxc-0)[4]

NAME↑       PF IMAGE                                                    READY STATE     INIT  RESTARTS PROBES(L:
logrotate   ● percona/percona-xtradb-cluster-operator:1.11.0-logcollector true  Running   false        0 off:off
logs        ● percona/percona-xtradb-cluster-operator:1.11.0-logcollector true  Running   false        0 off:off
pxc         ● percona/percona-xtradb-cluster:8.0.27-18.1                  true  Running   false        0 on:on
pxc-init    ● percona/percona-xtradb-cluster-operator:1.11.0              true  Completed true         0 off:off

<pod> <containers>
```


PERCONA

K9s (3)

Some interesting shortcuts:

- :node → show nodes
- :namespace → show namespaces
- :pod → show pods
- :service → show services

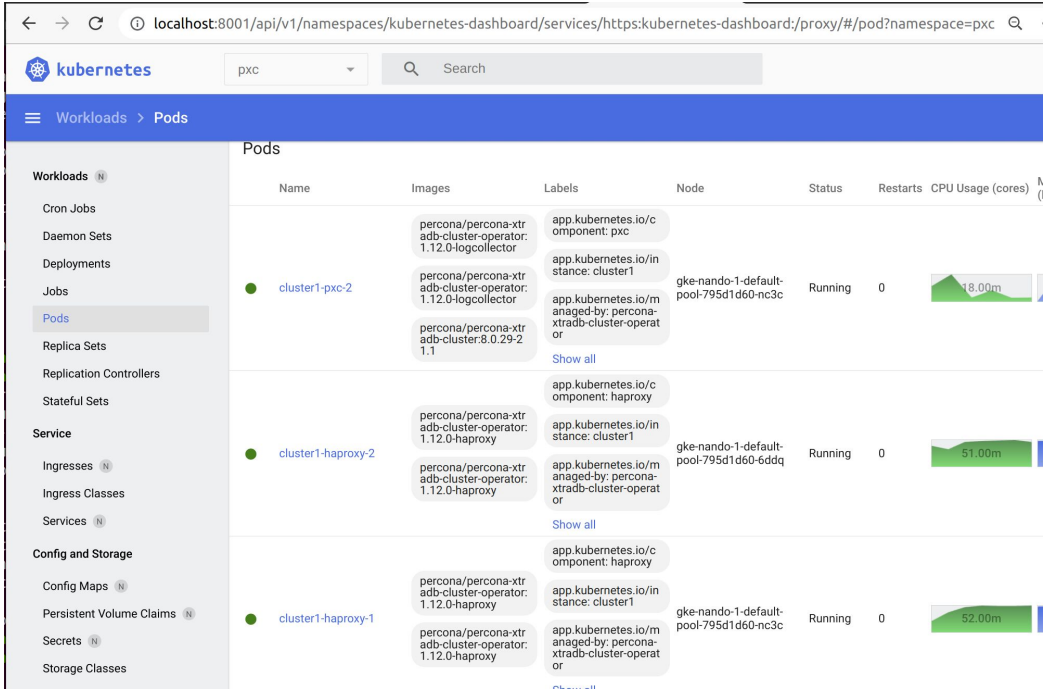
➔ Can edit configuration
➔ Can access logs
➔ Can open shell in container
➔ Can redirect TCP ports

Kubernetes Dashboard

A "(...) general-purpose, web-based UI for Kubernetes clusters.

- https://github.com/kubernetes/dashboard

* Requires the creation of a special Service Account granted with *ClusterRoleBinding* privilege.



PERCONA

Accessing the database

Secrets

https://docs.percona.com/percona-operator-for-mysql/pxc/users.html

```
$ kubectl get secrets
NAME                                    TYPE                                  DATA
AGE
cluster1-secrets                         Opaque                                7
8d
cluster1-ssl                            kubernetes.io/tls                     3
8d
cluster1-ssl-internal                   kubernetes.io/tls                     3
8d
default-token-mvlwr                     kuber
8d
internal-cluster1                       Opaqu
8d
percona-xtradb-cluster-operator-token-x2qc9 kuber
8d
```

```
$ kubectl get secrets cluster1-secrets -o yaml
apiVersion: v1
data:
  clustercheck: elNva09De1VQMmU3TURTVg==
  monitor: R3lIMlk5bksyPmQxS3ZkRlI5tOA==
  operator: OElUSkVfZj9HPWNlW2xrKGZi
  proxyadmin: NHlAPUFpQWpNbHBHWlZDVjx9Pw==
  replication: MU9qbkJaSkNpRUZWbUY3aA==
  root: WW1wVVZpVERZN1RKc1lCaXJXCg==
  xtrabackup: S2xHQlNNOEVyNlNwU1BXMkY=
(...)
```

First, retrieve the encoded password for the MySQL *root* user

```
$ echo "WW1wVVZpVERZN1RKc1lCaXJXCg== " | base64
-d
YmpUViTDY7TJsYBirW
```


PERCONA

Accessing the database from within K8s

The “unconventional” way for K8s: directly to one of the nodes

```
$ kubectl exec -it cluster1-pxc-1 -c pxc -- bash

bash-4.4$ mysql -uroot -pYmpUViTDY7TJsYBirW
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 7915
Server version: 8.0.27-18.1 Percona XtraDB Cluster (GPL), Release rel18, Revision ac35177, WSREP
version 26.4.3
…
```

Accessing the database from within K8s

The “unconventional” way for K8s: directly to one of the nodes

one-liner

```
$ kubectl exec -it cluster1-pxc-1 -c pxc -- mysql -uroot -pYmpUViTDY7TJsYBirW
```


PERCONA

Accessing the database from within K8s

https://docs.percona.com/percona-operator-for-mysql/pxc/haproxy-conf.html

Connecting to the cluster through HAproxy

primary

proxy protool

```
$ kubectl get service cluster1-haproxy
NAME                TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)
AGE
cluster1-haproxy    ClusterIP   10.108.6.176   <none>      3306/TCP,3309/TCP,33062/TCP,33060/TCP
```

replicas

```
$ kubectl get service cluster1-haproxy-replicas
NAME                        TYPE        CLUSTER-IP     EXTERNAL-IP    PORT(S)    AGE
cluster1-haproxy-replicas    ClusterIP   10.108.3.134   <none>         3306/TCP   104m
```

```
$ kubectl get services
```


PERCONA

Accessing the database from within K8s

Connecting to the cluster through HAproxy

Docker image with the *mysql* 8.0 command-line client

```
$ kubectl run -i --rm --tty percona-client --image= percona:8.0 --restart=Never -- bash -il
```

```
$ mysql -h cluster1-haproxy -uroot -pYmpUViTDY7TJsYBirW -P3306 -e 'select @@hostname'
+----------------+
| @@hostname     |
+----------------+
| cluster1-pxc-0 |
+----------------+
```

round-robin

```
$ mysql -h cluster1-haproxy-replicas -uroot -pYmpUViTDY7TJsYBirW -P3306 -e 'select @@hostname'
+----------------+
| @@hostname     |
+----------------+
| cluster1-pxc-1 |
+----------------+
```


PERCONA

Access from your notebook: port-forwarding

```
$ kubectl port-forward svc/cluster1-haproxy 8080:3306
Forwarding from 127.0.0.1:8080 -> 3306
Forwarding from [::1]:8080 -> 3306


$ mysql -h 127.0.0.1 -P 8080 -uroot -pYmpUViTDY7TJsYBirW
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 7552
Server version: 8.0.29-21.1 Percona XtraDB Cluster (GPL), Release rel21, Revision 250bc93, WSREP
version 26.4.3
```


PERCONA

Making customizations

PERCONA

Access from outside K8s: Expose

- Uses the K8s load balancer utility
- Allows external access

Edit `deploy/cr.yaml`, uncommenting these 3 lines:

```
 pxc:
    (...)
    expose:
      enabled: true
      type:
LoadBalancer
```

But the right thing to do is to access the database through **HAproxy***:

```
 haproxy:
    (...)
    serviceType: LoadBalancer
    externalTrafficPolicy: Cluster
    serviceAnnotations:
      networking.gke.io/load-balancer-type:
"Internal"
    (...)
    replicasServiceType: LoadBalancer
    replicasExternalTrafficPolicy: Cluster
    replicasServiceAnnotations:
      networking.gke.io/load-balancer-type:
"Internal"
```

```
$ kubectl apply -f deploy/cr.yaml
```

* or ProxySQL, if that's what you chose instead.

PERCONA

Access from outside K8s: Expose

- Just for MySQL access, doesn't go through HAProxy or ProxySQL

```
$ kubectl get services
NAME                              TYPE           CLUSTER-IP      EXTERNAL-IP      PORT(S)                                AGE
cluster1-haproxy                  ClusterIP      10.108.6.176      10.128.0.118     3306/TCP,3309/TCP,33062/TCP,33060/TCP   147m
cluster1-haproxy-replicas         ClusterIP      10.108.3.134      10.128.0.119     3306/TCP                                147m
cluster1-pxc                      ClusterIP      None            <none>           3306/TCP,33062/TCP,33060/TCP            147m
cluster1-pxc-0                    LoadBalancer     10.108.0.55      34.122.165.169    3306:32650/TCP                          7m10s
cluster1-pxc-1                    LoadBalancer     10.108.10.192    35.202.255.108    3306:30203/TCP                          7m9s
cluster1-pxc-2                    LoadBalancer     10.108.6.6       34.170.91.92      3306:32272/TCP                          7m9s
cluster1-pxc-unready              ClusterIP      None            <none>           3306/TCP,33062/TCP,33060/TCP            147m
percona-xtradb-cluster-operator   ClusterIP      10.108.7.100    <none>           443/TCP                                 147m
```

Get more details about a specific service:

```
$ kubectl get services cluster1-pxc-0 -o  wide
```

```
$ kubectl get services cluster1-pxc-0 -o  yaml
```


PERCONA

Customizing MySQL

https://docs.percona.com/percona-operator-for-mysql/pxc/options.html

```
spec:
  pxc:
    configuration: |
      [mysqld]
#      wsrep_debug=CLIENT
#      wsrep_provider_options="gcache.size=1G; gcache.recover=yes"
      innodb_buffer_pool_size=2G
      tmp_table_size=32M
```

```
$ kubectl apply -f deploy/cr.yaml
```

Complete reference for the Custom Resource (CR) options:
https://docs.percona.com/percona-operator-for-mysql/pxc/operator.html

Upgrading MySQL root password

Upgrading MySQL *root* password:

not on macOS

```
$ echo -n "au78yEJEKGs6R96b" | base64
--wrap=0
YXU3OHlF  LR3M2Ujk2Yg==
```

```
$ kubectl patch secret/cluster1-secrets -p '{"data":{"root": "  YXU3OHlFSkVLR3M2Ujk2Yg== "}}'
```

OR:

```
$ kubectl edit
```

OR:

kubectl apply -f update_secret.yaml

```
apiVersion: v1
kind: Secret
metadata:
  name: cluster1-secrets
type: Opaque
data:
  root:  YXU3OHlFSkVLR3M2Ujk2Yg==
```

* Wait a few seconds/minutes to take effect!

```
$ kubectl exec -it cluster1-pxc-0 -c
pxc -- mysql -uroot -p au78yEJEKGs6R96b
```


PERCONA

Upgrading MySQL root password

An alternative way:

```
$ cat deploy/secrets2.yaml
apiVersion: v1
kind: Secret
metadata:
  name: cluster1-secrets
type: Opaque
stringData:
  root: YmpUViTDY7TJsYBirW
```

data
vs
stringData

```
$ kubectl apply -f deploy/secrets2.yaml
```


PERCONA

Cluster management

Cluster management

- Backup and restore
- Horizontal and vertical scaling
- Adding sidecar containers
- Pausing a cluster
- Deleting a cluster

PERCONA

Backup

https://docs.percona.com/percona-operator-for-mysql/pxc/backups.html

```
  backup:
   image:
percona/percona-xtradb-cluster-operator:1.12.0-pxc8.0-backup
(...)
   pitr:
      enabled: false
      storageName: STORAGE-NAME-HERE
      timeBetweenUploads: 60
(...)
   storages:
      s3-us-west:
        type: s3
        verifyTLS: true
(...)
        s3:
          bucket: S3-BACKUP-BUCKET-NAME-HERE
          credentialsSecret: my-cluster-name-backup-s3
          region: us-west-2
     fs-pvc:
        type: filesystem
```

```
(...)
   schedule:
      - name: "sat-night-backup"
        schedule: "0 0 * * 6"
        keep: 3
        storageName:  s3-us-west
      - name: "daily-backup"
        schedule: "0 0 * * *"
        keep: 5
        storageName:  fs-pvc
```


PERCONA

Backup

```
$ kubectl get pods
NAME                                                READY   STATUS                      RESTARTS   AGE
cluster1-haproxy-0                                  2/2     Running                     0          3d
cluster1-haproxy-1                                  2/2     Running                     0          3d
cluster1-haproxy-2                                  2/2     Running                     0          3d
cluster1-pxc-0                                      3/3     Running                     0          3d
cluster1-pxc-1                                      3/3     Running                     0          3d
cluster1-pxc-2                                      3/3     Running                     0          3d
percona-client                                      0/1     Completed                   0          3d
percona-xtradb-cluster-operator-5dbc998f8b-mm7wx    1/1     Running                     0          3d
xb-cron-cluster1-fs-pvc-20221126000-372f8-gzsqx     0/1     Completed                   0          2d18h
xb-cron-cluster1-fs-pvc-20221127000-372f8-l52qm     0/1     Completed                   0          42h
xb-cron-cluster1-fs-pvc-20221128000-372f8-rww6g     0/1     Completed                   0          18h
xb-cron-cluster1-s3-us-west-20221126000-3d2dv-9ttzw 0/1     CreateContainerConfigError  0          2d18h
```

```
$ kubectl get pxc-backups
NAME                                        CLUSTER    STORAGE    DESTINATION                                                    STATUS      COMPLETED
AGE
cron-cluster1-fs-pvc-20221126000-372f8      cluster1   fs-pvc     pvc/xb-cron-cluster1-fs-pvc-20221126000-372f8                  Succeeded   2d17h
2d17h
cron-cluster1-fs-pvc-20221127000-372f8      cluster1   fs-pvc     pvc/xb-cron-cluster1-fs-pvc-20221127000-372f8                  Succeeded   41h
41h
cron-cluster1-fs-pvc-20221128000-372f8      cluster1   fs-pvc     pvc/xb-cron-cluster1-fs-pvc-20221128000-372f8                  Succeeded   17h
```


PERCONA

Backup

1) Set-up credentials to S3 (deploy/backup-s3.yaml):

kubectl apply -f

```
apiVersion: v1
kind: Secret
metadata:
  name: cluster1-s3-credentials
type: Opaque
data:
  AWS_ACCESS_KEY_ID: <base64 encoded>
  AWS_SECRET_ACCESS_KEY: <base64 encoded>
```

2) Configure backup and storage on (deploy/cr.yaml):

kubectl apply -f

```
  backup:
    enabled: true
    (...)
    storages:
      s3-us-west:
        type: s3
        verifyTLS: true
    (...)
    s3:
      bucket: nando-s3
      credentialsSecret: cluster1-s3-credentials
      region: us-east-1
#     endpointUrl: https://…
```

3) Create a manual backup (deploy/backup/backup.yaml):

kubectl apply -f

```
apiVersion: pxc.percona.com/v1
kind: PerconaXtraDBClusterBackup
metadata:
  name: backup1
#  finalizers:
#    - delete-backup
spec:
  pxcCluster: cluster1
  storageName: s3-us-west
```

```
$ kubectl get pxc-backup
NAME      STORAGE      DESTINATION
STATE       COMPLETED   AGE
backup1   s3-us-west
s3://nando-s3/cluster1-2023-04-18-20:08:20-full   Succeeded
27m         27m
```


PERCONA

Restore

Restoring from a backup previously taken

OR

Restoring from a backup from a different source

kubectl apply -f

Configure restore (deploy/backup/restore.yaml):

```
apiVersion: pxc.percona.com/v1
kind: PerconaXtraDBClusterRestore
metadata:
  name: restore1
spec:
  pxcCluster: cluster1
  backupName : backup1
```

```
apiVersion: pxc.percona.com/v1
kind: PerconaXtraDBClusterRestore
metadata:
  name: restore1
spec:
  pxcCluster: cluster1
  backupSource:
    destination:
s3://nando-s3/cluster1-2023-04-19-16:52:35-full
    s3:
      bucket: nando-s3
      credentialsSecret: cluster1-s3-credentials
      region: us-east-1
```

```
$ kubectl get pxc-restore
NAME        CLUSTER     STATUS              COMPLETED
AGE
restore1    cluster1    Stopping Cluster
7s
restore1    cluster1    Restoring
```


PERCONA

Backup - binlogs

PITR: binlog streaming only works with S3-like storage

kubectl apply -f deploy/cr.yaml

```
 backup:
    image: percona/percona-xtradb-cluster-operator:1.12.0-pxc8.0-backup
#    backoffLimit: 6
#    serviceAccountName: percona-xtradb-cluster-operator
#    imagePullSecrets:
#      - name: private-registry-credentials
    pitr:
      enabled: true
      storageName: s3-us-west
      timeBetweenUploads: 60
```

```
$ kubectl get pods
NAME                                                  READY   STATUS      RESTARTS   AGE
cluster1-haproxy-0                                    2/2     Running     0          19m
cluster1-haproxy-1                                    2/2     Running     0          18m
cluster1-haproxy-2                                    2/2     Running     0          18m
cluster1-pitr-76b97bd96f-vzbzx                        1/1     Running     0
2m2s
cluster1-pxc-0                                        3/3     Running     0          19m
cluster1-pxc-1                                        3/3     Running     0          19m
cluster1-pxc-2                                        3/3     Running     0          17m
percona-xtradb-cluster-operator-77bf8b9df5-kgxr6      1/1     Running     0
106m
restore-job-restore1-cluster1-zd8dz                   0/1     Completed   0          20m
```


PERCONA

Restore - PITR

kubectl apply -f deploy/backup/restore.yaml

```yaml
apiVersion: pxc.percona.com/v1
kind: PerconaXtraDBClusterRestore
metadata:
  name: restore2
spec:
  pxcCluster: cluster1
  backupSource:
    destination: s3://nando-s3/cluster1-2023-04-19-16:52:35-full
    s3:
      bucket: nando-s3
      credentialsSecret: cluster1-s3-credentials
      region: us-east-1
  pitr:
    type: transaction
    gtid: "81386cf6-de2c-11ed-972e-5f2edd7af261:7"
    backupSource:
      storageName: s3-us-west
```

```
$ kubectl get pods
NAME
cluster1-haproxy-0
cluster1-haproxy-1
cluster1-haproxy-2
cluster1-pxc-0
cluster1-pxc-1
cluster1-pxc-2
percona-xtradb-cluster-operator-77bf8b9df5-6xkl7
pitr-job-restore2-cluster1-4mwx7
restore-job-restore2-cluster1-9754h
```

```
$ aws s3 ls s3://nando-s3/
                           PRE cluster1-2023-04-19-16:52:35-full.sst_info/
                           PRE  cluster1-2023-04-19-16:52:35-full /
(...)
2023-04-19 13:52:59      25909 cluster1-2023-04-19-16:52:35-full.md5
2023-04-19 13:55:37         38 last-binlog-set-81386cf6-de2c-11ed-972e-5f2edd7af261
```


PERCONA

Restore - retrieving the backup file

The operator includes a script that facilitates copying over backups to a local computer:

```
$ kubectl get pxc-backup
NAME                                  CLUSTER    STORAGE   DESTINATION                                    STATUS      COMPLETED   AGE
cron-cluster1-fs-pvc-20234200040-372f8   cluster1   fs-pvc    pvc/xb-cron-cluster1-fs-pvc-20234200040-372f8   Succeeded   17h         17h
```

```
$ deploy/backup/copy-backup.sh cron-cluster1-fs-pvc-20234200040-372f8/tmp/backup
Log: /tmp/tmp.tCcrpi1R1G/log
pvc/xb-cron-cluster1-fs-pvc-20234200040-372f8pod "backup-access" deleted
pod/backup-access created
Starting pod..[done]

Downloading started
tar: Removing leading `/' from member names
Downloading finished
pod "backup-access" deleted

You can recover data locally with following commands:
    $ service mysqld stop
    $ rm -rf /var/lib/mysql/*
    $ cat /tmp/backup/xtrabackup.stream | xbstream --decompress -x -C /var/lib/mysql
    $ xtrabackup --prepare --target-dir=/var/lib/mysql
    $ chown -R mysql:mysql /var/lib/mysql
    $ service mysqld start
```

```
$ ls /tmp/backup
lost+found  md5sum.txt
sst_info  xtrabackup.stream
```


PERCONA

Restoring the backup in a sidecar container

It's possible to prepare a backup that was taken locally and restore it on a separate container (you may want to do a copy of the pvc hosting the backup first!):

```
apiVersion: v1
kind: Pod
metadata:
  name: backup-access
spec:
  containers:
  - name: xtrabackupview
    image: percona/percona-xtrabackup:8.0.29
    command: ["/bin/sh"]
    args: ["-c", "while true; do trap 'exit 0' SIGINT SIGTERM SIGQUIT SIGKILL; done;"]
    volumeMounts:
    - name: backup
      mountPath: /backup
  restartPolicy: Never
  volumes:
  - name: backup
    persistentVolumeClaim:
      claimName: pvc-c8420f88-f74b-4cc8-b514-a12b54c0e7d4
```

$ kubectl apply -f backup-access.yaml

PERCONA

Restoring the backup in a sidecar container

Decompress and prepare the backup:

```
$ kubectl exec -it backup-access -- sh

sh-4.4# cd /backup
sh-4.4# xbstream -x < xtrabackup.stream
sh-4.4# xtrabackup --prepare --target-dir=/backup
sh-4.4# chown 1001:1001 -R /backup
```

You can now delete that container - the PVC will remain intact:

```
$ kubectl delete pod backup-access
```

Restoring the backup in a sidecar container

Edit the container definition file (*backup-access.yaml*) as to change the image used from *xtrabackup* to a Percona Server one:

```
spec.image: percona/percona-server:8.0.29
```

Deploy the container again:

```
$ kubectl apply -f backup-access.yaml
```

Get inside it, edit the my.cnf sample file included, commenting all variables that are PXC-specific:

```
$ kubectl exec -it backup-access -- sh
sh-4.4# vi /backup/backup-my.cnf
```

and, finally, start a *mysqld* instance with the datadir:

```
$ mysqld --defaults-file=/backup/backup-my.cnf --datadir=/backup
```

Increasing memory, CPU, and storage

https://docs.percona.com/percona-operator-for-mysql/pxc/scaling.html

Filter nando-1 Filter nodes

Name	CPU requested	CPU allocatable	Memory requested	Memory allocatable
gke-nando-1-default-pool-81437ff3-6j1f	2 CPU	3.92 CPU	2.71 GB	12.97 GB
gke-nando-1-default-pool-81437ff3-7w1x	2.11 CPU	3.92 CPU	2.71 GB	12.97 GB
gke-nando-1-default-pool-81437ff3-nsw1	2.09 CPU	3.92 CPU	2.68 GB	12.97 GB

Filter nando-1 Filter persistent volume claims

Name	Phase	Volume	Storage class	Namespace
datadir-cluster1-pxc-0	Bound	pvc-13818b82-b436-4869-a8eb-d7ea4a79728a	standard	pxc
datadir-cluster1-pxc-1	Bound	pvc-9b8b2e8b-fa16-404a-95ef-6ca4ae549c44	standard	pxc
datadir-cluster1-pxc-2	Bound	pvc-3804116c-5389-4cb8-bbf3-ca7d6c8133ce	standard	pxc

PERCONA

Increasing memory, CPU, and storage

```
$ kubectl get pvc
NAME                     STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
datadir-cluster1-pxc-0   Bound    pvc-13818b82-b436-4869-a8eb-d7ea4a79728a   6Gi        RWO            standard       17h
datadir-cluster1-pxc-1   Bound    pvc-9b8b2e8b-fa16-404a-95ef-6ca4ae549c44   6Gi        RWO            standard       16h
datadir-cluster1-pxc-2   Bound    pvc-3804116c-5389-4cb8-bbf3-ca7d6c8133ce   6Gi        RWO            standard       16h
```

```
$ kubectl exec -it cluster1-pxc-0 -c pxc -- df -h
Filesystem      Size  Used Avail Use% Mounted on
overlay          95G  6.0G   89G   7% /
tmpfs            64M     0   64M   0% /dev
tmpfs           7.4G     0  7.4G   0% /sys/fs/cgroup
/dev/sda1        95G  6.0G   89G   7% /tmp
shm              64M     0   64M   0% /dev/shm
/dev/sdb        5.9G  325M  5.5G   6% /var/lib/mysql
tmpfs            13G     0   13G   0% /etc/mysql/vault-keyring-secret
tmpfs            13G   32K   13G   1% /etc/mysql/mysql-users-secret
tmpfs            13G   12K   13G   1% /etc/mysql/ssl-internal
tmpfs            13G   12K   13G   1% /etc/mysql/ssl
tmpfs            13G   12K   13G   1% /run/secrets/kubernetes.io/serviceaccount
tmpfs           7.4G     0  7.4G   0% /proc/acpi
tmpfs           7.4G     0  7.4G   0% /proc/scsi
tmpfs           7.4G     0  7.4G   0% /sys/firmware
```


PERCONA

Increasing memory, CPU, and storage

```
$ cat deploy/cr.yaml
(...)
 pxc:
(...)
   resources:
      requests:
        memory: 1G            -> 2G
        cpu: 600m             -> 1
#       ephemeral-storage: 1G
#     limits:
#       memory: 1G
(...)
   volumeSpec:
(...)
      persistentVolumeClaim:
(...)
        resources:
          requests:
            storage: 6G       -> 10G
```

not enough!

```
$ kubectl apply -f deploy/cr.yaml
```

NOTE: On **AWS** *allowVolumeExpansion* is set to False by default

```
$ kubectl get storageclass
```

```
$ kubectl edit storageclass gp2
```

```
allowVolumeExpansion: true
```

```
$ kubectl edit pvc datadir-cluster1-pxc-2
```

```
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 10G
```

* The datadir PVs should be expanded to the new size automatically within a few minutes. If not, you can always perform a rolling restart.

PERCONA

Scaling the cluster up

1) First, you may need to increase the “number” of nodes in the K8s cluster:

```
$ gcloud container clusters resize nando-1 --project consultants-206215 --zone us-central1-a --num-nodes=5
```

2) Second, choose one of 3 ways to increase the “size” of the cluster:

a)

```
$ cat deploy/cr.yaml
(...)
 pxc:
(...)
  size: 5
```

```
$ kubectl apply -f deploy/cr.yaml
```

b)

```
$ kubectl scale --replicas=5 pxc/cluster1
```

c)

```
$ kubectl edit pxc cluster1

spec:
  pxc:
    size: 5
```


PERCONA

Scaling the cluster down

To scale down, revert the previous steps:

1) Decrease the “size” of the cluster:

```
$ kubectl scale --replicas=3 pxc/cluster1
```

2) Optionally, decrease the “number” of nodes in the K8s cluster:

```
$ gcloud container clusters resize nando-1 --project consultants-206215 --zone us-central1-a --num-nodes=3
```

Sidecars

https://docs.percona.com/percona-operator-for-mysql/pxc/sidecar.html

- Sidecar container allow access to the datafiles of another container
- Can be useful for debugging, monitoring and performance tuning
- pmm-client runs as a sidecar
- Sidecars run as part of the pods

Careful: it does restart the pods!

kubectl apply -f

Adding a sidecar container (`deploy/cr.yaml`):

```
spec:
  pxc:
    (...)
    sidecars:
   - name: sysbench
      image: perconalab/sysbench
      command: ["sleep", "30d"]
```

```
$ kubectl exec -it cluster1-pxc-2 -c sysbench -- bash
```


PERCONA

Sidecars

Running sysbench through HAproxy:

1) First, connect to the database and create the target database:

```
$ kubectl run -i --rm --tty percona-client1 --image=percona:8.0 --restart=Never -- bash -il
# mysql -h cluster1-haproxy -uroot -pYmpUViTDY7TJsYBirW -P3306 -e 'create database sbtest'
```

2) Then, on the *sysbench* session, create the schema and populate the data:

```
# LUA_PATH=/sysbench/sysbench-tpcc/?.lua sysbench-tpcc/tpcc.lua
--mysql-host=cluster1-haproxy --mysql-user=root --mysql-password=YmpUViTDY7TJsYBirW
--mysql-port=3306 --scale=10 --mysql-db=sbtest --db-driver=mysql --force-pk=1   prepare
```

Sidecars

To make things easier, you may

add an environment variable to the sidecar container (`deploy/cr.yaml`):

kubectl apply -f

```
spec:
  pxc:
    (...)
    sidecars:
    - name: sysbench
      command: ["sleep", "30d"]
      env:
      - name: "LUA_PATH"
        value: "/sysbench/sysbench-tpcc/?.lua"
      image: perconalab/sysbench
```

3) Run the workload:

```
$ kubectl exec -it cluster1-pxc-2 -c sysbench -- sysbench-tpcc/tpcc.lua
--mysql-host=cluster1-haproxy --mysql-user=root --mysql-password=YmpUViTDY7TJsYBirW
--mysql-port=3306 --scale=10 --mysql-db=sbtest --db-driver=mysql --force-pk=1   run
```

Sidecars

Or you can simply run Sysbench on a separate pod:

```
$ kubectl run sysbench1 --image=perconalab/sysbench --restart=Never
--env="LUA_PATH=/sysbench/sysbench-tpcc/?.lua" --command --
sysbench-tpcc/tpcc.lua --mysql-host=cluster1-haproxy --mysql-port=3306
--mysql-user=root --mysql-password=YmpUViTDY7TJsYBirW --scale=10
--mysql-db=sbtest --db-driver=mysql
--force-pk=1 run
```


PERCONA

Pausing the cluster

https://docs.percona.com/percona-operator-for-mysql/pxc/pause.html

Gracefully pause the cluster, for some maintenance or to “restart” it:

kubectl apply -f
(deploy/cr.yaml):

```
spec:
  (...)
  allowUnsafeConfigurations: false
  pause: true
```

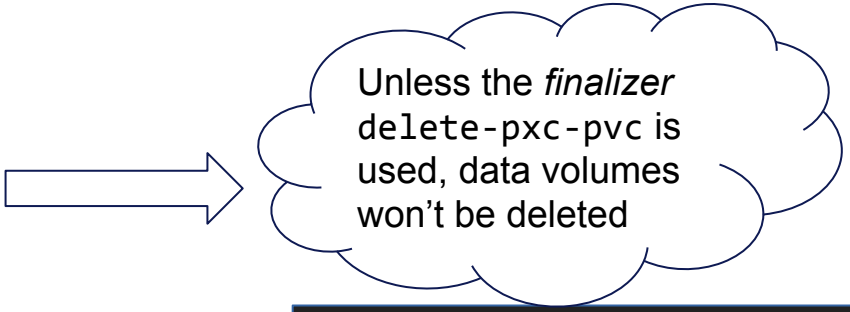
Setting it back to *false* starts the pods.

PERCONA

Deleting

Deleting the PXC cluster:

```
$ kubectl delete -f deploy/cr.yaml
```



Deleting the operator:

```
$ kubectl delete -f deploy/operator.yaml
```

```yaml
apiVersion: pxc.percona.com/v1
kind: PerconaXtraDBCluster
metadata:
  name: cluster1
  finalizers:
    - delete-pxc-pods-in-order
#     - delete-ssl
#     - delete-proxysql-pvc
#     - delete-pxc-pvc
```

Deleting the GKE cluster:

```
$ gcloud container clusters delete nando-1 --project consultants-206215 --zone us-central1-a
```


PERCONA

Deleting

Deleting the cluster

```
$ gcloud container clusters list
NAME         LOCATION       MASTER_VERSION   MASTER_IP      MACHINE_TYPE   NODE_VERSION     NUM_NODES  STATUS
training     us-central1-a  1.23.9-gke.900   34.72.11.250   n1-standard-4  1.23.9-gke.900   3          RUNNING
nando-1      us-central1-a  1.23.12-gke.100  34.121.252.81  n1-standard-4  1.23.12-gke.100  3          RUNNING
```

```
$ gcloud container clusters delete nando-1 --zone us-central1-a
The following clusters will be deleted.
 - [nando-1] in [us-central1-a]

Do you want to continue (Y/n)?  y

Deleting cluster nando-1...done.
Deleted
[https://container.googleapis.com/v1/projects/consultants-206215/zones/us-central1-a/clusters/nando-1].
```

Takes some time to complete!

PERCONA

Troubleshooting

Troubleshooting

- Crash recovery
- Debug and troubleshooting

PERCONA

Troubleshooting

Logs, logs, logs

```
$ kubectl logs cluster1-pxc-0 -c logs [ --previous]
```

```
$ kubectl logs cluster1-pxc-0 -c
pxc
```

```
$ kubectl logs
percona-xtradb-cluster-operator- 846bfc8f54-qfrhk
```

OR:

```
$ kubectl logs
deploy/percona-xtradb-cluster-operator
```

Describe:

```
$ kubectl describe pod
cluster-1-pxc-0
```

```
$ kubectl describe node
gke-node-1-default-pool-81437663-6j1f
```

Deployments (e.g.: changes applied to cr.yaml)

```
$ kubectl rollout status deployments percona-xtradb-cluster-operator
```


PERCONA

Troubleshooting

Percona tools: temporary *Percona-Toolkit* image available

```
$ kubectl run pt2 --image=perconalab/percona-server-mysql-operator:main-toolkit
--restart=Never --command sleep 20d
$ kubectl exec -it pt2 -- sh

sh-4.4$ pt-stalk --no-stalk --iterations=2 --sleep=30 --mysql-only --dest=/tmp/ptstalk --
--host=cluster1-haproxy --user=root --password=au78yEJEKGs6R96b
sh-4.4$ tar cvf /tmp/ptstalk-data.tar /tmp/ptstalk
sh-4.4$ exit

$ kubectl cp pt2:/tmp/ptstalk-data.tar ptstalk-data.tar
```

pt-k8s-debug-collector

```
$ pt-k8s-debug-collector --cluster pxc.percona.com/cluster1 --namespace pxc
```

cluster-dump.tar.gz

PERCONA

Troubleshooting

Inspecting a K8s node:

```
$ kubectl get pods -o wide
NAME                                          READY   STATUS    RESTARTS       AGE    IP            NODE                                    NOMINATED NODE
READINESS GATES
(...)
cluster1-pxc-2                                3/3     Running   0              145m   10.104.2.11   gke-nando-1-default-pool-e6acb66d-6lvz   <none>           <none>
(...)
```

```
$ kubectl debug node/ gke-nando-1-default-pool-e6acb66d-6lvz  -it --image=ubuntu  -- bash

Creating debugging pod node-debugger-gke-nando-1-default-pool-e6acb66d-6lvz-pbxc2 with container debugger on node
gke-nando-1-default-pool-e6acb66d-6lvz.
If you don't see a command prompt, try pressing enter.

root@gke-nando-1-default-pool-e6acb66d-6lvz:/# ps fax|grep "mysql\|haproxy"
   9909 ?        Ss     0:00  \_  haproxy -x /etc/haproxy/pxc/haproxy.sock -W -db -f /etc/haproxy/haproxy-global.cfg -f /etc/haproxy/pxc/haproxy.cfg -p
/etc/haproxy/pxc/haproxy.pid -S /etc/haproxy/pxc/haproxy-main.sock
  13629 ?        Sl     0:11  |   \_ haproxy -sf 486 -x /etc/haproxy/pxc/haproxy.sock -W -db -f /etc/haproxy/haproxy-global.cfg -f
/etc/haproxy/pxc/haproxy.cfg -p /etc/haproxy/pxc/haproxy.pid -S /etc/haproxy/pxc/haproxy-main.sock
  11311 ?        Ssl    0:00  \_ go-cron 0 0 * * * sh -c logrotate -s /opt/percona/logrotate/logrotate.status
/opt/percona/logrotate/logrotate-mysql.conf;/usr/bin/find /var/lib/mysql/ -name GRA_*.log -mtime +7 -delete
  11494 ?        Ssl    2:18  \_  mysqld
 231969 pts/0    S+     0:00      \_ grep --color=auto mysql\|haproxy
```

Sometimes it is possible to install additional software in this temporary, debugging container:

```
root@gke-nando-1-default-pool-e6acb66d-6lvz:/# apt-get update && apt-get install sysstat
```


PERCONA

Troubleshooting

In some particular circumstances, it may be necessary to scale the cluster down to 1 node (*unsafe configurations*), remove the data from the scaled-down nodes, and then scale the cluster back again - for example, to force the cluster to “rebootstrap” from node 0:

```
$ kubectl get pvc
NAME                                STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS
AGE
datadir-cluster1-pxc-0              Bound    pvc-76d2deb4-965a-4a4c-a23c-b12f869a00b8   10Gi       RWO            standard
25h
datadir-cluster1-pxc-1              Bound    pvc-83cce7ad-4691-4d60-87eb-fe853a53068e   10Gi       RWO            standard
```

```
spec:
 allowUnsafeConfigurations:  true
  pxc:
    size:  1
```

```
$ kubectl apply -f deploy/cr.yaml
```

```
$ kubectl delete pvc datadir-cluster1-pxc-1 \
datadir-cluster1-pxc-2
```

```
spec:
 allowUnsafeConfigurations:
false
  pxc:
    size:  3
```

```
$ kubectl apply -f deploy/cr.yaml
```


PERCONA

Troubleshooting

Avoid the restart-on-fail loop by preventing *mysqld* to start on the node:

```
$ kubectl exec -it cluster1-pxc-0 -c logs -- sh -c 'touch /var/lib/mysql/sleep-forever'
```

- Both the *pxc* and *logs* containers have access to the storage with the datadir
- Remove the *sleep-forever* file to allow *mysqld* to start again

There are special debug images for the officially supported PXC releases; they are identified by the *-debug* suffix. It is possible to restart the pods with them:

```
spec:
  pxc:
    image: percona/percona-xtradb-cluster:8.0.29-21.1-debug
```

```
$ kubectl apply -f deploy/cr.yaml
```


PERCONA

Extras

Extras

- ~~Deploying with a Helm chart~~
- ~~Installing PXC in multi-namespace mode~~
- ~~Multi-cluster and multi-region deployment~~
- Monitoring with PMM
- Upgrading

PERCONA

Administrative tasks: monitoring

- PMM → https://docs.percona.com/percona-operator-for-mysql/pxc/monitoring.html
- Add your PMM server's API key (value) to `deploy/secrets.yaml` as *pmmserverkey*

```
  pmm:
    enabled: true
    image: percona/pmm-client:2.32.0
    serverHost: <pmm-server>
#    serverUser: admin
#    pxcParams: "--disable-tablestats-limit=2000"
#    proxysqlParams: "--custom-labels=CUSTOM-LABELS"
    resources:
      requests:
        memory: 150M
        cpu: 300m
```


PERCONA

Administrative tasks: upgrading the operator

https://docs.percona.com/percona-operator-for-mysql/pxc/update.html

- Download the new Custom Resource Definition file (crd.yaml) and the Role-based access control file (rbac.yaml) and apply them

```
$ kubectl apply -f
https://raw.githubusercontent.com/percona/percona-xtradb-cluster-operator/v1.12.0/deploy/crd.yaml

$ kubectl apply -f
https://raw.githubusercontent.com/percona/percona-xtradb-cluster-operator/v1.12.0/deploy/rbac.yaml
```

- Apply a patch to the deployment, specifying the image to use

```
$ kubectl patch deployment percona-xtradb-cluster-operator \
-p'{"spec":{"template":{"spec":{"containers":[{"name":"percona-xtradb-cluster-operator
","image":"percona/percona-xtradb-cluster-operator:1.12.0"}]}}}}'
```

- Track the rollout progress with:

```
$ kubectl rollout status deployments percona-xtradb-cluster-operator
```


PERCONA

Administrative tasks: upgrading PXC

- `updateStrategy: SmartUpdate`
- `upgradeOptions.apply`:
  - `Recommended` → 8.0 for new clusters, otherwise preserve the existing branch
  - `8.0-recommended` or `5.7-recommended` →a way to start with 5.7
  - `Latest` → automatic upgrades pick the most recent version
  - `8.0-latest` or `5.7-latest` →a way to be using the latest in 5.7
  - <version-number> →specify a specific version of PXC
  - `Never` or `Disabled` → disables automatic upgrades

The *SmartUpdate* method relies on `versionServiceEndpoint` →https://check.percona.com

* *manual* and *semi-automatic* update methods reserved for Operator version 1.5.0 and earlier

PERCONA

Thank you!

fernando.laudares@percona.com

chetan.shivashankar@percona.com