MongoDB 社区版 on GKE部署说明

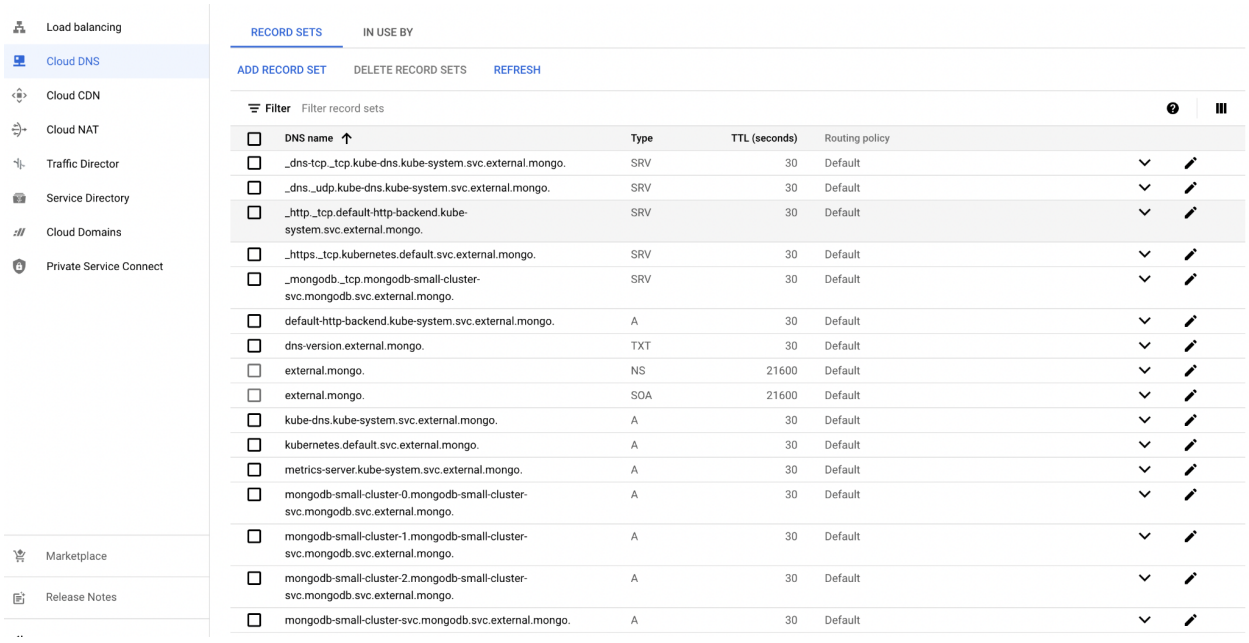
1. GKE集群安装

利用gcloud命令行工具一键创建GKE集群，在创建集群时，可通过“--cluster-dns”参数启用cloud dns，并通过--cluster-dns-scope参数将范围设置为VPC。启用该参数，可通过cloud dns维护kubernetes内pod与service的域名记录，而非Kubernetes默认的core-dns/kube-dns。启用该参数的好处时，可以让位于同一VPC内的GCE实例或者不同GKE集群之间的pod互相解析域名，实现服务发现机制的统一管理。

同时启用workload identity: 做velero 备份

[Using Cloud DNS for GKE | Kubernetes Engine Documentation]

```
gcloud container clusters create <cluster-name> \
  --cluster-dns clouddns --cluster-dns-scope vpc \
  --cluster-dns-domain <example.com>\
  --zone=us-central1-a \
  --machine-type=n2-standard-8 \
  --num-nodes=3
  --workload-pool=<PROJECT_ID>.svc.id.goog
```



Cloud DNS统一管理GKE集群内服务域名记录

使用PD-SSD作为默认的storage class, 并支持storage expansion:

```
kubectl patch sc  standard -p
'{"metadata":{"annotations":{"storageclass.kubernetes.io/is-default-class":"false"}
}}'


kubectl patch sc premium-rwo -p
'{"metadata":{"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}
}'
```

2. MongoDB Operator部署

2.1 克隆MongoDB Operator代码至本地，并切换至MongoDB Operator目录

```
git clone https://github.com/mongodb/mongodb-kubernetes-operator.git
```

2.2 创建CRD资源，并查看crd资源的创建

```
# 创建CRD资源
kubectl apply -f
config/crd/bases/mongodbcommunity.mongodb.com_mongodbcommunity.yaml
查看CRD资源的创建
kubectl get crd/mongodbcommunity.mongodbcommunity.mongodb.com
```

2.3 安装role与role-bindings，并查看资源的创建

```
# 创建role与role-bindings
kubectl apply -k config/rbac/ --namespace <my-namespace>
# 查看资源的创建
kubectl get role mongodb-kubernetes-operator --namespace <my-namespace>
kubectl get rolebinding mongodb-kubernetes-operator --namespace <my-namespace>
kubectl get serviceaccount mongodb-kubernetes-operator --namespace <my-namespace>
```

编辑“config/manager/manager.yaml”配置文件，添加CLUSTER_DOMAIN环境变量，将环境变量的值设置为创建GKE集群时“cluster-dns-domain”指定的域名。

```
env:
- name: WATCH_NAMESPACE
  valueFrom:
    fieldRef:
      fieldPath: metadata.namespace
- name: POD_NAME
  valueFrom:
    fieldRef:
      fieldPath: metadata.name
- name: OPERATOR_NAME
  value: mongodb-kubernetes-operator
- name: AGENT_IMAGE
  value: quay.io/mongodb/mongodb-agent:11.12.0.7388-1
- name: VERSION_UPGRADE_HOOK_IMAGE
  value: quay.io/mongodb/mongodb-kubernetes-operator-version-upgrade-post-start-hook:1.0.4
- name: READINESS_PROBE_IMAGE
  value: quay.io/mongodb/mongodb-kubernetes-readinessprobe:1.0.8
- name: MONGODB_IMAGE
  value: mongo
- name: MONGODB_REPO_URL
  value: docker.io
- name: CLUSTER_DOMAIN
  value: <example.com>
```

编辑operator配置文件

2.4 安装Operator

```
#安装Operator
kubectl create -f config/manager/manager.yaml --namespace <my-namespace>
#查看Operator创建
kubectl get pods --namespace <my-namespace>
```

3. MongoDB副本集部署

3.1 MongoDB副本集部署建议

2.1.1 配置建议

	小规模	中等规模	较大规模
CPU/Memory/Disk	2C/8GB/512GB SSD	4C/16GB/1TB SSD	8C/32G/2TB SSD
开源版本	4.4	4.4	4.4
副本数量	3	3	3

2.1.2 部署多套副本集建议

	部署
开发/测试	与应用开发测试相同的Cluster不同Namespace
生产	单独的Cluster

3.2 MongoDB配置

将“config/samples/mongodb.com_v1_mongodbcommunity_cr.yaml”文件中的

“<your-password-here>”替换成自己要设置的密码。

3.3 安装MongoDB副本集

```
kubectl apply -f config/samples/mongodb.com_v1_mongodbcommunity_cr.yaml --namespace
<my-namespace>
```

3.4 查看MongoDB副本集部署状态

```
kubectl get mongodbcommunity --namespace <my-namespace>
```

3.5 获取MongoDB副本集连接url

```
kubectl get secret <metadata.name>-<auth-db>-<username> -n <my-namespace> \
-o json | jq -r '.data | with_entries(.value |= @base64d)'
```

<metadata.name>, <auth-db>以及<username>的信息参照下表替换成对应的内容。

Variable	Description	Value in Sample
`<metadata.name>`	Name of the MongoDB database resource.	`example-mongodb`
`<auth-db>`	Authentication database where you defined the database user.	`admin`
`<username>`	Username of the database user.	`my-user`

输出内容类似如下，利用输出的字符串连接集群

```
{
  "connectionString.standard":
"mongodb://<username>:<password>@example-mongodb-0.example-mongodb-svc.mongodb.svc.
cluster.local:27017,example-mongodb-1.example-mongodb-svc.mongodb.svc.cluster.local
:27017,example-mongodb-2.example-mongodb-svc.mongodb.svc.cluster.local:27017/admin?
ssl=true",
  "connectionString.standardSrv":
"mongodb+srv://<username>:<password>@example-mongodb-svc.mongodb.svc.cluster.local/
admin?ssl=true",
  "password": "<password>",
  "username": "<username>"
}
```

3.6 可通过调整MongoDBCommunity内members的数量实现集群的扩缩容，通过调整version来实现对MongoDB集群版本的管理。

```
apiVersion: mongodbcommunity.mongodb.com/v1
kind: MongoDBCommunity
metadata:
  name: example-mongodb
spec:
  members: 3
  type: ReplicaSet
  version: "4.2.7"
  security:
    tls:
```

```
    enabled: true
    certificateKeySecretRef:
      name: <tls-secret-name>
    caConfigMapRef:
      name: <tls-ca-configmap-name>
```

4.MongoDB pod容量调整

通过查阅MongoDB Operator配置文档，可知MongoDB通过statefulset机制来对Pod的生命周期进行管理。因此Pod容量调整可通过设置MongoDB CRD描述中Statefulset的部分来实现。

```
statefulSet:
  spec:
    template:
      spec:
        # resources can be specified by applying an override
        # per container name.
        containers:
          - name: mongod
            resources:
              limits:
                cpu: "0.2"
                memory: 250M
              requests:
                cpu: "0.2"
                memory: 200M
          - name: mongodb-agent
            resources:
              limits:
                cpu: "0.2"
                memory: 250M
              requests:
                cpu: "0.2"
                memory: 200M
```

MongoDB CRD statefulset 部分描述

3.1、调整MongoDB 节点CPU与内存限制

在默认配置文件下安装MongoDB集群，Operator会为Master以及每个Replicaset节点设置CPU/Mem 的limit为0.2/250M，设置每个Replicaset节点的requests 0.2/200M。通过查寻其中一个Pod的描述，也可以验证此限制。

```
kubectl describe pod example-mongodb-0  -n mongodb
```

```
  Limits:
    cpu:     1
    memory:  500M
  Requests:
    cpu:     500m
    memory:  400M
```

接下来，将MongoDB的CPU/Mem limit上调至2/2048M，将CPU/Mem Request上调至1/1048M，应用配置文件，查看资源调整情况。

```
  statefulSet:
    spec:
      template:
        spec:
          # resources can be specified by applying an override
          # per container name.
          containers:
            - name: mongod
              resources:
                limits:
                  cpu: "2"
                  memory: 2048M
                requests:
                  cpu: "1"
                  memory: 1024M
```

MongoDB 配置文件

```
kubectl apply -f <mongdb-conf>
```

可参考这个[配置文件]

重新应用配置文件后，MongoDB Operator会触发statefuset重新启动pod。

```
^Cadmin_zzcao_altostrat_com@kubectl:~/mongodb-kubernetes-operator/config/samples$ kubectl get pod -n mongodb -w
NAME                                           READY   STATUS     RESTARTS   AGE
mongodb-kubernetes-operator-774b59c94f-nn5xl   1/1     Running    0          4h4m
mongodb-specify-pod-resources-0                2/2     Running    0          160m
mongodb-specify-pod-resources-1                2/2     Running    0          159m
mongodb-specify-pod-resources-2                0/2     Init:0/2   0          8s
```

自动重启MongoDB Pod

```
Limits:
  cpu:     2
  memory:  2048M
Requests:
  cpu:     1
  memory:  1024M
```

Pod配置调整生效

3.2、调整MongoDB 存储卷

MongoDB Operator通过创建PVC实现数据持久化存储，Operator会为每一个MongoDB节点分配两个PVC存储卷，一个存储卷用于存储数据(默认10G)，一个存储卷用于存储日志(默认2G)。

调整MongoDB存储卷的过程分为三步：

- 更改MongoDB Operator CRD定义文件，修改存储容量, 并重新应用该配置

```
kubectl apply -f config/samples/mongodb.com_v1_mongodbcommunity_cr.yaml
```

- 删除statefulset，MongoDB Operator会自动创建新的statefulset。

```
kubectl delete statefulset example-mongodb
```

- 更改pvc存储卷容量

```
kubectl patch pvc data-volume-example-mongodb-2 -p '{"spec"
:{"resources":{"requests":{"storage": "18Gi"}}}}'

kubectl get pvc
```

```
I have no name!@mongodb-specify-pod-resources-0:/$ df -h
Filesystem      Size  Used Avail Use% Mounted on
overlay          95G  6.2G   89G   7% /
tmpfs            64M     0   64M   0% /dev
tmpfs            15G     0   15G   0% /sys/fs/cgroup
/dev/sdb        9.8G  303M  9.5G   4% /data
/dev/sda1        95G  6.2G   89G   7% /hooks
shm              64M     0   64M   0% /dev/shm
/dev/sdc        2.0G  108K  1.9G   1% /var/log/mongodb-mms-automation
tmpfs            15G   12K   15G   1% /run/secrets/kubernetes.io/serviceaccount
tmpfs            15G     0   15G   0% /proc/acpi
tmpfs            15G     0   15G   0% /proc/scsi
tmpfs            15G     0   15G   0% /sys/firmware
```

存储卷调整之前

```
I have no name!@mongodb-specify-pod-resources-0:/$ df -h
Filesystem      Size  Used Avail Use% Mounted on
overlay          95G  6.2G   89G   7% /
tmpfs            64M     0   64M   0% /dev
tmpfs            15G     0   15G   0% /sys/fs/cgroup
/dev/sdc         28G  601M   28G   3% /data
/dev/sda1        95G  6.2G   89G   7% /hooks
shm              64M     0   64M   0% /dev/shm
/dev/sdb        2.0G  112K  1.9G   1% /var/log/mongodb-mms-automation
tmpfs            15G   12K   15G   1% /run/secrets/kubernetes.io/serviceaccount
tmpfs            15G     0   15G   0% /proc/acpi
tmpfs            15G     0   15G   0% /proc/scsi
tmpfs            15G     0   15G   0% /sys/firmware
```

存储卷调整之后

3.3、调整MongoDB 副本数量

通过更改MongoDB Operator CRD yaml文件中的member数量，可以实现replicaset数量的自动调整。

```yaml
# 调整前的配置
apiVersion: mongodbcommunity.mongodb.com/v1
kind: MongoDBCommunity
metadata:
 name: mongodb-specify-pod-resources
spec:
 members: 3
 type: ReplicaSet
 version: "4.4.0"
 security:
   authentication:
     modes: ["SCRAM"]
```

```
admin_zzcao_altostrat_com@kubectl:~/mongodb-kubernetes-operator/config/samples$ kubectl get pod -n mongodb
NAME                                           READY   STATUS    RESTARTS   AGE
mongodb-kubernetes-operator-774b59c94f-nn5xl   1/1     Running   0          4h24m
mongodb-specify-pod-resources-0                2/2     Running   0          7m39s
mongodb-specify-pod-resources-1                2/2     Running   0          19m
mongodb-specify-pod-resources-2                2/2     Running   0          20m
```

副本数量调整前

```yaml
# 调整后的配置
apiVersion: mongodbcommunity.mongodb.com/v1
kind: MongoDBCommunity
metadata:
 name: mongodb-specify-pod-resources
spec:
 members: 5
 type: ReplicaSet
 version: "4.4.0"
 security:
   authentication:
     modes: ["SCRAM"]
```

```
admin_zzcao_altostrat_com@kubectl:~/mongodb-kubernetes-operator/config/samples$ kubectl get pod -n mongodb
NAME                                           READY   STATUS    RESTARTS   AGE
mongodb-kubernetes-operator-774b59c94f-nn5xl   1/1     Running   0          4h30m
mongodb-specify-pod-resources-0                2/2     Running   0          13m
mongodb-specify-pod-resources-1                2/2     Running   0          25m
mongodb-specify-pod-resources-2                2/2     Running   0          25m
mongodb-specify-pod-resources-3                2/2     Running   0          3m17s
mongodb-specify-pod-resources-4                2/2     Running   0          46s
```

副本数量调整后

5.MongoDB备份和恢复

在比较过诸多MongoDB备份和恢复技术后，本文档建议采纳基于GKE的备份和恢复的底层技术，使用Velero开源工具实现MongoDB的备份和恢复。

4.1. Velero的安装和配置

4.1.1. 安装velero client

```
wget
https://github.com/vmware-tanzu/velero/releases/download/v1.8.1/velero-v1.8.1-linux
-arm.tar.gz
#解压软件包
tar zxvf velero-v1.8.1-linux-amd64.tar.gz
#将二进制文件移动至/usr/local/bin
sudo cp velero /usr/bin/velero
#验证软件包的安装
velero -h
```

4.1.2. 安装velero server

前提条件

- 创建GCS Bucket，velero server会将备份以及恢复的信息存储在GCS Bucket上。
- 启用workload identity，velero server 需要通过workload identity或者GCP service account key获取对GCS Bucket的操作权限。

详细操作请参考https://github.com/vmware-tanzu/velero-plugin-for-gcp#setup 。

```
velero install \
    --provider gcp \
    --plugins velero/velero-plugin-for-gcp:v1.4.0 \
    --bucket charles-mongodb-backup-test \
    --no-secret \
    --sa-annotations
iam.gke.io/gcp-service-account=velero@mongodb-on-gke.iam.gserviceaccount.com \
    --backup-location-config
serviceAccount=velero@mongodb-on-gke.iam.gserviceaccount.com \
```

在安装velero server时，velero client会通过Kubernetes API与Kubernetes集群进行交互，因此若环境中存在多个Kubernetes集群时，确保先将kubectl的context切换至目标Kubernetes集群。

完成velero server的安装后，velero server会以pod的形式运行在Kubernetes的velero namespace下。

```
admin_zzcao_altostrat_com@kubectl:~$ kubectl get pod -n velero
NAME                      READY   STATUS    RESTARTS   AGE
velero-55bd66b8cb-tc7z4   1/1     Running   0          38m
```

4.2. 基于Velero的备份和恢复的操作

4.2.1 创建velero备份任务

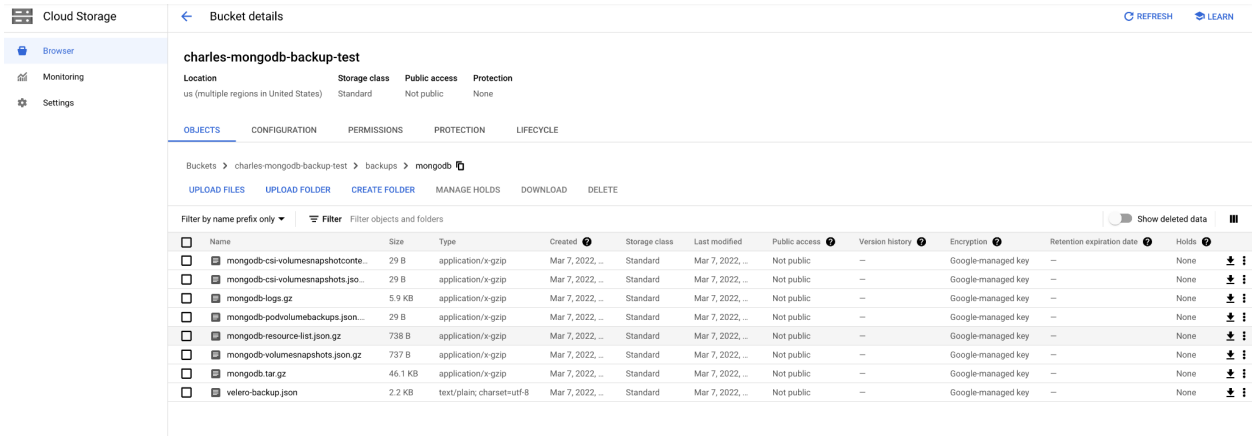
Velero 备份任务可通过 Velero client创建，velero client会与运行在kubernetes集群之上的velero server进行交互，velero server会将备份相关数据上传至GCS Bucket之上。velero会备份两种信息，etced数据以及pvc snapshot。

下面命令会将mongodb namespace上所有的资源都进行备份。

```
velero backup create mongodb-XXXX --include-namespaces mongodb
```

```
admin_zzcao_altostrat_com@kubectl:~$ velero backup describe mongodb
Name:         mongodb
Namespace:    velero
Labels:       velero.io/storage-location=default
Annotations:  velero.io/source-cluster-k8s-gitversion=v1.21.6-gke.1500
              velero.io/source-cluster-k8s-major-version=1
              velero.io/source-cluster-k8s-minor-version=21

Phase:  Completed

Errors:    0
Warnings:  0

Namespaces:
  Included:  mongodb
  Excluded:  <none>

Resources:
  Included:        *
  Excluded:        <none>
  Cluster-scoped:  auto

Label selector:  <none>

Storage Location:  default

Velero-Native Snapshot PVs:  auto

TTL:  720h0m0s

Hooks:  <none>

Backup Format Version:  1.1.0

Started:    2022-03-07 06:46:31 +0000 UTC
Completed:  2022-03-07 06:46:40 +0000 UTC

Expiration:  2022-04-06 06:46:31 +0000 UTC

Total items to be backed up:  46
Items backed up:              46

Velero-Native Snapshots:  6 of 6 snapshots completed successfully (specify --details for more information)
```

查看velero备份计划



GCS存储桶上查看备份数据信息

4.2.2 创建velero恢复任务

```
velero restore create --from-backup mongodb-XX
```

```
admin_zzcao_altostrat_com@kubectl:~$ velero restore describe mongodb-20220307065407
Name:         mongodb-20220307065407
Namespace:    velero
Labels:       <none>
Annotations:  <none>

Phase:                                  InProgress
Estimated total items to be restored:  45
Items restored so far:                 2

Started:    2022-03-07 06:54:07 +0000 UTC
Completed:  <n/a>

Backup:  mongodb

Namespaces:
  Included:  all namespaces found in the backup
  Excluded:  <none>

Resources:
  Included:        *
  Excluded:        nodes, events, events.events.k8s.io, backups.velero.io, restores.velero.io, resticrepositories.velero.io
  Cluster-scoped:  auto

Namespace mappings:  <none>

Label selector:  <none>

Restore PVs:  auto

Preserve Service NodePorts:  auto
```

查看velero恢复进度

```
admin_zzcao_altostrat_com@kubectl:~$ kubectl get all -n mongodb
NAME                                           READY   STATUS    RESTARTS   AGE
pod/mongodb-kubernetes-operator-774b59c94f-k7f7v   1/1     Running   0          52m
pod/mongodb-second-cluster-0                   2/2     Running   0          48m
pod/mongodb-second-cluster-1                   2/2     Running   0          52m
pod/mongodb-second-cluster-2                   2/2     Running   0          52m

NAME                                  TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)     AGE
service/mongodb-second-cluster-svc    ClusterIP   None         <none>        27017/TCP   52m

NAME                                          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mongodb-kubernetes-operator   1/1     1            1           52m

NAME                                                     DESIRED   CURRENT   READY   AGE
replicaset.apps/mongodb-kubernetes-operator-774b59c94f   1         1         1       52m

NAME                                       READY   AGE
statefulset.apps/mongodb-second-cluster    3/3     52m
```

Kubernetes成功恢复资源

```
admin_zzcao_altostrat_com@kubectl:~$ kubectl get pvc -n mongodb
NAME                                 STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
data-volume-mongodb-second-cluster-0   Bound    pvc-20c2fb73-6e97-4d68-8ada-f9bcefea65ca   501Gi      RWO            standard-rwo   53m
data-volume-mongodb-second-cluster-1   Bound    pvc-37144e82-10b6-442a-840f-62b0bae76c21   501Gi      RWO            standard-rwo   53m
data-volume-mongodb-second-cluster-2   Bound    pvc-4091dc64-dd47-4117-8496-e92385da3c5a   501Gi      RWO            standard-rwo   53m
logs-volume-mongodb-second-cluster-0   Bound    pvc-dc1b5e62-974b-47b2-82df-9f572f562d94   48Gi       RWO            standard-rwo   53m
logs-volume-mongodb-second-cluster-1   Bound    pvc-2c19bb3b-5502-4f49-a5eb-0c12ec65e4d7   48Gi       RWO            standard-rwo   53m
logs-volume-mongodb-second-cluster-2   Bound    pvc-25e5ec05-e1f5-4549-bbb7-00bb2a1a7753   48Gi       RWO            standard-rwo   53m
```

Kubernetes成功恢复PVC

```
mongodb-second-cluster:SECONDARY> db.stats(1024*1024*1024)
{
        "db" : "workload",
        "collections" : 3,
        "views" : 0,
        "objects" : 88516287,
        "avgObjSize" : 2797.9441070658554,
        "dataSize" : 230.65472356136888,
        "storageSize" : 133.48491668701172,
        "indexes" : 3,
        "indexSize" : 2.5137100219726562,
        "totalSize" : 135.99862670898438,
        "scaleFactor" : 1073741824,
        "fsUsedSize" : 137.09616088867188,
        "fsTotalSize" : 493.07440185546875,
        "ok" : 1,
        "$clusterTime" : {
                "clusterTime" : Timestamp(1646636720, 1),
                "signature" : {
                        "hash" : BinData(0,"tsFaF5XfrRhqjWvQ1K6DYKWS3VE="),
                        "keyId" : NumberLong("7070698346079322116")
                }
        },
        "operationTime" : Timestamp(1646636720, 1)
}
```

登录MongoDB查看数据恢复状态

4.3. 备份和恢复的基准时间

通过自动化脚本向MongoDB集群内分别注入100G与300G数据，集群的备份与恢复时间记录如下。

MongoDB容量	备份创建时间	集群恢复时间
100GB	5s	6 mins
300GB	5s	8 mins

4.4. 备份和恢复的测试用例

测试备份和恢复前，可以参考MongoDB造数步骤在MongoDB的部署中打入数据，并准备相应存储容量。

6.MongoDB日志与监控

可通过集成Google Cloud Managed Prometheus服务实现对MongoDB集群的监控。

5.1、安装Managed Prometheus Collector，Collector以Pod方式运行，是精简版的Prometheus。Collector收集exporter提供的指标，并上传至Managed Prometehus云端。

```
kubectl apply -f
https://raw.githubusercontent.com/GoogleCloudPlatform/prometheus-engine/v0.3.0/manifests/setup.yaml

kubectl apply -f
https://raw.githubusercontent.com/GoogleCloudPlatform/prometheus-engine/v0.3.1/manifests/operator.yaml
```

5.2、安装MongoDB Exporter，MongoDB Exporter用于收集MongoDB集群指标，并将指标提供给Managed Prometheus Collector。

```
helm install mongodb-exporter prometheus-community/prometheus-mongodb-exporter   -f value.yml  -n mongodb
```

Value.yml内容入下，将<mongodb-url>替换成3.5步骤的输出。

```
mongodb:
  uri: "<mongodb-url>"
serviceMonitor:
  enabled: false
```

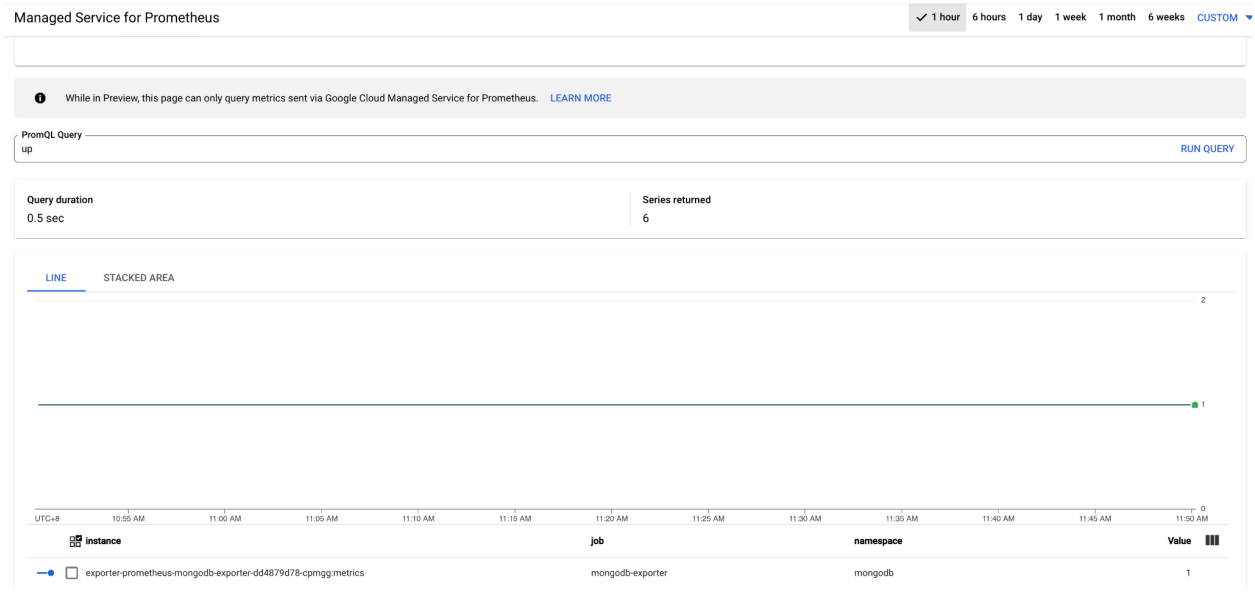
5.3、创建Pod Monitoring资源，PodMonitoring用于定义metric target，Prometehus collector通过pod monitoring定义收集相应endpoint的指标。

```
kubectl apply -f mongodb-exporter-pm.yml
```

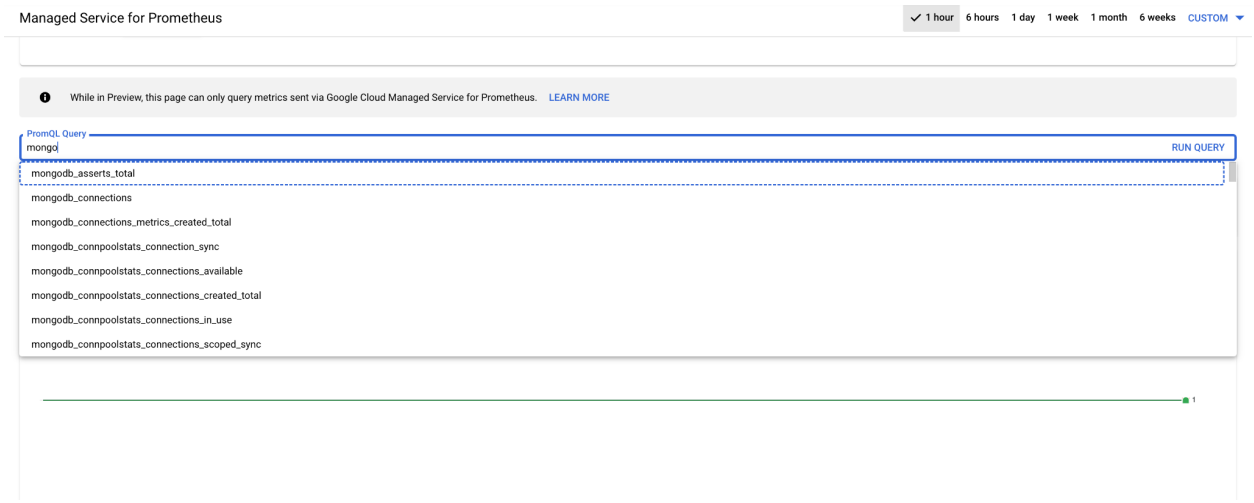
Mongodb-exporter-pm.yml 内容如下：

```
apiVersion: monitoring.googleapis.com/v1alpha1
kind: PodMonitoring
metadata:
  name: prom-example
spec:
  selector:
    matchLabels:
      app.kubernetes.io/instance: mongodb-exporter
  endpoints:
  - port: metrics
    interval: 30s
```

5.4、到Google Cloud Prometheus控制台上查看指标

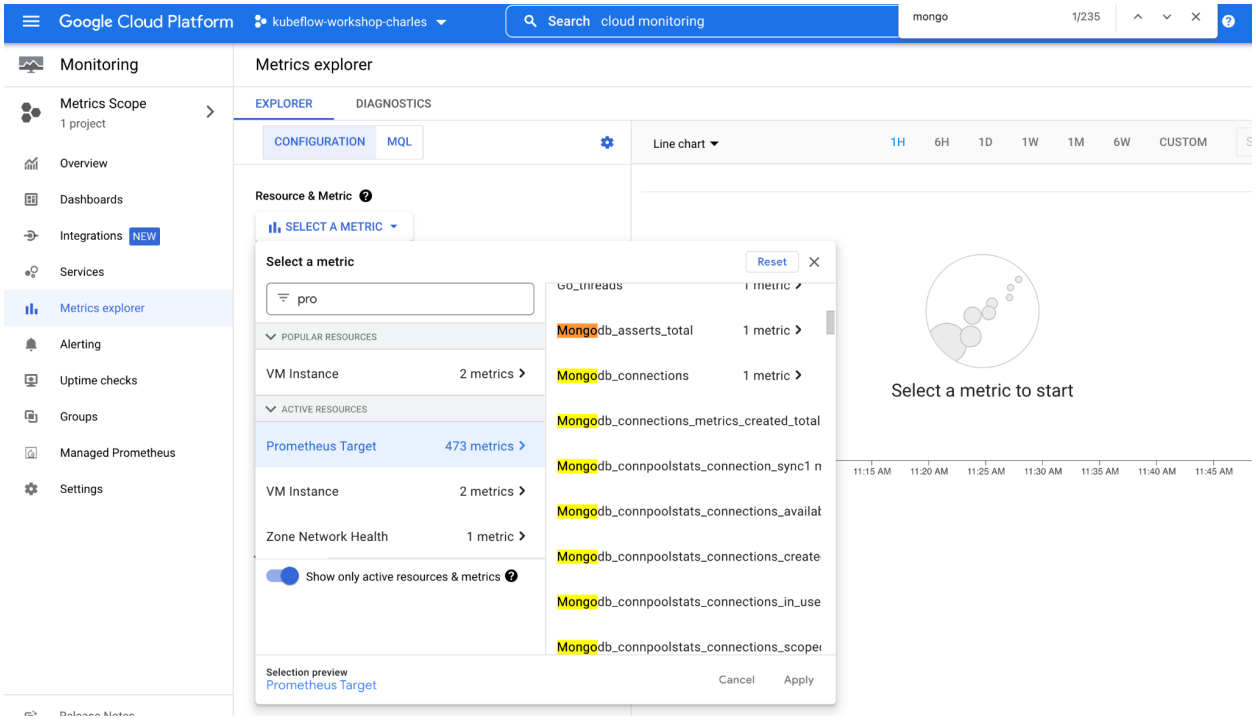


确认exporter实例已被Prometheus服务发现



查看mongodb相关指标

5.5、除了Prometheus控制台，也可以通过cloud monitoring的方式查看MongoDB指标



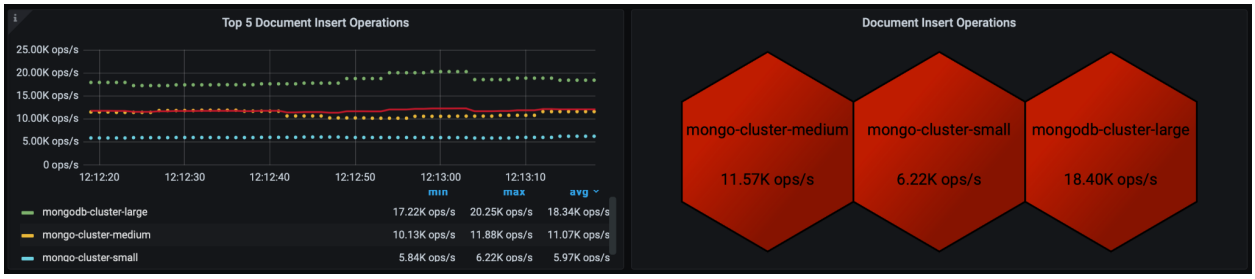
通过cloud monitoring的方式查看MongoDB指标

7.MongoDB性能基准测试

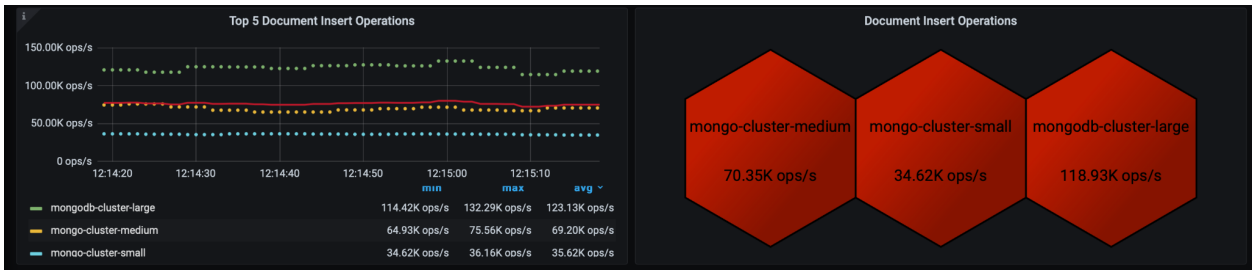
MongoDB性能基准测试参考(Percona)

本次测试采用开源工具POCDriver对不同规模的MongoDB集群进行性能测试，每个集群分别测试了单个batch插入1条文档，单个batch插入10条文档，单个batch插入100条文档，性能测试结果记录如。

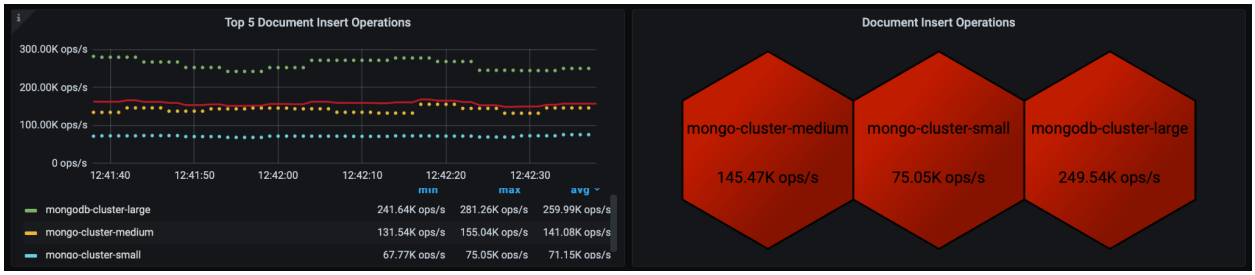
集群规模	集群配置	写QPS（batch:1）	写QPS（batch:10）	写QPS（batch:100）
small	2C/8GB /512GB SSD	6.22k	34.62k	75.05k
medium	4C/16GB/1TB SSD	11.57k	70.35k	145.47k
large	8C/32G/2TB SSD	18.40k	118.93k	249.54k



MongoDB集群Qps（1document/batch）

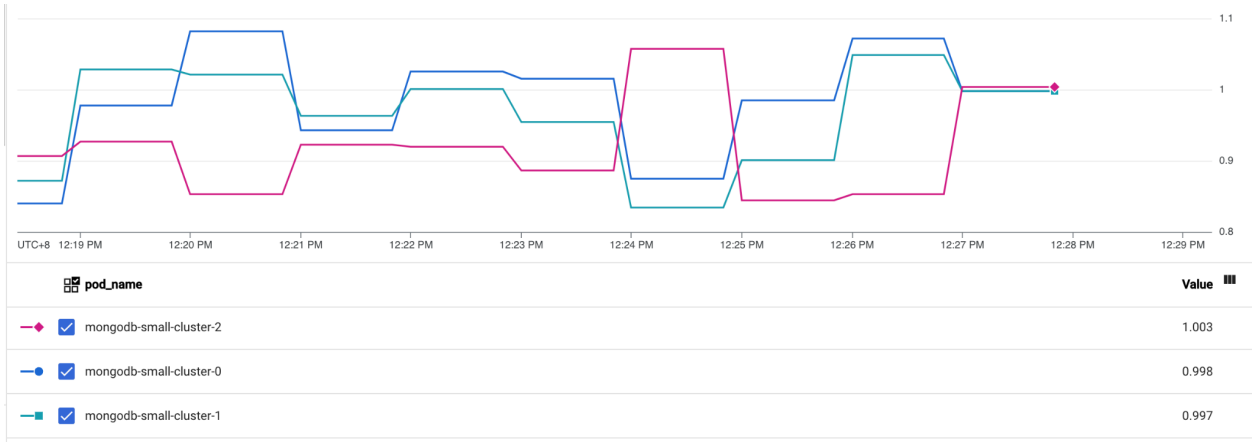


MongoDB集群Qps（10documents/batch）



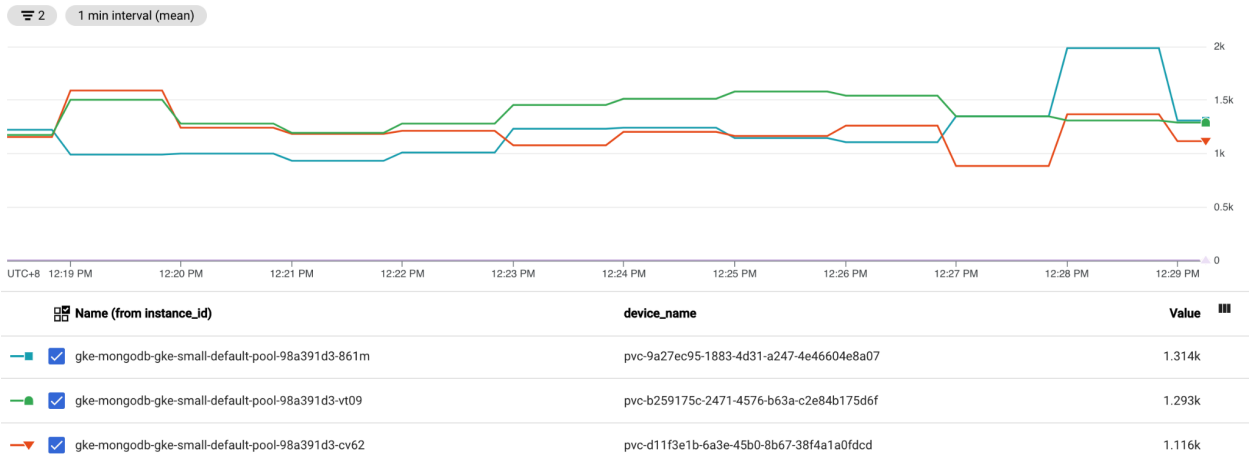
MongoDB集群Qps（100documents/batch）

容器与虚拟机资源占用情况

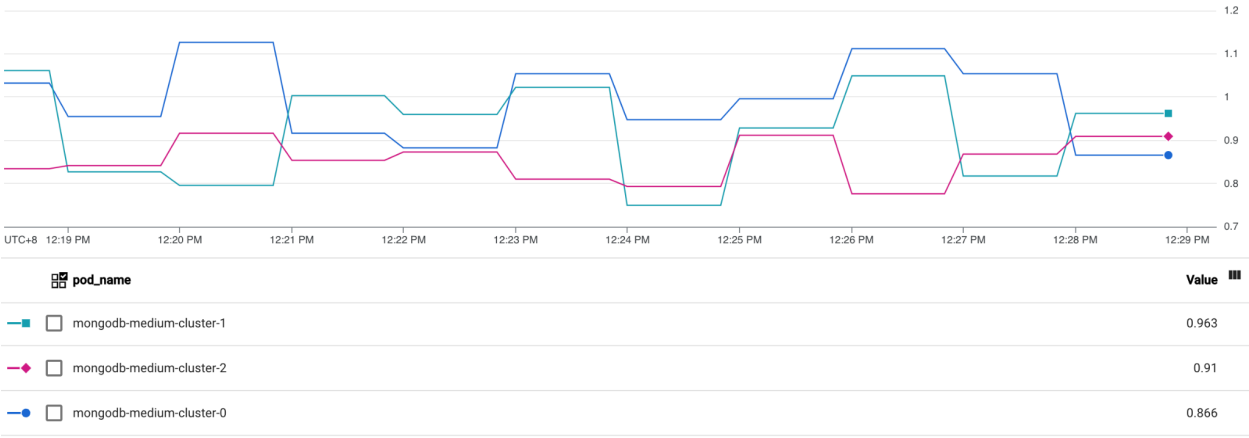


CPU利用率（small）

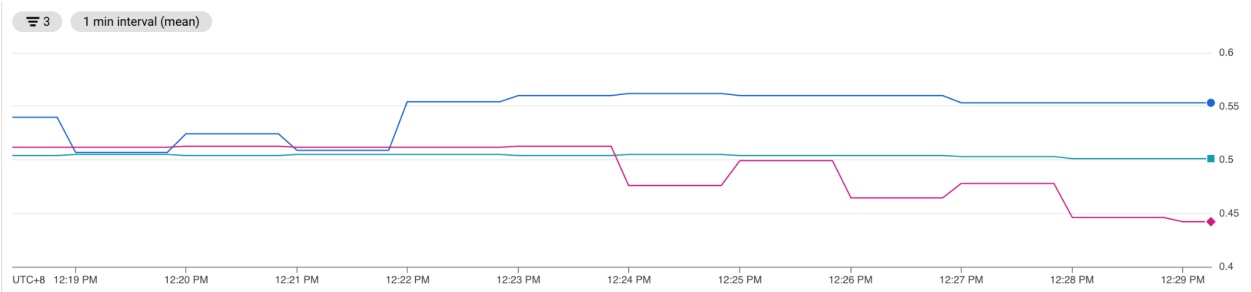
内存利用率（small）



存储IOPS（ Small ）

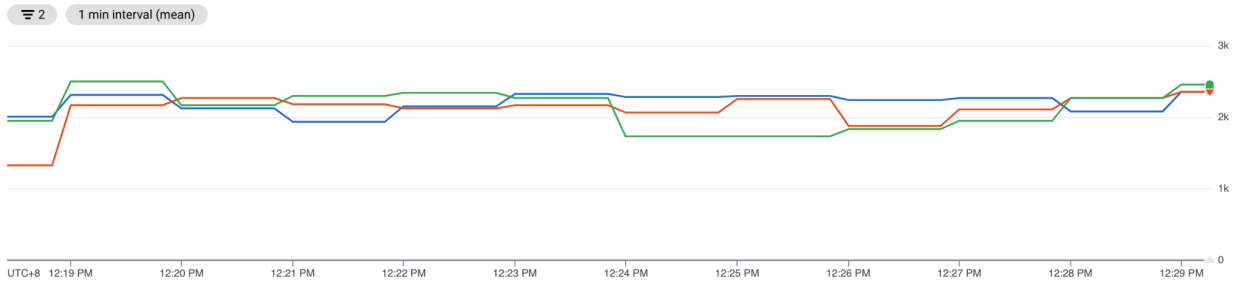


CPU利用率（medium）



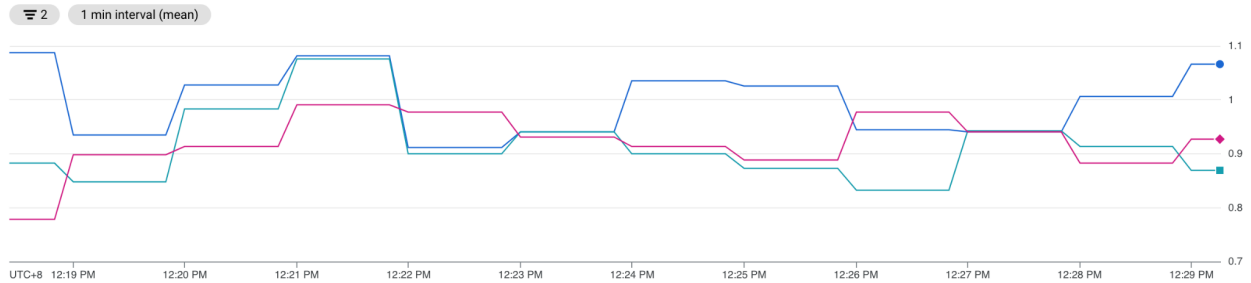
pod_name	Value
mongodb-medium-cluster-0	0.5534
mongodb-medium-cluster-1	0.5012
mongodb-medium-cluster-2	0.4423

内存利用率（medium）



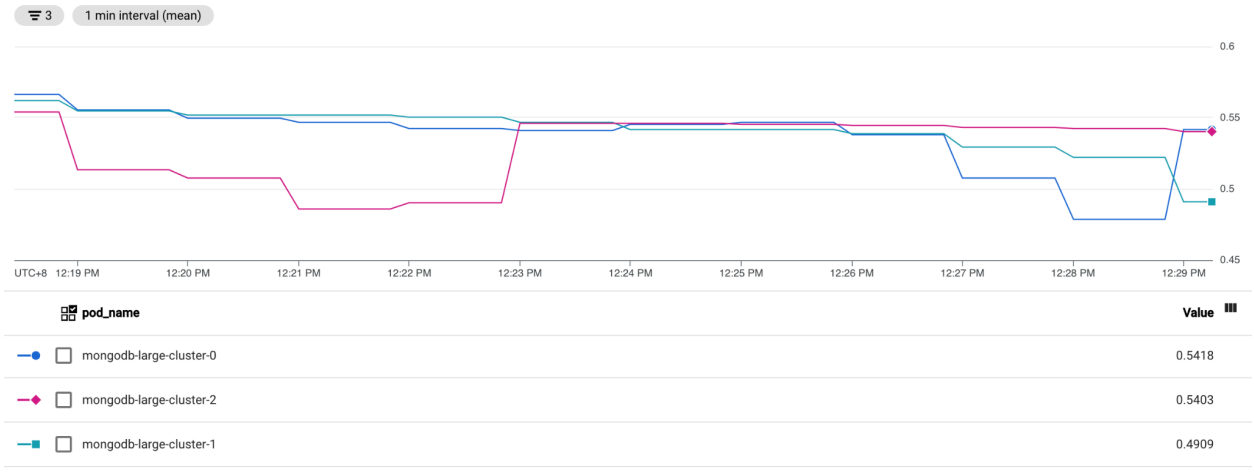
Name (from instance_id)	device_name	Value
gke-mongodb-gke-medium-default-pool-c4db6214-4kb0	pvc-b0978233-3ec0-46ed-8a9b-896c11dfdb5a	2.46k
gke-mongodb-gke-medium-default-pool-c4db6214-p844	pvc-7b809b5c-5e18-4e9c-8e97-d5a4be2d5f74	2.36k
gke-mongodb-gke-medium-default-pool-c4db6214-3hc1	pvc-68f02d78-b997-4125-a6a2-fcec38c2a2e1	2.35k

存储IOPS（medium）



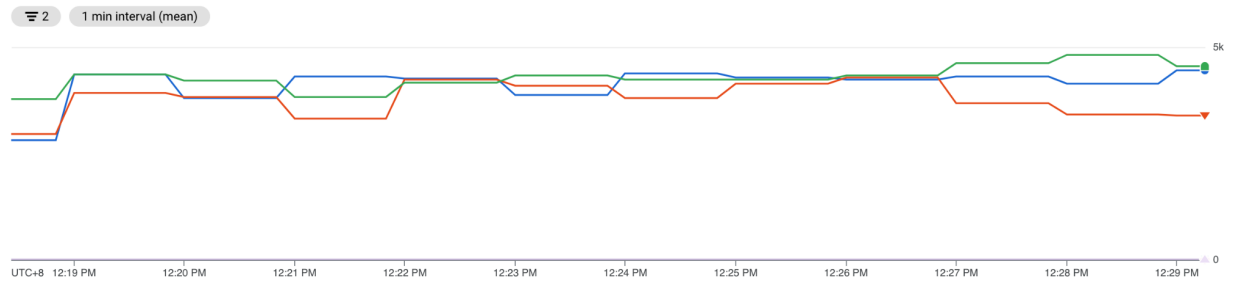
pod_name	Value
mongodb-large-cluster-0	1.066
mongodb-large-cluster-2	0.927
mongodb-large-cluster-1	0.87

CPU利用率（large）



pod_name	Value
mongodb-large-cluster-0	0.5418
mongodb-large-cluster-2	0.5403
mongodb-large-cluster-1	0.4909

内存利用率（large）



Name (from instance_id)	device_name	Value
gke-mongodb-gke-large-default-pool-ff649a6d-m71x	pvc-cac830b0-dd87-4131-93b5-ec74cb87033d	4.55k
gke-mongodb-gke-large-default-pool-ff649a6d-1gcx	pvc-00641405-a8de-464a-91d0-ea146b12bb27	4.45k
gke-mongodb-gke-large-default-pool-ff649a6d-79pn	pvc-e19738d9-fcb0-4438-8caa-e79358766c2b	3.38k

存储IOPS（ large ）

MongoDB性能测试方法请参考：

- 使用基准测试工具sysbench的文档
- 使用基准测试工具POCDriver的文档

1. 访问MongoDB on GKE

7.1. 从内部访问

7.2. 从外部访问

https://appfleet.com/blog/orchaestrating-mongodb-over-kubernetes/

8.附录:

8.1 参考的MongoDBCommunity配置文件

```
---
apiVersion: mongodbcommunity.mongodb.com/v1
kind: MongoDBCommunity
metadata:
  name: example-mongodb
spec:
  members: 3
  type: ReplicaSet
  version: "4.4.13"
  security:
    authentication:
      modes: ["SCRAM"]
  users:
    - name: my-user
      db: admin
      passwordSecretRef: # a reference to the secret that will be used to generate
the user's password
        name: my-user-password
      roles:
        - name: clusterAdmin
          db: admin
        - name: userAdminAnyDatabase
          db: admin
      scramCredentialsSecretName: my-scram
  statefulSet:
    spec:
      template:
        spec:
          # resources can be specified by applying an override
          # per container name.
          containers:
```

```
            - name: mongod
              resources:
                limits:
                  cpu: "2"
                  memory: 2000M
                requests:
                  cpu: "0.2"
                  memory: 200M
            - name: mongodb-agent
              resources:
                limits:
                  cpu: "0.2"
                  memory: 250M
                requests:
                  cpu: "0.2"
                  memory: 200M
      # Specifies a size for the data volume different from the default 10Gi
      volumeClaimTemplates:
        - metadata:
            name: data-volume
          spec:
            accessModes: [ "ReadWriteOnce" ]
            resources:
              requests:
                storage: 15Gi
  additionalMongodConfig:
    storage.wiredTiger.engineConfig.journalCompressor: zlib


# the user credentials will be generated from this secret
# once the credentials are generated, this secret is no longer required
---
apiVersion: v1
kind: Secret
metadata:
  name: my-user-password
type: Opaque
stringData:
  password: <your-password-here>
```

8.2 从vpc内部访问mongo cluster

- 获取连接串

```
k get secret example-mongodb-admin-my-user \
-o json | jq -r '.data | with_entries(.value |= @base64d)
```

```
{
  "connectionString.standard":
"mongodb://my-user:changeme@example-mongodb-0.example-mongodb-svc.default.svc.goodvm.local:27017,example-mongodb-1.example-mongodb-svc.default.svc.goodvm.local:27017,example-mongodb-2.example-mongodb-svc.default.svc.goodvm.local:27017/admin?ssl=false",
  "connectionString.standardSrv":
"mongodb+srv://my-user:changeme@example-mongodb-svc.default.svc.goodvm.local/admin?ssl=false",
  "password": "changeme",
  "username": "my-user"
}
```

- Mongosh 连接后创建管理员用户

```
mongosh
"mongodb://my-user:changeme@example-mongodb-0.example-mongodb-svc.default.svc.goodvm.local:27017,example-mongodb-1.example-mongodb-svc.default.svc.goodvm.local:27017,example-mongodb-2.example-mongodb-svc.default.svc.goodvm.local:27017/admin?ssl=false"
```

```
example-mongodb [primary] admin> db.createUser(
   {
    user: 'admin',
    pwd: 'password',
    roles: [ { role: 'root', db: 'admin' } ]
   }
 );
```

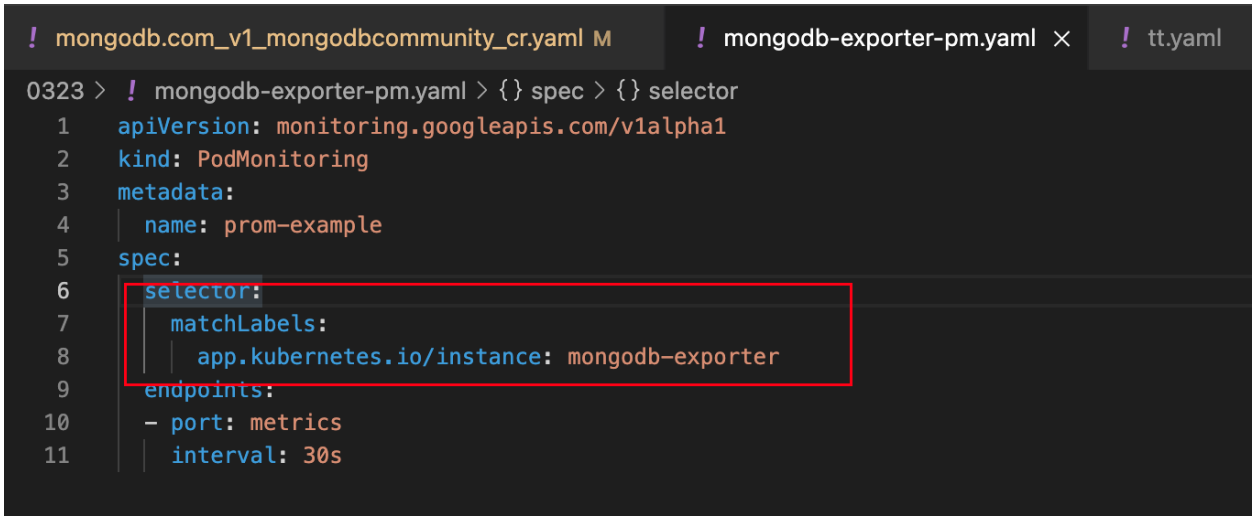
8.3 velero workload identity模式安装参考

```
kubectl create namespace velero
kubectl create serviceaccount velero --namespace velero
gcloud iam service-accounts add-iam-policy-binding
cliu201-sa@cliu201.iam.gserviceaccount.com --role roles/iam.workloadIdentityUser
--member "serviceAccount:cliu201.svc.id.goog[velero/velero]"
```

```
velero install \n
--provider gcp \n
--plugins velero/velero-plugin-for-gcp:v1.4.0 \n
--bucket cliu201-us \n
--no-secret \n
--sa-annotations
iam.gke.io/gcp-service-account=cliu201-sa@cliu201.iam.gserviceaccount.com \n
--backup-location-config serviceAccount=cliu201-sa@cliu201.iam.gserviceaccount.com

# 手动添加service account annotation
kubectl annotate serviceaccount velero
iam.gke.io/gcp-service-account=cliu201-sa@cliu201.iam.gserviceaccount.com
--namespace velero
```

8.4 mongodb-export service和collector的关联关系建立

```
❯ k describe svc mongodb-exporter-prometheus-mongodb-exporter
Name:              mongodb-exporter-prometheus-mongodb-exporter
Namespace:         default
Labels:            app.kubernetes.io/instance=mongodb-exporter
                   app.kubernetes.io/managed-by=Helm
                   app.kubernetes.io/name=prometheus-mongodb-exporter
                   helm.sh/chart=prometheus-mongodb-exporter-2.9.0
Annotations:       cloud.google.com/neg: {"ingress":true}
                   meta.helm.sh/release-name: mongodb-exporter
                   meta.helm.sh/release-namespace: default
Selector:          app.kubernetes.io/instance=mongodb-exporter,app.kubernetes.io/name=prometheus-mongodb-exporter
Type:              ClusterIP
IP Family Policy:  SingleStack
IP Families:       IPv4
IP:                10.44.13.5
IPs:               10.44.13.5
Port:              metrics  9216/TCP
TargetPort:        metrics/TCP
Endpoints:         10.40.2.5:9216
Session Affinity:  None
Events:            <none>
 ~/k8s/0323
❯
```


! mongodb.com_v1_mongodbcommunity_cr.yaml M
! mongodb-exporter-pm.yaml
! tt.yaml
0323 > ! mongodb-exporter-pm.yaml > {} spec > {} selector
1 apiVersion: monitoring.googleapis.com/v1alpha1
2 kind: PodMonitoring
3 metadata:
4 name: prom-example
5 spec:
6 selector:
7 matchLabels:
8 app.kubernetes.io/instance: mongodb-exporter
9 endpoints:
10 - port: metrics
11 interval: 30s