

Backup and Restore Milvus Data: From Standalone Cluster to Distributed Cluster

- Access the first Standalone Cluster and create a pod using `vector-db` namespace.

```
1  apiVersion: v1
2  kind: Pod
3  metadata:
4    name: milvus-backup-pod
5    namespace: vector-db
6  spec:
7    containers:
8      - name: python-container
9        image: python:3.8
10       command: [ "sleep", "infinity" ]
```

- Exec into the pod using

```
1  kubectl exec -it milvus-backup-pod -n vector-db -- bash
```

- Install `milvus-cli` and `vim`

```
1  apt-get update
2  pip install milvus-cli
3  apt-get install vim
```

- Create a new file to connect to the Milvus service.

```
1  vim connect-milvus.py
```

- Paste this script in the new Python file.

```
1  from pymilvus import connections, utility
2
3  # Connect to Milvus
4  connections.connect(
5      alias="default",
6      host='prod-milvus-milvus.vector-db.svc.cluster.local',
7      port='19530'
8  )
9
10 # Check connection by listing collections
11 try:
12     collections = utility.list_collections()
13     print("Successfully connected to Milvus. Collections:")
14     for collection in collections:
15         print(f"- {collection}")
16 except Exception as e:
17     print(f"Failed to connect to Milvus: {e}")
```

- Run the Python script using

```
1  python3 connect-milvus.py
```

- Once you run the script, the pod gets connected to the Milvus service and it will list all the collections which are available in the Milvus.
- Now we need to clone the Milvus Backup tool repository to take the backup of Milvus data in Minio.

```

1 apt-get update
2 apt install git -y
3 apt-get install wget -y
4 wget https://go.dev/dl/go1.21.4.linux-amd64.tar.gz -O go.tar.gz
5 tar -xzvf go.tar.gz -C /usr/local
6 echo export PATH=$HOME/go/bin:/usr/local/go/bin:$PATH >> ~/.profile
7 source ~/.profile
8 go version
9
10 git clone https://github.com/zilliztech/milvus-backup.git
11 cd milvus-backup
12 go get
13 go build

```

- Edit the backup configuration file

```

1 cd configs/
2 vim backup.yaml

```

```

1 # Configures the system log output.
2 log:
3   level: info # Only supports debug, info, warn, error, panic, or fatal. Default 'info'.
4   console: true # whether print log to console
5   file:
6     rootPath: "logs/backup.log"
7
8 http:
9   simpleResponse: true
10
11 # milvus proxy address, compatible to milvus.yaml
12 milvus:
13   address: prod-milvus-milvus.vector-db.svc.cluster.local
14   port: 19530
15   authorizationEnabled: false
16   # tls mode values [0, 1, 2]
17   # 0 is close, 1 is one-way authentication, 2 is two-way authentication.
18   tlsMode: 0
19   user: "username"
20   password: "password"
21
22 # Related configuration of minio, which is responsible for data persistence for Milvus.
23 minio:
24   # cloudProvider: "minio" # deprecated use storageType instead
25   storageType: "minio" # support storage type: local, minio, s3, aws, gcp, ali(aliyun), azure, tc(tencent)
26
27   address: prod-milvus-minio.vector-db.svc.cluster.local # Address of MinIO/S3
28   port: 9000 # Port of MinIO/S3
29   accessKeyID: minioadmin # accessKeyID of MinIO/S3
30   secretAccessKey: minioadmin # MinIO/S3 encryption string
31   useSSL: false # Access to MinIO/S3 with SSL
32   useIAM: false
33   iamEndpoint: ""
34
35   bucketName: "prod-milvus" # Milvus Bucket name in MinIO/S3, make it the same as your milvus instance
36   rootPath: "files" # Milvus storage root path in MinIO/S3, make it the same as your milvus instance
37
38   # only for azure
39   backupAccessKeyID: # accessKeyID of MinIO/S3

```

```

40 backupSecretAccessKey: # MinIO/S3 encryption string
41
42 backupBucketName: "prod-milvus-new" # Bucket name to store backup data. Backup data will store to
backupBucketName/backupRootPath
43 backupRootPath: "backup" # Rootpath to store backup data. Backup data will store to
backupBucketName/backupRootPath
44
45 backup:
46   maxSegmentGroupSize: 6G
47
48   parallelism:
49     # collection level parallelism to backup
50     backupCollection: 4
51     # thread pool to copy data. reduce it if blocks your storage's network bandwidth
52     copydata: 128
53     # Collection level parallelism to restore
54     restoreCollection: 2
55
56   # keep temporary files during restore, only use to debug
57   keepTempFiles: false
58
59   # Pause GC during backup through Milvus Http API.
60   gcPause:
61     enable: false
62     seconds: 7200
63   address: http://prod-milvus-milvus.vector-db.svc.cluster.local:9091

```

- Take the backup of Milvus Collection in Minio using

```

1 cd ..
2 ./milvus-backup create -n prod_milvus_new

```

- Backup of Milvus collection in Minio is successful.
- Now we will verify in Minio whether the collections are present or not.
- Download all the required packages (minio-client) to connect to Minio through pod.

```

1 apt-get update
2 apt install wget -y
3 wget https://dl.min.io/client/mc/release/linux-amd64/mc
4 chmod +x mc
5 mv mc /usr/local/bin/

```

- Connect to minio client using

```
1 mc alias set myminio http://prod-milvus-minio.vector-db.svc.cluster.local:9000 minioadmin minioadmin
```

- List the minio buckets

```
1 mc ls myminio
```

- If minio bucket called `prod-milvus-new` is present, then the Data has been successfully backed up in Minio.
- Now we will copy the collections from Minio to S3 buckets.
- In AWS s3 service in `us-east-2` region create an empty bucket with name `prod-milvus-new`
- Now connect to the AWS S3 service inside pod using

```
1 mc alias set s3 https://s3.us-east-2.amazonaws.com AWS_Access_Key_ID AWS_Access_Secret_Key
```

- Copy the Minio Bucket (`prod-milvus-new`) data to AWS S3 (`prod-milvus-new`) bucket.

```
1 mc mirror myminio/prod-milvus-new s3/prod-milvus-new
```

- Verify in AWS S3 service UI whether the data has been copied or not in the bucket.
- To Restore Milvus Data in the Distributed Milvus Cluster follow the below steps
- Create a new Cluster from Scratch.
- Take the access of the Cluster
- Install Cert Manager using

```
1 kubectl apply -f https://github.com/jetstack/cert-manager/releases/download/v1.5.3/cert-manager.yaml
```

- Check if the cert manager pods are running or not using

```
1 kubectl get pods -n cert-manager
```

- Install Milvus Operator using Helm

```
1 helm install milvus-operator \  
2   -n milvus-operator --create-namespace \  
3   --wait --wait-for-jobs \  
4   https://github.com/zilliztech/milvus-operator/releases/download/v1.0.0/milvus-operator-1.0.0.tgz
```

- You will see the output similar to the following after the installation process ends.

```
1 NAME: milvus-operator  
2 LAST DEPLOYED: Thu Jul  7 13:18:40 2022  
3 NAMESPACE: milvus-operator  
4 STATUS: deployed  
5 REVISION: 1  
6 TEST SUITE: None  
7 NOTES:  
8 Milvus Operator Is Starting, use `kubectl get -n milvus-operator deploy/milvus-operator` to check if its  
  successfully installed  
9 If Operator not started successfully, check the checker's log with `kubectl -n milvus-operator logs  
  job/milvus-operator-checker`  
10 Full Installation doc can be found in https://github.com/zilliztech/milvus-  
  operator/blob/main/docs/installation/installation.md  
11 Quick start with `kubectl apply -f https://raw.githubusercontent.com/zilliztech/milvus-  
  operator/main/config/samples/milvus_minimum.yaml`  
12 More samples can be found in https://github.com/zilliztech/milvus-operator/tree/main/config/samples  
13 CRD Documentation can be found in https://github.com/zilliztech/milvus-operator/tree/main/docs/CRD
```

- You can check if the Milvus Operator pod is running as follows:

```
1 kubectl get pods -n milvus-operator
```

- Deploy the Milvus Cluster using Kind Milvus

```
1 kubectl create ns vector-db  
2 vim milvus-cluster.yaml
```

```
1 apiVersion: milvus.io/v1beta1  
2 kind: Milvus  
3 metadata:  
4   annotations:  
5     milvus.io/dependency-values-merged: "true"  
6   creationTimestamp: "2024-07-15T13:20:19Z"
```

```
7   finalizers:
8     - milvus.milvus.io/finalizer
9   generation: 6
10  labels:
11    app: milvus
12    milvus.io/operator-version: 0.9.17
13  name: prod-milvus
14  namespace: vector-db
15  resourceVersion: "525373"
16  uid: 5a6ecdb4-5674-4683-8091-2d98f0331dbd
17 spec:
18   components:
19     dataCoord:
20       paused: false
21       replicas: 1
22     dataNode:
23       paused: false
24       replicas: 1
25     disableMetric: false
26     image: milvusdb/milvus:v2.4.5
27     imageUpdateMode: rollingUpgrade
28     indexCoord:
29       paused: false
30       replicas: 1
31     indexNode:
32       paused: false
33       replicas: 1
34     metricInterval: ""
35     paused: false
36     proxy:
37       paused: false
38       replicas: 1
39       serviceType: LoadBalancer
40     queryCoord:
41       paused: false
42       replicas: 1
43     queryNode:
44       paused: false
45       replicas: 1
46     rootCoord:
47       paused: false
48       replicas: 1
49     standalone:
50       paused: false
51       replicas: 0
52       serviceType: ClusterIP
53   config: {}
54   dependencies:
55     customMsgStream: null
56     etcd:
57       endpoints:
58         - prod-milvus-etcd.vector-db:2379
59       external: false
60       inCluster:
61         deletionPolicy: Retain
62       values:
63         auth:
64           rbac:
```

```
65         enabled: false
66     autoCompactionMode: revision
67     autoCompactionRetention: "1000"
68     enabled: true
69     extraEnvVars:
70     - name: ETCD_QUOTA_BACKEND_BYTES
71       value: "4294967296"
72     - name: ETCD_HEARTBEAT_INTERVAL
73       value: "500"
74     - name: ETCD_ELECTION_TIMEOUT
75       value: "2500"
76     image:
77       pullPolicy: IfNotPresent
78       repository: milvusdb/etcd
79       tag: 3.5.5-r4
80     livenessProbe:
81       enabled: true
82       timeoutSeconds: 10
83     name: etcd
84     pdb:
85       create: false
86     persistence:
87       accessMode: ReadWriteOnce
88       enabled: true
89       size: 10Gi
90       storageClass: null
91     readinessProbe:
92       enabled: true
93       periodSeconds: 20
94       timeoutSeconds: 10
95     replicaCount: 3
96     service:
97       peerPort: 2380
98       port: 2379
99       type: ClusterIP
100   kafka:
101     external: false
102   msgStreamType: pulsar
103   natsmq:
104     persistence:
105       persistentVolumeClaim:
106         spec: null
107   pulsar:
108     endpoint: prod-milvus-pulsar-proxy.vector-db:6650
109     external: false
110     inCluster:
111       deletionPolicy: Retain
112     values:
113       affinity:
114         anti_affinity: false
115       autorecovery:
116         resources:
117           requests:
118             cpu: 1
119             memory: 512Mi
120       bookkeeper:
121         configData:
122           PULSAR_GC: |
```

```

123         -Dio.netty.leakDetectionLevel=disabled -Dio.netty.recycler.linkCapacity=1024 -XX:+UseG1GC -
XX:MaxGCPauseMillis=10 -XX:+ParallelRefProcEnabled -XX:+UnlockExperimentalVMOptions -XX:+DoEscapeAnalysis -
XX:ParallelGCThreads=32 -XX:ConcGCThreads=32 -XX:G1NewSizePercent=50 -XX:+DisableExplicitGC -XX:-ResizePLAB -
XX:+ExitOnOutOfMemoryError -XX:+PerfDisableSharedMem -XX:+PrintGCDetails
124         PULSAR_MEM: |
125             -Xms4096m -Xmx4096m -XX:MaxDirectMemorySize=8192m
126         nettyMaxFrameSizeBytes: "104867840"
127     pdb:
128         usePolicy: false
129     replicaCount: 3
130     resources:
131         requests:
132             cpu: 1
133             memory: 2048Mi
134     volumes:
135         journal:
136             name: journal
137             size: 100Gi
138         ledgers:
139             name: ledgers
140             size: 200Gi
141     broker:
142         component: broker
143         configData:
144             PULSAR_GC: |
145                 -Dio.netty.leakDetectionLevel=disabled -Dio.netty.recycler.linkCapacity=1024 -
XX:+ParallelRefProcEnabled -XX:+UnlockExperimentalVMOptions -XX:+DoEscapeAnalysis -XX:ParallelGCThreads=32 -
XX:ConcGCThreads=32 -XX:G1NewSizePercent=50 -XX:+DisableExplicitGC -XX:-ResizePLAB -
XX:+ExitOnOutOfMemoryError
146         PULSAR_MEM: |
147             -Xms4096m -Xmx4096m -XX:MaxDirectMemorySize=8192m
148         backlogQuotaDefaultLimitGB: "8"
149         backlogQuotaDefaultRetentionPolicy: producer_exception
150         defaultRetentionSizeInMB: "-1"
151         defaultRetentionTimeInMinutes: "10080"
152         maxMessageSize: "104857600"
153         subscriptionExpirationTimeMinutes: "3"
154         ttlDurationDefaultInSeconds: "259200"
155     pdb:
156         usePolicy: false
157     podMonitor:
158         enabled: false
159     replicaCount: 1
160     resources:
161         requests:
162             cpu: 1.5
163             memory: 4096Mi
164     components:
165         autorecovery: true
166         bookkeeper: true
167         broker: true
168         functions: false
169         proxy: true
170         pulsar_manager: false
171         toolset: false
172         zookeeper: true
173     enabled: true
174     fullnameOverride: ""

```

```
175     images:
176     autorecovery:
177         pullPolicy: IfNotPresent
178         repository: apache/pulsar/pulsar
179         tag: 2.8.2
180     bookie:
181         pullPolicy: IfNotPresent
182         repository: apache/pulsar/pulsar
183         tag: 2.8.2
184     broker:
185         pullPolicy: IfNotPresent
186         repository: apache/pulsar/pulsar
187         tag: 2.8.2
188     proxy:
189         pullPolicy: IfNotPresent
190         repository: apache/pulsar/pulsar
191         tag: 2.8.2
192     pulsar_manager:
193         pullPolicy: IfNotPresent
194         repository: apache/pulsar/pulsar-manager
195         tag: v0.1.0
196     zookeeper:
197         pullPolicy: IfNotPresent
198         repository: apache/pulsar/pulsar
199         tag: 2.8.2
200     maxMessageSize: "5242880"
201     monitoring:
202         alert_manager: false
203         grafana: false
204         node_exporter: false
205         prometheus: false
206     name: pulsar
207     persistence: true
208     proxy:
209         configData:
210             PULSAR_GC: |
211                 -XX:MaxDirectMemorySize=2048m
212             PULSAR_MEM: |
213                 -Xms2048m -Xmx2048m
214             httpNumThreads: "100"
215     pdb:
216         usePolicy: false
217     podMonitor:
218         enabled: false
219     ports:
220         pulsar: 6650
221     replicaCount: 1
222     resources:
223         requests:
224             cpu: 1
225             memory: 2048Mi
226     service:
227         type: ClusterIP
228     pulsar_manager:
229         service:
230             type: ClusterIP
231     pulsar_metadata:
232         component: pulsar-init
```



```
233     image:
234       repository: apache/pulsar/pulsar
235       tag: 2.8.2
236   rbac:
237     enabled: false
238     limit_to_namespace: true
239     psp: false
240   zookeeper:
241     configData:
242       PULSAR_GC: |
243         -Dcom.sun.management.jmxremote -Djute.maxbuffer=10485760 -XX:+ParallelRefProcEnabled -
XX:+UnlockExperimentalVMOptions -XX:+DoEscapeAnalysis -XX:+DisableExplicitGC -XX:+PerfDisableSharedMem -
Dzookeeper.forceSync=no
244       PULSAR_MEM: |
245         -Xms1024m -Xmx1024m
246     pdb:
247       usePolicy: false
248   resources:
249     requests:
250       cpu: 0.3
251       memory: 1024Mi
252   rocksmq:
253     persistence:
254       persistentVolumeClaim:
255         spec: null
256   storage:
257     endpoint: prod-milvus-minio.vector-db:9000
258     external: false
259     inCluster:
260       deletionPolicy: Retain
261     values:
262       accessKey: minioadmin
263       bucketName: milvus-bucket
264       enabled: true
265       existingSecret: ""
266       iamEndpoint: ""
267     image:
268       pullPolicy: IfNotPresent
269       tag: RELEASE.2023-03-20T20-16-18Z
270   livenessProbe:
271     enabled: true
272     failureThreshold: 5
273     initialDelaySeconds: 5
274     periodSeconds: 5
275     successThreshold: 1
276     timeoutSeconds: 5
277   mode: distributed
278   name: minio
279   persistence:
280     accessMode: ReadWriteOnce
281     enabled: true
282     existingClaim: ""
283     size: 500Gi
284     storageClass: null
285   podDisruptionBudget:
286     enabled: false
287   readinessProbe:
288     enabled: true
```

```
289         failureThreshold: 5
290         initialDelaySeconds: 5
291         periodSeconds: 5
292         successThreshold: 1
293         timeoutSeconds: 1
294     region: ""
295     resources:
296         requests:
297             memory: 2Gi
298     rootPath: file
299     secretKey: minioadmin
300     service:
301         port: 9000
302         type: ClusterIP
303     startupProbe:
304         enabled: true
305         failureThreshold: 60
306         initialDelaySeconds: 0
307         periodSeconds: 10
308         successThreshold: 1
309         timeoutSeconds: 5
310     useIAM: false
311     useVirtualHost: false
312     secretRef: prod-milvus-minio
313     type: MinIO
314     hookConfig: null
315     mode: cluster
316 status:
317     componentsDeployStatus:
318         datacoord:
319             generation: 1
320             image: milvusdb/milvus:v2.4.5
321             status:
322                 availableReplicas: 1
323                 conditions:
324                     - lastTransitionTime: "2024-07-15T13:25:47Z"
325                       lastUpdateTime: "2024-07-15T13:25:47Z"
326                       message: Deployment has minimum availability.
327                       reason: MinimumReplicasAvailable
328                       status: "True"
329                       type: Available
330                     - lastTransitionTime: "2024-07-15T13:25:36Z"
331                       lastUpdateTime: "2024-07-15T13:25:47Z"
332                       message: ReplicaSet "prod-milvus-milvus-datacoord-8549f5c97f" has successfully
333                         progressed.
334                       reason: NewReplicaSetAvailable
335                       status: "True"
336                       type: Progressing
337             observedGeneration: 1
338             readyReplicas: 1
339             replicas: 1
340             updatedReplicas: 1
341         datanode:
342             generation: 1
343             image: milvusdb/milvus:v2.4.5
344             status:
345                 availableReplicas: 1
346                 conditions:
```

```

347 - lastTransitionTime: "2024-07-15T13:25:48Z"
348   lastUpdateTime: "2024-07-15T13:25:48Z"
349   message: Deployment has minimum availability.
350   reason: MinimumReplicasAvailable
351   status: "True"
352   type: Available
353 - lastTransitionTime: "2024-07-15T13:25:36Z"
354   lastUpdateTime: "2024-07-15T13:25:48Z"
355   message: ReplicaSet "prod-milvus-milvus-datanode-85688b669b" has successfully
356     progressed.
357   reason: NewReplicaSetAvailable
358   status: "True"
359   type: Progressing
360   observedGeneration: 1
361   readyReplicas: 1
362   replicas: 1
363   updatedReplicas: 1
364 indexcoord:
365   generation: 1
366   image: milvusdb/milvus:v2.4.5
367   status:
368     availableReplicas: 1
369     conditions:
370     - lastTransitionTime: "2024-07-15T13:25:47Z"
371       lastUpdateTime: "2024-07-15T13:25:47Z"
372       message: Deployment has minimum availability.
373       reason: MinimumReplicasAvailable
374       status: "True"
375       type: Available
376     - lastTransitionTime: "2024-07-15T13:25:36Z"
377       lastUpdateTime: "2024-07-15T13:25:47Z"
378       message: ReplicaSet "prod-milvus-milvus-indexcoord-bf9fff78f" has successfully
379         progressed.
380       reason: NewReplicaSetAvailable
381       status: "True"
382       type: Progressing
383     observedGeneration: 1
384     readyReplicas: 1
385     replicas: 1
386     updatedReplicas: 1
387 indexnode:
388   generation: 1
389   image: milvusdb/milvus:v2.4.5
390   status:
391     availableReplicas: 1
392     conditions:
393     - lastTransitionTime: "2024-07-15T13:25:47Z"
394       lastUpdateTime: "2024-07-15T13:25:47Z"
395       message: Deployment has minimum availability.
396       reason: MinimumReplicasAvailable
397       status: "True"
398       type: Available
399     - lastTransitionTime: "2024-07-15T13:25:36Z"
400       lastUpdateTime: "2024-07-15T13:25:47Z"
401       message: ReplicaSet "prod-milvus-milvus-indexnode-8444968896" has successfully
402         progressed.
403       reason: NewReplicaSetAvailable
404       status: "True"

```

```
405     type: Progressing
406     observedGeneration: 1
407     readyReplicas: 1
408     replicas: 1
409     updatedReplicas: 1
410 proxy:
411   generation: 1
412   image: milvusdb/milvus:v2.4.5
413   status:
414     availableReplicas: 1
415     conditions:
416     - lastTransitionTime: "2024-07-15T13:25:57Z"
417       lastUpdateTime: "2024-07-15T13:25:57Z"
418       message: Deployment has minimum availability.
419       reason: MinimumReplicasAvailable
420       status: "True"
421       type: Available
422     - lastTransitionTime: "2024-07-15T13:25:36Z"
423       lastUpdateTime: "2024-07-15T13:25:57Z"
424       message: ReplicaSet "prod-milvus-milvus-proxy-5f85ff97b8" has successfully
425         progressed.
426       reason: NewReplicaSetAvailable
427       status: "True"
428       type: Progressing
429     observedGeneration: 1
430     readyReplicas: 1
431     replicas: 1
432     updatedReplicas: 1
433 querycoord:
434   generation: 1
435   image: milvusdb/milvus:v2.4.5
436   status:
437     availableReplicas: 1
438     conditions:
439     - lastTransitionTime: "2024-07-15T13:25:47Z"
440       lastUpdateTime: "2024-07-15T13:25:47Z"
441       message: Deployment has minimum availability.
442       reason: MinimumReplicasAvailable
443       status: "True"
444       type: Available
445     - lastTransitionTime: "2024-07-15T13:25:36Z"
446       lastUpdateTime: "2024-07-15T13:25:47Z"
447       message: ReplicaSet "prod-milvus-milvus-querycoord-c69f8d6bf" has successfully
448         progressed.
449       reason: NewReplicaSetAvailable
450       status: "True"
451       type: Progressing
452     observedGeneration: 1
453     readyReplicas: 1
454     replicas: 1
455     updatedReplicas: 1
456 querynode:
457   generation: 2
458   image: milvusdb/milvus:v2.4.5
459   status:
460     availableReplicas: 1
461     conditions:
462     - lastTransitionTime: "2024-07-15T13:26:37Z"
```

```
463     lastUpdateTime: "2024-07-15T13:26:37Z"
464     message: Deployment has minimum availability.
465     reason: MinimumReplicasAvailable
466     status: "True"
467     type: Available
468 - lastTransitionTime: "2024-07-15T13:25:36Z"
469     lastUpdateTime: "2024-07-15T13:26:37Z"
470     message: ReplicaSet "prod-milvus-milvus-querynode-0-d56bdfc8f" has successfully
471         progressed.
472     reason: NewReplicaSetAvailable
473     status: "True"
474     type: Progressing
475     observedGeneration: 2
476     readyReplicas: 1
477     replicas: 1
478     updatedReplicas: 1
479 rootcoord:
480     generation: 1
481     image: milvusdb/milvus:v2.4.5
482     status:
483         availableReplicas: 1
484         conditions:
485         - lastTransitionTime: "2024-07-15T13:25:47Z"
486             lastUpdateTime: "2024-07-15T13:25:47Z"
487             message: Deployment has minimum availability.
488             reason: MinimumReplicasAvailable
489             status: "True"
490             type: Available
491         - lastTransitionTime: "2024-07-15T13:25:36Z"
492             lastUpdateTime: "2024-07-15T13:25:47Z"
493             message: ReplicaSet "prod-milvus-milvus-rootcoord-845b66bcc9" has successfully
494                 progressed.
495             reason: NewReplicaSetAvailable
496             status: "True"
497             type: Progressing
498         observedGeneration: 1
499         readyReplicas: 1
500         replicas: 1
501         updatedReplicas: 1
502 standalone:
503     generation: 1
504     image: milvusdb/milvus:v2.4.5
505     status:
506         conditions:
507         - lastTransitionTime: "2024-07-15T13:25:36Z"
508             lastUpdateTime: "2024-07-15T13:25:36Z"
509             message: Deployment has minimum availability.
510             reason: MinimumReplicasAvailable
511             status: "True"
512             type: Available
513         - lastTransitionTime: "2024-07-15T13:25:36Z"
514             lastUpdateTime: "2024-07-15T13:25:36Z"
515             message: ReplicaSet "prod-milvus-milvus-standalone-7fb8548bdc" has successfully
516                 progressed.
517             reason: NewReplicaSetAvailable
518             status: "True"
519             type: Progressing
520     observedGeneration: 1
```

```

521 conditions:
522 - lastTransitionTime: "2024-07-15T13:21:37Z"
523   message: Etcd endpoints is healthy
524   reason: EtcdReady
525   status: "True"
526   type: EtcdReady
527 - lastTransitionTime: "2024-07-15T13:21:08Z"
528   reason: StorageReady
529   status: "True"
530   type: StorageReady
531 - lastTransitionTime: "2024-07-15T13:25:36Z"
532   message: MsgStream is ready
533   reason: MsgStreamReady
534   status: "True"
535   type: MsgStreamReady
536 - lastTransitionTime: "2024-07-15T13:27:05Z"
537   message: All Milvus components are healthy
538   reason: ReasonMilvusHealthy
539   status: "True"
540   type: MilvusReady
541 - lastTransitionTime: "2024-07-15T13:27:05Z"
542   message: Milvus components are all updated
543   reason: MilvusComponentsUpdated
544   status: "True"
545   type: MilvusUpdated
546 endpoint: 4.213.203.14:19530
547 ingress:
548   loadBalancer: {}
549 observedGeneration: 6
550 replicas:
551   dataCoord: 1
552   dataNode: 1
553   indexCoord: 1
554   indexNode: 1
555   proxy: 1
556   queryCoord: 1
557   rootCoord: 1
558 rollingModeVersion: 2
559 status: Healthy

```

- Apply this file in `vector-db` namespace using

```
1 kubectl create -n vector-db -f milvus-cluster.yaml
```

- Check if all the pods are in running state

```
1 kubectl get pods -n vector-db
```

- Once your Milvus cluster is ready, the status of all pods in the Milvus cluster should be similar to the following. (milvus pod takes some time to be in running state)

	NAME	READY	STATUS	RESTARTS	AGE
2	prod-milvus-etcd-0	1/1	Running	0	14m
3	prod-milvus-etcd-1	1/1	Running	0	14m
4	prod-milvus-etcd-2	1/1	Running	0	14m
5	prod-milvus-milvus-datacoord-6c7bb4b488-k9htl	1/1	Running	0	6m
6	prod-milvus-milvus-datanode-5c686bd65-wxtmf	1/1	Running	0	6m
7	prod-milvus-milvus-indexcoord-586b9f4987-vb7m4	1/1	Running	0	6m
8	prod-milvus-milvus-indexnode-5b9787b54-xclbx	1/1	Running	0	6m

9	prod-milvus-milvus-proxy-84f67cdb7f-pg6wf	1/1	Running	0	6m
10	prod-milvus-milvus-querycoord-865cc56fb4-w2jmn	1/1	Running	0	6m
11	prod-milvus-milvus-querynode-5bcb59f6-nhqqw	1/1	Running	0	6m
12	prod-milvus-milvus-rootcoord-fdcccfc84-9964g	1/1	Running	0	6m
13	prod-milvus-minio-0	1/1	Running	0	14m
14	prod-milvus-minio-1	1/1	Running	0	14m
15	prod-milvus-minio-2	1/1	Running	0	14m
16	prod-milvus-minio-3	1/1	Running	0	14m
17	prod-milvus-pulsar-bookie-0	1/1	Running	0	14m
18	prod-milvus-pulsar-bookie-1	1/1	Running	0	14m
19	prod-milvus-pulsar-bookie-init-h6tfz	0/1	Completed	0	14m
20	prod-milvus-pulsar-broker-0	1/1	Running	0	14m
21	prod-milvus-pulsar-broker-1	1/1	Running	0	14m
22	prod-milvus-pulsar-proxy-0	1/1	Running	0	14m
23	prod-milvus-pulsar-proxy-1	1/1	Running	0	14m
24	prod-milvus-pulsar-pulsar-init-d2t56	0/1	Completed	0	14m
25	prod-milvus-pulsar-recovery-0	1/1	Running	0	14m
26	prod-milvus-pulsar-toolset-0	1/1	Running	0	14m
27	prod-milvus-pulsar-zookeeper-0	1/1	Running	0	14m
28	prod-milvus-pulsar-zookeeper-1	1/1	Running	0	13m
29	prod-milvus-pulsar-zookeeper-2	1/1	Running	0	13m

- Create a pod in `vector-db` namespace.

```

1 apiVersion: v1
2 kind: Pod
3 metadata:
4   name: milvus-backup-restore-pod
5   namespace: vector-db
6 spec:
7   containers:
8   - name: ubuntu-container
9     image: ubuntu:20.04
10    command: [ "sleep", "infinity" ]
11    securityContext:
12      runAsUser: 0
13      runAsGroup: 0

```

- Exec into the pod using

```
1 kubectl exec -it milvus-backup-restore-pod -n vector-db -- bash
```

- Install `milvus-cli` and `vim`

```

1 apt-get update
2 apt-get install pip -y
3 pip install milvus-cli
4 apt-get install vim

```

- Create a new file to connect to the Milvus service.

```
1 vim connect-milvus.py
```

- Paste this script in the new Python file.

```

1 from pymilvus import connections, utility
2
3 # Connect to Milvus
4 connections.connect(
5     alias="default",

```

```

6     host='prod-milvus-milvus.vector-db.svc.cluster.local',
7     port='19530'
8 )
9
10 # Check connection by listing collections
11 try:
12     collections = utility.list_collections()
13     print("Successfully connected to Milvus. Collections:")
14     for collection in collections:
15         print(f"- {collection}")
16 except Exception as e:
17     print(f"Failed to connect to Milvus: {e}")

```

- Run the Python script using

```
1 python3 connect-milvus.py
```

- Once you run the script, the pod gets connected to the Milvus service and it will list all the collections which are available in the Milvus.
- Download all the required packages (minio-client) to connect to Minio through pod.

```

1 apt-get update
2 apt install wget -y
3 wget https://dl.min.io/client/mc/release/linux-amd64/mc
4 chmod +x mc
5 mv mc /usr/local/bin/

```

- Connect to minio client using

```
1 mc alias set myminio http://prod-milvus-minio.vector-db.svc.cluster.local:9000 minioadmin minioadmin
```

- Create an empty minio bucket using

```
1 mc mb myminio/prod-milvus-new
```

- Connect to AWS S3 service from pod using

```
1 mc alias set s3 https://s3.us-east-2.amazonaws.com AWS_Access_Key_ID AWS_Access_Secret_Key
```

- Copy the S3 bucket (prod-milvus-new) data to the newly created minio bucket (prod-milvus-new) in second standalone cluster using

```
1 mc mirror s3/prod-milvus-new myminio/prod-milvus-new
```

- Now we need to clone the Milvus Backup tool repository to restore the Milvus Data in Distributed Milvus cluster.

```

1 apt-get update
2 apt install git -y
3 apt-get install wget -y
4 wget https://go.dev/dl/go1.21.4.linux-amd64.tar.gz -O go.tar.gz
5 tar -xzf go.tar.gz -C /usr/local
6 echo export PATH=$HOME/go/bin:/usr/local/go/bin:$PATH >> ~/.profile
7 source ~/.profile
8 go version
9
10 git clone https://github.com/zilliztech/milvus-backup.git
11 cd milvus-backup
12 go get
13 go build

```

- Edit the backup configuration file


```
1 cd configs/
2 vim backup.yaml
```

```
1 # Configures the system log output.
2 log:
3   level: info # Only supports debug, info, warn, error, panic, or fatal. Default 'info'.
4   console: true # whether print log to console
5   file:
6     rootPath: "logs/backup.log"
7
8 http:
9   simpleResponse: true
10
11 # milvus proxy address, compatible to milvus.yaml
12 milvus:
13   address: prod-milvus-milvus.vector-db.svc.cluster.local
14   port: 19530
15   authorizationEnabled: false
16   # tls mode values [0, 1, 2]
17   # 0 is close, 1 is one-way authentication, 2 is two-way authentication.
18   tlsMode: 0
19   user: "username"
20   password: "password"
21
22 # Related configuration of minio, which is responsible for data persistence for Milvus.
23 minio:
24   # cloudProvider: "minio" # deprecated use storageType instead
25   storageType: "minio" # support storage type: local, minio, s3, aws, gcp, ali(aliyun), azure, tc(tencent)
26
27   address: prod-milvus-minio.vector-db.svc.cluster.local # Address of MinIO/S3
28   port: 9000 # Port of MinIO/S3
29   accessKeyID: minioadmin # accessKeyID of MinIO/S3
30   secretAccessKey: minioadmin # MinIO/S3 encryption string
31   useSSL: false # Access to MinIO/S3 with SSL
32   useIAM: false
33   iamEndpoint: ""
34
35   bucketName: "prod-milvus" # Milvus Bucket name in MinIO/S3, make it the same as your milvus instance
36   rootPath: "files" # Milvus storage root path in MinIO/S3, make it the same as your milvus instance
37
38   # only for azure
39   backupAccessKeyID: # accessKeyID of MinIO/S3
40   backupSecretAccessKey: # MinIO/S3 encryption string
41
42   backupBucketName: "prod-milvus-new" # Bucket name to store backup data. Backup data will store to
43   backupBucketName/backupRootPath
44   backupRootPath: "backup" # Rootpath to store backup data. Backup data will store to
45   backupBucketName/backupRootPath
46
47 backup:
48   maxSegmentGroupSize: 6G
49
50   parallelism:
51     # collection level parallelism to backup
52     backupCollection: 4
53     # thread pool to copy data. reduce it if blocks your storage's network bandwidth
54     copydata: 128
55     # Collection level parallelism to restore
```

```
54     restoreCollection: 2
55
56     # keep temporary files during restore, only use to debug
57     keepTempFiles: false
58
59     # Pause GC during backup through Milvus Http API.
60     gcPause:
61         enable: false
62         seconds: 7200
63     address: http://prod-milvus-milvus.vector-db.svc.cluster.local:9091
```

- Now restore the Milvus Data using

```
1 cd ..
2 ./milvus-backup restore -n prod_milvus_new
```

- Restoration of data is successful in Distributed Milvus Cluster.
- To verify whether the same collections exist or not in the Distributed Milvus cluster after restoring data, run the python script

```
1 python3 connect-milvus.py
```

- If the milvus collections from Standalone cluster exactly matches to milvus collections from Distributed cluster then our Backup and Restore procedure was successful.
- We can also check from UI side whether collections with data exist or not in the Distributed Milvus Cluster.