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

- Create a new Cluster from Scratch.
- Take the access of the Cluster
- Install Cert Manager using

```
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

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

- Install Milvus Operator using Helm

```
helm install milvus-operator \
  -n milvus-operator --create-namespace \
  --wait --wait-for-jobs \
  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.

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

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

- Deploy the Milvus Cluster using Kind Milvus

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

```
apiVersion: milvus.io/v1beta1
kind: Milvus
metadata:
  annotations:
    milvus.io/dependency-values-merged: "true"
  creationTimestamp: "2024-07-15T13:20:19Z"
  finalizers:
```

```yaml
  - milvus.milvus.io/finalizer
  generation: 6
  labels:
    app: milvus
    milvus.io/operator-version: 0.9.17
  name: prod-milvus
  namespace: vector-db
  resourceVersion: "525373"
  uid: 5a6ecdb4-5674-4683-8091-2d98f0331dbd
spec:
  components:
    dataCoord:
      paused: false
      replicas: 1
    dataNode:
      paused: false
      replicas: 1
    disableMetric: false
    image: milvusdb/milvus:v2.4.5
    imageUpdateMode: rollingUpgrade
    indexCoord:
      paused: false
      replicas: 1
    indexNode:
      paused: false
      replicas: 1
    metricInterval: ""
    paused: false
    proxy:
      paused: false
      replicas: 1
      serviceType: LoadBalancer
    queryCoord:
      paused: false
      replicas: 1
    queryNode:
      paused: false
      replicas: 1
    rootCoord:
      paused: false
      replicas: 1
    standalone:
      paused: false
      replicas: 0
      serviceType: ClusterIP
  config: {}
  dependencies:
    customMsgStream: null
    etcd:
      endpoints:
      - prod-milvus-etcd.vector-db:2379
      external: false
      inCluster:
        deletionPolicy: Retain
        values:
          auth:
            rbac:
              enabled: false
```

```
          autoCompactionMode: revision
          autoCompactionRetention: "1000"
          enabled: true
          extraEnvVars:
          - name: ETCD_QUOTA_BACKEND_BYTES
            value: "4294967296"
          - name: ETCD_HEARTBEAT_INTERVAL
            value: "500"
          - name: ETCD_ELECTION_TIMEOUT
            value: "2500"
          image:
            pullPolicy: IfNotPresent
            repository: milvusdb/etcd
            tag: 3.5.5-r4
          livenessProbe:
            enabled: true
            timeoutSeconds: 10
          name: etcd
          pdb:
            create: false
          persistence:
            accessMode: ReadWriteOnce
            enabled: true
            size: 10Gi
            storageClass: null
          readinessProbe:
            enabled: true
            periodSeconds: 20
            timeoutSeconds: 10
          replicaCount: 3
          service:
            peerPort: 2380
            port: 2379
            type: ClusterIP
      kafka:
        external: false
      msgStreamType: pulsar
      natsmq:
        persistence:
          persistentVolumeClaim:
            spec: null
      pulsar:
        endpoint: prod-milvus-pulsar-proxy.vector-db:6650
        external: false
        inCluster:
          deletionPolicy: Retain
          values:
            affinity:
              anti_affinity: false
            autorecovery:
              resources:
                requests:
                  cpu: 1
                  memory: 512Mi
            bookkeeper:
              configData:
                PULSAR_GC: |
```

```
                -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
              PULSAR_MEM: |
                -Xms4096m -Xmx4096m -XX:MaxDirectMemorySize=8192m
              nettyMaxFrameSizeBytes: "104867840"
            pdb:
              usePolicy: false
            replicaCount: 3
            resources:
              requests:
                cpu: 1
                memory: 2048Mi
            volumes:
              journal:
                name: journal
                size: 100Gi
              ledgers:
                name: ledgers
                size: 200Gi
          broker:
            component: broker
            configData:
              PULSAR_GC: |
                -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
              PULSAR_MEM: |
                -Xms4096m -Xmx4096m -XX:MaxDirectMemorySize=8192m
              backlogQuotaDefaultLimitGB: "8"
              backlogQuotaDefaultRetentionPolicy: producer_exception
              defaultRetentionSizeInMB: "-1"
              defaultRetentionTimeInMinutes: "10080"
              maxMessageSize: "104857600"
              subscriptionExpirationTimeMinutes: "3"
              ttlDurationDefaultInSeconds: "259200"
            pdb:
              usePolicy: false
            podMonitor:
              enabled: false
            replicaCount: 1
            resources:
              requests:
                cpu: 1.5
                memory: 4096Mi
          components:
            autorecovery: true
            bookkeeper: true
            broker: true
            functions: false
            proxy: true
            pulsar_manager: false
            toolset: false
            zookeeper: true
          enabled: true
          fullnameOverride: ""
```

```
    images:
      autorecovery:
        pullPolicy: IfNotPresent
        repository: apachepulsar/pulsar
        tag: 2.8.2
      bookie:
        pullPolicy: IfNotPresent
        repository: apachepulsar/pulsar
        tag: 2.8.2
      broker:
        pullPolicy: IfNotPresent
        repository: apachepulsar/pulsar
        tag: 2.8.2
      proxy:
        pullPolicy: IfNotPresent
        repository: apachepulsar/pulsar
        tag: 2.8.2
      pulsar_manager:
        pullPolicy: IfNotPresent
        repository: apachepulsar/pulsar-manager
        tag: v0.1.0
      zookeeper:
        pullPolicy: IfNotPresent
        repository: apachepulsar/pulsar
        tag: 2.8.2
    maxMessageSize: "5242880"
    monitoring:
      alert_manager: false
      grafana: false
      node_exporter: false
      prometheus: false
    name: pulsar
    persistence: true
    proxy:
      configData:
        PULSAR_GC: |
          -XX:MaxDirectMemorySize=2048m
        PULSAR_MEM: |
          -Xms2048m -Xmx2048m
        httpNumThreads: "100"
      pdb:
        usePolicy: false
      podMonitor:
        enabled: false
      ports:
        pulsar: 6650
      replicaCount: 1
      resources:
        requests:
          cpu: 1
          memory: 2048Mi
      service:
        type: ClusterIP
    pulsar_manager:
      service:
        type: ClusterIP
    pulsar_metadata:
      component: pulsar-init
```

```
            image:
              repository: apachepulsar/pulsar
              tag: 2.8.2
          rbac:
            enabled: false
            limit_to_namespace: true
            psp: false
          zookeeper:
            configData:
              PULSAR_GC: |
                -Dcom.sun.management.jmxremote -Djute.maxbuffer=10485760 -XX:+ParallelRefProcEnabled -
XX:+UnlockExperimentalVMOptions -XX:+DoEscapeAnalysis -XX:+DisableExplicitGC -XX:+PerfDisableSharedMem -
Dzookeeper.forceSync=no
              PULSAR_MEM: |
                -Xms1024m -Xmx1024m
            pdb:
              usePolicy: false
            resources:
              requests:
                cpu: 0.3
                memory: 1024Mi
    rocksmq:
      persistence:
        persistentVolumeClaim:
          spec: null
    storage:
      endpoint: prod-milvus-minio.vector-db:9000
      external: false
      inCluster:
        deletionPolicy: Retain
        values:
          accessKey: minioadmin
          bucketName: milvus-bucket
          enabled: true
          existingSecret: ""
          iamEndpoint: ""
          image:
            pullPolicy: IfNotPresent
            tag: RELEASE.2023-03-20T20-16-18Z
          livenessProbe:
            enabled: true
            failureThreshold: 5
            initialDelaySeconds: 5
            periodSeconds: 5
            successThreshold: 1
            timeoutSeconds: 5
          mode: distributed
          name: minio
          persistence:
            accessMode: ReadWriteOnce
            enabled: true
            existingClaim: ""
            size: 500Gi
            storageClass: null
          podDisruptionBudget:
            enabled: false
          readinessProbe:
            enabled: true
```

```
            failureThreshold: 5
            initialDelaySeconds: 5
            periodSeconds: 5
            successThreshold: 1
            timeoutSeconds: 1
          region: ""
          resources:
            requests:
              memory: 2Gi
          rootPath: file
          secretKey: minioadmin
          service:
            port: 9000
            type: ClusterIP
          startupProbe:
            enabled: true
            failureThreshold: 60
            initialDelaySeconds: 0
            periodSeconds: 10
            successThreshold: 1
            timeoutSeconds: 5
          useIAM: false
          useVirtualHost: false
      secretRef: prod-milvus-minio
      type: MinIO
  hookConfig: null
  mode: cluster
status:
  componentsDeployStatus:
    datacoord:
      generation: 1
      image: milvusdb/milvus:v2.4.5
      status:
        availableReplicas: 1
        conditions:
        - lastTransitionTime: "2024-07-15T13:25:47Z"
          lastUpdateTime: "2024-07-15T13:25:47Z"
          message: Deployment has minimum availability.
          reason: MinimumReplicasAvailable
          status: "True"
          type: Available
        - lastTransitionTime: "2024-07-15T13:25:36Z"
          lastUpdateTime: "2024-07-15T13:25:47Z"
          message: ReplicaSet "prod-milvus-milvus-datacoord-8549f5c97f" has successfully
            progressed.
          reason: NewReplicaSetAvailable
          status: "True"
          type: Progressing
        observedGeneration: 1
        readyReplicas: 1
        replicas: 1
        updatedReplicas: 1
    datanode:
      generation: 1
      image: milvusdb/milvus:v2.4.5
      status:
        availableReplicas: 1
        conditions:
```

```
        - lastTransitionTime: "2024-07-15T13:25:48Z"
          lastUpdateTime: "2024-07-15T13:25:48Z"
          message: Deployment has minimum availability.
          reason: MinimumReplicasAvailable
          status: "True"
          type: Available
        - lastTransitionTime: "2024-07-15T13:25:36Z"
          lastUpdateTime: "2024-07-15T13:25:48Z"
          message: ReplicaSet "prod-milvus-milvus-datanode-85688b669b" has successfully
            progressed.
          reason: NewReplicaSetAvailable
          status: "True"
          type: Progressing
        observedGeneration: 1
        readyReplicas: 1
        replicas: 1
        updatedReplicas: 1
    indexcoord:
      generation: 1
      image: milvusdb/milvus:v2.4.5
      status:
        availableReplicas: 1
        conditions:
        - lastTransitionTime: "2024-07-15T13:25:47Z"
          lastUpdateTime: "2024-07-15T13:25:47Z"
          message: Deployment has minimum availability.
          reason: MinimumReplicasAvailable
          status: "True"
          type: Available
        - lastTransitionTime: "2024-07-15T13:25:36Z"
          lastUpdateTime: "2024-07-15T13:25:47Z"
          message: ReplicaSet "prod-milvus-milvus-indexcoord-bf9fff78f" has successfully
            progressed.
          reason: NewReplicaSetAvailable
          status: "True"
          type: Progressing
        observedGeneration: 1
        readyReplicas: 1
        replicas: 1
        updatedReplicas: 1
    indexnode:
      generation: 1
      image: milvusdb/milvus:v2.4.5
      status:
        availableReplicas: 1
        conditions:
        - lastTransitionTime: "2024-07-15T13:25:47Z"
          lastUpdateTime: "2024-07-15T13:25:47Z"
          message: Deployment has minimum availability.
          reason: MinimumReplicasAvailable
          status: "True"
          type: Available
        - lastTransitionTime: "2024-07-15T13:25:36Z"
          lastUpdateTime: "2024-07-15T13:25:47Z"
          message: ReplicaSet "prod-milvus-milvus-indexnode-8444968896" has successfully
            progressed.
          reason: NewReplicaSetAvailable
          status: "True"
```

```
      type: Progressing
    observedGeneration: 1
    readyReplicas: 1
    replicas: 1
    updatedReplicas: 1
  proxy:
    generation: 1
    image: milvusdb/milvus:v2.4.5
    status:
      availableReplicas: 1
      conditions:
      - lastTransitionTime: "2024-07-15T13:25:57Z"
        lastUpdateTime: "2024-07-15T13:25:57Z"
        message: Deployment has minimum availability.
        reason: MinimumReplicasAvailable
        status: "True"
        type: Available
      - lastTransitionTime: "2024-07-15T13:25:36Z"
        lastUpdateTime: "2024-07-15T13:25:57Z"
        message: ReplicaSet "prod-milvus-milvus-proxy-5f85ff97b8" has successfully
          progressed.
        reason: NewReplicaSetAvailable
        status: "True"
        type: Progressing
      observedGeneration: 1
      readyReplicas: 1
      replicas: 1
      updatedReplicas: 1
  querycoord:
    generation: 1
    image: milvusdb/milvus:v2.4.5
    status:
      availableReplicas: 1
      conditions:
      - lastTransitionTime: "2024-07-15T13:25:47Z"
        lastUpdateTime: "2024-07-15T13:25:47Z"
        message: Deployment has minimum availability.
        reason: MinimumReplicasAvailable
        status: "True"
        type: Available
      - lastTransitionTime: "2024-07-15T13:25:36Z"
        lastUpdateTime: "2024-07-15T13:25:47Z"
        message: ReplicaSet "prod-milvus-milvus-querycoord-c69f8d6bf" has successfully
          progressed.
        reason: NewReplicaSetAvailable
        status: "True"
        type: Progressing
      observedGeneration: 1
      readyReplicas: 1
      replicas: 1
      updatedReplicas: 1
  querynode:
    generation: 2
    image: milvusdb/milvus:v2.4.5
    status:
      availableReplicas: 1
      conditions:
      - lastTransitionTime: "2024-07-15T13:26:37Z"
```

```
      lastUpdateTime: "2024-07-15T13:26:37Z"
      message: Deployment has minimum availability.
      reason: MinimumReplicasAvailable
      status: "True"
      type: Available
    - lastTransitionTime: "2024-07-15T13:25:36Z"
      lastUpdateTime: "2024-07-15T13:26:37Z"
      message: ReplicaSet "prod-milvus-milvus-querynode-0-d56bdfc8f" has successfully
        progressed.
      reason: NewReplicaSetAvailable
      status: "True"
      type: Progressing
    observedGeneration: 2
    readyReplicas: 1
    replicas: 1
    updatedReplicas: 1
rootcoord:
  generation: 1
  image: milvusdb/milvus:v2.4.5
  status:
    availableReplicas: 1
    conditions:
    - lastTransitionTime: "2024-07-15T13:25:47Z"
      lastUpdateTime: "2024-07-15T13:25:47Z"
      message: Deployment has minimum availability.
      reason: MinimumReplicasAvailable
      status: "True"
      type: Available
    - lastTransitionTime: "2024-07-15T13:25:36Z"
      lastUpdateTime: "2024-07-15T13:25:47Z"
      message: ReplicaSet "prod-milvus-milvus-rootcoord-845b66bcc9" has successfully
        progressed.
      reason: NewReplicaSetAvailable
      status: "True"
      type: Progressing
    observedGeneration: 1
    readyReplicas: 1
    replicas: 1
    updatedReplicas: 1
standalone:
  generation: 1
  image: milvusdb/milvus:v2.4.5
  status:
    conditions:
    - lastTransitionTime: "2024-07-15T13:25:36Z"
      lastUpdateTime: "2024-07-15T13:25:36Z"
      message: Deployment has minimum availability.
      reason: MinimumReplicasAvailable
      status: "True"
      type: Available
    - lastTransitionTime: "2024-07-15T13:25:36Z"
      lastUpdateTime: "2024-07-15T13:25:36Z"
      message: ReplicaSet "prod-milvus-milvus-standalone-7fb8548bdc" has successfully
        progressed.
      reason: NewReplicaSetAvailable
      status: "True"
      type: Progressing
    observedGeneration: 1
```

```
  conditions:
  - lastTransitionTime: "2024-07-15T13:21:37Z"
    message: Etcd endpoints is healthy
    reason: EtcdReady
    status: "True"
    type: EtcdReady
  - lastTransitionTime: "2024-07-15T13:21:08Z"
    reason: StorageReady
    status: "True"
    type: StorageReady
  - lastTransitionTime: "2024-07-15T13:25:36Z"
    message: MsgStream is ready
    reason: MsgStreamReady
    status: "True"
    type: MsgStreamReady
  - lastTransitionTime: "2024-07-15T13:27:05Z"
    message: All Milvus components are healthy
    reason: ReasonMilvusHealthy
    status: "True"
    type: MilvusReady
  - lastTransitionTime: "2024-07-15T13:27:05Z"
    message: Milvus components are all updated
    reason: MilvusComponentsUpdated
    status: "True"
    type: MilvusUpdated
  endpoint: 4.213.203.14:19530
  ingress:
    loadBalancer: {}
  observedGeneration: 6
  replicas:
    dataCoord: 1
    dataNode: 1
    indexCoord: 1
    indexNode: 1
    proxy: 1
    queryCoord: 1
    rootCoord: 1
  rollingModeVersion: 2
  status: Healthy
```

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

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

- Check if all the pods are in running state

```
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
prod-milvus-etcd-0                             1/1     Running     0          14m
prod-milvus-etcd-1                             1/1     Running     0          14m
prod-milvus-etcd-2                             1/1     Running     0          14m
prod-milvus-milvus-datacoord-6c7bb4b488-k9htl    1/1     Running     0          6m
prod-milvus-milvus-datanode-5c686bd65-wxtmf      1/1     Running     0          6m
prod-milvus-milvus-indexcoord-586b9f4987-vb7m4   1/1     Running     0          6m
prod-milvus-milvus-indexnode-5b9787b54-xclbx     1/1     Running     0          6m
```

```
prod-milvus-milvus-proxy-84f67cdb7f-pg6wf        1/1     Running     0          6m
prod-milvus-milvus-querycoord-865cc56fb4-w2jmn   1/1     Running     0          6m
prod-milvus-milvus-querynode-5bcb59f6-nhqqw      1/1     Running     0          6m
prod-milvus-milvus-rootcoord-fdcccfc84-9964g     1/1     Running     0          6m
prod-milvus-minio-0                              1/1     Running     0          14m
prod-milvus-minio-1                              1/1     Running     0          14m
prod-milvus-minio-2                              1/1     Running     0          14m
prod-milvus-minio-3                              1/1     Running     0          14m
prod-milvus-pulsar-bookie-0                      1/1     Running     0          14m
prod-milvus-pulsar-bookie-1                      1/1     Running     0          14m
prod-milvus-pulsar-bookie-init-h6tfz             0/1     Completed   0          14m
prod-milvus-pulsar-broker-0                      1/1     Running     0          14m
prod-milvus-pulsar-broker-1                      1/1     Running     0          14m
prod-milvus-pulsar-proxy-0                       1/1     Running     0          14m
prod-milvus-pulsar-proxy-1                       1/1     Running     0          14m
prod-milvus-pulsar-pulsar-init-d2t56             0/1     Completed   0          14m
prod-milvus-pulsar-recovery-0                    1/1     Running     0          14m
prod-milvus-pulsar-toolset-0                     1/1     Running     0          14m
prod-milvus-pulsar-zookeeper-0                   1/1     Running     0          14m
prod-milvus-pulsar-zookeeper-1                   1/1     Running     0          13m
prod-milvus-pulsar-zookeeper-2                   1/1     Running     0          13m

```

- Populate some collection with data in Milvus from UI side by sharing the loadbalancer IP and port of milvus service with UI team.
- Create a pod in `vector-db` namespace

```
apiVersion: v1
kind: Pod
metadata:
  name: milvus-backup-pod
  namespace: vector-db
spec:
  containers:
  - name: python-container
    image: python:3.8
    command: [ "sleep", "infinity" ]
```

- Exec into the pod using

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

- Install `milvus-cli` and `vim`

```
apt-get update
pip install milvus-cli
apt-get install vim
```

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

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

- Paste this script in the new Python file.

```
from pymilvus import connections, utility

# Connect to Milvus
connections.connect(
    alias="default",
    host='prod-milvus-milvus.vector-db.svc.cluster.local',
    port='19530'
```

```
)

# Check connection by listing collections
try:
    collections = utility.list_collections()
    print("Successfully connected to Milvus. Collections:")
    for collection in collections:
        print(f"- {collection}")
except Exception as e:
    print(f"Failed to connect to Milvus: {e}")
```

- Run the Python script using

```
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.

```
apt-get update
apt install git -y
apt-get install wget -y
wget https://go.dev/dl/go1.21.4.linux-amd64.tar.gz -O go.tar.gz
tar -xzvf go.tar.gz -C /usr/local
echo export PATH=$HOME/go/bin:/usr/local/go/bin:$PATH >> ~/.profile
source ~/.profile
go version

git clone https://github.com/zilliztech/milvus-backup.git
cd milvus-backup
go get
go build
```

- Edit the backup configuration file

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

```
# Configures the system log output.
log:
  level: info # Only supports debug, info, warn, error, panic, or fatal. Default 'info'.
  console: true # whether print log to console
  file:
    rootPath: "logs/backup.log"

http:
  simpleResponse: true

# milvus proxy address, compatible to milvus.yaml
milvus:
  address: prod-milvus-milvus.vector-db.svc.cluster.local
  port: 19530
  authorizationEnabled: false
  # tls mode values [0, 1, 2]
  # 0 is close, 1 is one-way authentication, 2 is two-way authentication.
  tlsMode: 0
  user: "username"
  password: "password"

# Related configuration of minio, which is responsible for data persistence for Milvus.
```

```yaml
minio:
  # cloudProvider: "minio" # deprecated use storageType instead
  storageType: "minio" # support storage type: local, minio, s3, aws, gcp, ali(aliyun), azure, tc(tencent)

  address: prod-milvus-minio.vector-db.svc.cluster.local  # Address of MinIO/S3
  port: 9000   # Port of MinIO/S3
  accessKeyID: minioadmin # accessKeyID of MinIO/S3
  secretAccessKey: minioadmin  # MinIO/S3 encryption string
  useSSL: false # Access to MinIO/S3 with SSL
  useIAM: false
  iamEndpoint: ""

  bucketName: "prod-milvus" # Milvus Bucket name in MinIO/S3, make it the same as your milvus instance
  rootPath: "files" # Milvus storage root path in MinIO/S3, make it the same as your milvus instance

  # only for azure
  backupAccessKeyID: # accessKeyID of MinIO/S3
  backupSecretAccessKey:  # MinIO/S3 encryption string

  backupBucketName: "prod-milvus-new"  # Bucket name to store backup data. Backup data will store to backupBucketName/backupRootPath
  backupRootPath: "backup"  # Rootpath to store backup data. Backup data will store to backupBucketName/backupRootPath

backup:
  maxSegmentGroupSize: 6G

  parallelism:
    # collection level parallelism to backup
    backupCollection: 4
    # thread pool to copy data. reduce it if blocks your storage's network bandwidth
    copydata: 128
    # Collection level parallelism to restore
    restoreCollection: 2

  # keep temporary files during restore, only use to debug
  keepTempFiles: false

    # Pause GC during backup through Milvus Http API.
  gcPause:
    enable: false
    seconds: 7200
    address: http://prod-milvus-milvus.vector-db.svc.cluster.local:9091
```

- Take the backup of Milvus Collection in Minio using

```
cd ..
./milvus-backup create -n prod_milvus_new
```

- Backup of Milvus collection in Minio is successfull.
- 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.

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

- Connect to minio client using

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

- List the minio buckets

```
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

```
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.

```
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.
- Hence the Backup of Milvus Instance from Distributed Milvus Cluster is Completed.

- Now take the access of Second Standalone Cluster where we want to restore the Milvus Data
- Create a pod using `vector-db` namespace.

```
apiVersion: v1
kind: Pod
metadata:
  name: milvus-backup-restore-pod
  namespace: vector-db
spec:
  containers:
  - name: ubuntu-container
    image: ubuntu:20.04
    command: [ "sleep", "infinity" ]
    securityContext:
      runAsUser: 0
      runAsGroup: 0
```

- Exec into the pod using

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

- Install `milvus-cli` and `vim`

```
apt-get update
apt-get install pip -y
pip install milvus-cli
apt-get install vim
```

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

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

- Paste this script in the new Python file.

```
from pymilvus import connections, utility

```

```
# Connect to Milvus
connections.connect(
    alias="default",
    host='prod-milvus-milvus.vector-db.svc.cluster.local',
    port='19530'
)

# Check connection by listing collections
try:
    collections = utility.list_collections()
    print("Successfully connected to Milvus. Collections:")
    for collection in collections:
        print(f"- {collection}")
except Exception as e:
    print(f"Failed to connect to Milvus: {e}")
```

- Run the Python script using

```
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.

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

- Connect to minio client using

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

- Create an empty minio bucket using

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

- Connect to AWS S3 service from pod using

```
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

```
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 second standalone cluster.

```
apt-get update
apt install git -y
apt-get install wget -y
wget https://go.dev/dl/go1.21.4.linux-amd64.tar.gz -O go.tar.gz
tar -xzvf go.tar.gz -C /usr/local
echo export PATH=$HOME/go/bin:/usr/local/go/bin:$PATH >> ~/.profile
source ~/.profile
go version

git clone https://github.com/zilliztech/milvus-backup.git
cd milvus-backup
go get
go build
```

- Edit the backup configuration file

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

```
# Configures the system log output.
log:
  level: info # Only supports debug, info, warn, error, panic, or fatal. Default 'info'.
  console: true # whether print log to console
  file:
    rootPath: "logs/backup.log"

http:
  simpleResponse: true

# milvus proxy address, compatible to milvus.yaml
milvus:
  address: prod-milvus-milvus.vector-db.svc.cluster.local
  port: 19530
  authorizationEnabled: false
  # tls mode values [0, 1, 2]
  # 0 is close, 1 is one-way authentication, 2 is two-way authentication.
  tlsMode: 0
  user: "username"
  password: "password"

# Related configuration of minio, which is responsible for data persistence for Milvus.
minio:
  # cloudProvider: "minio" # deprecated use storageType instead
  storageType: "minio" # support storage type: local, minio, s3, aws, gcp, ali(aliyun), azure, tc(tencent)

  address: prod-milvus-minio.vector-db.svc.cluster.local  # Address of MinIO/S3
  port: 9000   # Port of MinIO/S3
  accessKeyID: minioadmin # accessKeyID of MinIO/S3
  secretAccessKey: minioadmin  # MinIO/S3 encryption string
  useSSL: false # Access to MinIO/S3 with SSL
  useIAM: false
  iamEndpoint: ""

  bucketName: "prod-milvus" # Milvus Bucket name in MinIO/S3, make it the same as your milvus instance
  rootPath: "files" # Milvus storage root path in MinIO/S3, make it the same as your milvus instance

  # only for azure
  backupAccessKeyID: # accessKeyID of MinIO/S3
  backupSecretAccessKey:  # MinIO/S3 encryption string

  backupBucketName: "prod-milvus-new"  # Bucket name to store backup data. Backup data will store to backupBucketName/backupRootPath
  backupRootPath: "backup"  # Rootpath to store backup data. Backup data will store to backupBucketName/backupRootPath

backup:
  maxSegmentGroupSize: 6G

  parallelism:
    # collection level parallelism to backup
    backupCollection: 4
    # thread pool to copy data. reduce it if blocks your storage's network bandwidth
```

```
    copydata: 128
    # Collection level parallelism to restore
    restoreCollection: 2

  # keep temporary files during restore, only use to debug
  keepTempFiles: false

    # Pause GC during backup through Milvus Http API.
  gcPause:
    enable: false
    seconds: 7200
    address: http://prod-milvus-milvus.vector-db.svc.cluster.local:9091
```

- Now restore the Milvus Data using

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

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

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

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