Altibase aku Sample Guide for Kubernetes

Overview

This document presents a sample guide to using Kubernetes StatefulSet using Altibase aku.

- For aku, refer to the aku section in the Altibase Utilities manual.
- The contents of this document are for sample purposes only, and should be modified according to each purpose and environment in the actual environment.
- test environment
  - Kubernetes: v1.24.2
  - Altibase: v7.1.0.8.8
  - Docker: v20.10.17

Create Altibase Docker image

- Prepare the altibase_home directory to be copied to the Docker image.
  - Install Altibase for Linux. It is not necessary to create an Altibase database.
  - When an Altibase database is created, the following operations are required

- When an Altibase database is created, the following operations are required.
    - Delete all files in $ALTIBASE_HOME/arch_logs.
    - Delete all files in $ALTIBASE_HOME/dbs.
    - Delete all files in $ALTIBASE_HOME/logs.
    - Delete all files in $ALTIBASE_HOME/trc.
- Create an Altibase Docker image using the Dockerfile below.
- You can also use https://hub.docker.com/r/altibase/7.1-bare without creating a Docker image.

```
# file : Dockerfile

FROM ubuntu:18.04
MAINTAINER Altibase

RUN sed -e '56 i\root\t\t soft\t nofile\t\t 1048576 \nroot\t\t hard\t nofile\t\t
1048576 \nroot\t\t soft\t nproc\t\t unlimited \nroot\t\t hard\t nproc\t\t
unlimited \n' -i /etc/security/limits.conf; \
echo "vm.swappiness = 1" >> /etc/sysctl.conf; \
echo "kernel.sem = 20000 32000 512 5029" >> /etc/sysctl.conf;

COPY ./altibase_home /home/altibase/altibase_home
```

Using PersistentVolume

- In this example, 4 volumes are set to different paths, but you can set them to the same path in whole or in part. This is because each pod creates its own subdirectory using the hostname.
- This example uses NFS volumes. Modifications are required to suit your environment.
- You can use any other type of volume that guarantees persistence.

Write a PersistentVolume yaml file

```
# file : altibase-pv.yaml

apiVersion: v1
kind: PersistentVolume
metadata:
  name: altibase-pv-a
spec:
  capacity:
    storage: 100Gi
  accessModes:
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  nfs:
    server: 192.168.1.121
    path: /home/altibase/nfs/test/a
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: altibase-pv-b
spec:
  capacity:
    storage: 100Gi
  accessModes:
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  nfs:
    server: 192.168.1.121
    path: /home/altibase/nfs/test/b
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: altibase-pv-c
spec:
  capacity:
    storage: 100Gi
  accessModes:
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  nfs:
    server: 192.168.1.121
    path: /home/altibase/nfs/test/c
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: altibase-pv-d
spec:
  capacity:
    storage: 100Gi
  accessModes:
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  nfs:
    server: 192.168.1.121
    path: /home/altibase/nfs/test/d
```

Create PersistentVolume

```
$ kubectl create -f altibase-pv.yaml
persistentvolume/altibase-pv-a created
persistentvolume/altibase-pv-b created
persistentvolume/altibase-pv-c created
persistentvolume/altibase-pv-d created
```

Confirm PersistentVolume creation

```
$ kubectl get pv -o wide
NAME                    CAPACITY    ACCESS MODES    RECLAIM POLICY    STATUS
CLAIM                                                STORAGECLASS      REASON    AGE
VOLUMEMODE
altibase-pv-a           100Gi       RWX             Retain            Available
                                                                           30s
Filesystem
altibase-pv-b           100Gi       RWX             Retain            Available
                                                                           30s
Filesystem
altibase-pv-c           100Gi       RWX             Retain            Available
                                                                           30s
Filesystem
altibase-pv-d           100Gi       RWX             Retain            Available
                                                                           30s
Filesystem
```

Using ConfigMap

- In this example, license, set_altibase.env, aku.conf, sample_schema.sql, and entry_point.sh files are managed as ConfigMap.
- Notes for managing altibase.properties file with ConfigMap are described in entry_point.sh file.
- To use Kubernetes, a hostname-based license must be issued from Altibase.
- In this example, 4 pods are created, so 4 licenses are required.

Write a ConfigMap yaml file

```
# file : altibase-cm.yaml

apiVersion: v1
kind: ConfigMap
metadata:
  name: altibase-cm
data:
  license: |
    # You need four hostname based Altibase licenses.

0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111

2222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222

3333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333

  set_altibase.env: |
    export ALTIBASE_HOME=/home/altibase/altibase_home
    export ALTIBASE_NLS_USE=UTF8
    export ALTIBASE_PORT_NO=20300
    export ALTIBASE_REPLICATION_PORT_NO=20301
    export ALTIBASE_ADMIN_MODE=1            # aku requirement
    export ALTIBASE_REMOTE_SYSDBA_ENABLE=1  # aku requirement
    export PATH=${ALTIBASE_HOME}/bin:${PATH}
    export LD_LIBRARY_PATH=${ALTIBASE_HOME}/lib:${LD_LIBRARY_PATH};

  aku.conf: |
    AKU_SYS_PASWWORD                = "manager"
    AKU_STS_NAME                    = "altibase-sts"
    AKU_SVC_NAME                    = "altibase-svc"
    AKU_SERVER_COUNT                = 4
    AKU_QUERY_TIMEOUT               = 3600
    AKU_PORT_NO                     = 20300
    AKU_REPLICATION_PORT_NO         = 20301
    AKU_FLUSH_AT_START              = 1
    AKU_FLUSH_TIMEOUT_AT_START      = 300
    AKU_FLUSH_AT_END                = 1
    AKU_ADDRESS_CHECK_COUNT         = 30
    AKU_DELAY_START_COMPLETE_TIME = 0

    REPLICATIONS = (
        REPLICATION_NAME_PREFIX = "AKU_REP"
        SYNC_PARALLEL_COUNT     = 1
        (
            (
                USER_NAME       = "SYS"
                TABLE_NAME      = "T1"
            ),
            (
                USER_NAME       = "SYS"
                TABLE_NAME      = "T2"
            ),
            (
                USER_NAME       = "SYS"
                TABLE_NAME      = "T3"
            )
        )
    )

  sample_schema.sql: |
    CREATE TABLE T1 ( I1 INTEGER PRIMARY KEY, I2 INTEGER );
    CREATE TABLE T2 ( I1 INTEGER, I2 INTEGER, I3 CHAR(100) )
    PARTITION BY RANGE( I1 )
    (
        PARTITION P1 VALUES LESS THAN (100),
        PARTITION P2 VALUES LESS THAN (200),
        PARTITION P3 VALUES DEFAULT
    );
    CREATE TABLE T3 ( I1 INTEGER, I2 INTEGER, I3 CHAR(100), I4 INTEGER);
    ALTER TABLE T2 ADD PRIMARY KEY ( I1, I2 );
    ALTER TABLE T3 ADD PRIMARY KEY ( I1, I3 );

  entry_point.sh: |
    #!/bin/bash
    . /home/altibase/config_map/set_altibase.env
    MY_POD_NAME=${HOSTNAME}

    function PodTerminate()
    {
      echo `date` "${MY_POD_NAME} aku end : begin" >>
/ALTIBASE/${MY_POD_NAME}.log
      ${ALTIBASE_HOME}/bin/aku -p end >> /ALTIBASE/${MY_POD_NAME}.log
      echo `date` "${MY_POD_NAME} aku end : finish" >>
/ALTIBASE/${MY_POD_NAME}.log
    }
    trap PodTerminate SIGTERM

    cp /home/altibase/config_map/license ${ALTIBASE_HOME}/conf/license
    cp /home/altibase/config_map/aku.conf ${ALTIBASE_HOME}/conf/aku.conf
    #If you need to change altibase.properties, you need to set
altibase.properties as a ConfigMap. After that, you need to uncomment following
line.
    #cp /home/altibase/config_map/altibase.properties
${ALTIBASE_HOME}/conf/altibase.properties
    DB_DIR="/ALTIBASE/${MY_POD_NAME}"
    DB_DIR_SED="\/ALTIBASE\/${MY_POD_NAME}"
    #set path for arch_logs, dbs, logs and trc directories.
    echo `date` "${MY_POD_NAME} sed -i 's/?/${DB_DIR_SED}/g'
${ALTIBASE_HOME}/conf/altibase.properties"  >> /ALTIBASE/${MY_POD_NAME}.log
    sed -i "s/?/${DB_DIR_SED}/g" ${ALTIBASE_HOME}/conf/altibase.properties

    while (true)
    do
      if [ -d "${DB_DIR}" ];then
        echo `date` "${MY_POD_NAME} Altibase database path exists. [${DB_DIR}] "
>> /ALTIBASE/${MY_POD_NAME}.log
      else
        echo `date` "${MY_POD_NAME} Create Altibase database path. [${DB_DIR}] "
>> /ALTIBASE/${MY_POD_NAME}.log
        mkdir -p ${DB_DIR}
        sleep 1
      fi

      if [ -f "${DB_DIR}/dbs/SYS_TBS_MEM_DATA-0-0" ] ;then
        echo `date` "${MY_POD_NAME} Altibase database exists. " >>
/ALTIBASE/${MY_POD_NAME}.log
        break
      else
        echo `date` "${MY_POD_NAME} Create Altibase database. " >>
/ALTIBASE/${MY_POD_NAME}.log
        rm -rf ${DB_DIR}/*
        mkdir -p ${DB_DIR}/arch_logs
        mkdir -p ${DB_DIR}/dbs
        mkdir -p ${DB_DIR}/logs
        mkdir -p ${DB_DIR}/trc
        chown -R ${USER}:${USER} ${HOME}
        chown -R ${USER}:${USER} ${DB_DIR}
        ${ALTIBASE_HOME}/bin/server create UTF8 UTF8
        sleep 5
        if [ -f "${DB_DIR}/dbs/SYS_TBS_MEM_DATA-0-0" ] ;then
          break
        else
          echo `date` "${MY_POD_NAME} ${DB_DIR}/dbs/SYS_TBS_MEM_DATA-0-0 file is
NOT!!! created."
          continue
        fi
      fi
    done

    echo `date` "${MY_POD_NAME} altibase server start " >>
/ALTIBASE/${MY_POD_NAME}.log
    ${ALTIBASE_HOME}/bin/server start

    exec_command="${ALTIBASE_HOME}/bin/isql -silent -s localhost -u sys -p
manager -sysdba "

    $exec_command<<EOF>> .result
      set linesize 100
      set pagesize 50
      select count(*) from system_.sys_tables_ where table_name='T1' or
table_name='T2' or table_name='T3';
      exit;
    EOF
    result_count=$(tail -2 .result| head -1| awk '{print $1}')
    cat .result >> /ALTIBASE/${MY_POD_NAME}.log
    echo `date` "${MY_POD_NAME} result_count: [${result_count}] " >>
/ALTIBASE/${MY_POD_NAME}.log
    rm .result

    if [ ${result_count} -ne 3 ];then
      ${ALTIBASE_HOME}/bin/is -sysdba -f
/home/altibase/config_map/sample_schema.sql >> /ALTIBASE/${MY_POD_NAME}.log
    fi

    echo `date` "${MY_POD_NAME} aku start " >> /ALTIBASE/${MY_POD_NAME}.log
    ${ALTIBASE_HOME}/bin/aku -p start >> /ALTIBASE/${MY_POD_NAME}.log

    while (true)
    do
      sleep 1
    done
```

Create ConfigMap

```
$ kubectl create -f altibase-cm.yaml
configmap/altibase-cm created
```

Confirm ConfigMap creation

```
$ kubectl get cm -o wide
NAME                DATA   AGE
altibase-cm         5      32s
```

Using Service

- This example uses a Kubernetes headless service.

Write a Service yaml file

```yaml
# file : altibase-svc.yaml

apiVersion: v1
kind: Service
metadata:
  name: altibase-svc
spec:
  type: ClusterIP
  clusterIP: None
  publishNotReadyAddresses: true
  ports:
  - name: service-port
    port: 20300
    targetPort: 20300
  - name: replication-port
    port: 20301
    targetPort: 20301
  selector:
    app: altibase-sts
```

Create Service

```
$ kubectl create -f altibase-svc.yaml
service/altibase-svc created
```

Confirm Service creation

```
$ kubectl get svc -o wide
NAME                           TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)
         AGE    SELECTOR
altibase-svc                   ClusterIP   None         <none>
20300/TCP,20301/TCP   37s    app=altibase-sts
```

Using StatefulSet

- In this example, the StatefulSet creates 4 Pods.
- Altibase Kubernetes Utility supports up to 4 pods.

Write a StatefulSet yaml file

```yaml
# file : altibase-sts.yaml

apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: altibase-sts
spec:
  serviceName: altibase-svc
  replicas: 4
  podManagementPolicy: OrderedReady
  selector:
    matchLabels:
      app: altibase-sts
  template:
    metadata:
      labels:
        app: altibase-sts
    spec:
      terminationGracePeriodSeconds: 60
      containers:
      - name: altibase-sts
        image: altibase/7.1-bare
        command:
        - /bin/bash
        - "-c"
        - /home/altibase/config_map/entry_point.sh
        ports:
        - containerPort: 20300
          protocol: TCP
        - containerPort: 20301
          protocol: TCP
        resources:
          requests:
            cpu: 250m
          limits:
            cpu: 3
        startupProbe:
          exec:
            command:
            - cat
            - /tmp/aku_start_completed
          failureThreshold: 3600
          periodSeconds: 10
        volumeMounts:
        - name: altibase-pv
          mountPath: /ALTIBASE
        - name: altibase-cm
          mountPath: /home/altibase/config_map
          readOnly: true
      volumes:
        - name: altibase-cm
          configMap:
            name: altibase-cm
            defaultMode: 0777
            items:
            - key: "license"
              path: "license"
            - key: "set_altibase.env"
              path: "set_altibase.env"
            - key: "aku.conf"
              path: "aku.conf"
            - key: "sample_schema.sql"
              path: "sample_schema.sql"
            - key: "entry_point.sh"
```

```
              path: "entry_point.sh"
  volumeClaimTemplates:
    - metadata:
        name: altibase-pv
      spec:
        accessModes: [ "ReadWriteMany" ]
        resources:
          requests:
            storage: 10Gi
```

Create StatefulSet

```
$ kubectl create -f altibase-sts.yaml
statefulset.apps/altibase-sts created
```

Confirm StatefulSet and Pod creation

```
$ kubectl get sts -o wide
NAME                 READY   AGE    CONTAINERS               IMAGES
altibase-sts         4/4     12m    altibase-sts             altibase/7.1-bare

$ kubectl get pod -o wide
NAME                   READY   STATUS     RESTARTS         AGE    IP
NODE      NOMINATED NODE   READINESS GATES
altibase-sts-0         1/1     Running    0                16m    10.244.1.91
worker2   <none>           <none>
altibase-sts-1         1/1     Running    0                16m    10.244.2.118
worker1   <none>           <none>
altibase-sts-2         1/1     Running    0                15m    10.244.1.92
worker2   <none>           <none>
altibase-sts-3         1/1     Running    0                14m    10.244.2.119
worker1   <none>           <none>
```

Check Altibase replication operation

- The altibase-sts-3 pod connects to the local Altibase with isql and inputs data to the T1 table.
- Access the altibase of the altibase-sts-2 pod with isql from the altibase-sts-3 pod and search the T1 table.

```
$ kubectl exec -it altibase-sts-3 -- /bin/bash
root@altibase-sts-3:/# . /home/altibase/config_map/set_altibase.env
root@altibase-sts-3:/# is
-----------------------------------------------------------------
     Altibase Client Query utility.
     Release Version 7.1.0.8.8
     Copyright 2000, ALTIBASE Corporation or its subsidiaries.
     All Rights Reserved.
-----------------------------------------------------------------
ISQL_CONNECTION = TCP, SERVER = localhost, PORT_NO = 20300
iSQL> insert into t1 values(1,1);
iSQL> exit

root@altibase-sts-3:/# isql -s altibase-sts-2.altibase-svc -u sys -p manager
-----------------------------------------------------------------
```

```
     Altibase Client Query utility.
     Release Version 7.1.0.8.8
     Copyright 2000, ALTIBASE Corporation or its subsidiaries.
     All Rights Reserved.
-----------------------------------------------------------------
ISQL_CONNECTION = TCP, SERVER = altibase-sts-2.altibase-svc, PORT_NO = 20300
iSQL> select * from T1;
I1          I2
---------------------------
1           1
1 row selected.
```

Check Scale-down operation

- Scale-down with replicas=3.
- Connect to the altibase-sts-2 pod.
- Check that the XSN value of replication AKU_REP_23 is -1, which is the reset state.

```
$ kubectl scale sts altibase-sts --replicas=3
statefulset.apps/altibase-sts scaled
$
$ kubectl exec -it altibase-sts-2 -- /bin/bash
root@altibase-sts-2:/# . /home/altibase/config_map/set_altibase.env
root@altibase-sts-2:/# is
-----------------------------------------------------------------
     Altibase Client Query utility.
     Release Version 7.1.0.8.8
     Copyright 2000, ALTIBASE Corporation or its subsidiaries.
     All Rights Reserved.
-----------------------------------------------------------------
ISQL_CONNECTION = TCP, SERVER = localhost, PORT_NO = 20300
iSQL> select REPLICATION_NAME, XSN from system_.sys_replications_;
REPLICATION_NAME                          XSN
------------------------------------------------------------------
AKU_REP_02                                1591138
AKU_REP_12                                1591138
AKU_REP_23                                -1
3 rows selected.
```

Check Scale-up operation

- Connect to the altibase-sts-0 pod and insert additional data into the T1 table.
- Scale-up with replicas=4.
- Connect to the altibase-sts-3 pod and check if the data additionally inserted in the T1 table is reflected in the scaled-up pod.

```
$ kubectl exec -it altibase-sts-0 -- /bin/bash
root@altibase-sts-0:/# . /home/altibase/config_map/set_altibase.env
root@altibase-sts-0:/# is
-----------------------------------------------------------------
     Altibase Client Query utility.
     Release Version 7.1.0.8.8
     Copyright 2000, ALTIBASE Corporation or its subsidiaries.
     All Rights Reserved.
-----------------------------------------------------------------
ISQL_CONNECTION = TCP, SERVER = localhost, PORT_NO = 20300
iSQL> insert into t1 values(2,2);
1 row inserted.
iSQL> exit
root@altibase-sts-0:/# exit
$
$ kubectl scale sts altibase-sts --replicas=4
statefulset.apps/altibase-sts scaled
$
$ kubectl exec -it altibase-sts-3 -- /bin/bash
root@altibase-sts-3:/# . /home/altibase/config_map/set_altibase.env
root@altibase-sts-3:/# is
-----------------------------------------------------------------
     Altibase Client Query utility.
     Release Version 7.1.0.8.8
     Copyright 2000, ALTIBASE Corporation or its subsidiaries.
     All Rights Reserved.
-----------------------------------------------------------------
ISQL_CONNECTION = TCP, SERVER = localhost, PORT_NO = 20300
iSQL> select * from T1;
I1          I2
---------------------------
1           1
2           2
2 row selected.
```