Etcd Cluster Setup

Configuration of etcd using StateFulset:

```yaml
apiVersion: v1
kind: Service
metadata:
  name: etcd
spec:
  type: ClusterIP
  ports:
  - port: 2379
    name: client
  - port: 2380
    name: peer
  selector:
    app: etcd
---
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: etcd
  labels:
    app: etcd
spec:
  serviceName: etcd
  replicas: 3
  selector:
    matchLabels:
      app: etcd
  template:
    metadata:
      labels:
        app: etcd
    spec:
      containers:
      - name: etcd
        image: quay.io/coreos/etcd:latest
        ports:
        - containerPort: 2379
          name: client
        - containerPort: 2380
          name: peer
        volumeMounts:
        - name: data
          mountPath: /var/run/etcd
        command:
          - /bin/sh
          - -c
          - |
            PEERS="etcd-0=http://etcd-0.etcd:2380,etcd-1=http://etcd-
1.etcd:2380,etcd-2=http://etcd-2.etcd:2380"
            exec etcd --name ${HOSTNAME} \
              --listen-peer-urls http://0.0.0.0:2380 \
              --listen-client-urls http://0.0.0.0:2379 \
              --advertise-client-urls http://${HOSTNAME}.etcd:2379 \
              --initial-advertise-peer-urls http://${HOSTNAME}:2380 \
              --initial-cluster-token etcd-cluster-1 \
              --initial-cluster ${PEERS} \
              --initial-cluster-state new \
              --data-dir /var/run/etcd/default.etcd
  volumeClaimTemplates:
  - metadata:
      name: data
    spec:
      storageClassName: longhorn
      accessModes: [ "ReadWriteOnce" ]
      resources:
        requests:
          storage: 1Gi
```

In above configuration we are creating a service for internal communication of etcd pods (happens on port 2379) and external communication between patroni and etcd (happens on port 2380). A statefulset of 3 replicas is configured, with configuration for etcd. Persistent volume (PV) of 1GB for each pod is claimed using PVC.

`kubectl apply -f file_name.yaml`

Patroni Cluster Setup

```
apiVersion: v1
kind: Service
metadata:
  name: patroni
  namespace: default
spec:
  type: LoadBalancer
  ports:
    - name: postgresql
      port: 5432
  selector:
    app: patroni
```

Service of load balancer type is created for patroni cluster and external IP is assigned from IpAddress-Pool. Below configurations are for IpAddress-Pool.

```
apiVersion: v1
kind: ConfigMap
metadata:
  namespace: metallb-system
  name: config
data:
  config: |
    address-pools:
    - name: default
      protocol: layer2
      addresses:
      - 10.1.41.206-10.1.41.210
---

apiVersion: metallb.io/v1beta1
kind: L2Advertisement
metadata:
  name: empty
  namespace: metallb-system
```

IpAddress-pool assign the ip from addresses range. When a service in Kubernetes is assigned an external IP, MetalLB running in L2 mode advertises that IP on the network so that devices in the same Layer 2 domain (like switches and routers) know how to reach it.
(Without L2Advertisement we can not access the patroni cluster outside Kubernetes cluster).

Configuration for patroni (ConfigMap)

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: patroni-config
  namespace: default
data:
  patroni.yml: |
    scope: postgres-ha
    namespace: /service/
    name: patroni

    etcd:
      hosts: etcd-0.etcd:2379,etcd-1.etcd:2379,etcd-2.etcd:2379



    bootstrap:
      dcs:
        ttl: 30
        loop_wait: 10
        retry_timeout: 10
        maximum_lag_on_failover: 1048576
        postgresql:
          use_pg_rewind: true
          parameters:
            max_connections: 100
            max_locks_per_transaction: 64
            max_worker_processes: 8
            wal_level: replica
            hot_standby: "on"
            wal_keep_size: 1024
            archive_mode: "on"
            archive_timeout: 1800s

    postgresql:
      listen: "*"
      connect_address:
"${HOSTNAME}.patroni.${PATRONI_KUBERNETES_NAMESPACE}.svc.cluster.local:5432"
      data_dir: /var/lib/postgresql/data
      bin_dir: /usr/lib/postgresql/15/bin
      authentication:
        superuser:
          username: postgres
          password: postgres
        replication:
          username: replicator
          password: replicator
      parameters:
        archive_mode: "on"
        archive_command: 'cp %p /var/lib/postgresql/archive/%f'

    tags:
      nofailover: false
      noloadbalance: false
      clonefrom: false
```

Config map is used to store essential configuration for patroni pods.

Statefulset for Patroni

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: patroni
  namespace: default
spec:
  serviceName: "patroni"
  replicas: 2
  selector:
    matchLabels:
      app: patroni
  template:
    metadata:
      labels:
        app: patroni
    spec:
      containers:
        - name: patroni
          image: adeee11/patroni:latest
          ports:
            - containerPort: 5432
          env:
            - name: PATRONI_KUBERNETES_NAMESPACE
              valueFrom:
                fieldRef:
                  fieldPath: metadata.namespace
            - name: PATRONI_KUBERNETES_USE_ENDPOINTS
              value: "true"
            - name: PATRONI_ETCD_HOSTS
              value: "etcd-0.etcd:2379,etc-1.etcd:2379,etcd-2.etcd:2379"
          volumeMounts:
            - name: postgresql-data
              mountPath: /var/lib/postgresql/data
            - name: config
              mountPath: /etc/patroni
              subPath: patroni.yml
      volumes:
        - name: config
          configMap:
            name: patroni-config
  volumeClaimTemplates:
    - metadata:
        name: postgresql-data
      spec:
        storageClassName: longhorn
        accessModes: [ "ReadWriteOnce" ]
        resources:
          requests:
            storage: 1Gi
```

A statefulset of 2 replicas is configured, attached the configMap using volume for patroni and configuration. Persistent volume (PV) of 1GB for each pod is claimed using PVC.

`kubectl apply -f <name>.yaml`

Accessing the Cluster

We can access the cluster using external ip (Load Balancer) assigned by IpAddress-pool using following command.

```
psql -h <external-ip> -p 5432 -U postgres
Password = postgres
```

If Load Balancer connect you to replicator (Read-only pod) for first time, exit and login back you. For second time you will be connected to leader (Read-write pod). This is how Load is balanced.
That All!