Simple Deployment using Kubernetes for Exploring COmanage Registry

Follow these instructions for a simple deployment of Registry that uses basic authentication with default username and password suitable for a first exploration and evaluation of Registry.

The instructions use minikube and assume a Linux environment, though other minikube environments should work.

> These instructions are not suitable for a production deployment.

1. Install minikube following the directions, start a cluster, make sure you can use the kubectl command to manage the cluster.
2. Create a directory to store database state (adjust the name and permissions as necessary for your minikube environment):

   ```
   sudo mkdir -p /mnt/var/lib/postgresql/data
   sudo chmod -R 0777 /mnt/var/lib/postgresql/data
   ```

3. Create the file namespace.yaml with contents

   ```
   ---
   apiVersion: v1
   kind: Namespace
   metadata:
     name: comanage
   ```

   and then apply it with

   ```
   kubectl create -f namespace.yaml
   ```

4. Set the context to use the new namespace:

   ```
   kubectl config set-context --current --namespace=comanage
   ```

5. Create the file postgres-secrets.yaml with contents

   ```
   ---
   apiVersion: v1
   kind: Secret
   metadata:
     namespace: comanage
     name: postgres-secrets
   type: Opaque
   stringData:
     POSTGRES_PASSWORD: password
     COMANAGE_REGISTRY_DATABASE_USER_PASSWORD: password
   ```

   and then apply it with with

   ```
   kubectl apply -f postgres-secrets.yaml
   ```

6. Create the file postgres-configs.yaml with contents

```
---
apiVersion: v1
kind: ConfigMap
metadata:
  namespace: comanage
  name: postgres-configs
data:
  init-user-db.sh: |
    #!/bin/bash
    set -e

    psql -v ON_ERROR_STOP=1 --username "$POSTGRES_USER" --dbname "$POSTGRES_DB" <<-EOSQL
        CREATE USER registry_user PASSWORD '${COMANAGE_REGISTRY_DATABASE_USER_PASSWORD}';
        CREATE DATABASE registry;
        GRANT ALL PRIVILEGES ON DATABASE registry TO registry_user;
    EOSQL
```

and then apply it with

```
kubectl apply -f postgres-configs.yaml
```

7. Create the file postgres-pvc.yaml with contents

```
---
apiVersion: v1
kind: PersistentVolume
metadata:
  namespace: comanage
  name: postgres-pv
spec:
  storageClassName: manual
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: /mnt/var/lib/postgresql/data
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  namespace: comanage
  name: postgres-pv-claim
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
```

and then apply it with

```
kubectl apply -f postgres-pvc.yaml
```

8. Create the file postgres-service.yaml with contents

```
---
apiVersion: v1
kind: Service
metadata:
  namespace: comanage
  name: comanage-registry-database
spec:
  selector:
    app: postgres
  ports:
    - port: 5432
---
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: comanage
  name: postgres
spec:
  replicas: 1
  selector:
    matchLabels:
      app: postgres
  template:
    metadata:
      labels:
        app: postgres
    spec:
      containers:
        - name: postgres
          image: postgres:14
          imagePullPolicy: "IfNotPresent"
          env:
            - name: POSTGRES_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: postgres-secrets
                  key: POSTGRES_PASSWORD
                  optional: false
            - name: COMANAGE_REGISTRY_DATABASE_USER_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: postgres-secrets
                  key: COMANAGE_REGISTRY_DATABASE_USER_PASSWORD
                  optional: false
          volumeMounts:
            - name: init-user-db
              mountPath: /docker-entrypoint-initdb.d/init-user-db.sh
              subPath: init-user-db.sh
            - name: postgres-state
              mountPath: /var/lib/postgresql/data
          ports:
            - containerPort: 5432
      volumes:
        - name: init-user-db
          configMap:
            name: postgres-configs
        - name: postgres-state
          persistentVolumeClaim:
            claimName: postgres-pv-claim
```

and apply it with

```
kubectl apply -f postgres-service.yaml
```

9. Create the file registry-service.yaml with contents

```
---
apiVersion: v1
kind: Service
metadata:
  namespace: comanage
  name: registry
spec:
  type: NodePort
  selector:
    app: registry
  ports:
    - name: https
      port: 443
    - name: http
      port: 80
---
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: comanage
  name: registry
spec:
  replicas: 1
  selector:
    matchLabels:
      app: registry
  template:
    metadata:
      labels:
        app: registry
    spec:
      containers:
        - name: registry
          image: comanageproject/comanage-registry:4.1.2-basic-auth-1
          imagePullPolicy: "IfNotPresent"
          env:
            - name: COMANAGE_REGISTRY_VIRTUAL_HOST_FQDN
              value: "localhost"
            - name: COMANAGE_REGISTRY_VIRTUAL_HOST_PORT
              value: "8443"
            - name: COMANAGE_REGISTRY_DATABASE_PORT
              value: "5432"
```

and apply it with

```
kubectl apply -f registry-service.yaml
```

10. It may take some time for the PostgreSQL and Registry images to be pulled down and for the services to be initialized. Run

```
kubectl get pods
```

and wait for both the postgres and registry pods to have Running STATUS and be READY.

Once the pods (containers) are running you can monitor the Registry pod output using

```
kubectl logs -l app=registry
```

11. Use the kubectl port-forwarding capability to forward port 8443 on your localhost to port 443 of the registry service (other techniques for exposing the registry service on your localhost are available):

```
kubectl port-forward service/registry 8443:443
```

12. Browse to https://localhost:8443/registry/
13. Click through the warning from your web browser about self-signed HTTPS certificates.

14. Click LOGIN and login using the Username "registry.admin" and password "password".
15. To stop the pods (containers):

```
kubectl delete pod,svc --all
```

16. To stop and then delete the minikube cluster:

```
minikube stop
minikube delete
```