Part 4

Deploying WordPress and MySQL on Kubernetes

Installing dependancies:

Download rancher.io/local-path storage class:

`kubectl apply -f https://raw.githubusercontent.com/rancher/local-path-provisioner/master/deploy/local-path-storage.yaml`

Check with `kubectl get storageclass`

Make this storage class (local-path) the default:

`kubectl patch storageclass local-path -p '{"metadata": {"annotations": {"storageclass.kubernetes.io/is-default-class":"true"}}}'`

1. Create a PersistentVolumeClaim? for MySQL:

MySQL needs persistent storage to store its data. Save the following YAML to a file named `mysql-pvc.yaml`:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mysql-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

Apply the PVC:

`kubectl apply -f mysql-pvc.yaml`

2. Deploy MySQL:

Save the following YAML to a file named `mysql-deployment.yaml`:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mysql
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
      - name: mysql
        image: mysql:5.7
        env:
        - name: MYSQL_ROOT_PASSWORD
          value: "password"
        - name: MYSQL_DATABASE
          value: "wordpress"
        ports:
        - containerPort: 3306
        volumeMounts:
        - name: mysql-persistent-storage
          mountPath: /var/lib/mysql
      volumes:
      - name: mysql-persistent-storage
        persistentVolumeClaim:
          claimName: mysql-pvc
```

Apply the Deployment:

`kubectl apply -f mysql-deployment.yaml`

3. Create a Service for MySQL:

This will allow WordPress to communicate with MySQL. Save the following YAML to a file named `mysql-service.yaml`:

```
apiVersion: v1
kind: Service
metadata:
  name: mysql
spec:
  selector:
    app: mysql
  ports:
    - protocol: TCP
      port: 3306
      targetPort: 3306
```

Apply the Service: `kubectl apply -f mysql-service.yaml`

4. Deploy WordPress:

Save the following YAML to a file named wordpress-deployment.yaml:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wordpress
spec:
  replicas: 1
  selector:
    matchLabels:
      app: wordpress
  template:
    metadata:
      labels:
        app: wordpress
    spec:
      containers:
      - name: wordpress
        image: wordpress:latest
        env:
        - name: WORDPRESS_DB_HOST
          value: mysql
        - name: WORDPRESS_DB_USER
          value: "root"
        - name: WORDPRESS_DB_PASSWORD
          value: "password"
```

```
        ports:
        - containerPort: 80
```

Apply the Deployment: `kubectl apply -f wordpress-deployment.yaml`

5. Create a Service for WordPress?:

This will expose WordPress to external traffic. Save the following YAML to a file named `wordpress-service.yaml`:

```
apiVersion: v1
kind: Service
metadata:
  name: wordpress
spec:
  selector:
    app: wordpress
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  type: NodePort
```

Apply the Service:

`kubectl apply -f wordpress-service.yaml`

6. Access WordPress?:

Since we used a NodePort service, WordPress should be accessible on node's IP at a dynamically allocated port above 30000. To find the NodePort assigned to WordPress:

`kubectl get svc wordpress`

Then, in a web browser, access `WordPress`:

`http://< INTERNAL-IP>:<NODE_PORT>`