Sure Shashank! Let’s walk through how a **Spring Boot application (in one container)** communicates with a **database (in another container)**—and how the whole flow works **from the `kubectl apply` command to actual running pods**, assuming we are **not using a single pod but separate Kubernetes resources**.

🧩 Components Involved

We’ll use:

- **Spring Boot App** (e.g., `employee-service`)
- **Database** (e.g., `MySQL`)
- Each runs in its own **container**, inside separate **pods**
- We’ll define everything in Kubernetes using YAML files:
  - Deployment for Spring Boot app
  - Deployment for MySQL
  - Service to expose MySQL
  - ConfigMap/Secrets for environment variables

✅ Step-by-Step: From `kubectl apply` to Running App

---

1. Prepare Docker Images

Build a Docker image for your Spring Boot application.

```
./mvnw clean package
docker build -t my-spring-boot-app .
docker push <your-dockerhub>/my-spring-boot-app
```

-

You don’t need to build an image for MySQL; use the official image:

```
image: mysql:8
```

-

---

2. Define YAML Files

a. MySQL Deployment and Service

```
# 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:8
        env:
        - name: MYSQL_ROOT_PASSWORD
          value: root123
        - name: MYSQL_DATABASE
          value: mydb
        ports:
        - containerPort: 3306
---
apiVersion: v1
kind: Service
metadata:
  name: mysql
spec:
  selector:
    app: mysql
  ports:
  - port: 3306
    targetPort: 3306
  clusterIP: None  # headless service (optional)
```

---

b. Spring Boot Deployment

```
# springboot-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: employee-service
spec:
  replicas: 1
  selector:
    matchLabels:
      app: employee-service
  template:
    metadata:
      labels:
        app: employee-service
    spec:
      containers:
      - name: employee-service
        image: <your-dockerhub>/my-spring-boot-app
        ports:
        - containerPort: 8080
        env:
        - name: SPRING_DATASOURCE_URL
          value: jdbc:mysql://mysql:3306/mydb
        - name: SPRING_DATASOURCE_USERNAME
          value: root
        - name: SPRING_DATASOURCE_PASSWORD
```

```
          value: root123
---
apiVersion: v1
kind: Service
metadata:
  name: employee-service
spec:
  selector:
    app: employee-service
  ports:
  - port: 8080
    targetPort: 8080
  type: NodePort
```

---

3. Run: `kubectl apply`

```
kubectl apply -f mysql-deployment.yaml
kubectl apply -f springboot-deployment.yaml
```

---

4. What Happens Internally

a. `kubectl apply` Command

- Sends the YAML files to **Kubernetes API Server**
- API Server verifies the syntax and authenticity
- Stores resource definitions in **etcd** (Kubernetes key-value store)

b. Scheduler

- The **Kube Scheduler** looks for suitable **nodes** to place the pods (MySQL pod, Spring Boot pod)

c. Kubelet

- Each node runs a **Kubelet** agent
- Kubelet sees a new pod scheduled and tells **Container Runtime (like Docker or containerd)** to pull the image
- Containers start running in their pods

d. Service Discovery

- Kubernetes **Service (mysql)** assigns a DNS name (`mysql.default.svc.cluster.local`)
- Spring Boot uses `jdbc:mysql://mysql:3306/mydb` → DNS automatically resolves to the MySQL pod IP

e. Internal Networking

- Each pod gets its own IP address
- Services create **iptables rules** so traffic is routed to the correct pod behind the service

f. Database Connection

- Spring Boot tries to connect to MySQL at startup
- If connection is successful, app starts fully

---

5. Verify Everything

```
kubectl get pods
kubectl logs <springboot-pod-name>
kubectl exec -it <mysql-pod-name> -- mysql -uroot -p
kubectl get svc
```

---

6. Exposing Application

Since we used `NodePort`, you can access it:
`http://<NodeIP>:<NodePort>`

-

🔁 Summary of Flow

```
kubectl apply → API Server → etcd
              └──→ Scheduler → picks a node
                              └──→ Kubelet → Container runtime
                                               └──→ Starts container
Spring Boot connects to MySQL via Service
```

Would you like me to provide the actual full working code and YAMLs in a downloadable ZIP?