Understanding, deploy and create springboot project like “CRUDE Operation “Apache2, Apache tomcat 8.05 with PostgreSQL 9.5 help of dcoker how to LoadBalancer, and ClusterIP,Ingress withe command and code and example on ubutnu 22.04 ?

ChatGPT

To deploy a Spring Boot CRUD application using Apache Tomcat, Docker, and Kubernetes (with LoadBalancer, ClusterIP, and Ingress), follow these steps. I'll guide you through setting up a simple Spring Boot CRUD application, containerizing it with Docker, and deploying it on a Kubernetes cluster on Ubuntu 22.04.

**Prerequisites:**

1. **Ubuntu 22.04**
2. **Docker** installed
3. **Kubernetes** installed (using Minikube or a cloud provider)
4. **Kubectl** installed
5. **Spring Boot CLI** or **Maven** installed for building the Spring Boot application

**Step 1: Create a Spring Boot CRUD Application**

Create a simple Spring Boot CRUD application. Here's an example using Maven:

1. **Create a new Spring Boot project using Spring Initializr or Maven:**

sh

Copy code

mvn archetype:generate -DgroupId=com.example -DartifactId=crud -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

cd crud

1. **Update pom.xml to include Spring Boot dependencies:**

xml

Copy code

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.7.0</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

1. **Create a simple CRUD REST controller and entity:**

java

Copy code

// src/main/java/com/example/crud/CrudApplication.java

package com.example.crud;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class CrudApplication {

public static void main(String[] args) {

SpringApplication.run(CrudApplication.class, args);

}

}

// src/main/java/com/example/crud/Person.java

package com.example.crud;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

@Entity

public class Person {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private Long id;

private String name;

// Getters and setters...

}

// src/main/java/com/example/crud/PersonRepository.java

package com.example.crud;

import org.springframework.data.repository.CrudRepository;

public interface PersonRepository extends CrudRepository<Person, Long> {

}

// src/main/java/com/example/crud/PersonController.java

package com.example.crud;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

import java.util.Optional;

@RestController

@RequestMapping("/persons")

public class PersonController {

@Autowired

private PersonRepository personRepository;

@PostMapping

public Person createPerson(@RequestBody Person person) {

return personRepository.save(person);

}

@GetMapping("/{id}")

public Optional<Person> getPerson(@PathVariable Long id) {

return personRepository.findById(id);

}

@PutMapping("/{id}")

public Person updatePerson(@PathVariable Long id, @RequestBody Person person) {

person.setId(id);

return personRepository.save(person);

}

@DeleteMapping("/{id}")

public void deletePerson(@PathVariable Long id) {

personRepository.deleteById(id);

}

}

1. **Build the project:**

sh

Copy code

mvn clean package

**Step 2: Containerize the Application with Docker**

1. **Create a Dockerfile:**

Dockerfile

Copy code

# Use the official Maven image to create a build artifact

FROM maven:3.8.4-openjdk-11 as builder

WORKDIR /app

COPY . .

RUN mvn clean package -DskipTests

# Use the official OpenJDK image to run the application

FROM openjdk:11-jre-slim

WORKDIR /app

COPY --from=builder /app/target/crud-0.0.1-SNAPSHOT.jar app.jar

EXPOSE 8080

ENTRYPOINT ["java", "-jar", "app.jar"]

1. **Build the Docker image:**

sh

Copy code

docker build -t spring-boot-crud .

**Step 3: Set Up Kubernetes Resources**

1. **Create a Kubernetes deployment and service:**

yaml

Copy code

# deployment.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: spring-boot-crud

spec:

replicas: 3

selector:

matchLabels:

app: spring-boot-crud

template:

metadata:

labels:

app: spring-boot-crud

spec:

containers:

- name: spring-boot-crud

image: spring-boot-crud:latest

ports:

- containerPort: 8080

---

# service.yaml

apiVersion: v1

kind: Service

metadata:

name: spring-boot-crud

spec:

selector:

app: spring-boot-crud

ports:

- protocol: TCP

port: 80

targetPort: 8080

type: LoadBalancer

1. **Create an Ingress resource (if using Ingress):**

yaml

Copy code

# ingress.yaml

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: spring-boot-crud-ingress

annotations:

nginx.ingress.kubernetes.io/rewrite-target: /

spec:

rules:

- host: <your-domain>

http:

paths:

- path: /

pathType: Prefix

backend:

service:

name: spring-boot-crud

port:

number: 80

**Step 4: Deploy to Kubernetes**

1. **Apply the Kubernetes manifests:**

sh

Copy code

kubectl apply -f deployment.yaml

kubectl apply -f service.yaml

kubectl apply -f ingress.yaml # if using Ingress

1. **Verify the deployment:**

sh

Copy code

kubectl get deployments

kubectl get services

kubectl get pods

kubectl get ingress # if using Ingress

**Access the Application**

* **LoadBalancer:** Once the service is created, get the external IP using:

sh

Copy code

kubectl get svc

* **Ingress:** Configure your DNS to point to the external IP of your ingress controller.

This setup creates a Spring Boot CRUD application, containerizes it with Docker, and deploys it to a Kubernetes cluster with LoadBalancer and Ingress configurations. Adjust the domain and other specifics as needed for your environment.

4o

Top of Form



Bottom of Form

ChatGPT can make mistakes. Check important info.