**EXERCISES ON KUBERNETES**

1. Create a simple deployment of the given app with name of your choice and 3 replicas of pods. Check the status of pod by sending request. App should be accessed from outside the cluster.

dep.yaml

**Note: Replace usn with your usn starting as “ms” i.e. exclude “1” from your usn**

**apiVersion: apps/v1**

**kind: Deployment**

**metadata:**

**name: usn-nginx-deployment**

**namespace: usn**

**labels:**

**app: usn-nginx**

**spec:**

**replicas: 3**

**selector:**

**matchLabels:**

**app: usn-nginx**

**template:**

**metadata:**

**labels:**

**app: usn-nginx**

**spec:**

**containers:**

**- name: nginx**

**image: 172.1.14.168:5001/nginx**

**ports:**

**- containerPort: 80**

**Command to create name space:**

kubectl create namespace ms99cs001

**Command to deploy:**

kubectl apply -f dep.yaml

**Command to check pods:**

kubectl get pods --namespace=ms99cs001

**Command to expose**

kubectl expose deployment usn-nginx-deployment --type=NodePort --name=usn-nginx-service --namespace=ms99cs001

To get exposed port:

kubectl get svc --namespace=ms99cs001

**Open the browser and type :**

http://172.1.14.168:<NodePort>

1. Demonstrate the updation of image in live container in a pod using command line.

**apiVersion: apps/v1**

**kind: Deployment**

**metadata:**

**name: usn-nginx-deployment**

**namespace: usn**

**labels:**

**app: usn-nginx**

**spec:**

**replicas: 3**

**selector:**

**matchLabels:**

**app: usn-nginx**

**template:**

**metadata:**

**labels:**

**app: usn-nginx**

**spec:**

**containers:**

**- name: nginx**

**image: 172.1.14.168:5001/nginx**

**ports:**

**- containerPort: 80**

kubectl set image deployment/usn-nginx-deployment nginx=newImageusn --namespace=ms99cs001

kubectl describe deploy usn-nginx-deployment --namespace=ms99cs001 | grep newImageusn

1. Perform the following.
2. Create 3 pods with names nginx1, nginx2,nginx3. All of them should have the label app=v1 Show all labels of the pods.
3. Get only the 'app=v2' pods.
4. Remove the 'app' label from the pods we created before

kubectl run ms99cs001-nginx1 --image=nginx --restart=Never --labels=app=ms99cs001-v1 --namespace=ms99cs001

kubectl run ms99cs001-nginx2 --image=nginx --restart=Never --labels=app=ms99cs001-v1 --namespace=ms99cs001

kubectl run ms99cs001-nginx3 --image=nginx --restart=Never --labels=app=ms99cs001-v1 --namespace=ms99cs001

kubectl get po --show-labels --namespace=ms99cs001

kubectl get po -l app=ms99cs001-v2 --namespace=ms99cs001

kubectl label po ms99cs001-nginx1 ms99cs001-nginx2 ms99cs001-nginx3 app- --namespace=ms99cs001

1. Create a Pod with ubuntu image and a command to echo “YOUR\_NAME” which overrides the default CMD/ENTRYPOINT of the image. Delete pod.

**dep\_ubuntu\_pod1.yaml**

apiVersion: **apps/v1**

kind: Pod

metadata:

name: ubuntu

**namespace: usn**

labels:

app: ubuntu

spec:

containers:

- name: ubuntu

image: 172.1.14.168:5001/ubuntu

command: ["/bin/bash"]

args: ["-c", "echo MSRIT"]

**kubectl apply -f dep\_ubuntu\_pod1.yaml**

**kubectl logs ubuntu –-namespace=usn**

**kubectl delete pod ubuntu –namespace=usn**

1. Create a Pod that runs one container. The configuration file for the Pod defines a command and arguments by using environment variables and Delete pod.

**dep\_ubuntu\_pod.yaml**

apiVersion: v1

kind: Pod

metadata:

name: ubuntunew

**namespace: usn**

labels:

app: ubuntunew

spec:

containers:

- name: ubuntunew

image: 172.1.14.168:5001/ubuntu

env:

- name: MESSAGE

value: "Hello MSRIT"

command: ["/bin/echo"]

args: ["$(MESSAGE)"]

**kubectl apply -f dep\_ubuntu\_pod.yaml**

**kubectl logs ubuntunew –-namespace=usn**

**kubectl delete pod ubuntunew –namespace=usn**

**////spring boot**

**// BookApplication**

**package com.example.book;**

**import org.springframework.boot.SpringApplication;**

**import org.springframework.boot.autoconfigure.SpringBootApplication;**

**@SpringBootApplication**

**public class BookApplication {**

**public static void main(String[] args) {**

**SpringApplication.run(BookApplication.class, args);**

**}**

**}**

**--------------------------------------------------------------------------------------------**

**// Controller**

**package com.example.book.controller;**

**import com.example.book.model.BookModel;**

**import com.example.book.service.BookService;**

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

**import java.util.List;**

**@RestController**

**public class BookController {**

**private final BookService bookService;**

**public BookController(BookService bookService) {**

**this.bookService = bookService;**

**}**

**@PostMapping("/create")**

**public BookModel createBook(@RequestBody BookModel book){**

**return bookService.create(book);**

**}**

**@GetMapping("/getAllBooks")**

**public List<BookModel> getAllBooks() {**

**return bookService.getAllBooks();**

**}**

**@GetMapping("/get/{bookID}")**

**public BookModel getBook(@PathVariable String bookID) {**

**return bookService.getBook(bookID);**

**}**

**@DeleteMapping("/deleteBook/{bookId}")**

**public String deleteBook(@PathVariable String bookId) {**

**bookService.delete(bookId);**

**return "Book Deleted";**

**}**

**@DeleteMapping("/deleteAll")**

**public String deleteBooks() {**

**bookService.deleteAll();**

**return "All books data deleted";**

**}**

**@PutMapping("/updateBook/{bookId}")**

**public BookModel updateBook(@RequestBody BookModel book, @PathVariable String bookId) {**

**return bookService.update(book, bookId);**

**}**

**}**

**---------------------------------------------------------------------------**

**// Model**

**package com.example.book.model;**

**import jakarta.persistence.Column;**

**import jakarta.persistence.Entity;**

**import jakarta.persistence.Id;**

**import jakarta.persistence.Table;**

**@Entity**

**@Table**

**public class BookModel {**

**@Id**

**private String Id;**

**@Column**

**private String author;**

**public BookModel() {**

**}**

**public BookModel(String id, String author) {**

**Id = id;**

**this.author = author;**

**}**

**public String getId() {**

**return Id;**

**}**

**public void setId(String id) {**

**Id = id;**

**}**

**public String getAuthor() {**

**return author;**

**}**

**public void setAuthor(String author) {**

**this.author = author;**

**}**

**}**

**---------------------------------------------------------------------**

**// Repository**

**package com.example.book.repository;**

**import com.example.book.model.BookModel;**

**import org.springframework.data.jpa.repository.JpaRepository;**

**import org.springframework.stereotype.Repository;**

**@Repository**

**public interface BookRepository extends JpaRepository<BookModel, String> {**

**}**

**----------------------------------------------------------------------**

**// Service**

**package com.example.book.service;**

**import com.example.book.model.BookModel;**

**import com.example.book.repository.BookRepository;**

**import org.springframework.stereotype.Service;**

**import java.util.List;**

**@Service**

**public class BookService {**

**private final BookRepository bookRepository;**

**public BookService(BookRepository bookRepository) {**

**this.bookRepository = bookRepository;**

**}**

**//create an entity book**

**public BookModel create(BookModel book) {**

**return bookRepository.save(book);**

**}**

**//read all books**

**public List<BookModel> getAllBooks() {**

**return bookRepository.findAll();**

**}**

**//read one book**

**public BookModel getBook(String bookId){**

**return bookRepository.findById(bookId).orElse(null);**

**}**

**//update**

**public BookModel update(BookModel book, String id) {**

**BookModel b1 = bookRepository.findById(id).get();**

**b1.setAuthor(book.getAuthor());**

**bookRepository.save(b1);**

**return b1;**

**}**

**//delete by id**

**public void delete(String id) {**

**bookRepository.deleteById(id);**

**}**

**//delete all**

**public void deleteAll() {**

**bookRepository.deleteAll();**

**}**

**}**

**------------------------------------------------------------------------------------**

**// resources/application.properties**

**spring.jpa.hibernate.ddl-auto = update**

**spring.datasource.url= jdbc:mysql://localhost:3306/bookstore**

**spring.datasource.username= root**

**#TODO: Change this**

**spring.datasource.password=**

**spring.datasource.driver-class-name= com.mysql.cj.jdbc.Driver**