

Installing MInKube on your windows machine  
  
<https://www.youtube.com/watch?v=x6jje4w3ZEs&list=PLhfxuQVMs-nyQIX8597QLKKhmpDvNX7tu&index=1>

Install chocolatey  
PS D:\Nilesh\github\sb-bank-application\Kubernetes> Set-ExecutionPolicy Bypass -Scope Process -Force; [System.Net.ServicePointManager]::SecurityProtocol = [System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1'))

PS C:\Windows\system32> choco

Chocolatey v2.2.2

PS C:\Windows\system32> choco install kubernetes-cli

PS C:\Windows\system32> choco install minikube

PS C:\Windows\system32> systeminfo

**PS C:\Windows\system32> minikube start**

PS C:\Windows\system32> minikube status

PS C:\Windows\system32> minikube dashboard

PS C:\Windows\system32> kubectl get pods

PS C:\Windows\system32> minikube stop

PS C:\Windows\system32> minikube status

PS C:\Windows\system32> minikube delete  
  
  
**PS C:\Windows\system32> minikube start --driver=hyperv**

PS C:\Windows\system32> minikube status

PS C:\Windows\system32> minikube dashboard

PS C:\Windows\system32> kubectl get pods

PS C:\Windows\system32> minikube stop

PS C:\Windows\system32> minikube status

PS C:\Windows\system32> minikube delete

https://www.youtube.com/watch?v=HFl2dzhVuUo&t=24s

**SERVICE REGISTRY – EUREKA SERVER** <properties>

<java.version>17</java.version>

<spring-cloud.version>2022.0.4</spring-cloud.version>

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>

</dependency>

</dependencies>

<dependencyManagement>

<dependencies>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-dependencies</artifactId>

<version>${spring-cloud.version}</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

service-registry\src\main\resources\application.yml

server:

port: 8761

spring:

application:

name: service-registry

eureka:

instance:

hostname: localhost

client:

register-with-eureka: false

fetch-registry: false

serviceUrl:

defaultZone: http://${eureka.instance.hostname}:${server.port}/eureka/

package com.example.serviceregistry;

import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

@SpringBootApplication

@EnableEurekaServer

public class ServiceRegistryApplication {

public static void main(String[] args) {

SpringApplication.run(ServiceRegistryApplication.class, args);

}

}

<http://localhost:8761/>

**DEPARTMENT-SERVICE** <properties>

<java.version>17</java.version>

<spring-cloud.version>2022.0.4</spring-cloud.version>

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>

</dependency>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

</dependencies>

<dependencyManagement>

<dependencies>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-dependencies</artifactId>

<version>${spring-cloud.version}</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

department-service\src\main\resources\application.yml

server:

port: 8001

spring:

application:

name: department-service

eureka:

client:

serviceUrl:

defaultZone: http://localhost:8761/eureka/

package com.example;

@SpringBootApplication

@EnableDiscoveryClient

public class DepartmentServiceApplication {

public static void main(String[] args) {

SpringApplication.run(DepartmentServiceApplication.class, args);

}

}

**CONFIG-SERVER**

<properties>

<java.version>17</java.version>

<spring-cloud.version>2022.0.4</spring-cloud.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-config-server</artifactId>

</dependency>

</dependencies>

<dependencyManagement>

<dependencies>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-dependencies</artifactId>

<version>${spring-cloud.version}</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

config-server\src\main\resources\application.yml

server:

port: 8088

spring:

profiles:

active: native

package com.example.configserver;

import org.springframework.cloud.config.server.EnableConfigServer;

@SpringBootApplication

@EnableConfigServer

public class ConfigServerApplication {

public static void main(String[] args) {

SpringApplication.run(ConfigServerApplication.class, args);

}

}

config-server\src\main\resources\config\department-service.yaml  
The service name and the file name should be same

**Move these configurations from**department-service\src\main\resources\application.yml  
to  
config-server\src\main\resources\config\department-service.yaml  
  
server:

port: 8001

eureka:

client:

serviceUrl:

defaultZone: http://localhost:8761/eureka/

**Update department-service\src\main\resources\application.yml**spring:

application:

name: department-service

config:

import: optional:configserver:http://localhost:8088/

**Update department-service\pom.xml**<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-config</artifactId>

</dependency>  
  
**Update config-server\src\main\resources\application.yml**spring:

profiles:

active: native

cloud:

config:

server:

bootstrap: true

**ZIPKIN SERVER**

docker run -d -p 9411:9411 openzipkin/zipkin

http://localhost:9411/zipkin/  
  
Add zipkin dependency to department-service\pom.xml

<dependency>

<groupId>io.micrometer</groupId>

<artifactId>micrometer-tracing-bridge-brave</artifactId>

</dependency>

<dependency>

<groupId>io.zipkin.reporter2</groupId>

<artifactId>zipkin-reporter-brave</artifactId>

</dependency>

Update config-server\src\main\resources\config\department-service.yaml

management:

tracing:

sampling:

probability: 1.0

**BUSINESS LOGIC DEPARTMENT SERVICE**Create empty packages  
com.example.model  
com.example.controller  
com.example.repository  
  
package com.example.model;

import java.util.ArrayList;

import java.util.List;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@NoArgsConstructor

public class Department {

private Long id;

private String name;

private List<Employee> employees = new ArrayList<>();

public Department(Long id, String name){

this.id = id;

this.name = name;

}

}

package com.example.model;

public record Employee(Long id, Long departmentId, String name, int age, String position) {

}

package com.example.repository;

@Repository

public class DepartmentRepository {

private List<Department> departments = new ArrayList<>();

public Department addDepartment(Department department) {

departments.add(department);

return department;

}

public Department findById(Long id) {

return departments.stream().filter(department -> department.getId().equals(id))

.findFirst()

.orElseThrow();

}

public List<Department> findAll(){

return departments;

}

}

package com.example.controller;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

@RestController

@RequestMapping("/department")

public class DepartmentController {

@Autowired

private DepartmentRepository departmentRepository;

private static final Logger LOGGER = LoggerFactory.getLogger(DepartmentController.class);

@PostMapping

public Department add(@RequestBody Department department) {

LOGGER.info("Department add : {}", department);

return departmentRepository.addDepartment(department);

}

@GetMapping

public List<Department> findAll(){

LOGGER.info("Get all department");

return departmentRepository.findAll();

}

@GetMapping("/{id}")

public Department findById(@PathVariable("id") Long departmentId) {

LOGGER.info("Get department for Id : {}", departmentId);

return departmentRepository.findById(departmentId);

}

}

POST <http://localhost:8001/department>  
{

    "id": 1,

    "name": "IT"

}

POST <http://localhost:8001/department>  
{

    "id": 2,

    "name": "CS"

}

GET <http://localhost:8001/department>

GET <http://localhost:8001/department/1>

**EMPLOYEE SERVICE**

<properties>

<java.version>17</java.version>

<spring-cloud.version>2022.0.4</spring-cloud.version>

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

<dependency>

<groupId>io.micrometer</groupId>

<artifactId>micrometer-tracing-bridge-brave</artifactId>

</dependency>

<dependency>

<groupId>io.zipkin.reporter2</groupId>

<artifactId>zipkin-reporter-brave</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-config</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>

</dependency>

</dependencies>

<dependencyManagement>

<dependencies>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-dependencies</artifactId>

<version>${spring-cloud.version}</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

employee-service\src\main\resources\application.yml

spring:

application:

name: employee-service

config:

import: optional:configserver:http://localhost:8088/

Create config-server\src\main\resources\config\employee-service.yaml  
server:

port: 8002

eureka:

client:

serviceUrl:

defaultZone: http://localhost:8761/eureka/

management:

tracing:

sampling:

probability: 1.0

package com.example;

import org.springframework.cloud.client.discovery.EnableDiscoveryClient;

@SpringBootApplication

@EnableDiscoveryClient

public class EmployeeServiceApplication {

public static void main(String[] args) {

SpringApplication.run(EmployeeServiceApplication.class, args);

}

}

Create empty packages  
com.example.model  
com.example.controller  
com.example.repository  
  
package com.example.model;

public record Employee(Long id, Long departmentId, String name, int age, String position) {

}

package com.example.repository;

@Repository

public class EmployeeRepository {

private List<Employee> employees

= new ArrayList<>();

public Employee add(Employee employee) {

employees.add(employee);

return employee;

}

public Employee findById(Long id) {

return employees.stream()

.filter(a -> a.id().equals(id))

.findFirst()

.orElseThrow();

}

public List<Employee> findAll() {

return employees;

}

public List<Employee> findByDepartment(Long departmentId) {

return employees.stream()

.filter(a -> a.departmentId().equals(departmentId))

.toList();

}

package com.example.controller;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

@RestController

@RequestMapping("/employee")

public class EmployeeController {

private static final Logger LOGGER

= LoggerFactory.getLogger(EmployeeController.class);

@Autowired

EmployeeRepository repository;

@PostMapping

public Employee add(@RequestBody Employee employee) {

LOGGER.info("Employee add: {}", employee);

return repository.add(employee);

}

@GetMapping

public List<Employee> findAll() {

LOGGER.info("Employee find");

return repository.findAll();

}

@GetMapping("/{id}")

public Employee findById(@PathVariable("id") Long id) {

LOGGER.info("Employee find: id={}", id);

return repository.findById(id);

}

@GetMapping("/department/{departmentId}")

public List<Employee> findByDepartment(@PathVariable("departmentId") Long departmentId) {

LOGGER.info("Employee find: departmentId={}", departmentId);

return repository.findByDepartment(departmentId);

}

POST <http://localhost:8002/employee>  
{

    "id": 1,

    "departmentId": 2,

    "name": "XYZ",

    "age": 12,

    "position": "manager"

}

GET <http://localhost:8002/employee>  
  
GET <http://localhost:8002/employee/1>  
  
GET <http://localhost:8002/employee/department/2>

**CALL EMPLOYEE SERVICE FROM DEPARTMENT SERVICE USING HTTP-EXCHANGER**package com.example.client;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.service.annotation.GetExchange;

import org.springframework.web.service.annotation.HttpExchange;

@HttpExchange

public interface EmployeeClient {

@GetExchange("/employee//department/{departmentId}")

public List<Employee> findByDepartment(@PathVariable("departmentId") Long departmentId);

}

The method declaration is copied from

package com.example.controller;  
@RestController

@RequestMapping("/employee")

public class EmployeeController {

...

@GetMapping("/department/{departmentId}")

public List<Employee> findByDepartment(@PathVariable("departmentId") Long departmentId) {

LOGGER.info("Employee find: departmentId={}", departmentId);

return repository.findByDepartment(departmentId);

}

}

department-service\pom.xml

<dependency>

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

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

</dependency>

**RestClient is going to be deprecated so start using reactive web client**

package com.example.config;

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

import org.springframework.cloud.client.loadbalancer.reactive.LoadBalancedExchangeFilterFunction;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.web.reactive.function.client.WebClient;

import org.springframework.web.reactive.function.client.support.WebClientAdapter;

import org.springframework.web.service.invoker.HttpServiceProxyFactory;

import com.example.client.EmployeeClient;

@Configuration

public class WebClientConfig {

@Autowired

private LoadBalancedExchangeFilterFunction filterFunction;

@Bean

WebClient employeeWebClient() {

return WebClient.builder()

.baseUrl("http://employee-service")

.filter(filterFunction)

.build();

}

@Bean

EmployeeClient employeeClient() {

HttpServiceProxyFactory httpServiceProxyFactory

= HttpServiceProxyFactory

.builder(WebClientAdapter.forClient(employeeWebClient()))

.build();

return httpServiceProxyFactory.createClient(EmployeeClient.class);

}

}

package com.example.model;

import lombok.NoArgsConstructor;

@NoArgsConstructor

public class Department {

private Long id;

private String name;

private List<Employee> employees = new ArrayList<>();

public Department(Long id, String name){

this.id = id;

this.name = name;

}

//setters and getters

}

package com.example.controller;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

@RestController

@RequestMapping("/department")

public class DepartmentController {

@Autowired

private DepartmentRepository departmentRepository;

@Autowired

private EmployeeClient employeeClient;

private static final Logger LOGGER = LoggerFactory.getLogger(DepartmentController.class);

@PostMapping

public Department add(@RequestBody Department department) {

LOGGER.info("Department add : {}", department);

return departmentRepository.addDepartment(department);

}

@GetMapping

public List<Department> findAll(){

LOGGER.info("Get all department");

return departmentRepository.findAll();

}

@GetMapping("/{id}")

public Department findById(@PathVariable("id") Long departmentId) {

LOGGER.info("Get department for Id : {}", departmentId);

return departmentRepository.findById(departmentId);

}

@GetMapping("/with-employees")

public List<Department> findAllWithEmployees(){

LOGGER.info("Get all department");

List<Department> departments = departmentRepository.findAll();

departments.forEach(department ->

department.setEmployees(employeeClient.findByDepartment(department.getId()))

);

return departments;

}

}

POST <http://localhost:8002/employee>  
{

    "id": 1,

    "departmentId": 1,

    "name": "XYZ",

    "age": 12,

    "position": "manager"

}

{

    "id": 2,

    "departmentId": 2,

    "name": "XYZ",

    "age": 12,

    "position": "manager"

}

{

    "id": 3,

    "departmentId": 2,

    "name": "XYZ",

    "age": 12,

    "position": "manager"

}

GET <http://localhost:8002/employee>

POST <http://localhost:8001/department>  
{

    "id": 1,

    "name": "CS"

}

{

    "id": 2,

    "name": "IT"

}

GET <http://localhost:8001/department/with-employees>

**API-GATEWAY** <properties>

<java.version>17</java.version>

<spring-cloud.version>2022.0.4</spring-cloud.version>

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

<groupId>io.micrometer</groupId>

<artifactId>micrometer-tracing-bridge-brave</artifactId>

</dependency>

<dependency>

<groupId>io.zipkin.reporter2</groupId>

<artifactId>zipkin-reporter-brave</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-config</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-gateway</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>

</dependency>

</dependencies>

<dependencyManagement>

<dependencies>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-dependencies</artifactId>

<version>${spring-cloud.version}</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

api-gateway\src\main\resources\application.yml

server:

port: 8060

eureka:

client:

serviceUrl:

defaultZone: http://localhost:8761/eureka/

management:

tracing:

sampling:

probability: 1.0

spring:

application:

name: api-gateway

config:

import: "optional:configserver:http://localhost:8088"

cloud:

gateway:

routes:

- id: employee-service

uri: lb://employee-service

predicates:

- Path=/employee/\*\*

- id: department-service

uri: lb://department-service

predicates:

- Path=/department/\*\*

package com.example;

import org.springframework.cloud.client.discovery.EnableDiscoveryClient;

@SpringBootApplication

@EnableDiscoveryClient

public class ApiGatewayApplication {

public static void main(String[] args) {

SpringApplication.run(ApiGatewayApplication.class, args);

}

}

POST <http://localhost:8060/employee>  
{

    "id": 1,

    "departmentId": 1,

    "name": "XYZ",

    "age": 12,

    "position": "manager"

}

{

    "id": 2,

    "departmentId": 2,

    "name": "XYZ",

    "age": 12,

    "position": "manager"

}

{

    "id": 3,

    "departmentId": 2,

    "name": "XYZ",

    "age": 12,

    "position": "manager"

}

POST <http://localhost:8060/department>  
{

    "id": 1,

    "name": "CS"

}

{

    "id": 2,

    "name": "IT"

}

GET <http://localhost:8060/department/with-employees>