Spring Cloud Course:

Learning Spring Cloud :

1. **Create a spring cloud server**

```
<dependency>
                    <groupId>org.springframework.cloud</groupId>
                    <artifactId>spring-cloud-starter-netflix-eureka-
server</artifactId>
            </dependency>
```

Add dependency for – Web,Dev tools .

Add below configuration is application.properties file

```
spring.application.name=discovery-server
server.port=${PORT:8761}
eureka.client.registerWithEureka=false
eureka.client.fetchRegistry=false
eureka.client.should-unregister-on-shutdown=true
```

Run Server open http://127.0.0.1:8761

2. **Create a spring cloud clients a**

```
<dependency>
                    <groupId>org.springframework.cloud</groupId>
                    <artifactId>spring-cloud-starter-netflix-eureka-
client</artifactId>
            </dependency
```

Add dependency devtools ,actuators

Spring Boot Cleint App.java

```
import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.builder.SpringApplicationBuilder;

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

import org.springframework.web.bind.annotation.RestController;
```

```
@SpringBootApplication
@EnableDiscoveryClient
public class SpringcloudEurekaclientApplication {

	public static void main(String[] args) {
		new SpringApplicationBuilder(SpringcloudEurekaclientApplication.class).web(true).run(args);
	}
}
```

3. ClientInsatnace.java file

```
import org.springframework.beans.factory.annotation.Value;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
@EnableDiscoveryClient
@SpringBootApplication
@RestController
public class ServiceApplication {

	@Value("${service.instance.name}")
	private String instance;

	public static void main(String[] args) {
		SpringApplication.run(ServiceApplication.class, args);
```

```
	}


	@RequestMapping("/")
	public String message() {
		return "Hello from " + instance;
	}
}
```

Application.properties

```
spring.application.name=client-service
eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka/
eureka.client.register-with-eureka=true
eureka.client.should-unregister-on-shutdown=true
service.instance.name=instance
```

To Run this application with multiple instances

Go to run as and create run configuration with 2 values

1. Server.port=8081 and 8082 for second instance
2. Service.insatnce.name=instance 1 and instance 2 for second instance

Application to access these both services :

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.web.client.RestTemplateBuilder;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;
import org.springframework.http.HttpMethod;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.RequestMapping;
```

```
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.client.RestTemplate;

import com.netflix.appinfo.InstanceInfo;
import com.netflix.discovery.EurekaClient;
@EnableDiscoveryClient
@SpringBootApplication
@RestController
public class SpringcloudEurekaclientAppApplication {
@Autowired
	private EurekaClient client;

@Autowired
RestTemplateBuilder resttemplate;
	public static void main(String[] args) {
		SpringApplication.run(SpringcloudEurekaclientAppApplication.class, args);
	}

	@RequestMapping("/")
	public String getServiceList()
	{
		RestTemplate template=resttemplate.build();
		InstanceInfo instanceInfo= client.getNextServerFromEureka("client-service", false);
		String baseUrl=instanceInfo.getHomePageUrl();
		ResponseEntity<String>
response=template.exchange(baseUrl,HttpMethod.GET,null,String.class);

		return response.getBody();
```

```
	}


}
```

```
<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>
```

Application.properties

```
spring.application.name=Client
eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka/
eureka.client.register-with-eureka=false
```

Now run server and two instances of instance client and one instance of Client Application .

http://127.0.0.1:8080/

Will return response from the instances .

Distributed Configuration Management

- Distributed configuration management is required to create a distributed config server

ConfigServer.java

```
@SpringBootApplication
@EnableConfigServer
@EnableDiscoveryClient
```

```
public class ConfigServerApplication {

	public static void main(String[] args) {

	SpringApplication.run(ConfigServerApplication.class,
args);
	}
}
```

```
<dependencies>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-actuator</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.cloud</groupId>
			<artifactId>spring-cloud-config-server</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.cloud</groupId>
			<artifactId>spring-cloud-starter-netflix-eureka-
server</artifactId>
			</dependency>
```

Application.properties

```
server.port=8800
spring.cloud.config.server.git.uri=https://github.com/rijuvan/spring-config-
repository.git  // Specify your git
spring.application.name=configserver
	eureka.client.service-url.defaultZone=http://localhost:8761/eureka
```

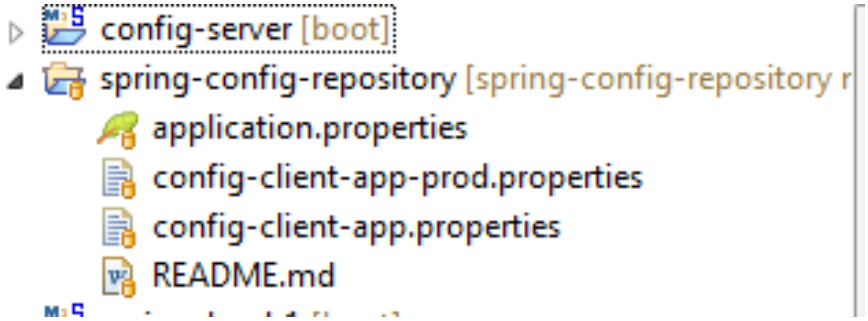
To Connect top GIT :

https://github.com/rijuvan/CDP-SpringBoot

Then Download all configuration files and create your own Git and upload

IN STS go to window -> show view – select Git

Click on working and right click and import
Work bench will look like :



Now create a config App:

```
@SpringBootApplication
@EnableDiscoveryClient
@RestController
public class ConfigClientAppApplication {
	@Autowired
	private ConfigClientAppConfiguration properties;

	@Value("${some.other.property}")
	private String someOther;

	public static void main(String[] args) {

	SpringApplication.run(ConfigClientAppApplication.clas
s, args);
	}

	@RequestMapping
	public String printConfig()
	{
		StringBuilder sb=new StringBuilder();
		sb.append(properties.getProperty());
		sb.append(" || ");
		sb.append(someOther);
		return sb.toString();

	}
}
```

Client Config Properties file :

```
@Component
@ConfigurationProperties(prefix="some")
public class ConfigClientAppConfiguration {
	private String property;

	public String getProperty() {
		return property;
	}

	public void setProperty(String property) {
		this.property = property;
	}


}
```

Create file bootstarp.properties in reources folder :

```
spring.application.name=config-client-app
spring.cloud.config.discovery.enabled=true
eureka.client.service-url.defaultZone=http://localhost:8761/eureka
```

```
<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-web</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-actuator</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>
```

Create a eureka server or use server created in previous application

```
@SpringBootApplication
@EnableEurekaServer
public class SpringcloudEurekaserverApplication {
```

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

SpringApplication.run(SpringcloudEurekaserverApplication
.class,args);
}
}
```

Application.properties

```
spring.application.name=discovery-server
eureka.client.register-with-eureka=false
eureka.client.fetch-registry=false
server.port=8761
eureka.server.enableSelfPreservation=false
```

```
<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>


<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>
```

- Now run Eureka Server first
- Then run Config Server
- Then Config Client

.....

Creating routing using Netflix Zuul:

1. Create a project with name gateway-service and add web, eureka discovery and Zuul dependency.
2. Create other project with name hello-service and add Web and Eureka discovery
3. Create other project with name goodbye-service and add Web and Eureka discovery
4. Create service discovery or use previous one

```
package com.soft.infg.gatewayservice;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;
import org.springframework.cloud.netflix.zuul.EnableZuulProxy;

@SpringBootApplication
@EnableZuulProxy
@EnableDiscoveryClient
public class GatewayServiceApplication {

	public static void main(String[] args) {
		SpringApplication.run(GatewayServiceApplication.class, args);
	}
}
```

```
spring.application.name=gatewayservice
eureka.client.service-url.defaultZone=http://127.0.0.1:8761/eureka
```

```
package com.soft.infg.helloservice;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@SpringBootApplication
@EnableDiscoveryClient
@RestController
public class HelloServiceApplication {

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

```
            SpringApplication.run(HelloServiceApplication.class, args);
        }

        @RequestMapping
        public String hello()

        {

            return "HelloService";

        }
    }
```

```
spring.application.name=hello
server.port=1111
    eureka.client.service-url.defaultZone=http://localhost:8761/eureka
```

```
@SpringBootApplication
@EnableDiscoveryClient
@RestController
public class GoodbyeServiceApplication {

        public static void main(String[] args) {
            SpringApplication.run(GoodbyeServiceApplication.class, args);
        }

        @RequestMapping
        public String bye()
        {
            return "Bye from Service!!";
        }
}
```

```
spring.application.name=goodbye
server.port=2222
    eureka.client.service-url.defaultZone=http://localhost:8761/eureka
```

Start Eureka Server
Start Gate Way Service
Start Hello Service
Start Good By Service

http://localhost:8080/hello
http://localhost:8080/goodbye

Client Side Load Balancing:

1. Create a ribbon time-timeservice with dependency eureka discovery and create two instances with server.port 4444 and server.port 5555.
2. Create ribbon client App with Web, Discovery and ribbon dependency .
3. Create or use discovery server.

```
@EnableDiscoveryClient
@SpringBootApplication
@RestController
public class RibbonTimeserviceApplication {
	@Value("${server.port}")
	private int port;
	public static void main(String[] args) {
		SpringApplication.run(RibbonTimeserviceApplication.class, args);
	}

@RequestMapping
	public String showTime()
	{
		return "The Current time is " + new Date().toString() + "Answered by
service running at port= " + port;
	}
}
```

```
spring.application.name="timeservice"
eureka.client.service-url.defaultZone=http://127.0.0.1:8761/eureka
```

```
@RestController
@EnableDiscoveryClient
@SpringBootApplication
public class RibbonTimeAppApplication {


		@Inject
		private RestTemplate restTemplate;

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

```
		SpringApplication.run(RibbonTimeAppApplication.class, args);
	}

	@GetMapping
	public String getTime() {
		return restTemplate.getForEntity("http://time-service",
String.class).getBody();
	}

	@Bean
	@LoadBalanced
	public RestTemplate restTemplate() {
		return new RestTemplate();
	}
```

- Start Discovery Server
- Start two instances of ribbon service on two ports
- Start Ribbon Client

Circuit breaker implementation:

1. **Create Service Weather Service with Web and Discovery client dependency**
2. **Create a weather – app with web , actuators ,hystrix and discovery client dependency .**

```
Start Discovery Server
Start two instances of ribbon service on two ports
Start Ribbon Client

package com.soft.infg.weatherservice;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import io.netty.util.internal.ThreadLocalRandom;

@EnableDiscoveryClient
@SpringBootApplication
@RestController
public class WeatherServiceApplication {

	private String weather[] = { "Suunny", "Cloudy", "Rainy", "Windy" };
```

```
	public static void main(String[] args) {
		SpringApplication.run(WeatherServiceApplication.class, args);
	}

	@RequestMapping("/weather")
	public String getWeather() {

		int rand = ThreadLocalRandom.current().nextInt(0, 4);
		return weather[rand];

	}
}
```

```
server.port=9000
spring.application.name=weather-service
eureka.client.service-url.defaultZone=http://localhost:8761/eureka/
```

```
package com.soft.infg.weatherapp;

import javax.inject.Inject;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.circuitbreaker.EnableCircuitBreaker;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;
import org.springframework.cloud.client.loadbalancer.LoadBalanced;
import org.springframework.context.annotation.Bean;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.client.RestTemplate;

@EnableDiscoveryClient
@SpringBootApplication
@EnableCircuitBreaker
@RestController
public class WeatherAppApplication {

	@Inject
	private WeatherService weatherService;
	public static void main(String[] args) {
		SpringApplication.run(WeatherAppApplication.class, args);
	}

	@RequestMapping("/current/weather")
	public String getWeather()
	{
		return weatherService.getWeather();
	}

	@Bean
	@LoadBalanced
	public RestTemplate restTemplate() {
		return new RestTemplate();
	}}
```

```
package com.soft.infg.weatherapp;

import javax.inject.Inject;

import org.springframework.stereotype.Service;
import org.springframework.web.client.RestTemplate;

import com.netflix.hystrix.contrib.javanica.annotation.HystrixCommand;
import com.netflix.ribbon.proxy.annotation.Hystrix;

@Service
public class WeatherService {

	@Inject
private RestTemplate restTemplate ;


	@HystrixCommand(fallbackMethod="unknown")
	public String getWeather()
	{
	return restTemplate.getForEntity("http://weather-service/weather",
String.class).getBody();
	}

	public String unknown()
	{
		return "Unknown";
	}

}
```

```
server.port=9090
spring.application.name=weather-app
eureka.client.service-url.defaultZone=http://localhost:8761/eureka/
```

- Start Discovery
- Start Weather Service
- Start Weather App and access http://localhost:9090/current/weather

- **Now to check the pattern stop weather service is showing unknown .**

Hystrix Dashboard:

- Create Application Web, hystrix dashboard dependency.

- Enable Management console in circuit breaker application :

WeatherApp -- Contains circuit breaker APP

```
server.port=9090
spring.application.name=weather-app
eureka.client.service-url.defaultZone=http://localhost:8761/eureka/
management.endpoints.web.exposure.include=hystrix.stream
```

Create Hystrix dashboard application :

```
@SpringBootApplication
@EnableHystrixDashboard
public class DashboardHystrixApplication {

	public static void main(String[] args) {
		SpringApplication.run(DashboardHystrixApplication.class, args);
	}
}
```

Open

http://127.0.0.1:8080/hystrix

Now enter below address (before that generate some traffic by calling circuit breaker app)

http://127.0.0.1:9090/actuator/hystrix.stream