SPRING CLOUD AND SPRING MICROSERVICES

Session 8

OBJECTIVES

- Explain how to use Spring Cloud
- Explain how to use Spring Microservices
- Explain how to use Spring Microservices with Spring Cloud

WHAT IS SPRING CLOUD?

- Spring Cloud is a tool-set that builds on the concepts of Spring Boot and enables developers to quickly build patterns used in distributed systems.

- Spring Cloud enables the developers to use applications and services to quickly create these patterns, which can be reused with different distributed systems.

UNDERSTANDING SPRING CLOUD (1-2)

- Spring Cloud provides multiple libraries that ease the process of developing, deploying, and operating distributed applications for the cloud.
- Spring Cloud can be of assistance in the following ways:

Spring Cloud enables applications to access cloud services.

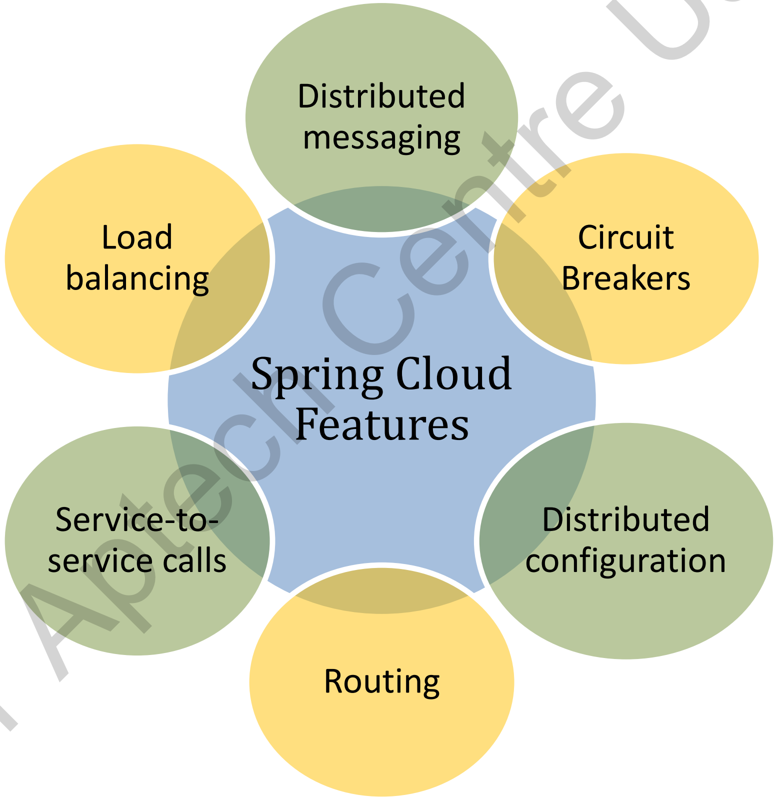
Spring Cloud also allows extension of its functionality to other services.

Spring Cloud works independently using Cloud Connectors, allowing full use of libraries.

Spring Cloud extends support to Spring and Spring Boot applications.

UNDERSTANDING SPRING CLOUD (2-2)

- Spring Cloud includes following features:



Spring Cloud Features

Distributed messaging

Circuit Breakers

Distributed configuration

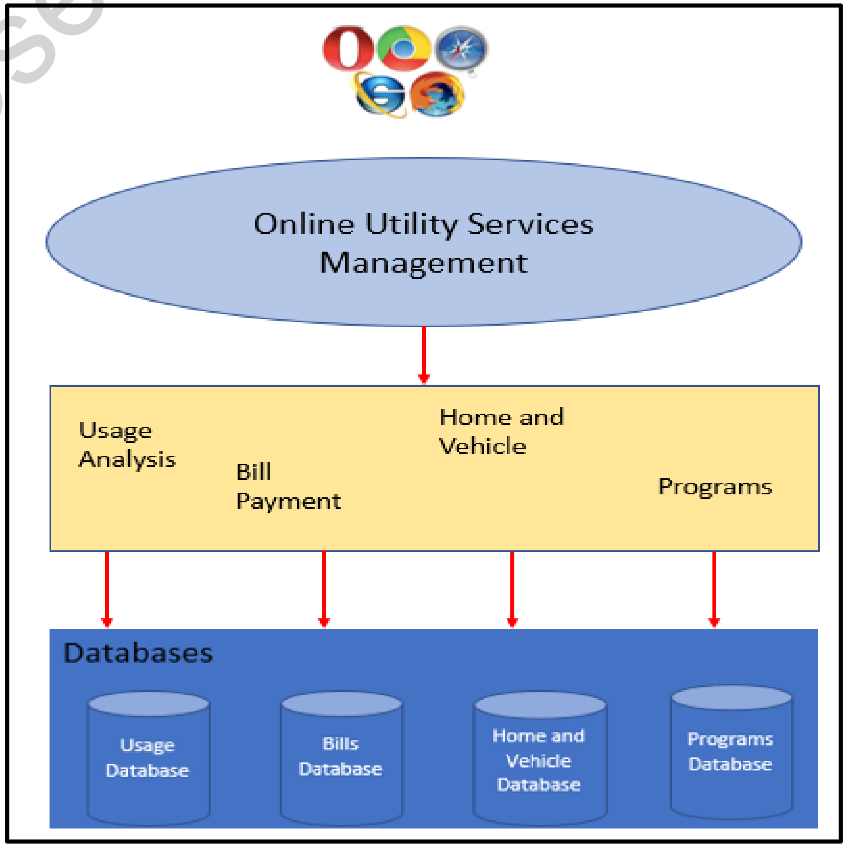
Routing

Service-to-service calls

Load balancing

DEMYSTIFYING MICROSERVICES

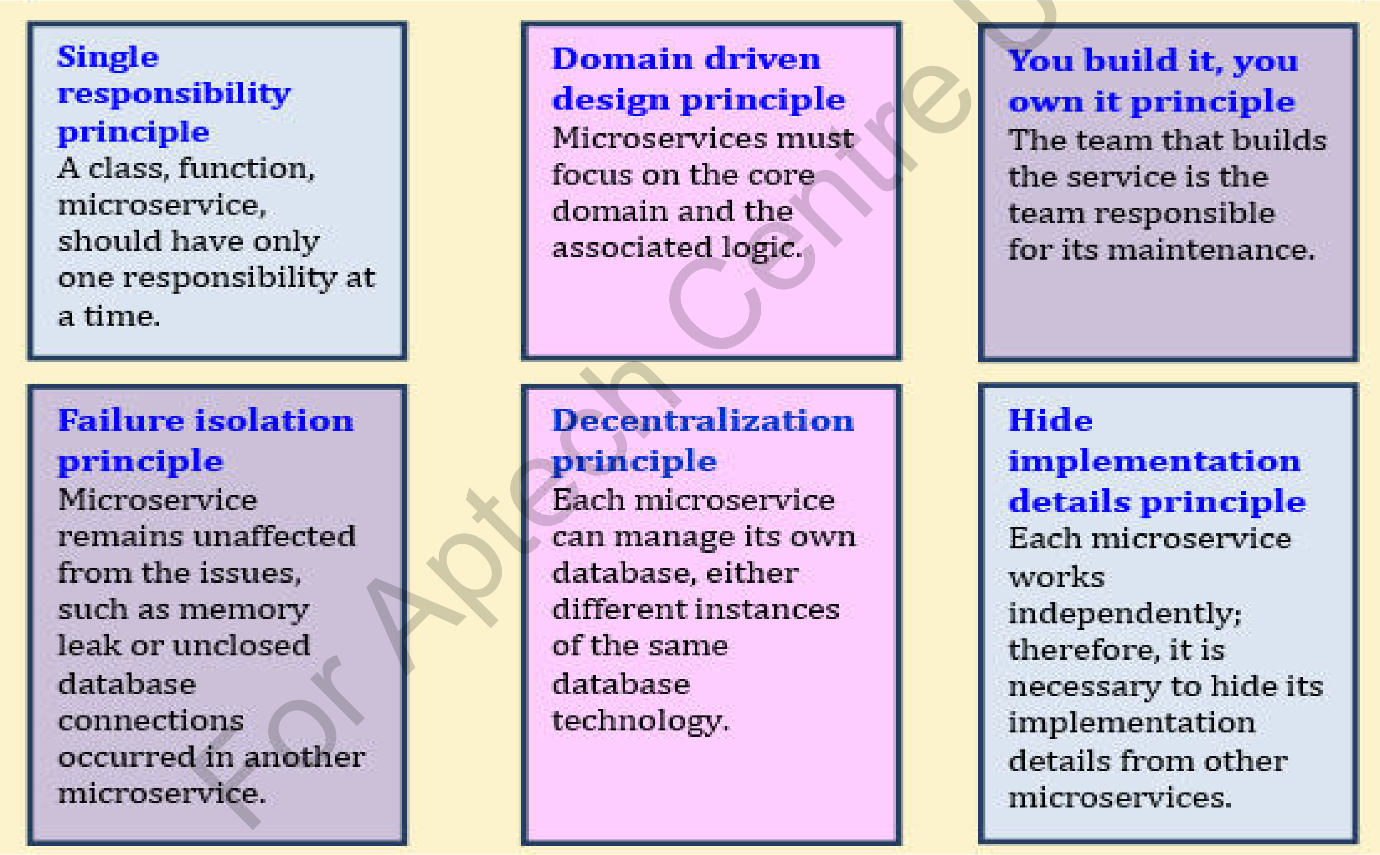
- Microservices is architecture as well as an approach that can be used to combine number of smaller components for building up large systems/applications.
- Microservices architecture is useful because it allows component-wise scalability instead of application scalability.



Microservices Used in an Application

PRINCIPLES AND CHARACTERISTICS OF MICROSERVICES(1-2)

- Microservices are written based on certain principles, such as:



Single responsibility principle
A class, function, microservice, should have only one responsibility at a time.

Domain driven design principle
Microservices must focus on the core domain and the associated logic.

You build it, you own it principle
The team that builds the service is the team responsible for its maintenance.

Failure isolation principle
Microservice remains unaffected from the issues, such as memory leak or unclosed database connections occurred in another microservice.

Decentralization principle
Each microservice can manage its own database, either different instances of the same database technology.

Hide implementation details principle
Each microservice works independently; therefore, it is necessary to hide its implementation details from other microservices.

PRINCIPLES AND CHARACTERISTICS OF MICROSERVICES(2-2)

- Microservices have the following characteristics:

Single Responsibility	No Sharing	Data Privacy	Private Implementation	Group deployment
• A microservice works on a Single Responsibility principle.	• A microservice does not share any data or functions with other services.	• A microservice controls and manages the data storage under it.	• The implementation of a microservice is not done through a shared library but through a deploy-time dependency.	• Microservices should be deployed in groups as they work best in groups.

HATEOAS ENABLED SPRING BOOT MICROSERVICE

- HATEOAS stands for Hypermedia as the Engine of Application State.

- It is used to give extra information about a REST API and is considered as an extra level upon REST.

REACTIVE SPRING BOOT MICROSERVICE

- Reactive programming style is generally event-driven and asynchronous.
- The most important technique in reactive programming is that it supports non-blocking executions of threads.
- Reactive Spring is based on project reactor which uses Spring platform and Spring cloud framework for developing non-blocking applications.

IMPLEMENTING SECURITY

- Architecture of microservices is highly distributed and is characterized by high network traffic.
- Security measures should be implemented before a microservice API is called otherwise, the service will not perform well and also expose the business data.
- Spring offers interesting features and frameworks for securing microservices; one of the commonly used is OAuth2.0.

ENABLING CROSS ORIGIN FOR MICROSERVICES INTERACTION

- Cross-Origin Resource Sharing allows a JavaScript code on a Web page to access a REST API called from a different domain/origin.
- When CORS is enabled, it first performs a security check whenever the browser requests a resource from a different origin.
- Following code snippet shows how to enable a microservice to enable cross-origin requests:

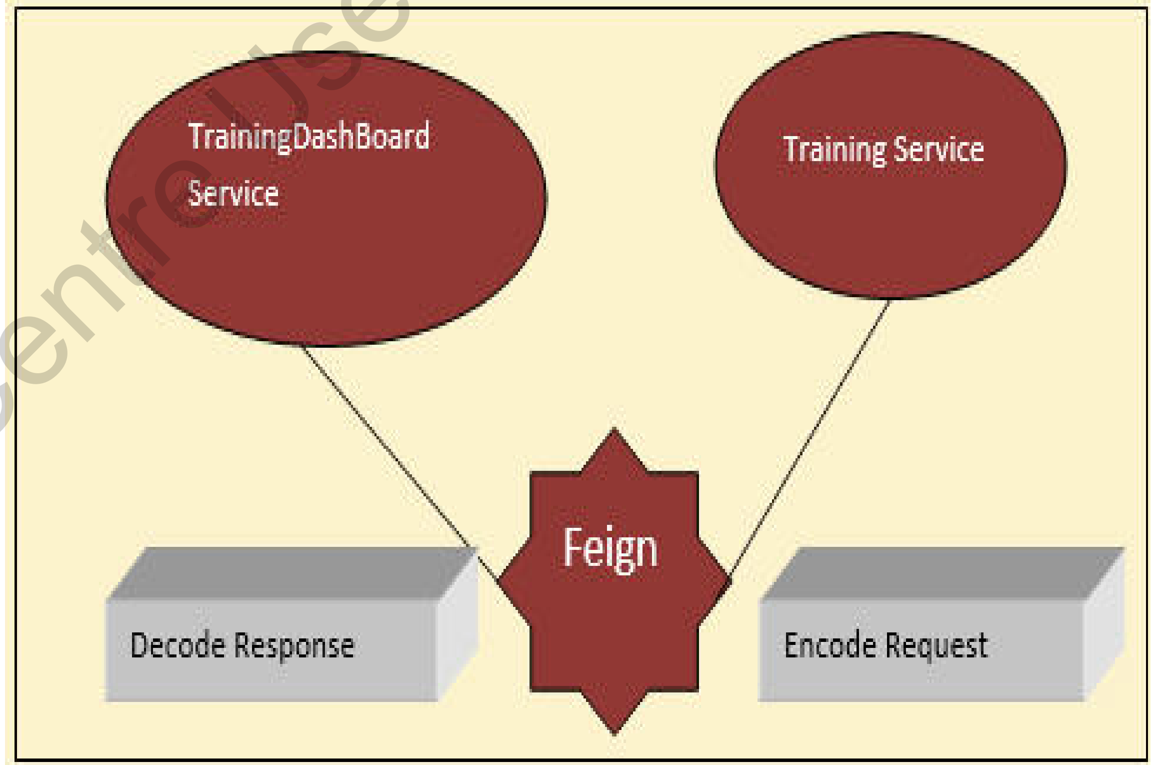
```
@RestController
class GreetingController{
@CrossOrigin
@RequestMapping("/")
Greet greet() {
     return new Greet("Hello World!");
   }
}
```

SCALING MICROSERVICES WITH SPRING CLOUD USING SPRING CLOUD CONFIG

- Spring Cloud Config is a Spring Boot application which provides server-side and client-side support for externalized configuration, primarily in a distributed system.
- Some of the Spring Cloud Config Server features are as follows:
  - ❑ Spring Cloud Config Server can be easily embedded in a Spring Boot Application using @EnableConfigServer.
  - ❑ Spring Cloud Config Server can encrypt and decrypt property values, both symmetrically and asymmetrically.
  - ❑ Spring Cloud Config Server uses HTTP and resource-based API for external configurations.

FEIGN AS DECLARATIVE REST CLIENT

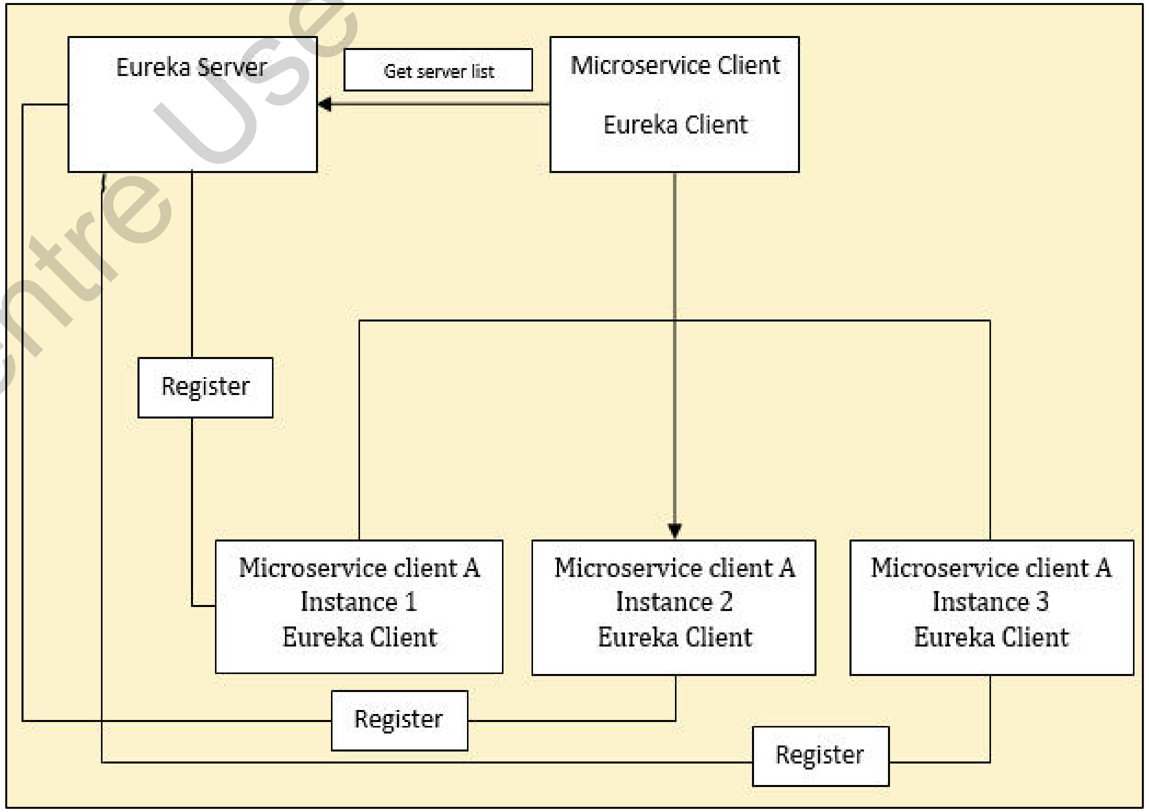
- Spring Cloud Feign is basically a Spring Cloud Netflix library, which makes it easier to configure Web service clients.
- Feign works on a declarative principle or in other words is a declarative Web service client.



Including Feign in a Project

EUREKA FOR REGISTRATION AND DISCOVERY

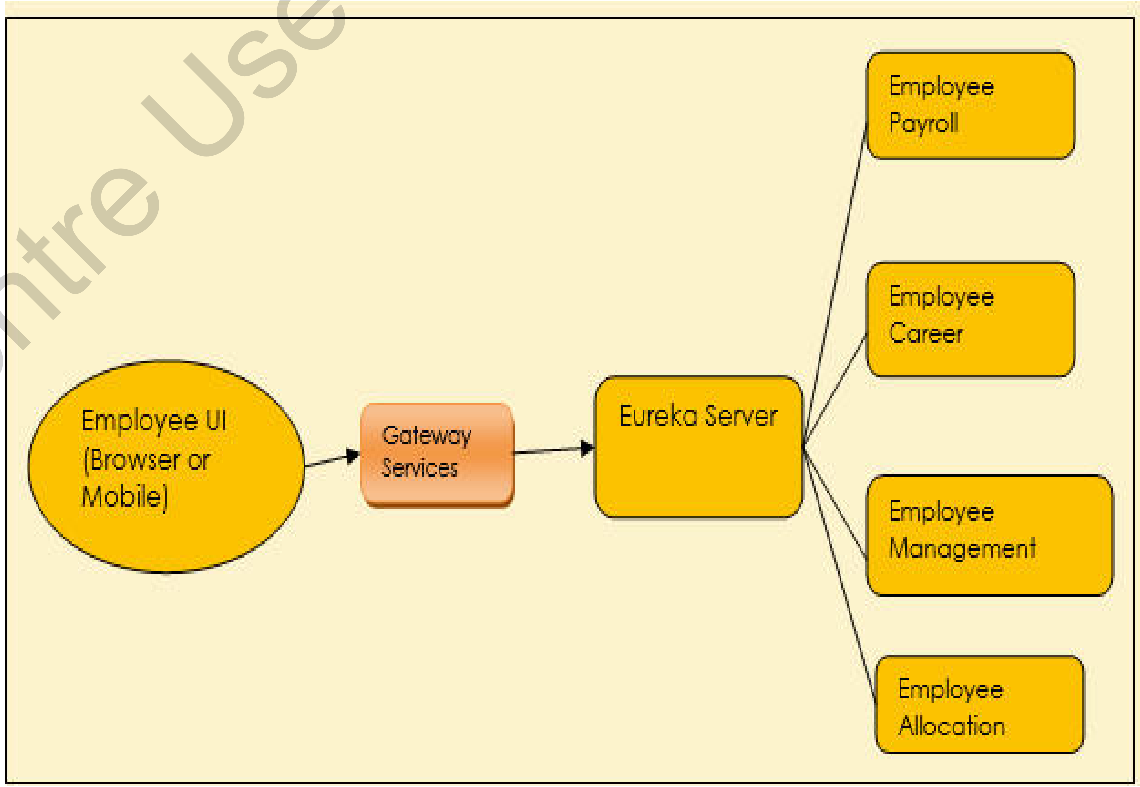
- Eureka or Eureka Server is a REST-based, AWS Discovery Service.
- Eureka includes Eureka Client, which is a Java based client component.
- Its main purpose is to locate services for balancing load and making sure that all service requests are directed to available instances.



Workflow of Eureka

ZUUL PROXY FOR THE API GATEWAY

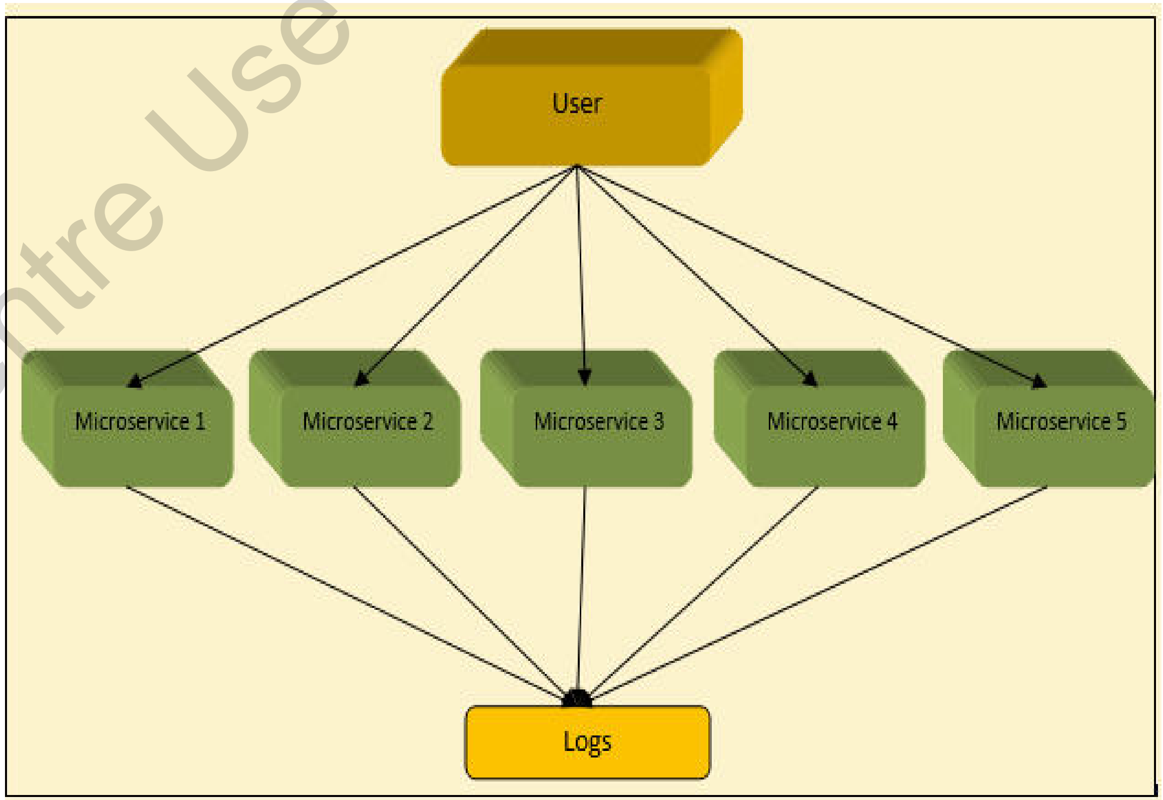
- Zuul is an edge service or a simple gateway service that manages all incoming requests.
- Zuul consists of four types of filters, which are:
  - ❑ Prefilters: These filters are invoked before the request is routed.
  - ❑ Post filters: These filters are invoked after the request has been routed to the relevant microservice.
  - ❑ Route filters: These filters are used to route a request to an appropriate microservice.
  - ❑ Error filters: These filters are invoked when an error occurs during request handling



Zuul-Enabled Services

LOGGING AND MONITORING MICROSERVICES

- Logging and monitoring are critical control systems of microservices.
- Monitoring microservices is essential to identify and troubleshoot problems.
- Logging provides the details of the transactions that have occurred, and this information is essential to find the source of an error and then debug it.



Logging in Microservices

CONTAINERIZING MICROSERVICES WITH DOCKER

- Containerization is commonly explained as 'next-generation virtualization' or 'virtualization of virtualization'.
- It enables developers to execute an application in a virtual environment, by compiling and storing all the application files and libraries, together at one location, as a package, in a container.
- Containerization's popularity can be largely attributed to the open source software Docker.

SUMMARY(1-2)

- Spring Cloud is a tool-set that builds on the concepts of Spring Boot and enables developers to quickly build patterns used in distributed systems.
- Microservices architecture is useful because it allows component-wise scalability instead of application scalability.
- Principles of microservices:
  - ❑ Single responsibility
  - ❑ Domain driven design principle
  - ❑ Self maintenance
  - ❑ Failure isolation principle
  - ❑ Decentralization principle
  - ❑ Hidden implementation details

SUMMARY(2-2)

- HATEOAS stands for Hypermedia as the Engine of Application State.
- Spring Cloud Config is a Spring Boot application which provides server-side and client-side support for externalized configuration.
- Eureka or Eureka Server is a REST-based, AWS Discovery Service.
- Zuul is an edge service or a simple gateway service that manages all incoming requests.
- Logging and monitoring are critical control systems of microservices.
- Containerization is the process of executing an application in a virtual environment by creating a package of its files and libraries and storing them in a container.