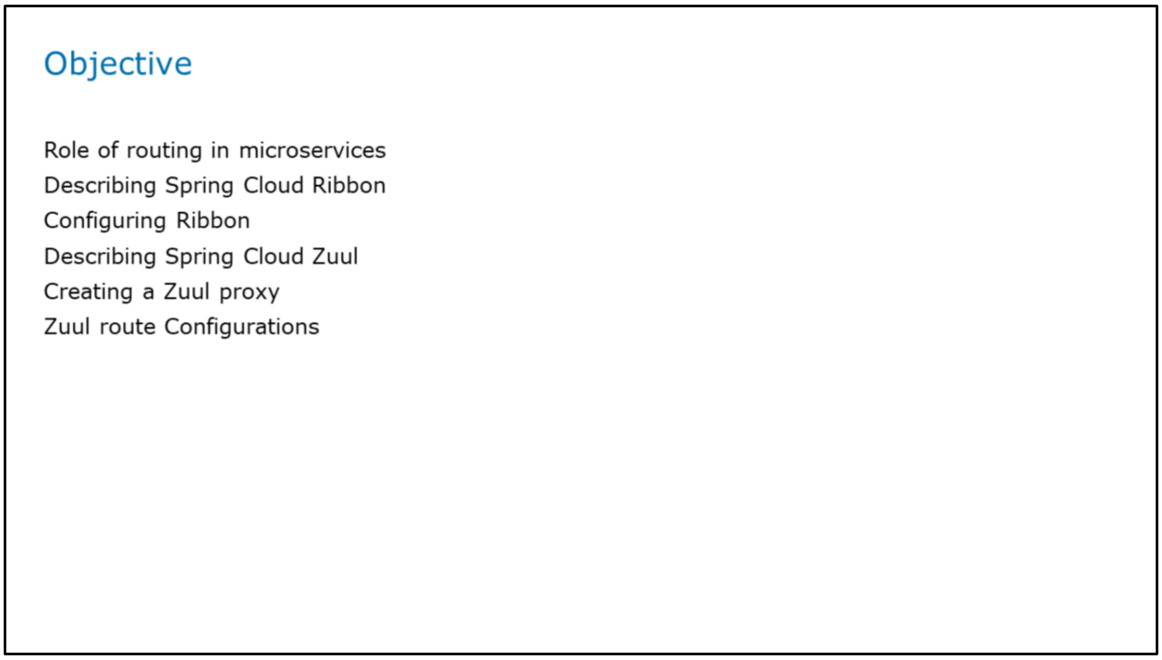
Add instructor notes here.

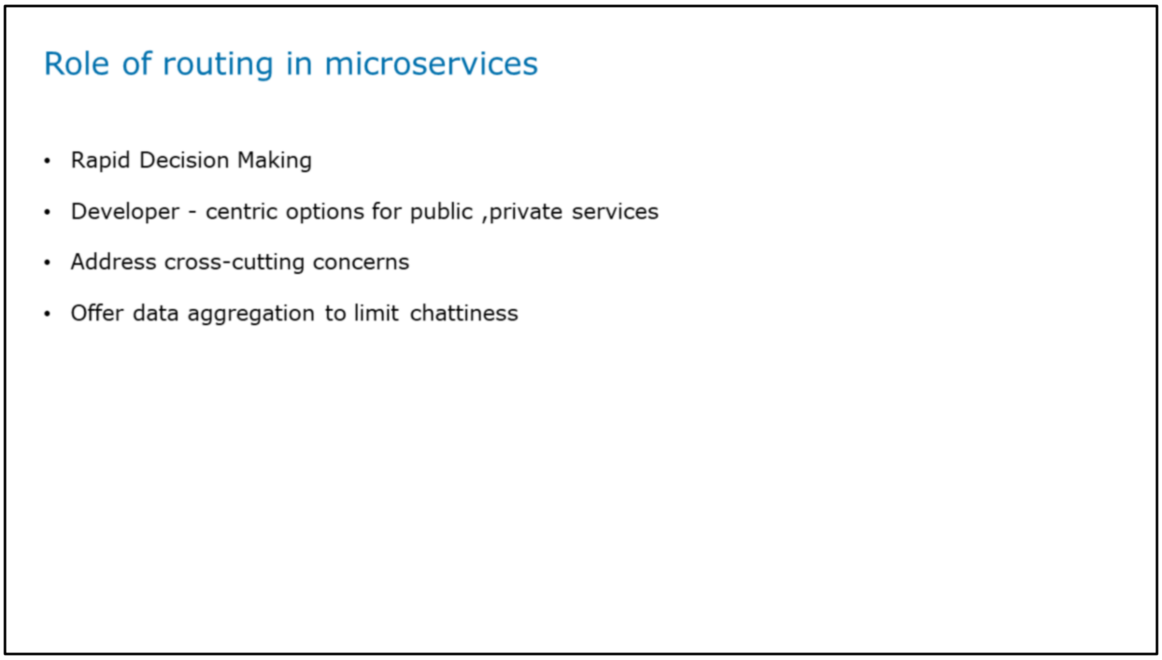
Routing Your Microservices Traffic

Capgemini



Objective

Role of routing in microservices
Describing Spring Cloud Ribbon
Configuring Ribbon
Describing Spring Cloud Zuul
Creating a Zuul proxy
Zuul route Configurations


Role of routing in microservices

- Rapid Decision Making
- Developer - centric options for public ,private services
- Address cross-cutting concerns
- Offer data aggregation to limit chattiness

f there are failures in your microservices ecosystem, then you need to fail fast by opening the circuit. This ensures that no additional calls are made to the failing service, once the circuit breaker is open. So we return an exception immediately. This pattern also monitor the system for failures and once things are back to normal, the circuit is closed to allow normal functionality.
This is a very common pattern to avoid cascading failure in your microservice ecosystem.
You can use some popular third-party libraries to implement circuit breaking in your application, such as Polly and Hystrix.
Retry Design Pattern
This pattern states that you can retry a connection automatically which has failed earlier due to an exception. This is very handy in case of temporary issues with one of your services. A lot of times a simple retry might fix the issue. The load balancer might point you to a different healthy server on the retry, and your call might be a success.
Timeout Design Pattern
This pattern states that you should not wait for a service response for an indefinite amount of time — throw an exception instead of waiting too long. This will ensure that you are not stuck in a state of limbo, continuing to consume application

resources. Once the timeout period is met, the thread is freed up.

Spring Cloud Ribbon

Ribbon is a client-side load balancer that gives you a lot of control over the behavior of HTTP and TCP clients. Feign already uses Ribbon, so, if you use @FeignClient, this section also applies.

Each load balancer is part of an ensemble of components that work together to contact a remote server on demand, and the ensemble has a name that you give it as an application developer (for example, by using the @FeignClient annotation). On demand, Spring Cloud creates a new ensemble as an ApplicationContext for each named client by using RibbonClientConfiguration. This contains (amongst other things) an ILoadBalancer, a RestClient, and a ServerListFilter.

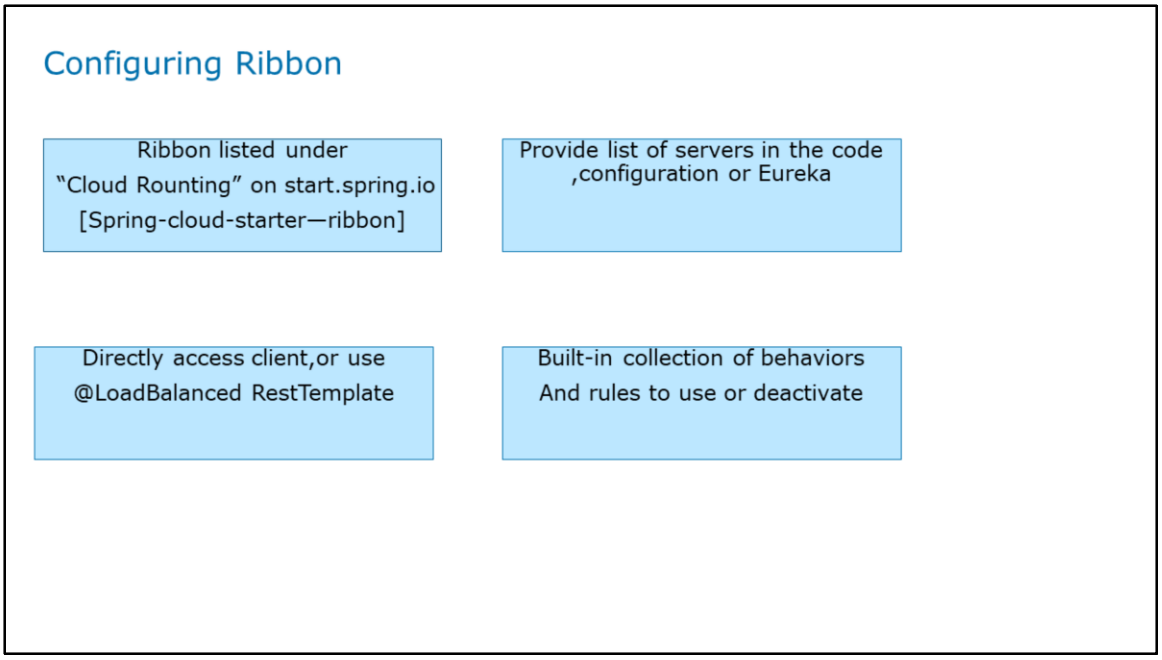
To include Ribbon in your project, use the starter with a group ID of org.springframework.cloud and an artifact ID of spring-cloud-starter-netflix-ribbon. See the Spring Cloud Project page for details on setting up your build system with the current Spring Cloud Release Train.

Key concepts –Spring Cloud Ribbon

Ribbon offers : storage of server addresses (“server list”),server freshness checks (“ping”) and server selection criteria (“rules”)

Activate in code with @LoadBalanced, @RibbonClient annotations

Extend or override by using configuration classes



Configuring Ribbon

Ribbon listed under
"Cloud Rounting" on start.spring.io
[Spring-cloud-starter—ribbon]

Provide list of servers in the code
,configuration or Eureka

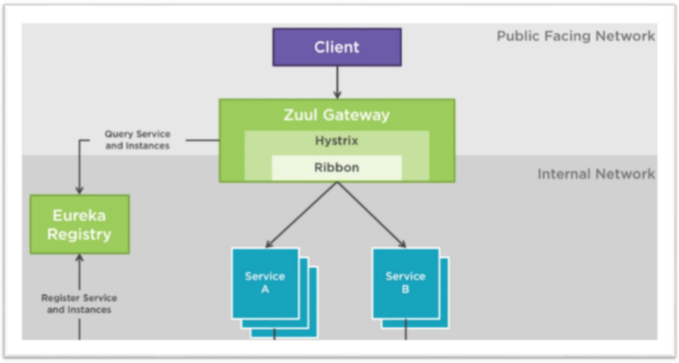
Directly access client,or use
@LoadBalanced RestTemplate

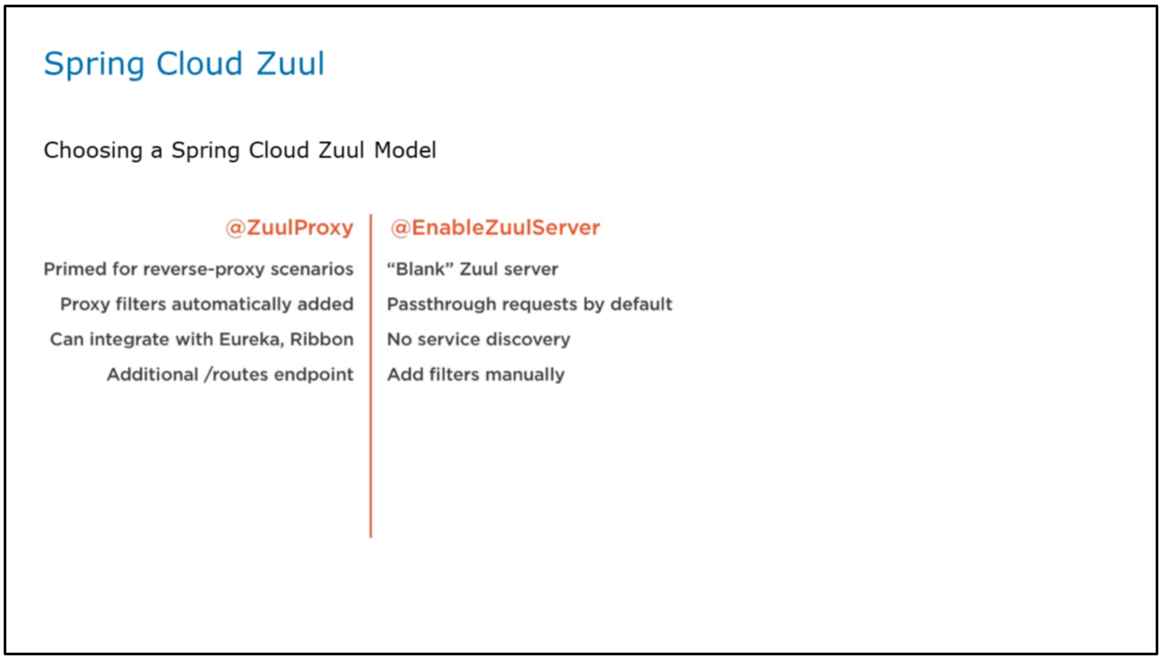
Built-in collection of behaviors
And rules to use or deactivate

Spring Cloud Zuul

Embedded proxy for routing traffic in a microservices architecture .

How Zuul Works :

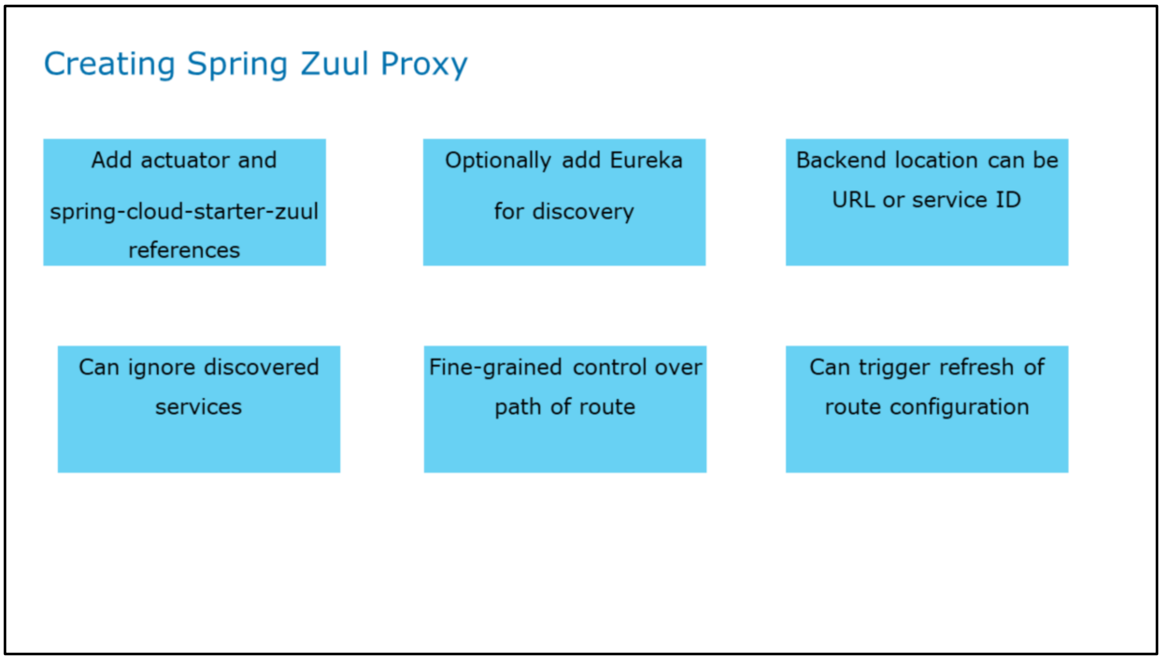




Spring Cloud Zuul

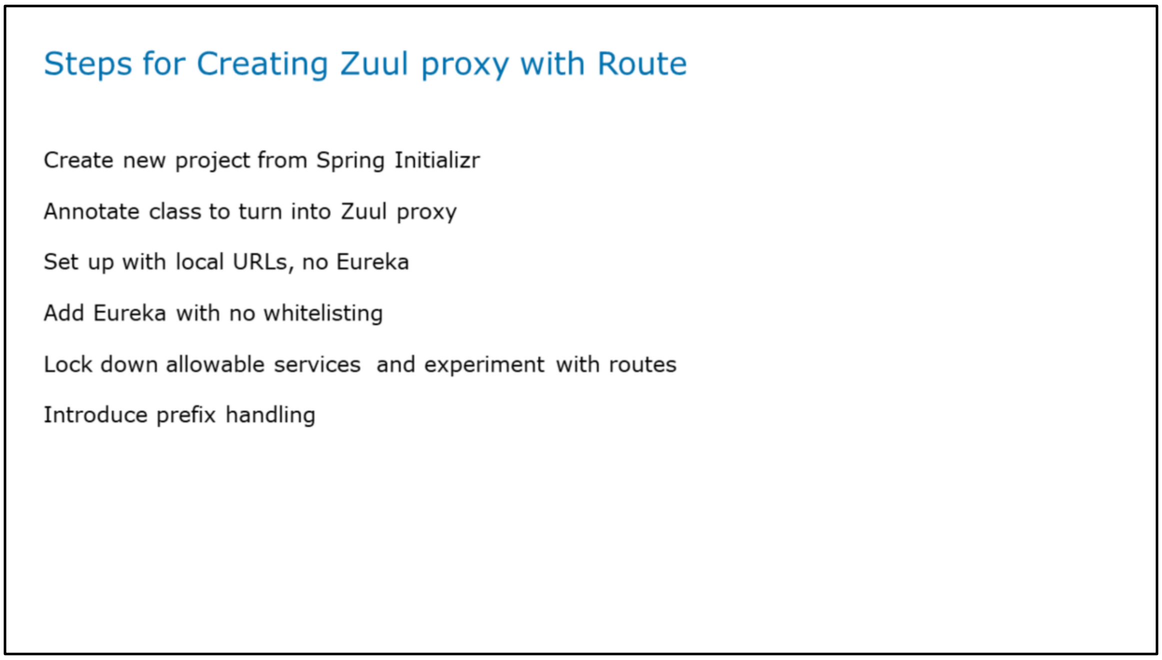
Choosing a Spring Cloud Zuul Model

@ZuulProxy	@EnableZuulServer
Primed for reverse-proxy scenarios	“Blank” Zuul server
Proxy filters automatically added	Passthrough requests by default
Can integrate with Eureka, Ribbon	No service discovery
Additional /routes endpoint	Add filters manually



Creating Spring Zuul Proxy

- Add actuator and spring-cloud-starter-zuul references
- Optionally add Eureka for discovery
- Backend location can be URL or service ID
- Can ignore discovered services
- Fine-grained control over path of route
- Can trigger refresh of route configuration



Steps for Creating Zuul proxy with Route

Create new project from Spring Initializr

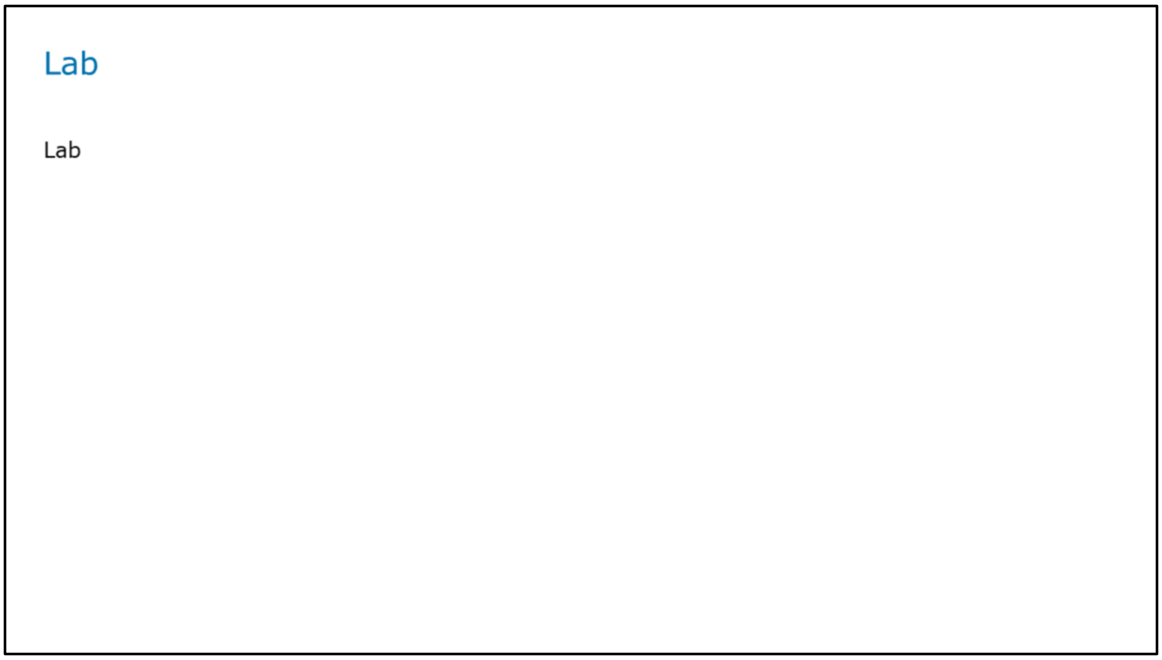
Annotate class to turn into Zuul proxy

Set up with local URLs, no Eureka

Add Eureka with no whitelisting

Lock down allowable services and experiment with routes

Introduce prefix handling



Lab

Lab