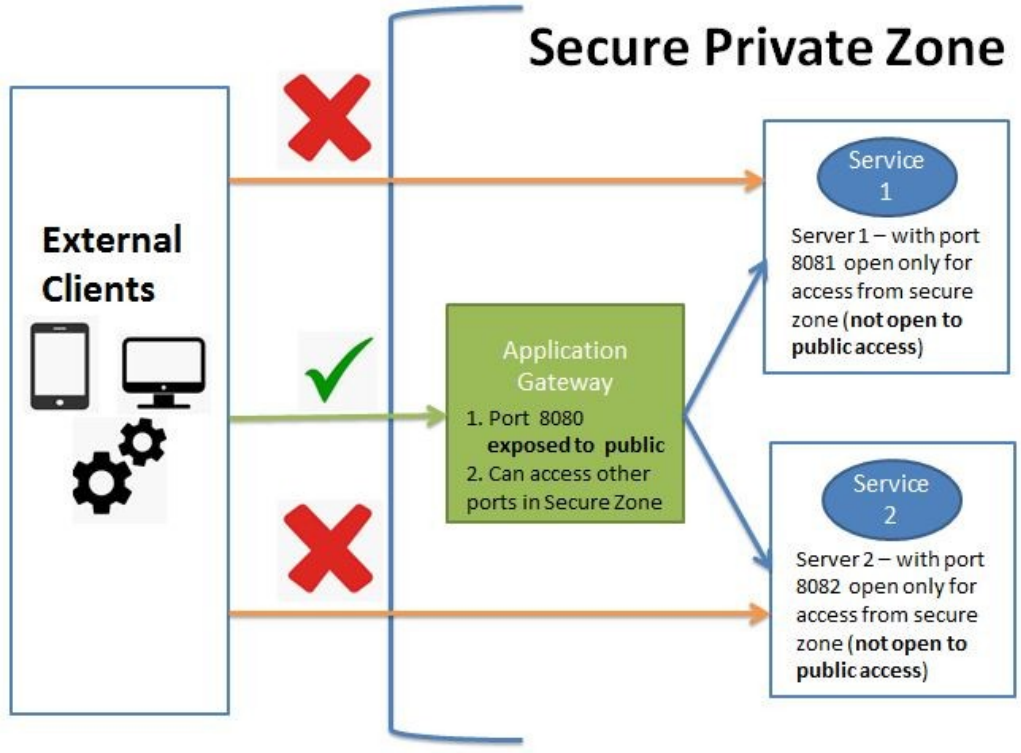
Spring Cloud Tutorial - Spring Cloud Gateway Hello World Example

Zuul is a blocking API. A blocking gateway api makes use of as many threads as the number of incoming requests. So this approach is more resource intensive. If no threads are available to process incoming request then the request has to wait in queue.
In this tutorial we will be implementing API Gateway using Spring Cloud Gateway. Spring Cloud Gateway is a non blocking API. When using non blocking API, a thread is always available to process the incoming request. These request are then processed asynchronously in the background and once completed the response is returned. So no incoming request never gets blocked when using Spring Cloud Gateway.
We will first look at what is API gateway and why are they needed. Then we will be exploring the Spring Cloud Gateway Architecture and implement an API Gateway using it.

What is an API Gateway? Why do we need it?

An API Gateway acts as a single entry point for a collection of microservices. Any external client cannot access the microservices directly but can access them only through the application gateway
In a real world scenario an external client can be any one of the three-

Mobile Application
Desktop Application
External Services or third party Apps



The advantages of this approach are as follows-

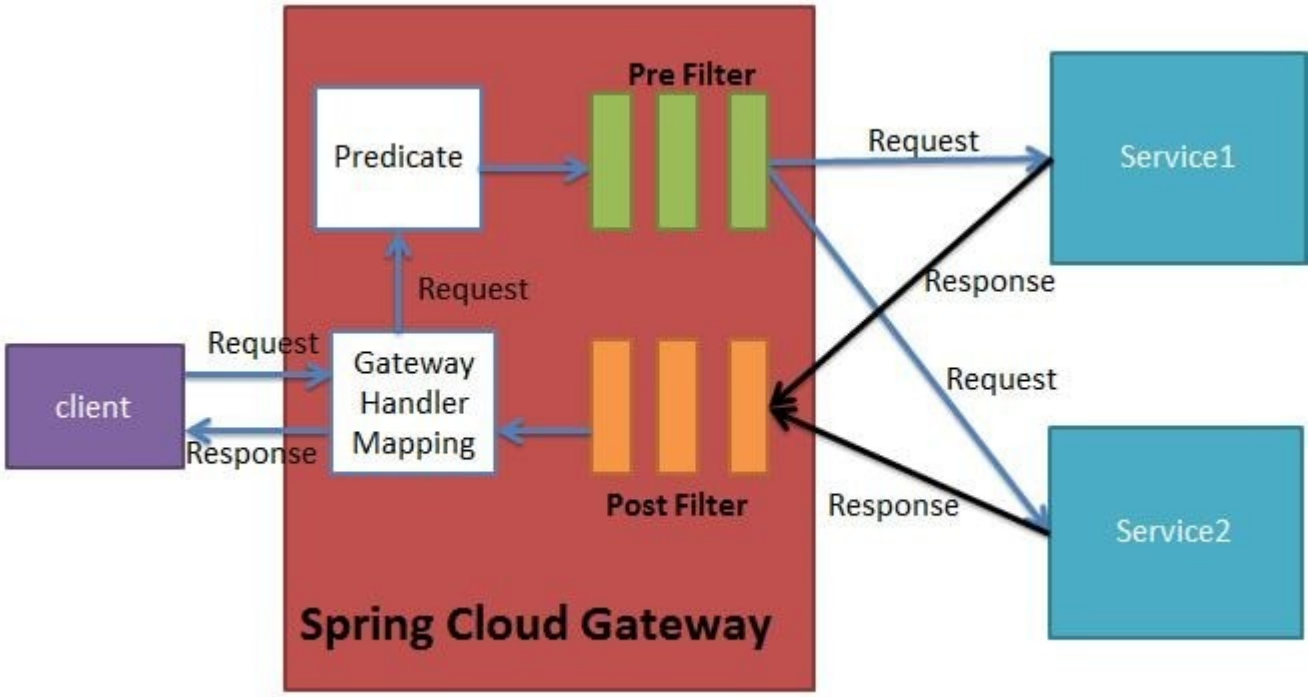
1. This improves the security of the microservices as we limit the access of external calls to all our services.
2. The cross cutting concerns like authentication, monitoring/metrics, and resiliency will be needed to be implemented only in the API Gateway as all our calls will be routed through it.
3. The client does not know about the internal architecture of our microservices system. Client will not be able to determine the location of the microservice instances.

4. Simplifies client interaction as he will need to access only a single service for all the requirements.
5. Spring Cloud Gateway Architecture

Spring Cloud Gateway is API Gateway implementation by Spring Cloud team on top of Spring reactive ecosystem.
It consists of the following building blocks-
Route:

- Route the basic building block of the gateway.
- It consists ofID destination URI Collection of predicates and a collection of filters
- A route is matched if aggregate predicate is true.

Predicate:
This is similar to Java 8 Function Predicate. Using this functionality we can match HTTP request, such as headers , url, cookies or parameters.
Filter:
These are instances Spring Framework GatewayFilter. Using this we can modify the request or response as per the requirement.



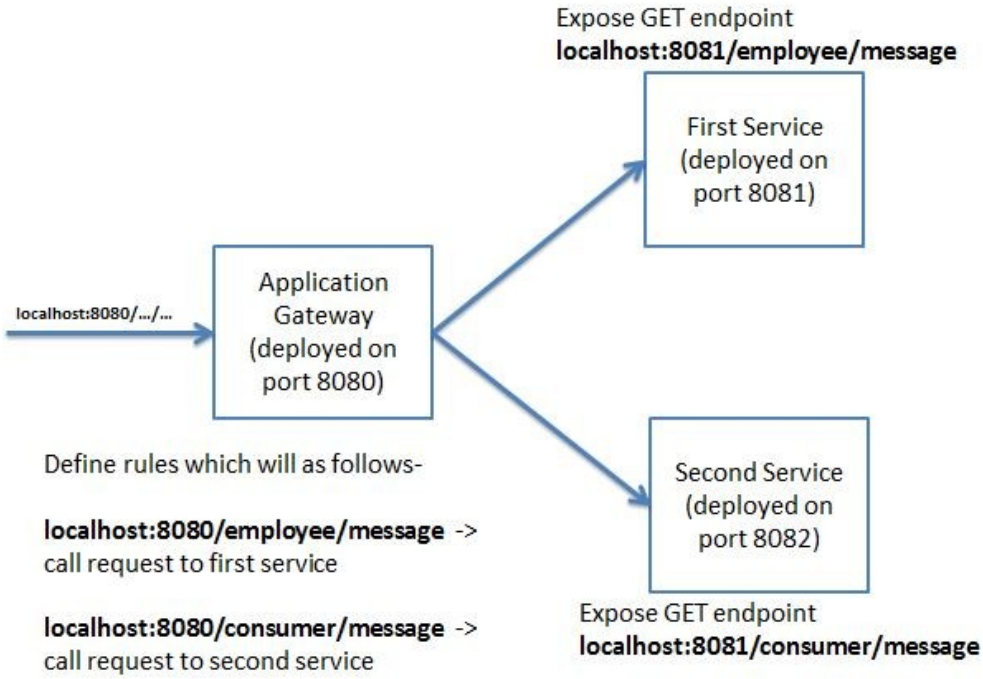
spring cloud gateway architecture
When the client makes a request to the Spring Cloud Gateway, the Gateway Handler Mapping first checks if the request matches a route. This matching is done using the predicates. If it matches the predicate then the request is sent to the filters.

Implementing Spring Cloud Gateway
Using Spring Cloud Gateway we can create routes in either of the two ways -

1. Use java based configuration to programmatically create routes
2. Use property based configuration(i.e application.properties or application.yml) to create routes.

In this tutorial we will be implementing Spring Cloud Gateway using both configurations.

We will be implementing Spring Cloud Gateway application which routes request to two other microservices depending on the url pattern.



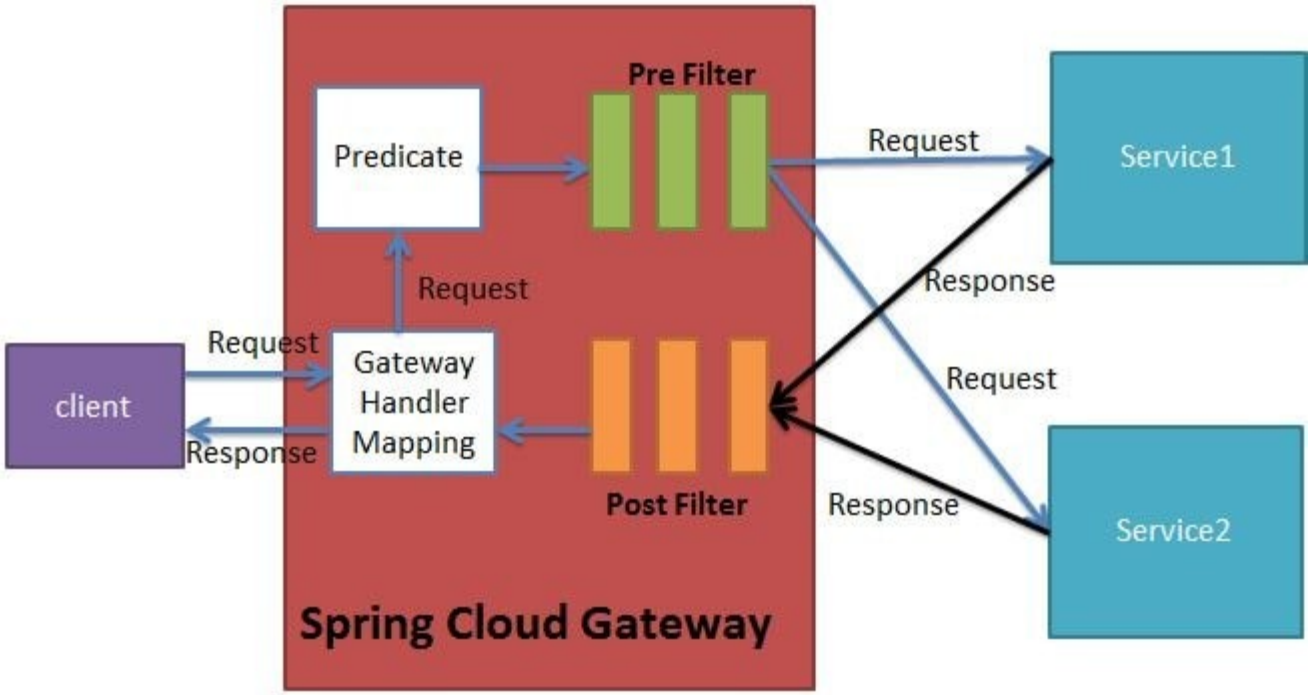
For lab refer cheetsheet attached

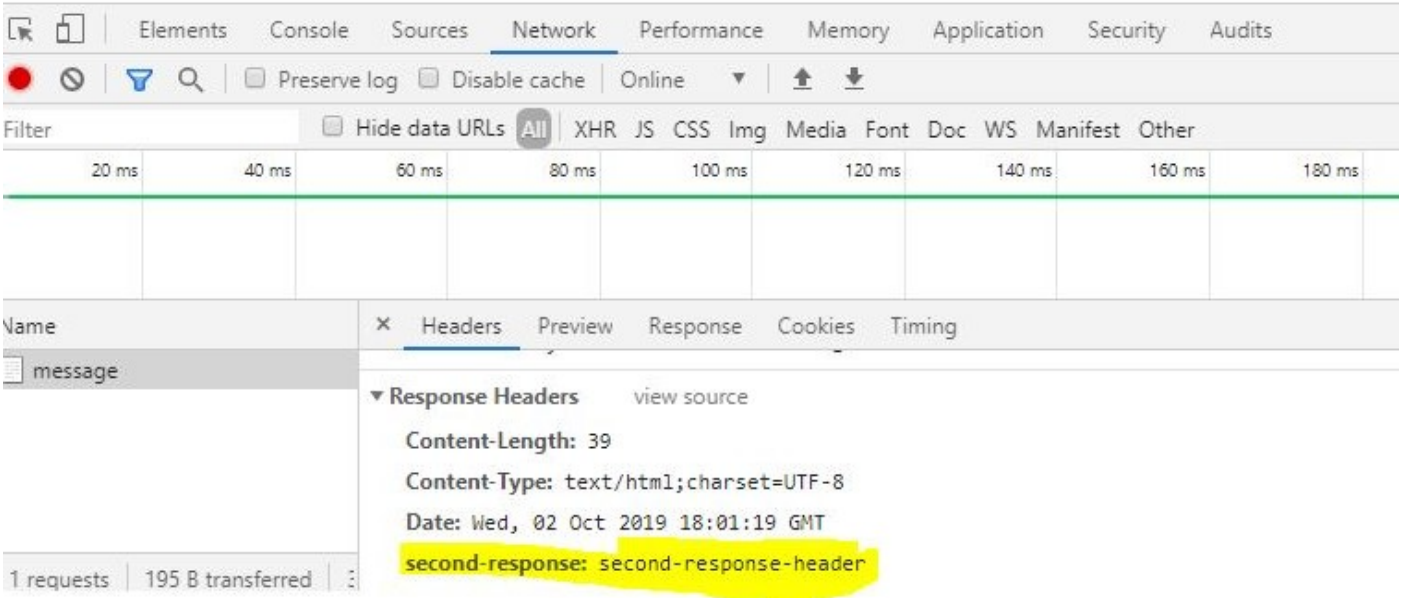
Spring Cloud Tutorial - Spring Cloud Gateway Filters Example

In this tutorial we will be making use of Spring Cloud provided filters and also create custom filters for our spring cloud gateway.
In the next tutorial we will be integrating Spring Cloud Gateway with Eureka Service Discovery.

- Using Predicates Spring Cloud Gateway determines which route should get called. Once decided the request is the routed to the intended microservice.
- Before routing this request we can apply some filters to the request. These filters are known as pre filters.
- After applying the filters the intended micoservice call is made and the response is returned back to the Spring Cloud Gateway which returns this response back to the caller.
- Before returning the response we can again apply some filters to this response. Such filters are called post filters.

spring cloud gateway architecture
As specified in the Spring Cloud Gateway Documentation, Spring Cloud provides a number of built in filters. Also we can create our own custom filter to suit our business requirement. In this tutorial we will be looking at the various Filters that can be used with Spring Cloud Gateway.


Pre Filter
Predicate
Request
Service1
Request
Response
Request
client
Request
Gateway
Handler
Mapping
Request
Response
Post Filter
Response
Service2
Spring Cloud Gateway


Elements
Console
Sources
Network
Performance
Memory
Application
Security
Audits
Preserve log
Disable cache
Online
Filter
Hide data URLs
All
XHR
JS
CSS
Img
Media
Font
Doc
WS
Manifest
Other
20 ms
40 ms
60 ms
80 ms
100 ms
120 ms
140 ms
160 ms
180 ms
Name
message
Headers
Preview
Response
Cookies
Timing
Response Headers
view source
Content-Length: 39
Content-Type: text/html;charset=UTF-8
Date: Wed, 02 Oct 2019 18:01:19 GMT
second-response: second-response-header
1 requests
195 B transferred