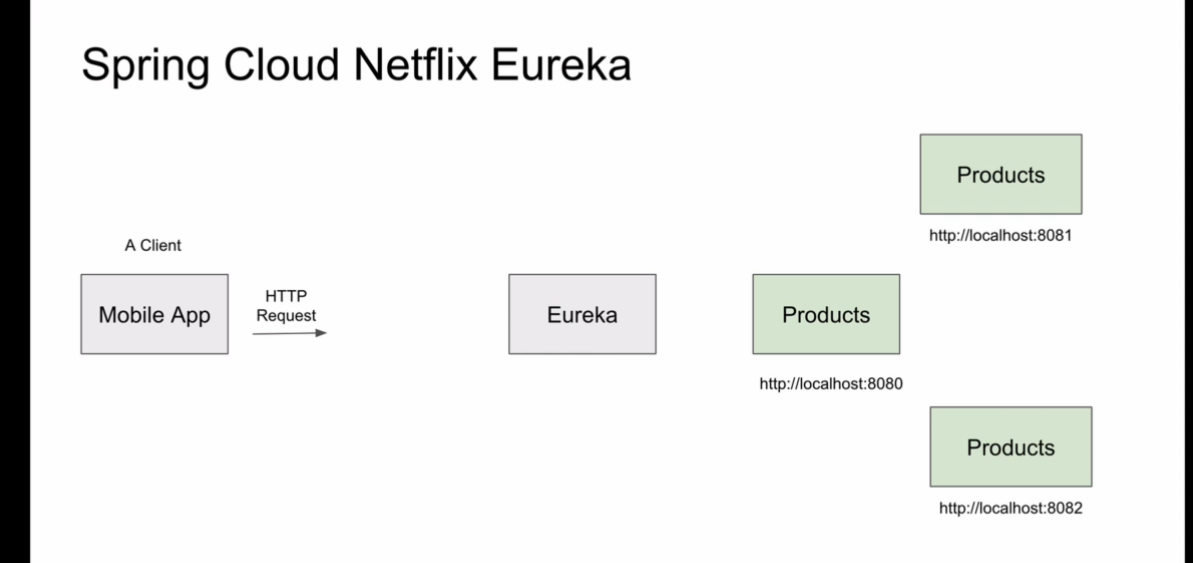
***Spring Cloud and Micro services:***

A single mobile application connect to 1 BE server

But what if we have multiple instance of that server

Then how the client server would knows all the service of the product instance on

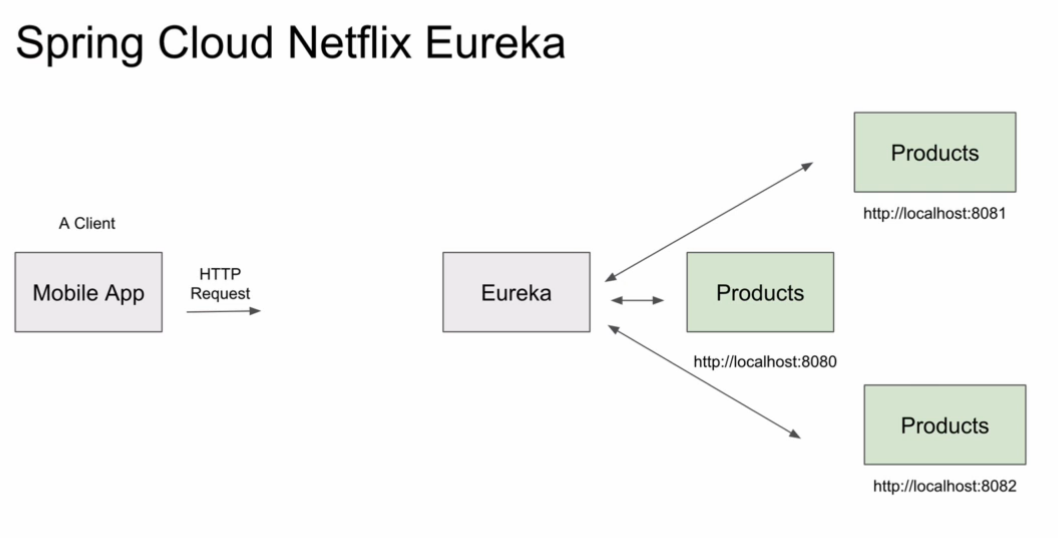
1. Different IP + Port



Eureka will know the address of all the micro services and it will act as a service discovery and it’s instances to the client with one

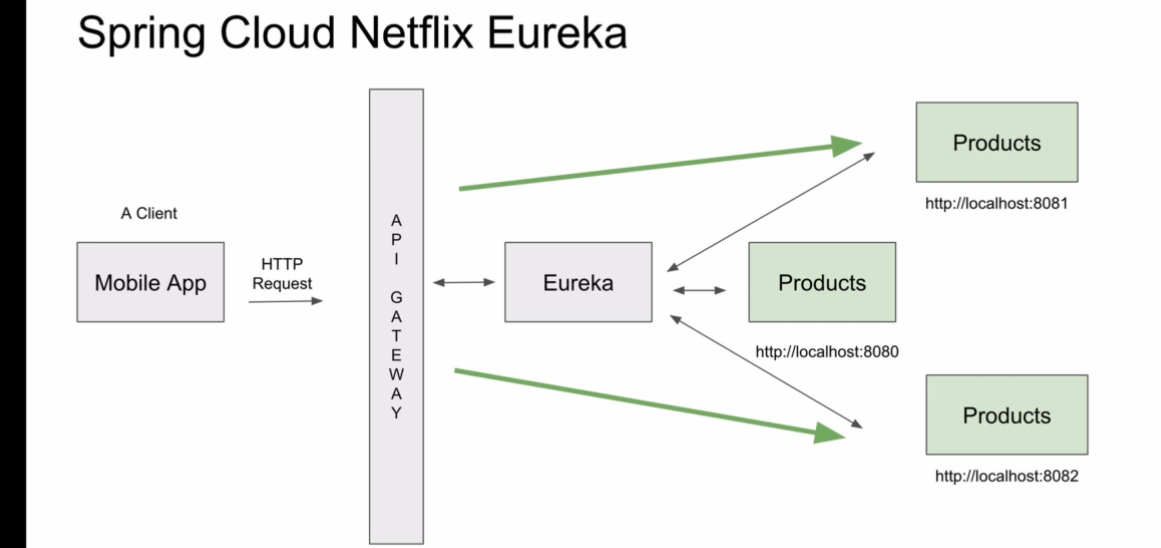
https://{IP\_ADDDRESS}:{PORT}/{EndPoint}

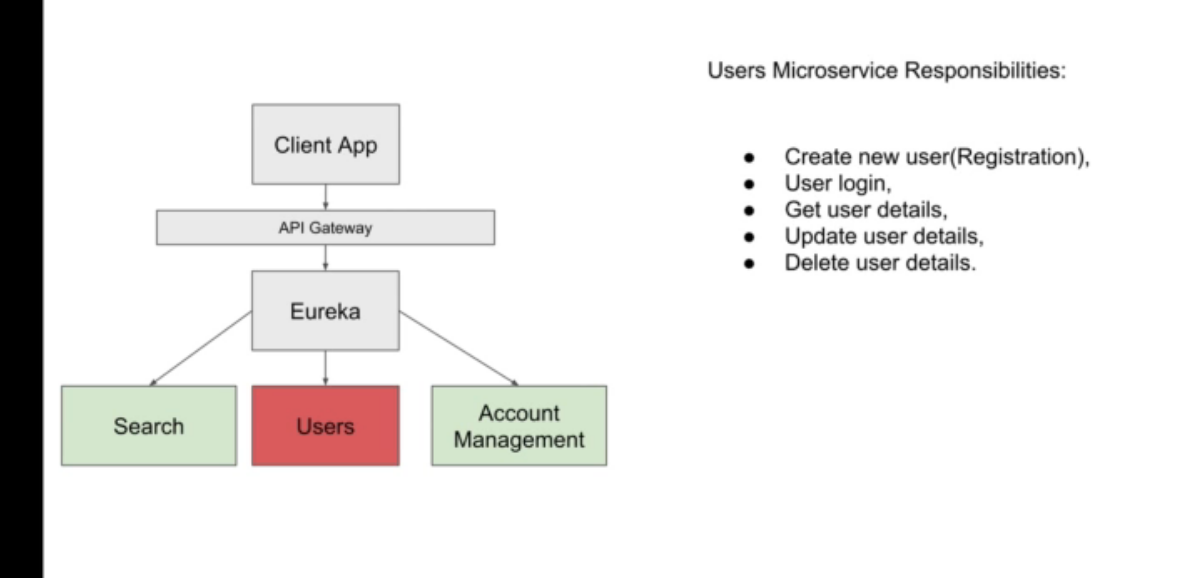
for eureka server!!



Basically we can say that eureka server is a API\_GATEWAY

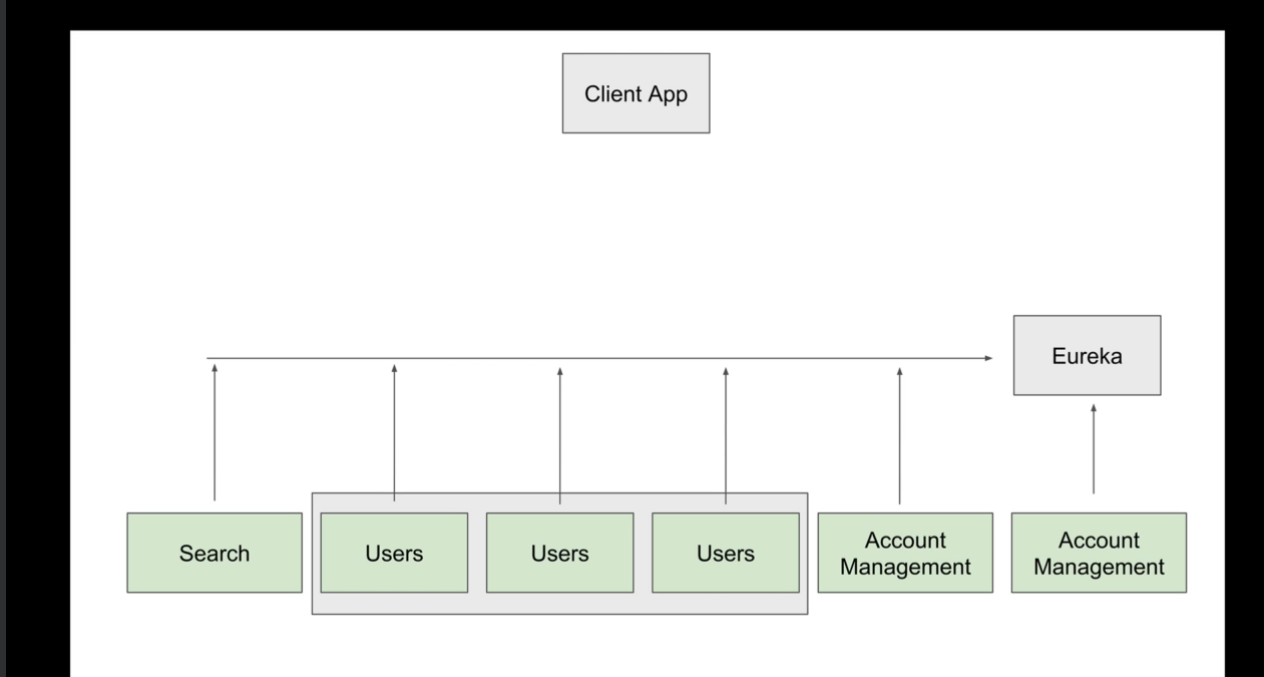
Given in the below diagram





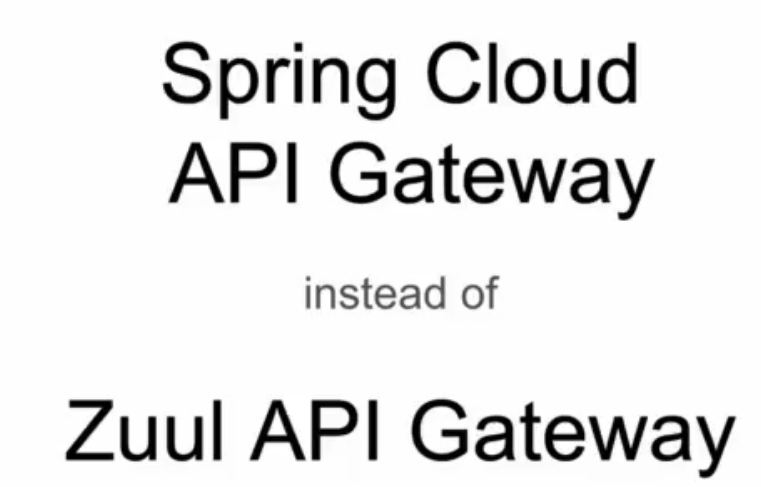
This is the architecture diagram of user micro services

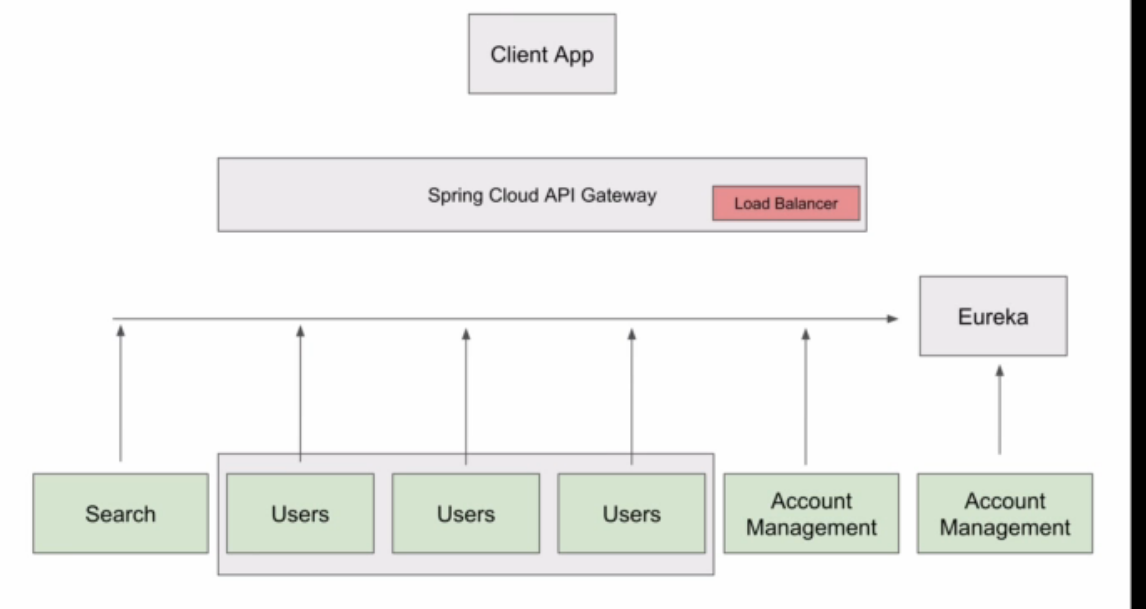
1. We will learn how to make micro services and how to register with eureka:



In this Eureka is registering with eurka server for service availability

1. We are going to go ahead with Spring Cloud Api gateway in Built

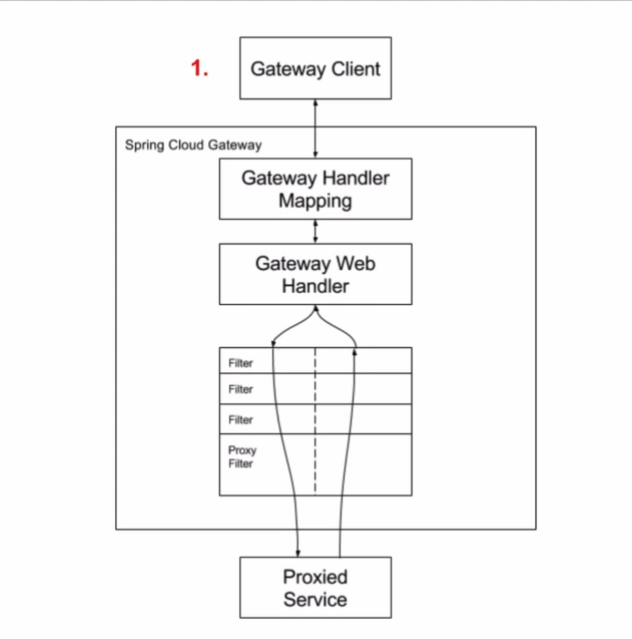




API GATEWAY within built load balancer to distribute the load around the different instance of the services for HA.

1. We can impl custom filter
2. We can impl the custom route for services

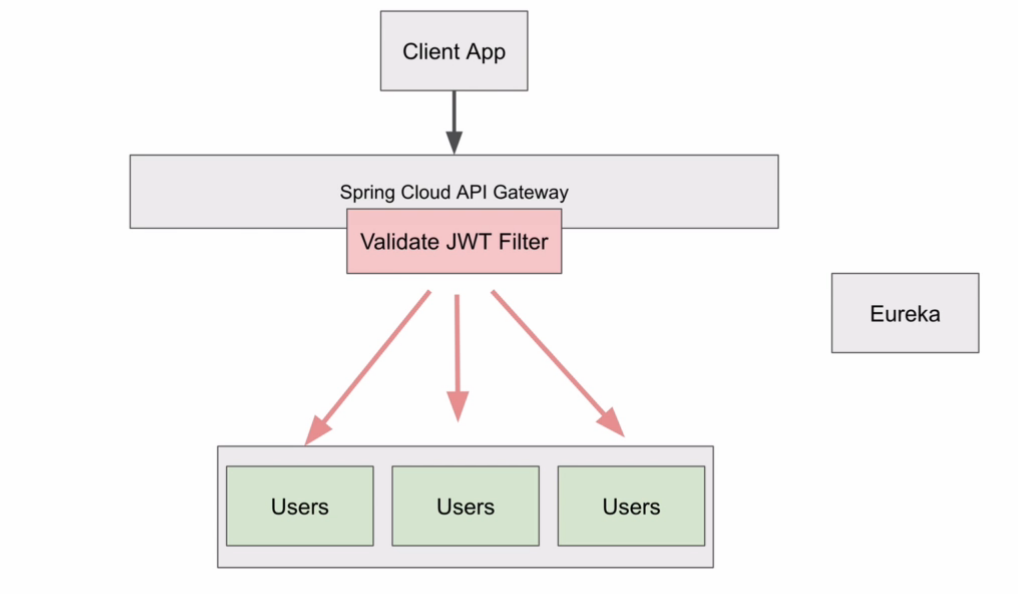
# How Spring Cloud API – Gateway Works internally



1. Gateway handler receive the request and gateway map handler will check the configure routed for the particular api hit
2. Gate way web handler will allow request to filter chain

There two arrows from proxied service one is going down and other one is going up

So after all the filter chain pass the request get back to the web handler.



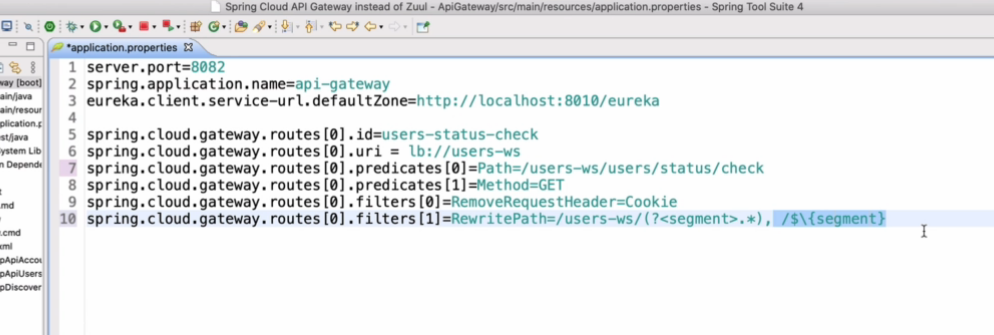
Api gateway filter the request for JWT and route the request to the particular services

Validate JWT filter if valid then request will pass otherwise it will not pass through the request

By Introducing the Spring cloud api gate way we can define the end point and entry point of execution for all the services and it will work as a router who will route from api gateway to the exact services

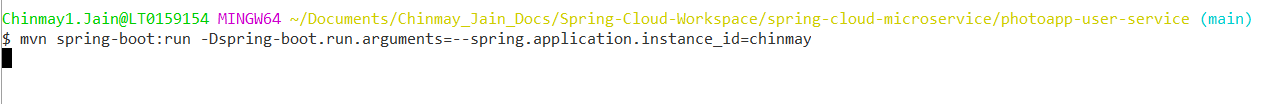
Instead of using **Eureka Discovery services Ip we can use the api Gateway End-point**

**Defining the Route in the configuration file is as follows :**

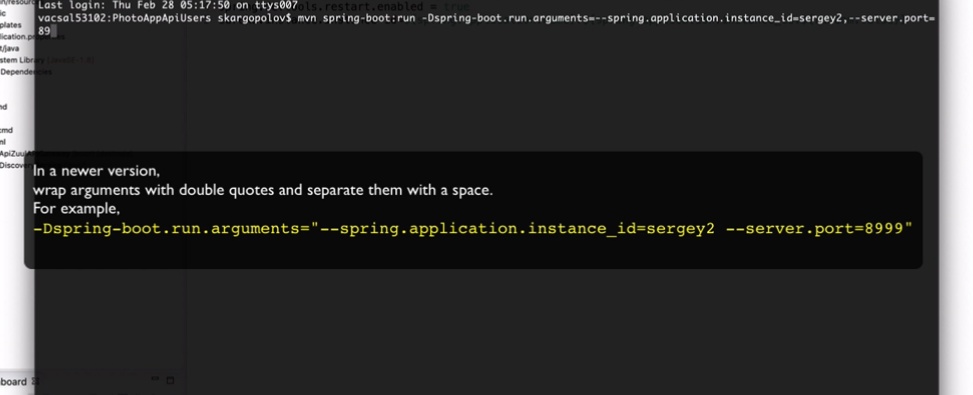
****

1. Route is a array type we can have any number of routes in the project
2. Route[0].id = will define the unique id for different services
3. Route[0].predicate[0]=path=”Whole path of the entire user api call”

To run from terminal and pass the argument

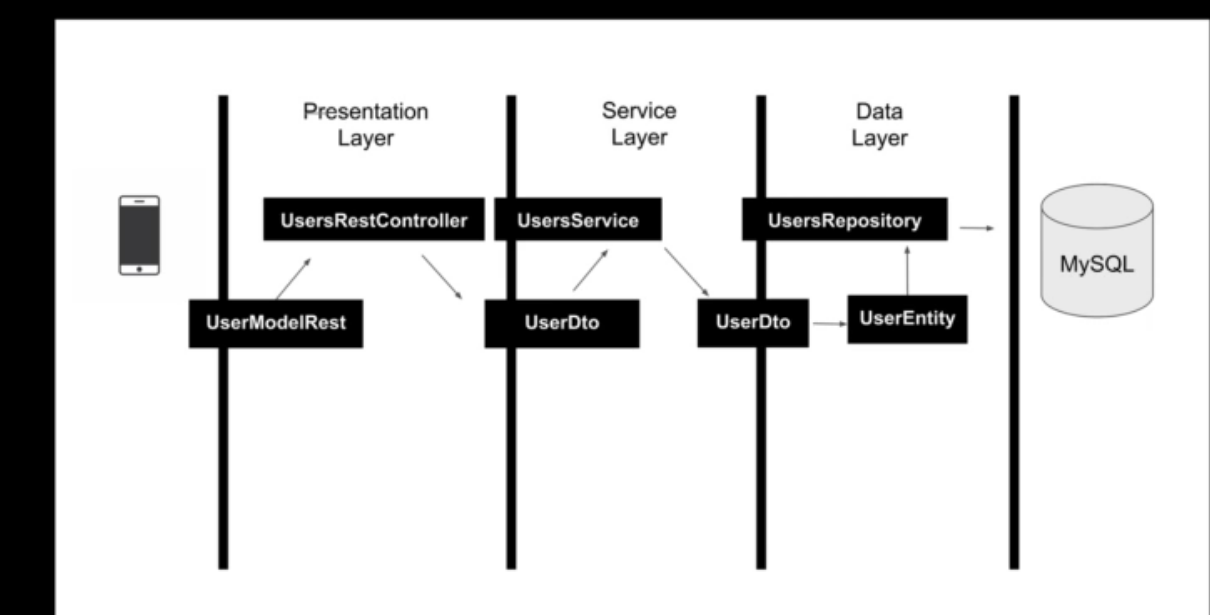


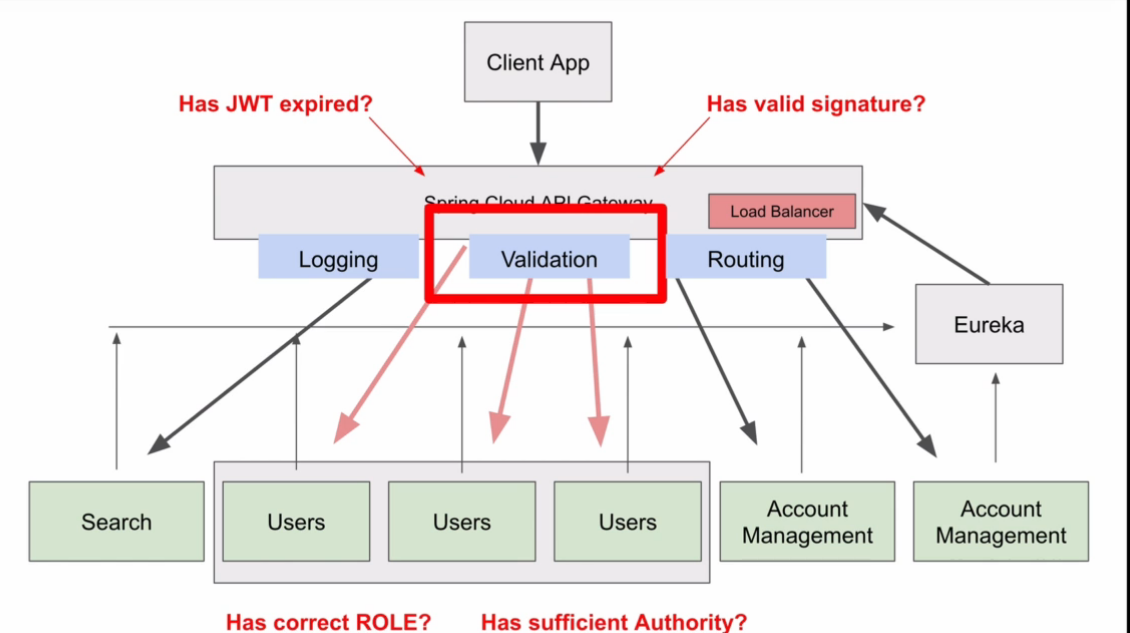
This one is for mvn cmd



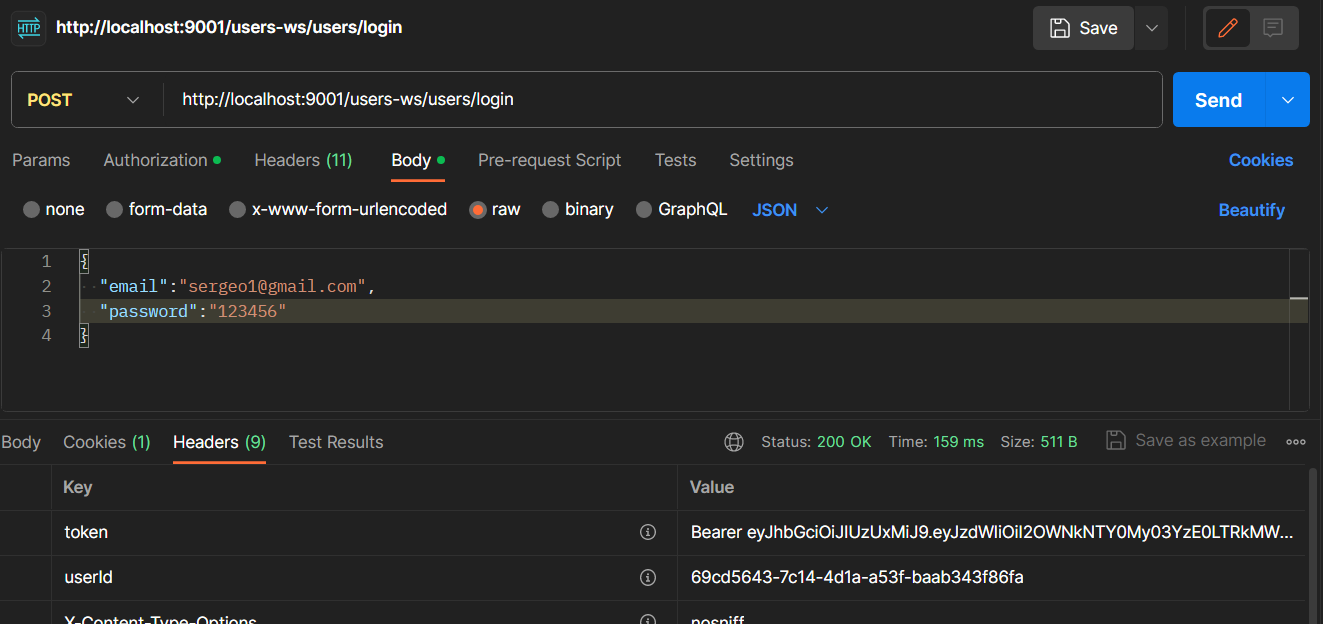
For gradle we can use the cmd :

***./gradlew bootRun -Pargs='--eureka.instance.instance-id=yourValue'***

******

******

***Here we will get into validation of JWT at api gate way level if we sign the token with key we will verify with the help of filters at Api gateway with   
  
HttpFilters – Prefilter and Post filter & GlobalFilter***

******

***As we can see that JWT token is Generated to generate JWT we have Some steps to follow in Spring Security***

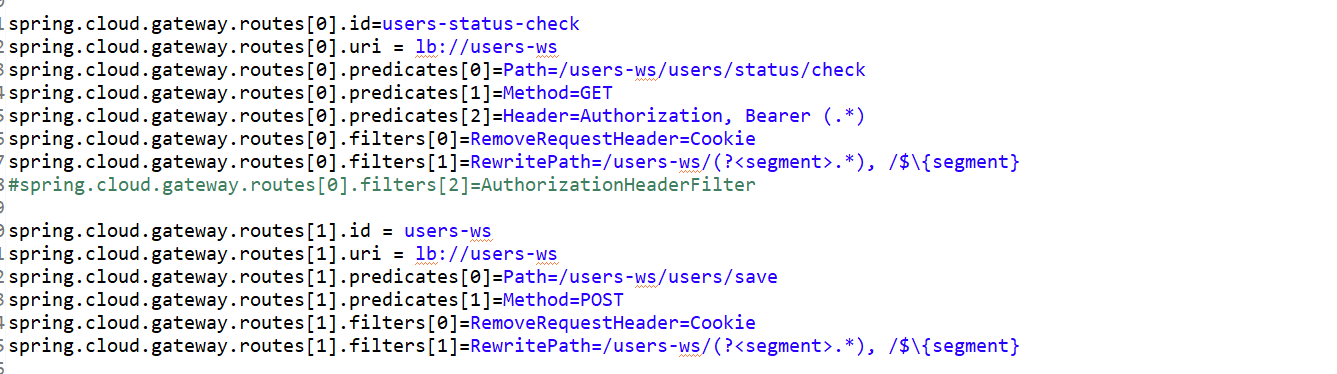
1. ***Create One custom class Authentication filter which will extend the*** UsernamePasswordAuthenticationFilter
2. ***In this class override one method attempAuthenctication which will return the Authentication object***

******

1. ***From this authentication we call the userService and connect the repository to check the credentials of user and load***
2. ******
3. ***This above code will generate the JWT according to your credentials.***

***# Now we will some configuration in api gateway headers if the headers doesn’t contains the value of JWT token then routing will no work in from Api gateway to Backend service.***

***Like this :***

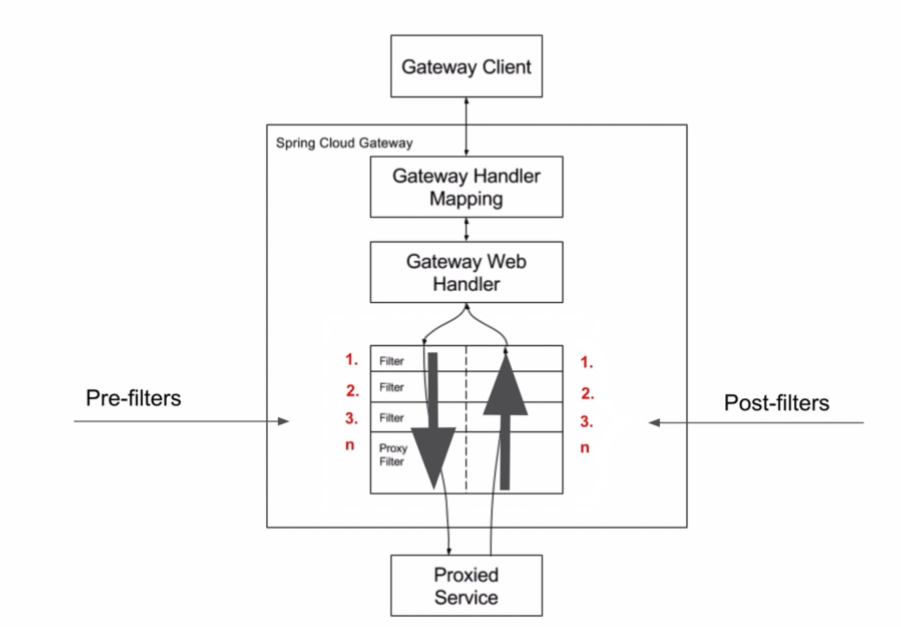
******

spring.cloud.gateway.routes[0].predicates[2]=Header=Authorization, Bearer (.\*)

this piece of line will indicate that if the Authorization token is not present in the headers then we will not allowed to route the request to the backend server from API gateway.

***# Global Filters In Spring Cloud***

1. ***PreFilter***
2. ***postFilter***

******

***First request will land onto to the Spring Cloud api gateway***

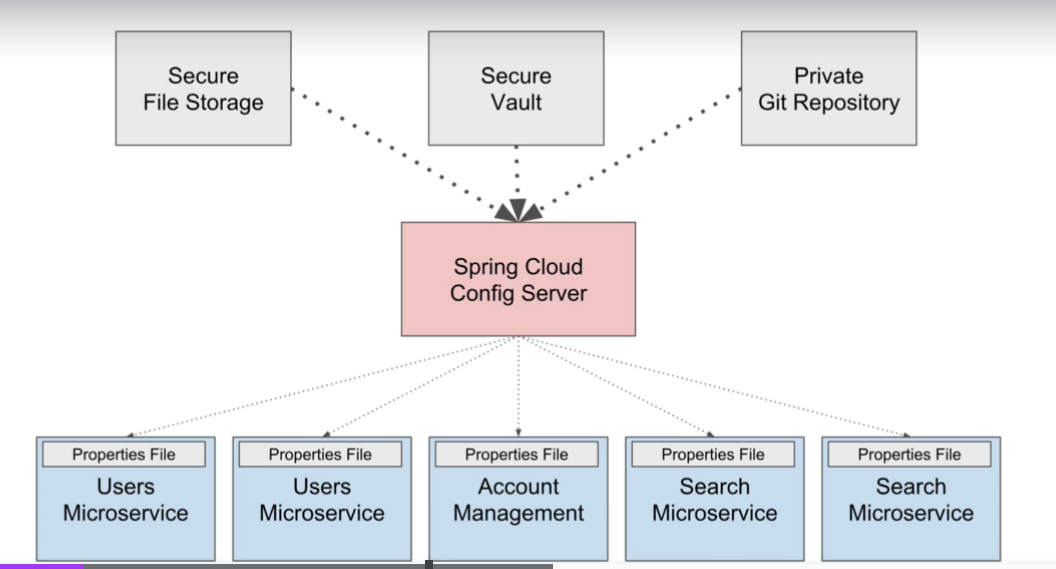
***And and we will verify the Header if we have Authorization header or not***

***Then we will call the Pre filter after the request and then proxy return the response then it will run the Post – Filter***

***Pre-Filter the calls will go from top to down what ever the @order()***

***Is define & post filter we will call the filter-chain bottom to top @order().***

***TOPIC: Spring Cloud Config Server –Git backend***

******

***Each of the Different service config file will present at one point only***

***Centralized Place to Access the Config file from single Source Only!!***

***#TOPIC: Distributed tracing with micrometer and Zipkin***