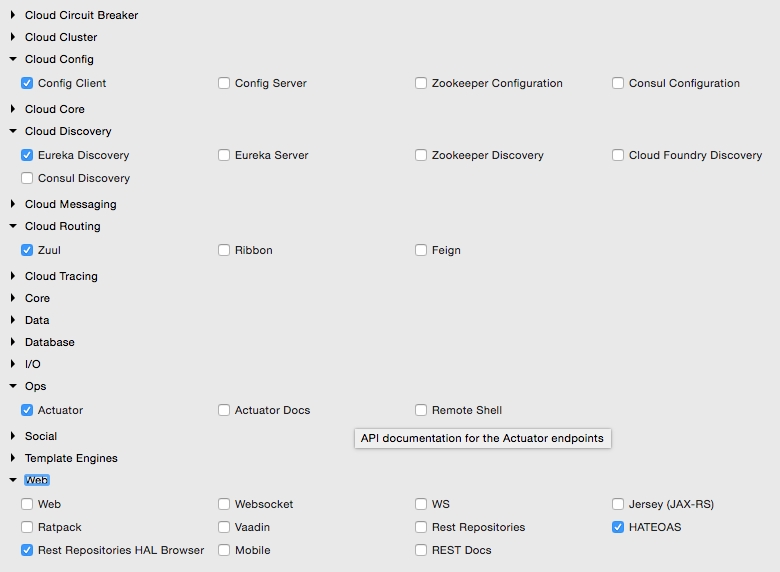
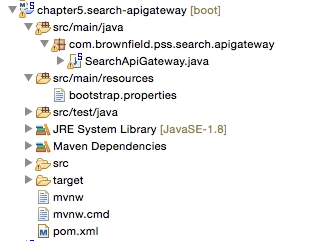
* Lab 16 - Setting Up Zuul
* Convert the microservices one by one. Start with Search API Gateway. Create a new Spring Starter project, and select **Zuul**, **Config Client**, **Actuator**, and **Eureka Discovery**:
* 
* The project structure for search-apigateway is shown in the following diagram:
* 
* The next step is to integrate the API gateway with Eureka and the Config server. Create a search-apigateway.property file with the contents given next, and commit to the Git repository.
* This configuration also sets a rule on how to forward traffic. In this case, any request coming on the /api endpoint of the API gateway should be sent to search-service:
* spring.application.name=search-apigateway  
  zuul.routes.search-apigateway.serviceId=search-service  
  zuul.routes.search-apigateway.path=/api/\*\*  
  eureka.client.serviceUrl.defaultZone:http://localhost:8761/eureka/
* search-service is the service ID of the Search service, and it will be resolved using the Eureka server.
* Update the bootstrap.properties file of search-apigateway as follows. There is nothing new in this configuration—a name to the service, the port, and the Config server URL:
* spring.application.name=search-apigateway  
  server.port=8095  
  spring.cloud.config.uri=http://localhost:8888
* Edit Application.java. In this case, the package name and the class name are also changed to com.brownfield.pss.search.apigateway and SearchApiGateway respectively. Also add @EnableZuulProxy to tell Spring Boot that this is a Zuul proxy:
* @EnableZuulProxy  
  @EnableDiscoveryClient  
  @SpringBootApplication  
  public class SearchApiGateway {
* Run this as a Spring Boot app. Before that, ensure that the Config server, the Eureka server, and the Search microservice are running.
* Change the website project's CommandLineRunner as well as BrownFieldSiteController to make use of the API gateway:
* Flight[] flights = searchClient.postForObject("http://search-apigateway/api/search/get", searchQuery, Flight[]