## Swagger Integration:

Swagger is an API description format for REST APIs. It is equally suitable both for designing new APIs and for documenting your existing APIs.

Here we are going to generate API documentation for our APIs with Swagger using Annotations in Spring Boot.

You can access API-Docs at ***http://{host}:{port}/v2/api-docs*** and UI at ***http:// {host}:{port}/swagger-ui.html***

#### Dependencies:

Add following dependencies to your pom file.



#### Code:

* Configuration Class
  + Add *@EnableSwagger2* annotation to the configuration class.
  + Create a bean that returns Docket object with API details.



* API Definition Class
  + Create API definition class that implements *ReaderListener* interface
  + We are using this class to define security schemes



* Controller Class
  + Annotation *@Api* at class level for displaying API description
  + Annotation *@ApiOperation* at resource level for displaying resource details like input/output type, security and all other details
  + Annotation *@ApiResponses* is used for setting response codes and messages



## Fault Tolerance

Here, we think about possible point of failures and how to avoid them. We will focus on unavailability of any service and how to avoid problems by them.

### Services

We need to define some mechanism so that if a service is down or unavailable due to heavy traffic, our application should not stop working or misbehave. We can resolve this issue by deploying multiple instances of one service. We use this method to scale our service and for load balancing.

For this we need to make following changes in our application:

* Deploy multiple instance of the service on different locations/ports
* Register all services with Eureka with same Service-ID
* Use Zuul or Ribbon for client side load balancing
* Add *@EnableRetry* annotation in Zuul class and below dependency in order to retry, it will retry for REST call if any instance of the service is down and call is sent to that instance.



### Eureka

As we know Eureka is used for service registration, so if Eureka service becomes unavailable due to some reason our application services will not be able to find and communicate with each other. We can overcome this problem by deploying multiple Eureka instances and letting them know about each other. They sync all services and details with one another and if one instance goes down, our application can still use different instances.

We need to make following changes for running two instances:



### Zuul

Zuul is used for service discovery and proxy services. Services communicate with each other using Zuul service. It is also used for load balancing as it internally uses Ribbon. We also deploy multiple instances of Zuul to avoid miscommunication if it gets unavailable.

In order to achieve this, we deploy multiple instances of Zuul on different servers/ports and we use Ribbon for client side load balancing on zuul service.

