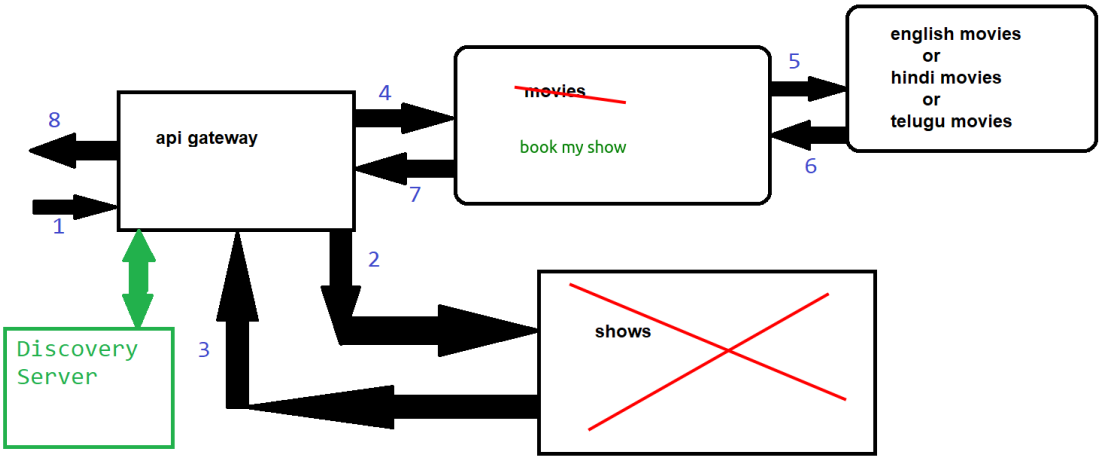
Microservices (SpringBoot 2.7.3)



eureka-server

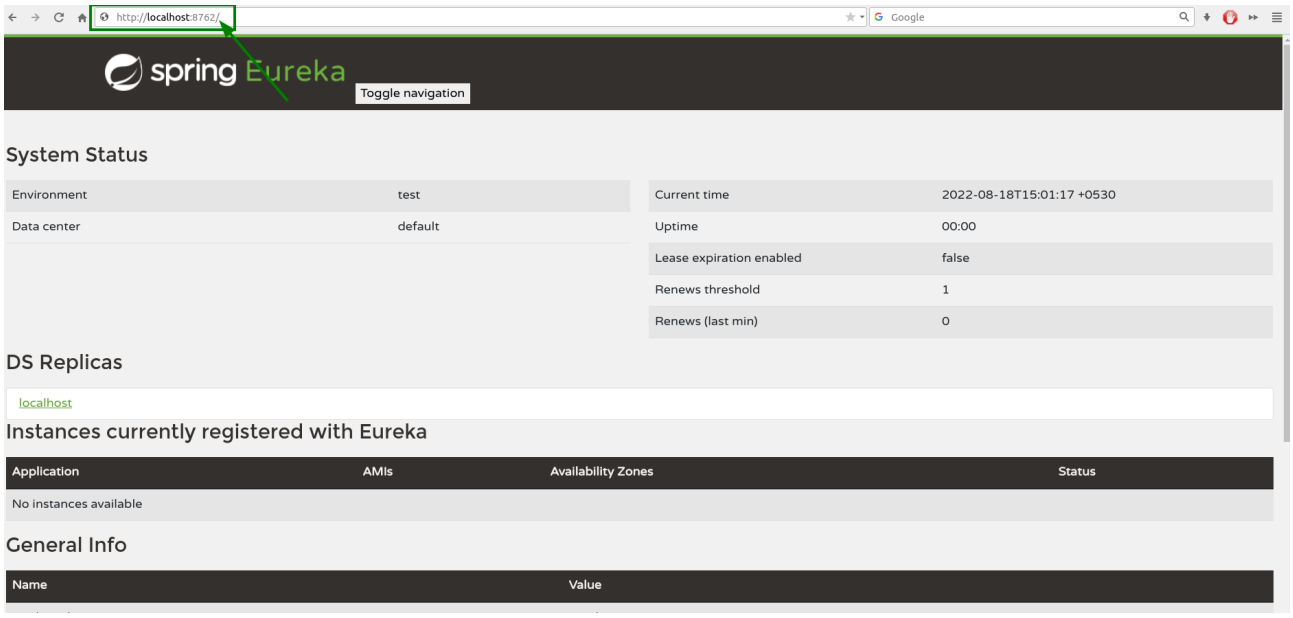
dependencies

spring-cloud-starter-netflix-eureka-server
spring-boot-starter-web

application.properties

server.port=8762
eureka.client.registerWithEureka= false
eureka.client.fetchRegistry= false
eureka.client.serviceUrl.defaultZone= http://localhost:8762/eureka

Class with main method should contain
@EnableEurekaServer



api-gateway

dependencies

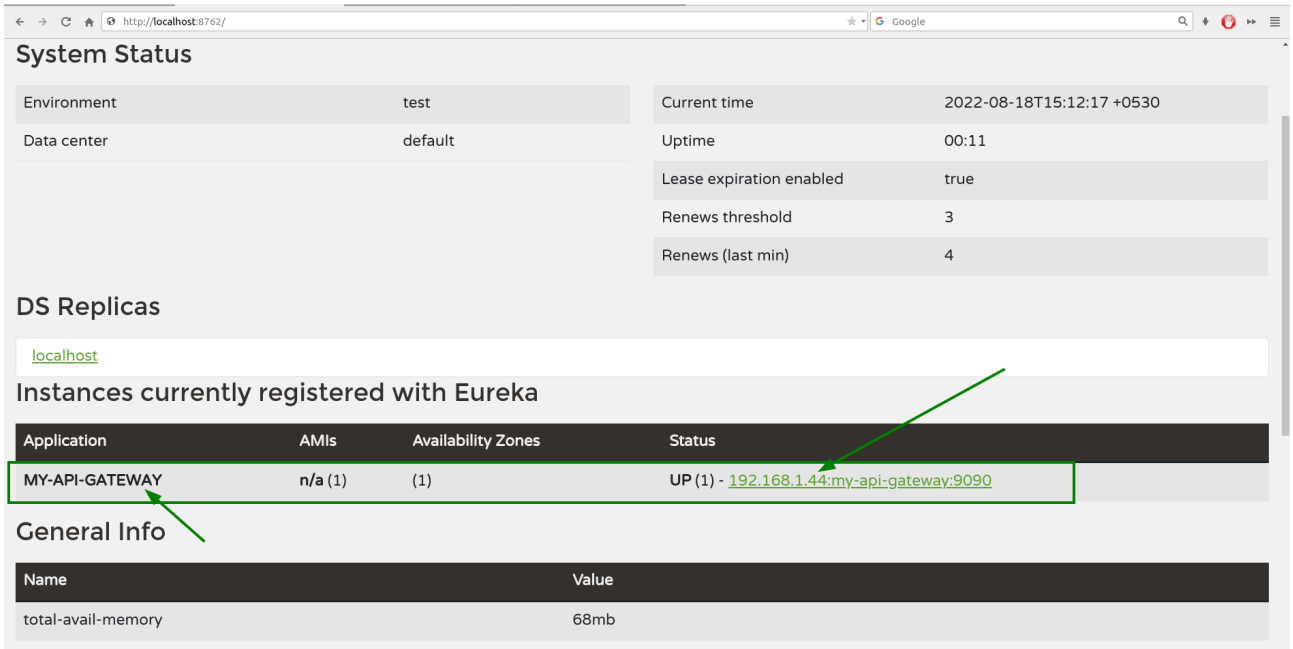
spring-cloud-starter-gateway
spring-cloud-starter-netflix-eureka-client
spring-boot-devtools

application.properties

server.port=9090
spring.application.name=my-api-gateway
eureka.client.serviceUrl.defaultZone= http://localhost:8762/eureka

 # After our-shows application is ready
# http://localhost:9090/shows/api/v1
spring.cloud.gateway.routes[0].uri=lb://OUR-SHOWS/
spring.cloud.gateway.routes[0].predicates[0]=Path=/shows/api/v1/**

 # After bollywood-movies application is ready
 # http://localhost:9090/api/v1/movies
spring.cloud.gateway.routes[1].uri=lb://BOLLYWOOD-MOVIES/
spring.cloud.gateway.routes[1].predicates[0]=Path=/api/v1/movies/**



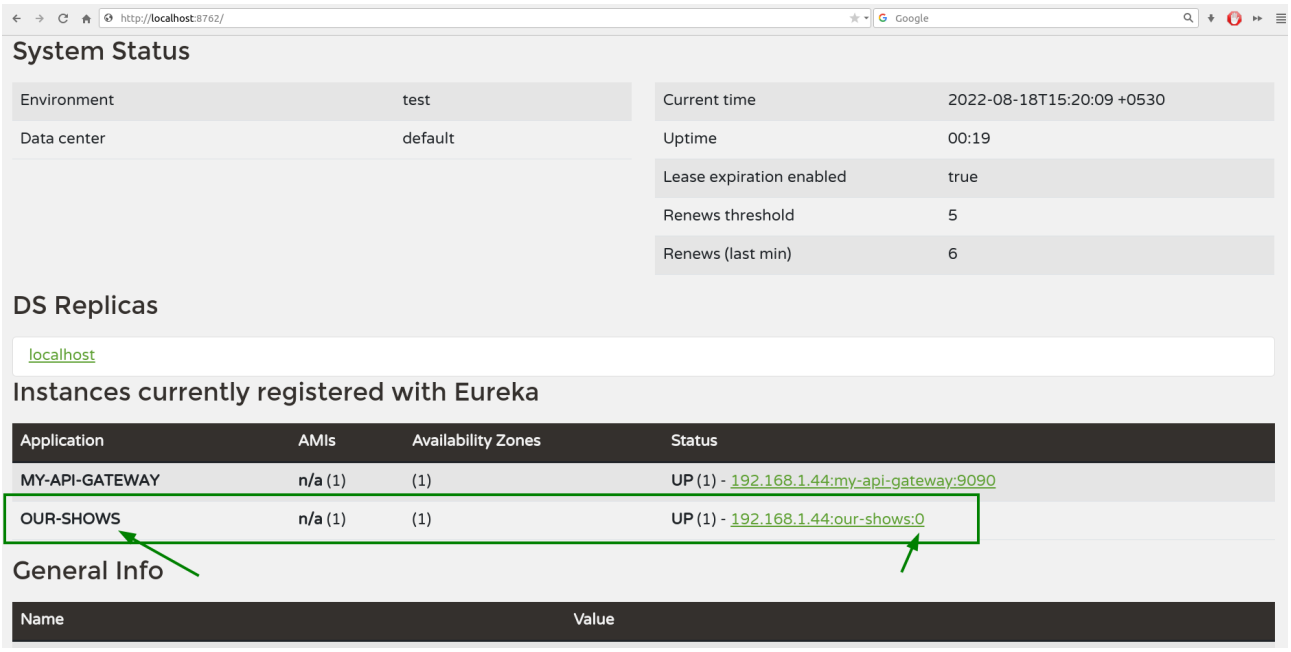
book-our-show

dependencies

spring-boot-starter-web
spring-cloud-starter-netflix-eureka-client
spring-boot-devtools

application.properties

server.port=0
spring.application.name=our-shows
eureka.client.serviceUrl.defaultZone= http://localhost:8762/eureka



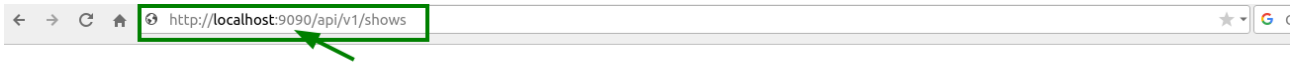

```
package com.demo.controllers;
import ...

@RestController
@RequestMapping("/shows/api/v1")
public class ShowsController {

	@GetMapping
	public List<Show> getDefaultShows(){
		List<Show> shows = new ArrayList<Show>();
		shows.add(new Show("Mahabharat", "Ram", 4.9));
		shows.add(new Show("Ramayan", "Valmiki", 4.7));

		return shows;
	}
}
```

```
package com.demo.models;

public class Show {
```

```
	private int id;
	private String title;
	private String director;
	private double rating;

	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getTitle() {
		return title;
	}
	public void setTitle(String title) {
		this.title = title;
	}
	public String getDirector() {
		return director;
	}
	public void setDirector(String director) {
		this.director = director;
	}
	public double getRating() {
		return rating;
	}
	public void setRating(double rating) {
		this.rating = rating;
	}
	@Override
	public String toString() {
		return "Show [id=" + id + ", title=" + title + ", director=" + director + ", rating=" + rating +
"]";
	}

	public Show(String title, String director, double rating) {
		super();
		this.title = title;
		this.director = director;
		this.rating = rating;
	}

	public Show(int id, String title, String director, double rating) {
		this(title, director, rating);
		this.id = id;
	}

	public Show() {
		super();
		// TODO Auto-generated constructor stub
	}
}
```

[{"id":0,"title":"Mahabharat","director":"Ram","rating":4.9},
{"id":0,"title":"Ramayan","director":"Valmiki","rating":4.7}]

Our-shows application is ready now
To consume via api gateway, update api gateway properties file.
And



[{"id":0,"title":"Mahabharat","director":"Ram","rating":4.9},
{"id":0,"title":"Ramayan","director":"Valmiki","rating":4.7}]

bollywood-service

dependencies

spring-boot-starter-data-jpa
spring-boot-starter-web
spring-cloud-starter-netflix-eureka-client
spring-boot-devtools
h2

application.properties

```
spring.datasource.url=jdbc:h2:mem:mydemodb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.H2Dialect

spring.h2.console.enabled=true
spring.h2.console.path=/h2-console

spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update

server.port=0
spring.application.name=bollywood-movies

eureka.client.serviceUrl.defaultZone= http://localhost:8762/eureka
```

write-crud-app

```
package com.demo.entities;

import java.time.LocalDate;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Movie {

	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY) // optional
	private int id;

	private String title;
	private String director;
	private double rating;
	private LocalDate releasedDate;

	public void setReleasedDate(LocalDate releasedDate) {
		this.releasedDate = releasedDate;
	}
	public LocalDate getReleasedDate() {
		return releasedDate;
```

```
	}
	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getTitle() {
		return title;
	}
	public void setTitle(String title) {
		this.title = title;
	}
	public String getDirector() {
		return director;
	}
	public void setDirector(String director) {
		this.director = director;
	}
	public double getRating() {
		return rating;
	}
	public void setRating(double rating) {
		this.rating = rating;
	}


	@Override
	public String toString() {
		return "Movie [id=" + id + ", title=" + title + ", director=" + director + ", rating=" + rating
				+ ", releasedDate=" + releasedDate + "]";
	}
	public Movie() {
		super();
		this.releasedDate = LocalDate.now();
	}
	public Movie(int id, String title, String director, double rating) {
		this(title, director, rating);
		this.id = id;
	}
	public Movie(String title, String director, double rating) {
		this();
		this.title = title;
		this.director = director;
		this.rating = rating;
	}
}


package com.demo.repositories;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;

import com.demo.entities.Movie;
```

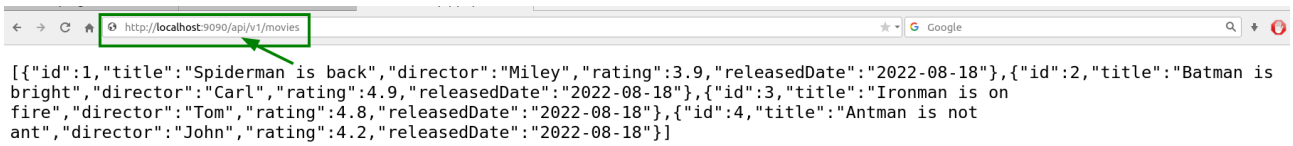
```
public interface MovieRepository extends JpaRepository<Movie, Integer> {
	@Query(nativeQuery = false, value = "select m from Movie m where m.title  like %?1%")
	List<Movie> findByTitle(String title);

}


package com.demo.service;

import java.util.List;
import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Configuration;
import org.springframework.stereotype.Component;
import org.springframework.stereotype.Controller;
import org.springframework.stereotype.Repository;
import org.springframework.stereotype.Service;

import com.demo.entities.Movie;
import com.demo.repositories.MovieRepository;

@Service
public class MovieService {

	@Autowired
	private MovieRepository repo;

	public Movie createMovie(Movie m) {
		// validation - movie rating should not be negative
		// validation - movie title and author should not be empty

		return repo.save(m);
	}

	public List<Movie> getAllMovies() {
		// logic

		return repo.findAll();
	}

	public Movie getMovieById(int id) throws Exception {
		// logic

		Optional<Movie> optional = repo.findById(id);
		if(optional.isEmpty()) {
			System.out.println("Movie with id ("+id+") not found in db");
			throw new Exception("Movie with id ("+id+") not found in db");
		} else {
			return optional.get();
		}
	}

	// find movies by title
	public List<Movie> getMoviesByTitle(String title) {
```

```
		System.out.println("Finding by title : "+title);
		if(title!=null && title.length()!=0) {
			return repo.findByTitle(title);
		} else {
			return getAllMovies();
		}
	}


	// update movie
	// delete movie

}

package com.demo.controllers;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import com.demo.entities.Movie;
import com.demo.service.MovieService;

@RestController
@CrossOrigin(origins = {"https://hoppscotch.io", "http://localhost:4200"})
@RequestMapping("/api/v1/movies")
public class MovieController {

	@Autowired
	private MovieService service;

	@GetMapping("")
	public List<Movie> getMovies() {
		return service.getAllMovies();
	}

	@GetMapping("/{id}")
	public Movie getMovieById(@PathVariable int id) throws Exception {
		return service.getMovieById(id);
	}

	@PostMapping("")
	public Movie saveMovie(@RequestBody Movie m) {
		return service.createMovie(m);
	}

	@GetMapping("/title") // ?title="batman is flying"
	public List<Movie> getMoviesByTitle(@RequestParam String title){
```

```
		return service.getMoviesByTitle(title);
	}
}
```

```
package com.demo;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

import com.demo.entities.Movie;
import com.demo.service.MovieService;

@SpringBootApplication
public class Application implements CommandLineRunner {

	@Autowired
	private MovieService service;

	@Override
	public void run(String... args) throws Exception {
		System.out.println("Service: "+service);

		service.createMovie(new Movie("Spiderman is back", "Miley", 3.9));
		service.createMovie(new Movie("Batman is bright", "Carl", 4.9));
		service.createMovie(new Movie("Ironman is on fire", "Tom", 4.8));
		service.createMovie(new Movie("Antman is not ant", "John", 4.2));
	}

	public static void main(String[] args) {
		SpringApplication.run(Application.class, args);
	}

}
```

Bollywood application is ready now
To consume via api gateway, update api gateway properties file.
And



Communications b/w microservices (rest client)

dependency

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-openfeign</artifactId>
</dependency>
```

class containing main method (anywhere in your classpath)

```
@EnableFeignClients
```

```
package com.demo.clients;

import java.util.List;

import org.springframework.cloud.openfeign.FeignClient;
import org.springframework.web.bind.annotation.GetMapping;

import com.demo.models.Show;

//@FeignClient(value = "comic", url = "http://localhost:9090/api/v1/movies")
@FeignClient("bollywood-movies")
public interface BollywoodMoviesClient {

    @GetMapping("/api/v1/movies")
    List<Show> getMovies();

    //    @GetMapping("/api/v1/movies/{movieId}")
    //    Show getMovieById(@PathVariable int movieId);

}
```

Consume FeignClient

```
@Autowired
private BollywoodMoviesClient client;

    ...
    client.getMovies();
```