

Spring Cloud Micro Services

Eldar

Table of Contents

references.....	2
בנושא spring cloud.....	2
https://cloud.spring.io/spring-cloud-commons/reference/html/#spring-cloud-context-application-context-services.....	2
What is Micro Service.....	3
از מה הולכים לעשות.....	4
Project 1.....	5
Configuration Server.....	5
צפיה בנתוני הקונפיגורציה דרך הדף.....	7
Intro to Services A, B.....	9
Service A.....	9
Project 2.....	12
Service A.....	12
Project 3.....	15
Service B.....	15
Discovery Server.....	17
הרכבת Consul.....	21
המשך עבודה.....	24
קביעת node name של consul.....	29
Load Balancing.....	34
לסיפור LB נכenis.....	36
LB Aspect.....	39
Circuit Breaker.....	43
versions for hystrix.....	48
API Gateway.....	51
בנייה הפרויקט.....	64
Gateway – המשך.....	75
Dashboard.....	76
יצירת הפרויקט.....	79
Feign Clients.....	86

references

אתר מקורי של spring cloud בנושא

<https://cloud.spring.io/spring-cloud-commons/reference/html/#spring-cloud-context-application-context-services>

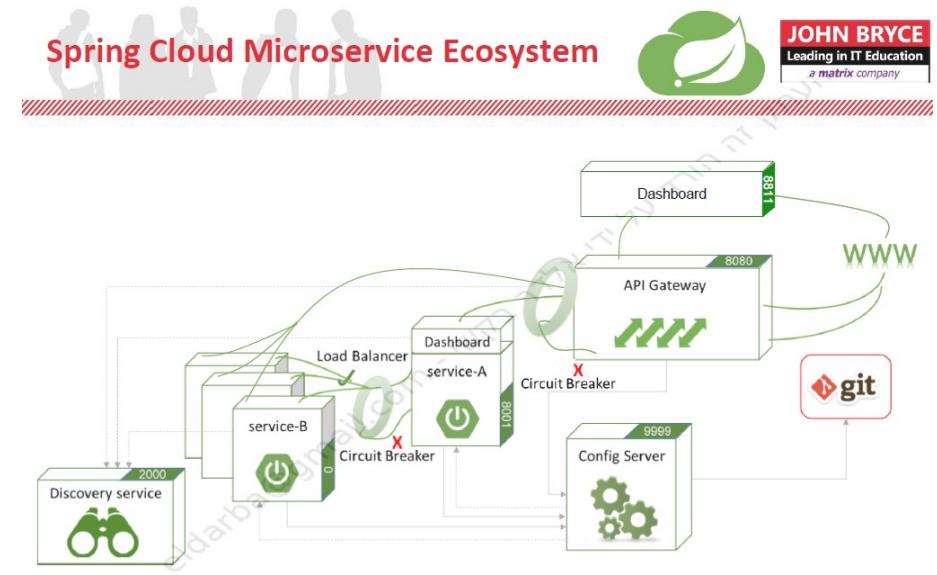
What is Micro Service

In service oriented architecture:

- Distributed components connected via http
- Each component is a micro service

از מה הולכים לעשות

מתחללים לעבוד ובסוף נקבל את מה שיש למיטה בתמונה.



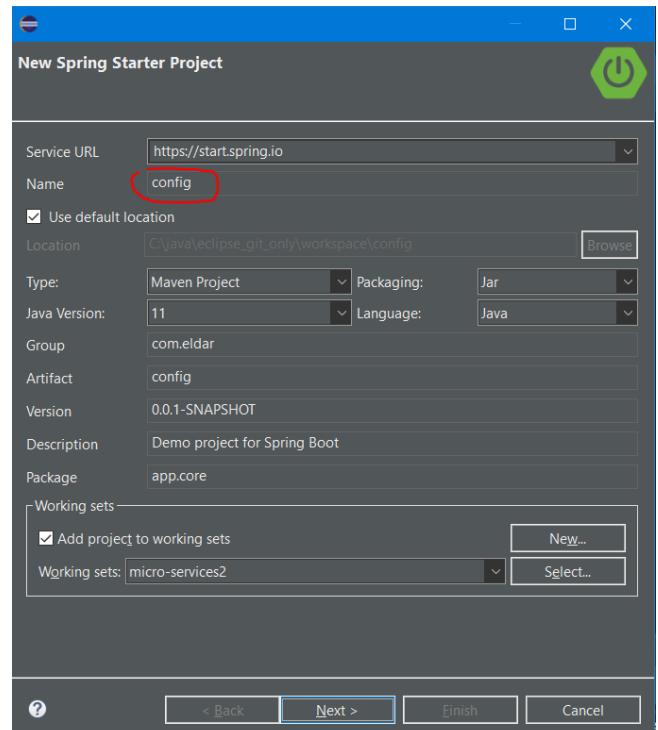
Project 1

Configuration Server

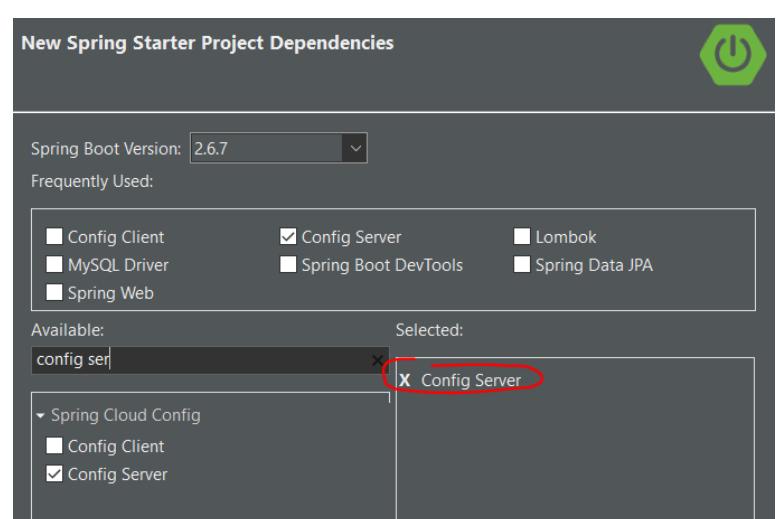
הפרויקט הראשון שלנו יהיה config server שידע לקבל בקשות ממיקרו סרבריס. הבקשה תגיע עם השם של המיקרו סרבריס, תרוץ לגיט ותביא משם את קובץ הקונפיגורציה.

יצירת הפרויקט:

שם הפרויקט: config

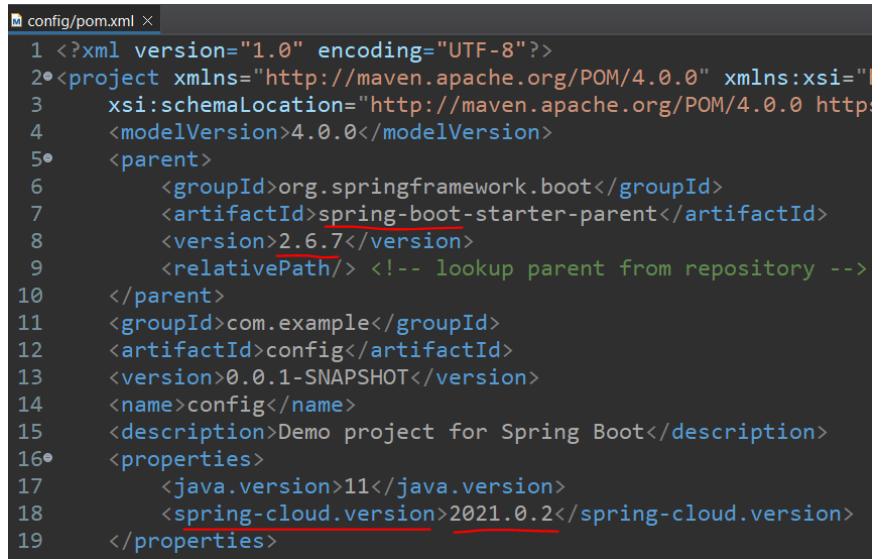


כל מה שצורך זה dependency אחד:



קובץ properties

שעבד לי טוב עם גרסאות



```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
3   <modelVersion>4.0.0</modelVersion>
4   <parent>
5     <groupId>org.springframework.boot</groupId>
6     <artifactId>spring-boot-starter-parent</artifactId>
7     <version>2.6.7</version>
8     <relativePath/> <!-- lookup parent from repository -->
9   </parent>
10  <groupId>com.example</groupId>
11  <artifactId>config</artifactId>
12  <version>0.0.1-SNAPSHOT</version>
13  <name>config</name>
14  <description>Demo project for Spring Boot</description>
15  <properties>
16    <java.version>11</java.version>
17    <spring-cloud.version>2021.0.2</spring-cloud.version>
18  </properties>
19</project>

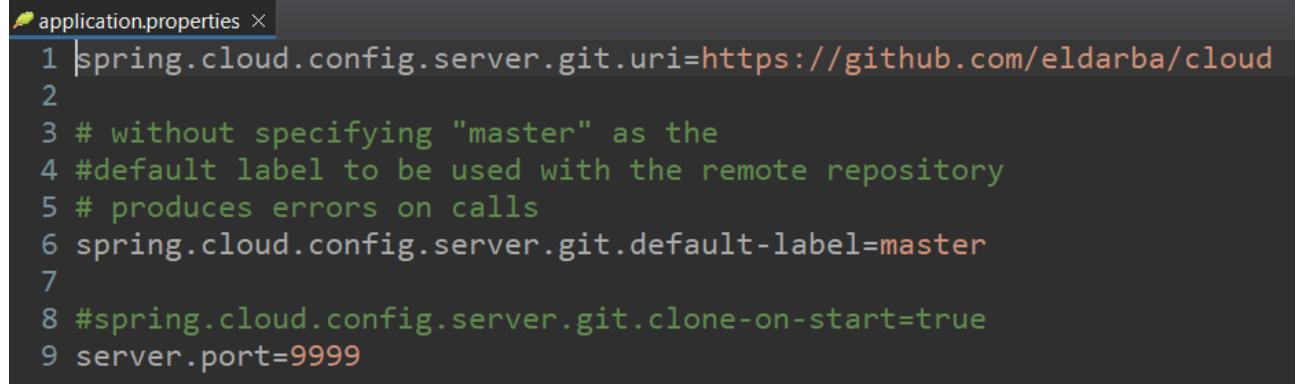
```

spring.cloud.config.server.git.uri=https://github.com/eldarba/cloud

spring.cloud.config.server.git.default-label=master

#spring.cloud.config.server.git.clone-on-start=true

server.port=9999



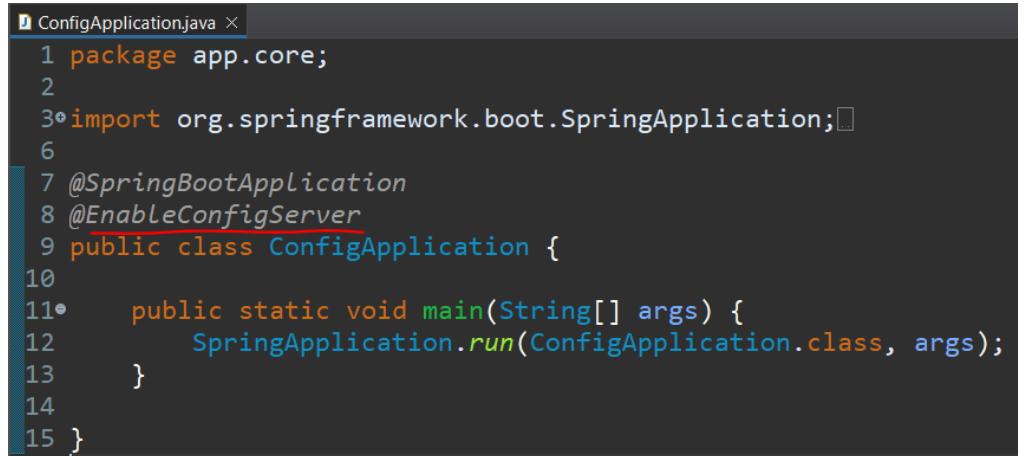
```

1 spring.cloud.config.server.git.uri=https://github.com/eldarba/cloud
2
3 # without specifying "master" as the
4 #default label to be used with the remote repository
5 # produces errors on calls
6 spring.cloud.config.server.git.default-label=master
7
8 #spring.cloud.config.server.git.clone-on-start=true
9 server.port=9999

```

אפשר לחת URI שהוא במערכת הקבצים של המחשב המקומי יא: וכאן הלאה...

עדכן את קובץ האפליקציה:



```

1 package app.core;
2
3 import org.springframework.boot.SpringApplication;
4
5 @SpringBootApplication
6 @EnableConfigServer
7 public class ConfigApplication {
8
9   public static void main(String[] args) {
10     SpringApplication.run(ConfigApplication.class, args);
11   }
12 }

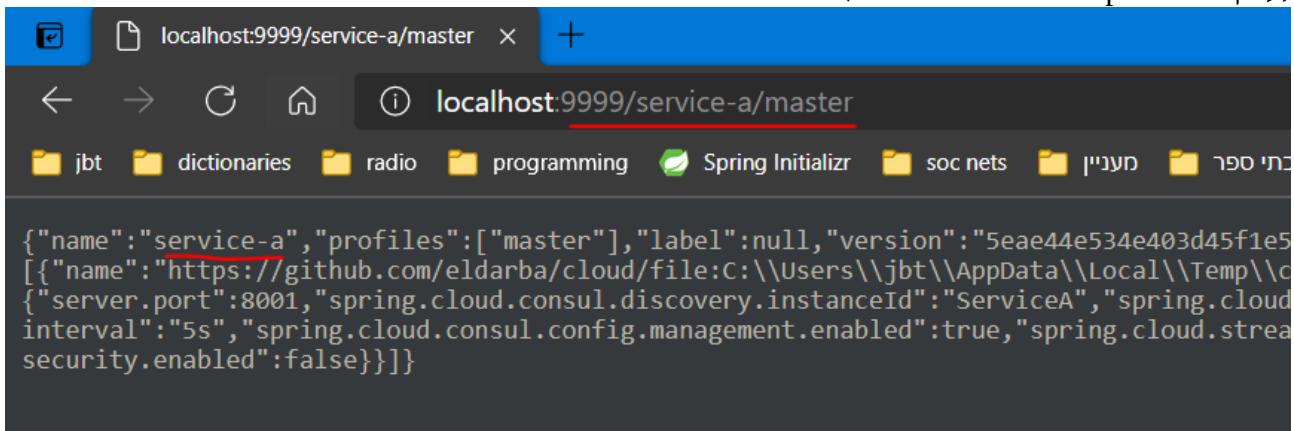
```

ונרץ...

צפיה בנתוני הקונפיגורציה דרך הדפסה

넓נה בדף:

(עדיף postman כי רואים יותר ברור)

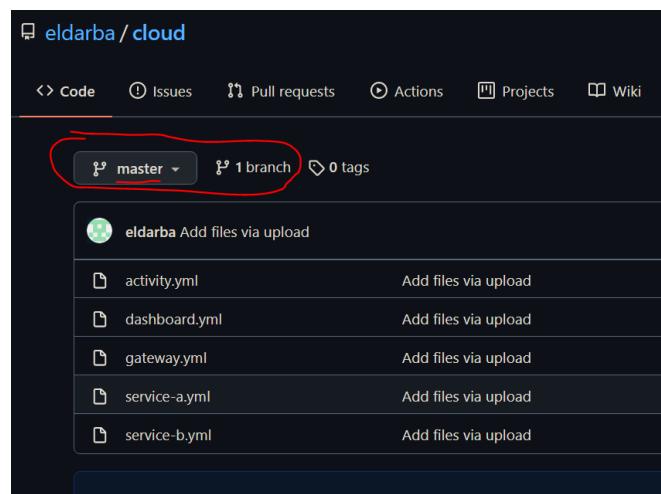


A screenshot of a browser window titled "localhost:9999/service-a/master". The address bar shows the URL. Below the address bar is a navigation bar with icons for back, forward, search, and refresh. The main content area displays a JSON object:

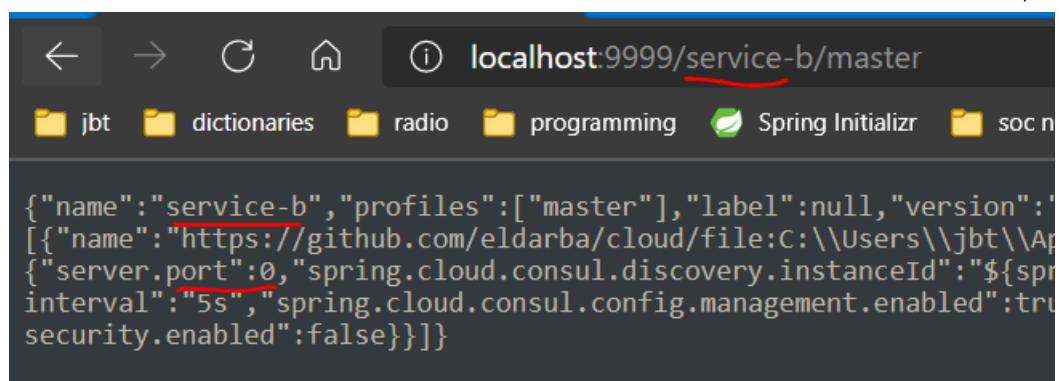
```
{"name": "service-a", "profiles": ["master"], "label": null, "version": "5eae44e534e403d45f1e5", [{"name": "https://github.com/eldarba/cloud/file:C:\\\\Users\\\\jbt\\\\AppData\\\\Local\\\\Temp\\\\cloud\\\\service-a\\\\service-a.yml"}, {"server.port": 8001, "spring.cloud.consul.discovery.instanceId": "ServiceA", "spring.cloud.consul.discovery.interval": "5s", "spring.cloud.consul.config.management.enabled": true, "spring.cloud.stream.security.enabled": false}]}]
```

נראה שאפשר להזמין את הקונפיגורציה לפי שם של service. מקבלים את זה ב JSON

חשוב שהשמות יהיו ב lower case. הקבצים ב git חייבים להיות ב lower case. אם נקרה לשירות ב spring בשם service-A הינה GIT תמיד תהיה באותיות קטנות (lower case).git אומר שפינה ל master configuration Master



נלך לראות גם את B



A screenshot of a browser window titled "localhost:9999/service-b/master". The address bar shows the URL. Below the address bar is a navigation bar with icons for back, forward, search, and refresh. The main content area displays a JSON object:

```
{"name": "service-b", "profiles": ["master"], "label": null, "version": "5eae44e534e403d45f1e5", [{"name": "https://github.com/eldarba/cloud/file:C:\\\\Users\\\\jbt\\\\AppData\\\\Local\\\\Temp\\\\cloud\\\\service-b\\\\service-b.yml"}, {"server.port": 0, "spring.cloud.consul.discovery.instanceId": "${spring.cloud.consul.discovery.instanceId}", "spring.cloud.consul.discovery.interval": "5s", "spring.cloud.consul.config.management.enabled": true, "spring.cloud.stream.security.enabled": false}]}]
```

ה 0 ב port=0 spring MVC אומר ל starter של tomcat להציג port. וכך B יציג פורטים מסוימים שאנו רוצים להציג כמה מופעים ממנו (שלא ייפול על אותו פורט)

כך זה נראה ב postman
החלק הצבוע הוא מה כתוב בתוך הקובץ yml שב git

```
[{"name": "service-a",  
 "profiles": [  
     "master"  
 ],  
 "label": null,  
 "version": "c46b419ea67199bb4bbebf236bcdeab6c864d350",  
 "state": null,  
 "propertySources": [  
     {  
         "name": "https://github.com/eldarba/cloud/file:C:\\Users\\jbt\\AppData\\Local\\Temp\\config-repo-9258076159410883445\\service-a.yml",  
         "source": {  
             "server.port": 8001,  
             "spring.cloud.consul.discovery.instanceId": "ServiceA",  
             "spring.cloud.consul.discovery.health-check-interval": "5s",  
             "spring.cloud.consul.config.management.enabled": true,  
             "spring.cloud.stream.bindings.output.binder": "kafka",  
             "spring.cloud.stream.bindings.output.destination": "cloudEX",  
             "management.security.enabled": false  
         }  
     }  
 ]}  
]
```

Intro to Services A, B

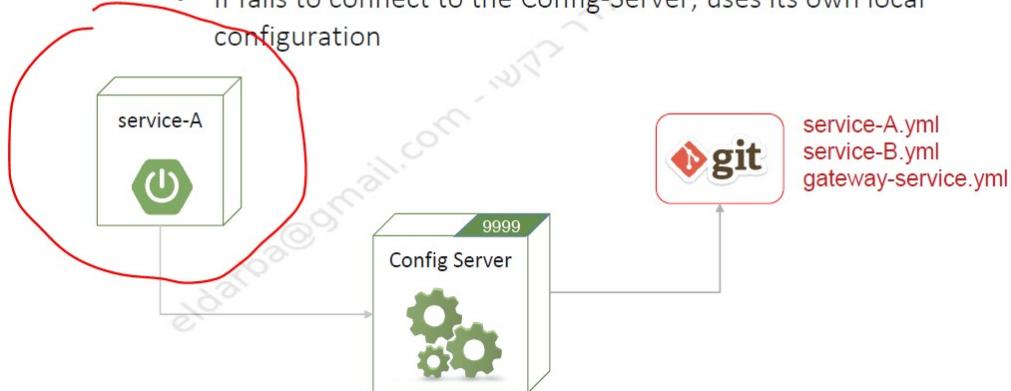
Service A

Configuration Server

JOHN BRYCE
Leading in IT Education
a matrix company

Any service that uses Config-server will:

- Try to get *.yml / *.properties loaded from GIT
- If fails to connect to the Config-Server, uses its own local configuration



תלויות

- כדי להגדיר WEB controllers צריך חתיכת קוד ש יודעת ל取ת config server מהgid לו קוראים לי כך וכך,
- לקבל הקונפיגורציה ולדרווס.

Configuration Server

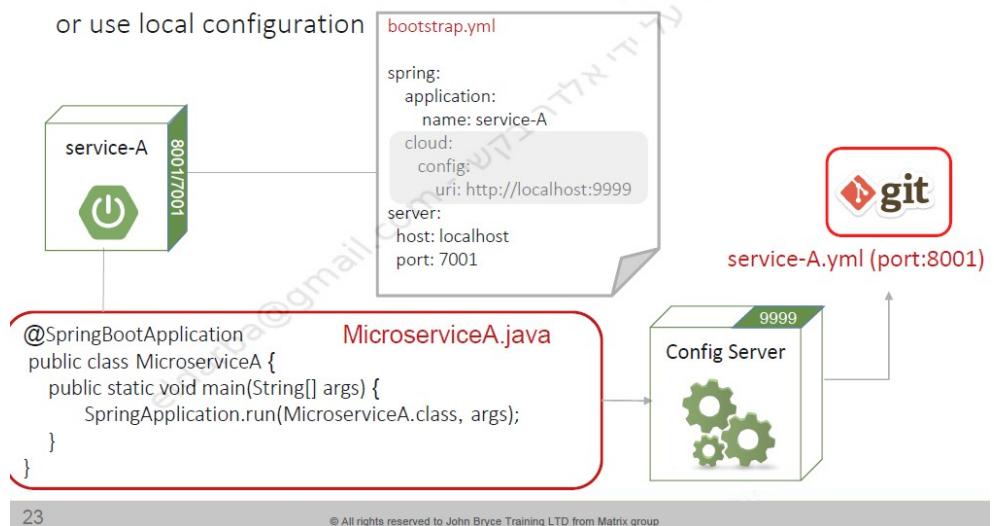
JOHN BRYCE
Leading in IT Education
a matrix company

- Client Microservice
- Maven dependencies:

```
pom.xml
...
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-cloud-starter-web</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-config</artifactId>
</dependency>
...
```

Configuration Server

Microservices are set to connect to config server
or use local configuration



פורט 7001 יכול אם ה GIT לא זמין.

לABI B

מ GIT מקבלים port=0 או B מגריל פורטימ.

Configuration Server

Running multiple Microservice instances

Assign zero value to port

Configuration server randomizes values



24

© All rights reserved to John Bryce Training LTD from Matrix group

לABI כל שירות שושאב קונפיגורציה, אם יש תקלת בשרת קונפיגורציה ולא נרצה שיעלת עם ההגדרות המיקומיות נגדיר:

Configuration Server



What happens when configuration server is not running?

- Service uses local configuration
- bootstrap.yml / bootstrap.properties

Setting service property:

spring.cloud.config.fail-fast=true

Causes service to fail with IllegalStateException in such cases

25

© All rights reserved to John Bryce Training LTD from Matrix group

ואז תיצור שגיאה והשירות לא יעלה.

Project 2

Service A

New Spring Starter Project

Service URL: https://start.spring.io

Name: service-a (circled)

Use default location

Location: C:\java\eclipse_git_only\workspace-microservices-temp\service-a

Type: Maven Project | Packaging: Jar

Java Version: 11 | Language: Java

Group: com.example

Artifact: service-a

Version: 0.0.1-SNAPSHOT

Description: Demo project for Spring Boot

Package: app.core

Working sets:

Add project to working sets

Dependencies

New Spring Starter Project Dependencies

Spring Boot Version: 2.6.7

Frequently Used:

Config Server

Available: Selected:

config cli

Spring Cloud Config

Config Client

VMware Tanzu Application Service

Config Client (TAS)

X Config Client
X Spring Web

Make Default Clear Selection

? < Back Next > Finish Cancel

The screenshot shows the 'New Spring Starter Project Dependencies' dialog. In the 'Available:' list, 'Config Client' is checked with a red circle around it. In the 'Selected:' list, 'Config Client' and 'Spring Web' are selected with red circles around them. The 'Selected:' list has a red border.

application.properties file

```
spring.application.name=service-a
spring.config.import=configserver:http://localhost:9999
server.port=7001
```

application.properties

```
1 spring.application.name=service-a
2 spring.config.import=configserver:http://localhost:9999
3 server.port=7001
4 |
```

Configuration Class

האנוטציה הזאת כבר לא חובה. מספיק שיש מימוש על ה classpath

```
ServiceAApplication.java ×
1 package app.core;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.cloud.client.discovery.EnableDiscoveryClient;
6
7 @SpringBootApplication
8 @EnableDiscoveryClient
9 public class ServiceAApplication {
10
11     public static void main(String[] args) {
12         SpringApplication.run(ServiceAApplication.class, args);
13     }
14
15 }
16
```



[@EnableDiscoveryClient] is no longer required. You can put a [DiscoveryClient] implementation on the classpath to cause the Spring Boot application to register with the service discovery server.

https://cloud.spring.io/spring-cloud-commons/multi/multi__spring_cloud_commons_abstractions.html

Controller

```
ControllerA.java ×
1 package app.core.controllers;
2
3 import org.springframework.web.bind.annotation.GetMapping;
4 import org.springframework.web.bind.annotation.RestController;
5
6 @RestController
7 public class ControllerA {
8
9     @GetMapping("/service/a")
10    public String handleA() {
11        return "Service A";
12    }
13
14 }
```

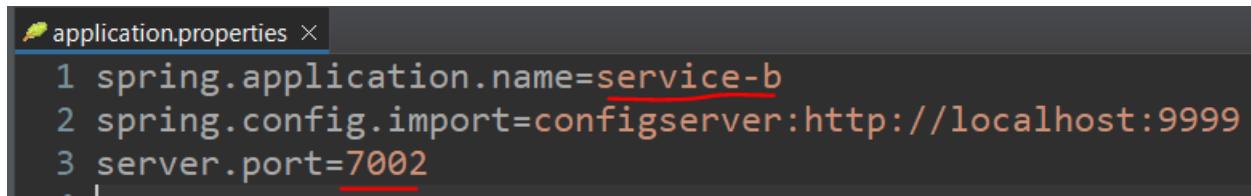
אפשר להריץ ולנשת לשירות לראות.
.config server מ Amor לקל פורט

Project 3

Service B

לעשות העתק הדבק ל A וazz רק:

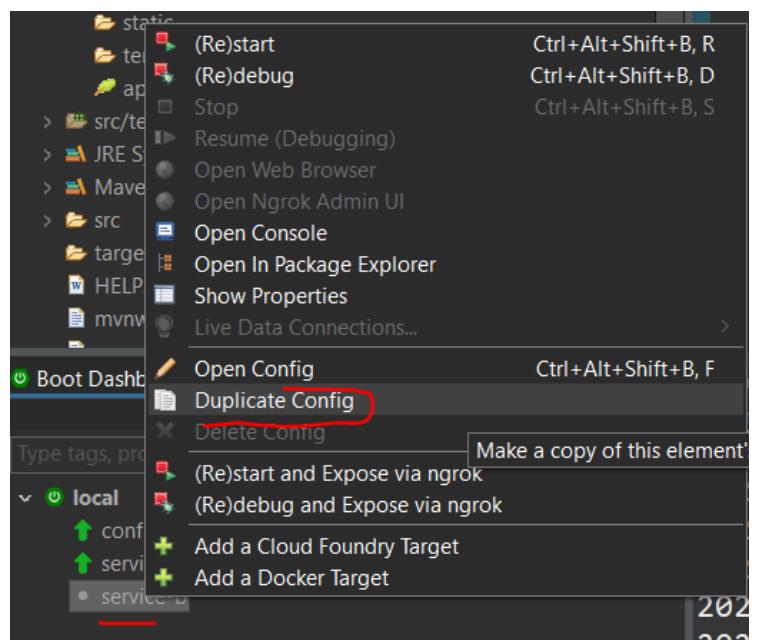
- לשנות את שם הפרויקט ל B (רצוי גם ב POM)
- לשנות את שמות המחלקות וה packages ל b (איפה שצריך)
- לעדכן את ה controller ל controller ב application.properties
- לעורוך את קובץ application.properties



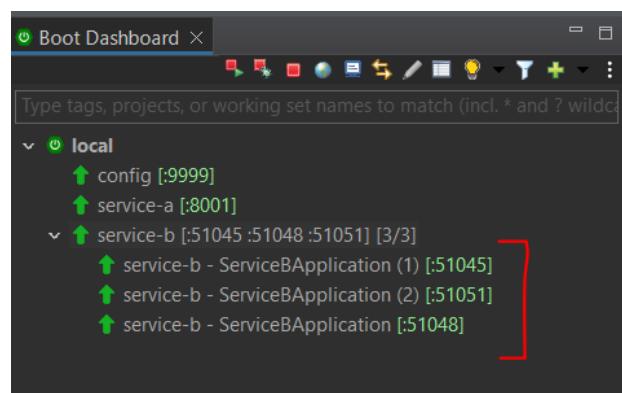
```
application.properties x
1 spring.application.name=service-b
2 spring.config.import=configserver:http://localhost:9999
3 server.port=7002
```

כעת ניתן להרים כמה מופעי B ולראות שככל אחד מהם מקבל port אחר באקראי.

עושים זאת ע"י לחצן ימני על הקונפיגורציה הקיימת וazz לשכפל אותה:



למשל נרים שלושה מופעים:



בקונפיגורציה הענן של B יש את המאפיין instanceId ומוארך לו שם האפליקציה וערך אקראי:

```
{  
    "name": "service-b",  
    "profiles": [  
        "master"  
    ],  
    "label": null,  
    "version": "c46b419ea67199bb4bbebf236bcdeab6c864d350",  
    "state": null,  
    "propertySources": [  
        {  
            "name": "https://github.com/eldarba/cloud/file:C:\\\\Users\\\\jbt\\\\AppData\\\\Local\\\\Temp\\\\config-repo",  
            "source": {  
                "server.port": 0,  
                "spring.cloud.consul.discovery.instanceId": "${spring.application.name}:${random.value}",  
                "spring.cloud.consul.discovery.health-check-interval": "5s",  
                "spring.cloud.consul.config.management.enabled": true,  
                "spring.cloud.stream.bindings.output.binder": "kafka",  
                "spring.cloud.stream.bindings.output.destination": "cloudEX",  
                "management.security.enabled": false  
            }  
        }  
    ]  
}
```

ונכל להזיריק את הערך שהתקבל ולשרשר בresponse של controller

```
ControllerB.java ×  
1 package app.core.controllers;  
2  
3 import org.springframework.beans.factory.annotation.Value;  
4 import org.springframework.web.bind.annotation.GetMapping;  
5 import org.springframework.web.bind.annotation.RestController;  
6  
7 @RestController  
8 public class ControllerB {  
9  
10     @Value("${spring.cloud.consul.discovery.instanceId}")  
11     private String instanceId;  
12  
13     @GetMapping("/service/b")  
14     public String handleB() {  
15         return "From: " + instanceId;  
16     }  
17  
18 }
```

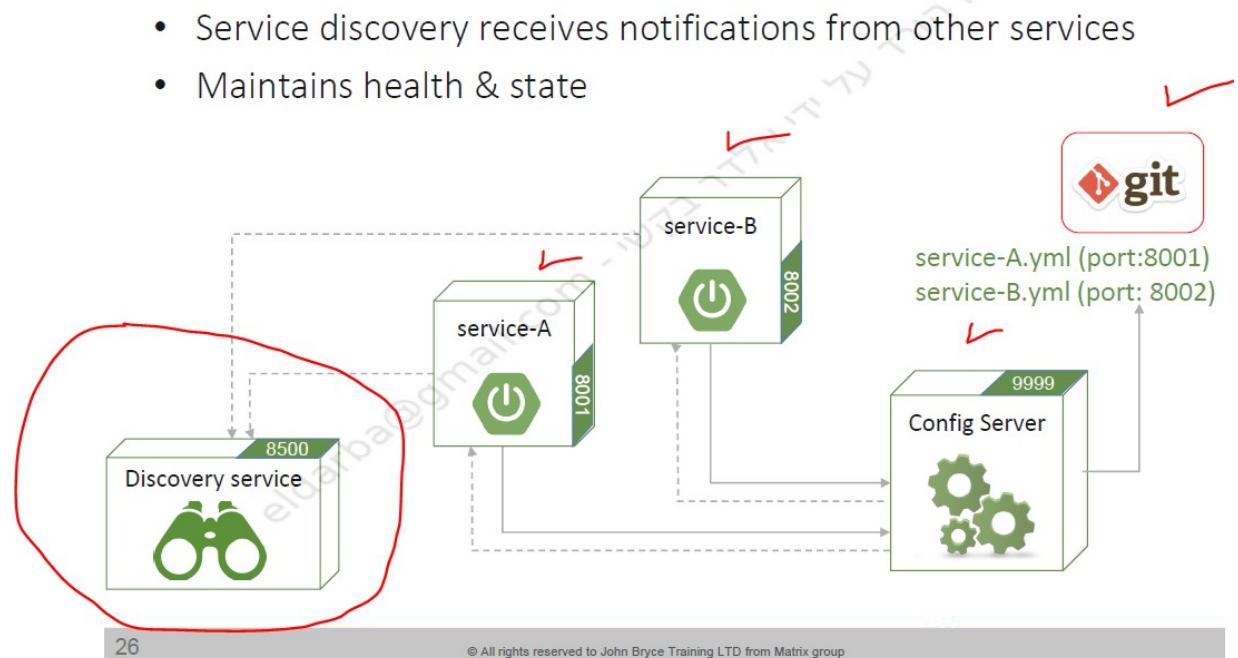
마חר ויהיו כמה מופעי B כדי שייהיו להם שמות שונים וכך משיגים זאת.
אפשר לפנות לשירותים ולקבל מענה מהפורטים השונים.

Discovery Server

עבור service discovery נשתמש ב .consul
זה מוצר של HashiCorp, Inc
יצא לשוק ב 2014
open source

Consul can do

- DNS-based service discovery
- distributed Key-value storage
- segmentation
- configuration

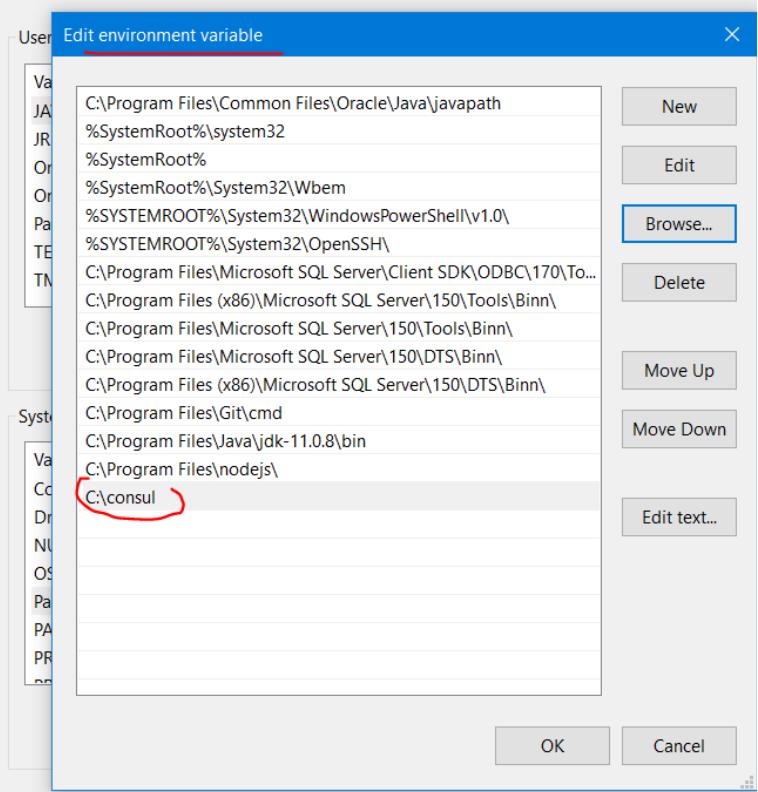


- Discovery server
 - Consul supports
 - Discovery services
 - Monitoring – services ,nodes and clusters
 - Configuration management (not recommended for production)
- Installing & Running Consul:
 - Download consul.exe from
<https://www.consul.io/downloads.html>
 - Add consul.exe to system path
 - Use the command: *consul agent -dev*



צרייך להוריד מהאתר. יורד קובץ ZIP ומחלצים.
<https://www.consul.io/downloads>
יש שם קובץ בינהי אחד שאפשר להגדיר אותו ב path

Environment Variables



לבדוק שצמינו

```
Command Prompt
Microsoft Windows [Version 10.0.19041.928]
(c) Microsoft Corporation. All rights reserved.

C:\Users\jbt>consul
Usage: consul [--version] [--help] <command> [<args>]

Available commands are:
  acl           Interact with Consul's ACLs
  agent         Runs a Consul agent
  catalog       Interact with the catalog
  config        Interact with Consul's Centralized Configuration
  connect       Interact with Consul Connect
  debug         Records a debugging archive for operators
  event         Fire a new event
  exec          Executes a command on Consul nodes
  ...
```

Run the Consul Agent

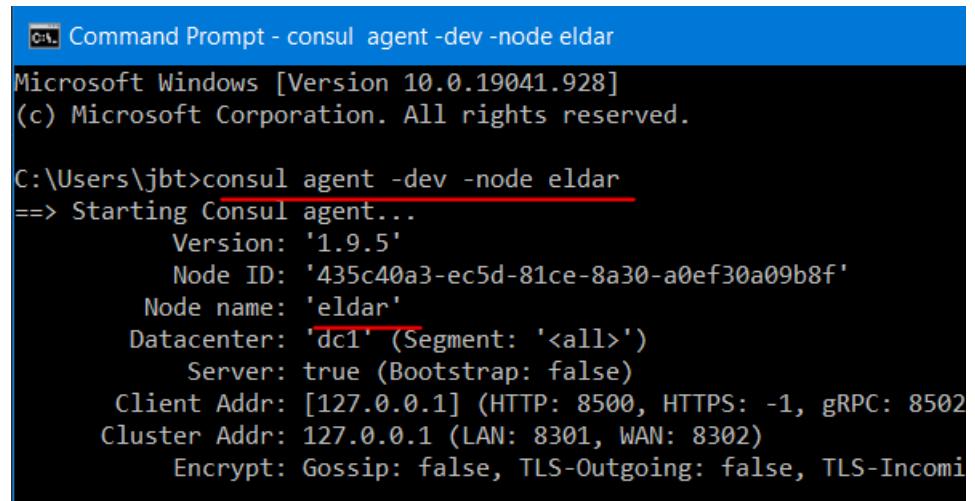


- Running consul:

```
Command Prompt - consul agent -dev
C:\Development\Consul>consul agent -dev
=> Starting Consul agent...
    Version: 'v1.7.1'
    Node ID: '0fe4358a-140d-051a-efde-935ff3d0cb9e'
    Node name: 'Ronik-T490'
    Datacenter: 'dc1' (Segment: '<all>')
    Server: true (Bootstrap: false)
    Client Addr: [127.0.0.1] (HTTP: 8500, HTTPS: -1, gRPC: 8502, DNS: 8600)
    Cluster Addr: 127.0.0.1 (LAN: 8301, WAN: 8302)
    Encrypt: Gossip: false, TLS-Outgoing: false, TLS-Incoming: false, Auto-Encrypt-TLS: false
```

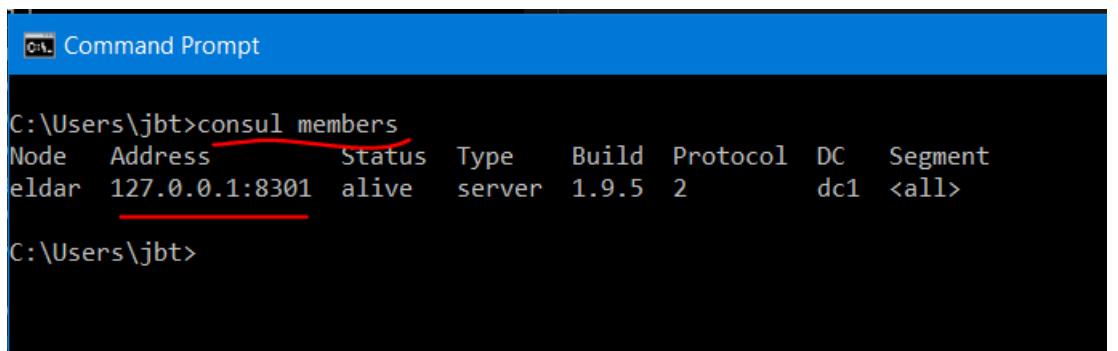
- Default address is: *http://localhost:8500*

הרכבת Consul



```
C:\Users\jbt>consul agent -dev -node eldar
==> Starting Consul agent...
      Version: '1.9.5'
      Node ID: '435c40a3-ec5d-81ce-8a30-a0ef30a09b8f'
      Node name: 'eldar'
      Datacenter: 'dc1' (Segment: '<all>')
      Server: true (Bootstrap: false)
      Client Addr: [127.0.0.1] (HTTP: 8500, HTTPS: -1, gRPC: 8502)
      Cluster Addr: 127.0.0.1 (LAN: 8301, WAN: 8302)
      Encrypt: Gossip: false, TLS-Outgoing: false, TLS-Incommi
```

ב CMD אחר אפשר להריץ



Node	Address	Status	Type	Build	Protocol	DC	Segment
eldar	127.0.0.1:8301	alive	server	1.9.5	2	dc1	<all>

כיבוי בצורה נכונה נעשה על ידי צירוף המקלים **.ctrl + C** זה מיידע חברים אחרים ב cluster שה node עזב. לאחרת חברים אחרים חושבים שהוא פשוט נכשל וימשיכו לניסיונות.



- Consul clients are
 - Our Microservices
 - Other Consul instances
- Spring provides a Client Consul starter which:
 - Connects to default/specified Consul location
 - Registers and broadcasts heartbeats
 - Sets refresh intervals (specified in *.properties/yml conf.)
 - heartbeats
 - refresh
 - timeouts



אחד מהם זה **consul**. מאחר ו-discovery זה נראה חשוב, מוצרי **consul** מנהיכים שאנו מרים כמה מופעים. לפחות שניים לפחות. והם יודעים לנטר אחד את השני. קונסול אחד מסתכל על אחר כדי לדעת מה קורה. אבל לא נעשה זאת זה בקורס אלא נרץ מופע אחד.

: discovery service שלנו יבוא להגיד לו שלום ומאותו רגע **consul** כשרת

- יבדוק אם השירותים שלנו קיימים
- ינהל רשימת השירותים הזמינים
- יעזר ל load balancer וכדומה

נסתכל על החלק הרולוונטי לקונסול בקובץ הקונפיגורציה ב GIT עבור השירות B

```
1 server:
2   port: 0
3
4   spring:
5     cloud:
6       consul:
7         discovery:
8           instanceId: ${spring.application.name}:${random.value}
9           health-check-interval: 5s
10      config:
11        management:
12          enabled: true
13      stream:
14        bindings:
15          output:
16            binder: kafka
17            destination: cloudEX
18    management:
19      security:
20        enabled: false
```

instanceId – נתנו הначיה להירשם עם שם אקראי

unhealthy.health-check לעשות בדיקה כל 5 שניות כדי לא להנגיש לבקשת שירותים שסומנו כ unhealthy. אפשר היה לתת חצי שעה ולא חמיש שניות אבל אז יכול להיות שה ribbon loadbalancer שלנו מכיל רשימה לא עדכנית. ככל שהמרוחכים קצרים בזמן כך יותר אמין מצד שני יש לזה מחיר של תובורה ברשת.

נראה שחרר פה הכתובת של consul. אבל ה starter של spring מ Chapman את קונסול ב 8500 מיINET localhost:8500 זאת בሪית המclid. אפשר לknfag את host ואת port.

המשר עובודה

נעדכן את השירותים A ו B להירשם אצל קונסול כך שהוא יעשה להם discovery ובדיקות בריאות.

אז יש לנו קונפיגורציה ב GIT שאומرت לשירות B לłączת ל consul

יש לנו גם consul באויר

עכשו חסר למייקרו סרביס את הקטע הקוד שעושה את השלום הזה על הגבעה. חסרים לו שני discovery server dependencies כדי שיכל לעשות שימוש בקונפיגורציה ולהתגלוות – רישום אצל

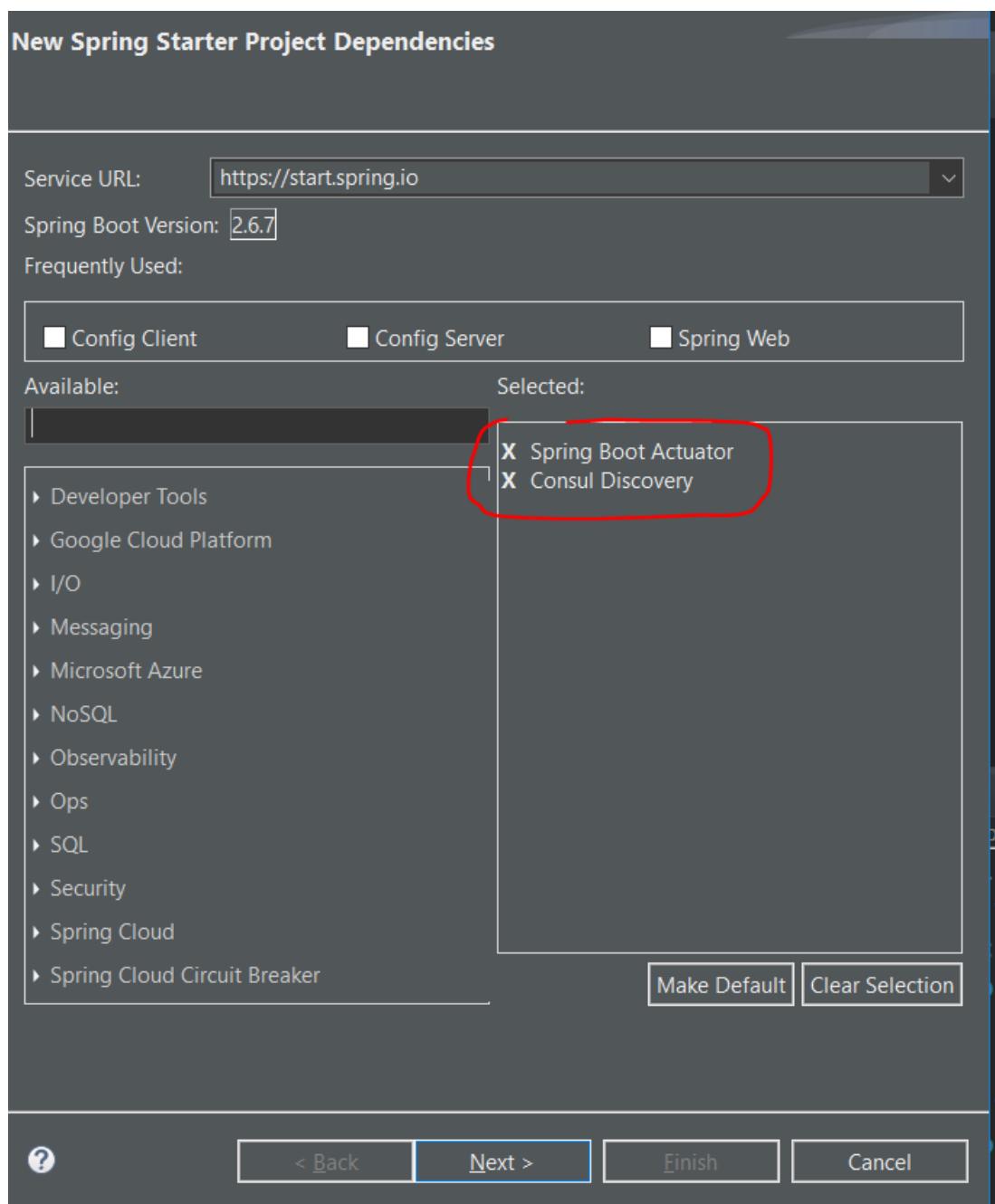


- Make your Microservice discoverable by adding the following Maven dependencies:

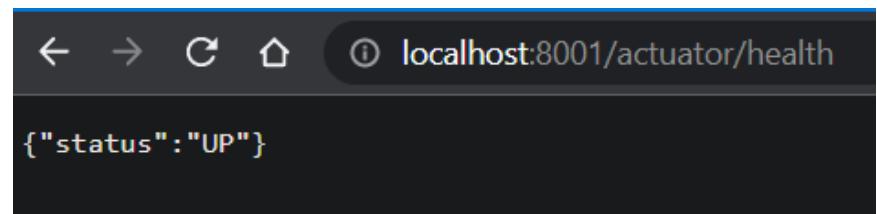
```
pom.xml
...
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-config</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-consul-discovery</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
...
1
2
```

consul discovery - זה שבא ועשה 'register' של consul discovery .1
health - זהו API של spring שיודיע לשדר events ו events משתמש בו לבדיקות דופק, -2
check ולחוות כל מיני אירועים שקשורים ל life-cycle של האפליקציה. ברור ש discovery רוצה את זה כדי לדעת עם השירות חי או מת. אז צריך שניהה מסוגלים לשדר את ה heart beat שלנו .actuator
הוכיחה ואת זה עושים באמצעות actuator

נוסיף למיקרו סרבריסס A ול B שני dependencies המשמש ב service discovery שמספק שירותים של ניטור.



עכשו גם נוכל לפנות שירות ל end point של actuator עבור שירות A



```
{"status": "UP"}
```

אפשר לעשות זאת לכל המופעים של שירות B

קביעת node name של consul

אם לא רוצים את שם ברירת מחדל כפי שמופיע כאן

The screenshot shows the Consul UI with the 'Nodes' tab selected. The main panel displays a single node entry:

54914_	Leader	2 Services	127.0.0.1
--------	--------	------------	-----------

ניתן לקבוע node name בזמן שימושם את :consul

```
C:\Users\jbt>consul agent -dev -node eldar
```

נקבל

The screenshot shows the Consul UI with the 'Nodes' tab selected. The main panel displays a single node entry:

eldar	Leader	1 Service	127.0.0.1
-------	--------	-----------	-----------

אם נלחץ על שם ה node שמוופיע (במקרה זה eldar) נראה פירוט עם מידע על המופעים שלו:

The screenshot shows the Consul UI for the 'eldar' node. The 'Health Checks' section contains the following data:

CheckID	Type	Notes
serfHealth	serf	-

The 'Service Instances' section contains the following data:

ServiceName	CheckID	Type	Notes
service-b	service:service-b-55b1c1ef00f6f3788633a635557355c9	http	-

וכמסתכלים על השירותים רואים בבירור את ה node name ואת ה hostname

The screenshot shows the Netflix OSS UI for a service named "service-a". On the left, there's a sidebar with the following menu items:

- Overview
- Services** (highlighted with a red circle)
- Nodes
- Key/Value
- Intentions
- ACCESS CONTROLS (with a red dot)
- Tokens

The main content area has a title "service-a" and a navigation bar with tabs: Instances (highlighted with a blue underline), Intentions, Routing, and Tags.

Below the navigation bar is a search bar with a "Search Across" dropdown and a "Health Status" dropdown.

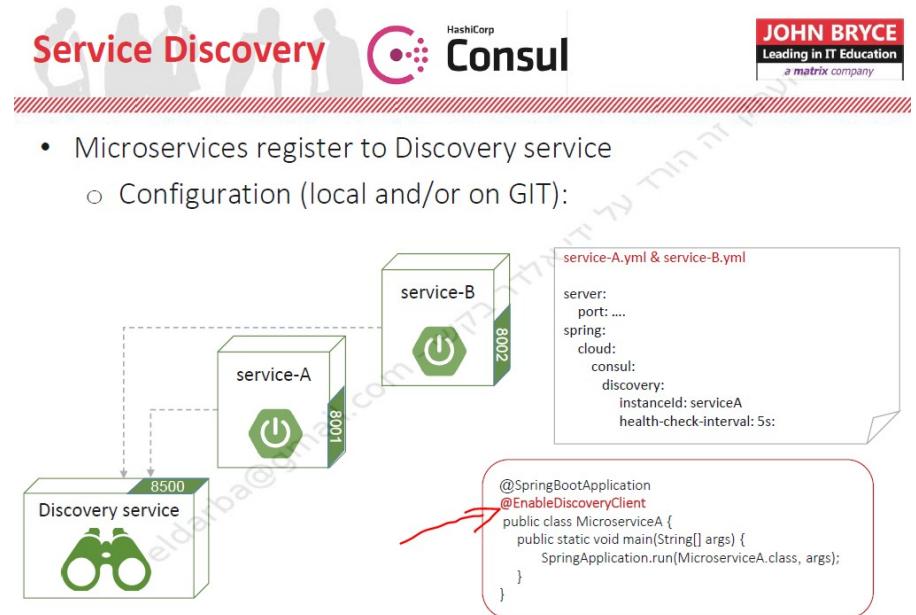
The main panel displays the service details for "ServiceA":

- All service checks passing (green checkmark)
- All node checks passing (green checkmark)
- A status entry for "eldar" with a location icon and "localhost:8001" (both highlighted with red circles)

הינו אמורים לשים ב client את האנומציה

```
6  
7 @SpringBootApplication  
8 @EnableDiscoveryClient  
9 public class ServiceAApplication {  
0  
1•   public static void main(String[] args) {  
2     SpringApplication.run(ServiceAApplication.class, args);  
3   }  
4  
5 }
```

אבל זה לא חובה כי זה enabled כברירת מחדל.





- More Consul client configuration

Property	Default Value
spring.cloud.consul.discovery.ip-address	localhost:8500
spring.cloud.consul.config.enabled	true
spring.cloud.consul.config.fail-fast	true
spring.cloud.consul.discovery.enabled	true
spring.cloud.consul.discovery.health-check-interval	10s
spring.cloud.consul.discovery.health-check-timeout	10s
spring.cloud.consul.discovery.instance-id	
spring.cloud.consul.discovery.instance-zone	

- Find list at:

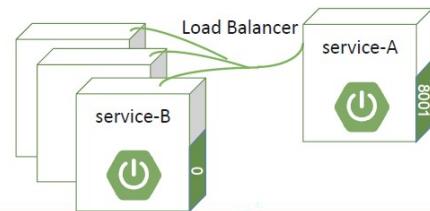
<https://cloud.spring.io/spring-cloud-consul/reference/html/appendix.html>

Load Balancing

Client Side Load Balancing

JOHN BRYCE
Leading in IT Education
a matrix company

- Spring Cloud Ribbon
 - Ribbon maintains load-balancing for domain-intra-communication
 - Tracks living instances & ignores failed instances
 - Maintains valid available server list
 - Can be fully configured both programmatically & configuration files
 - Can be easily wrap any RestTemplate activity done from one Microservice to another
 - Round-robin is used by default



36

© All rights reserved to John Bryce Training LTD from Matrix group

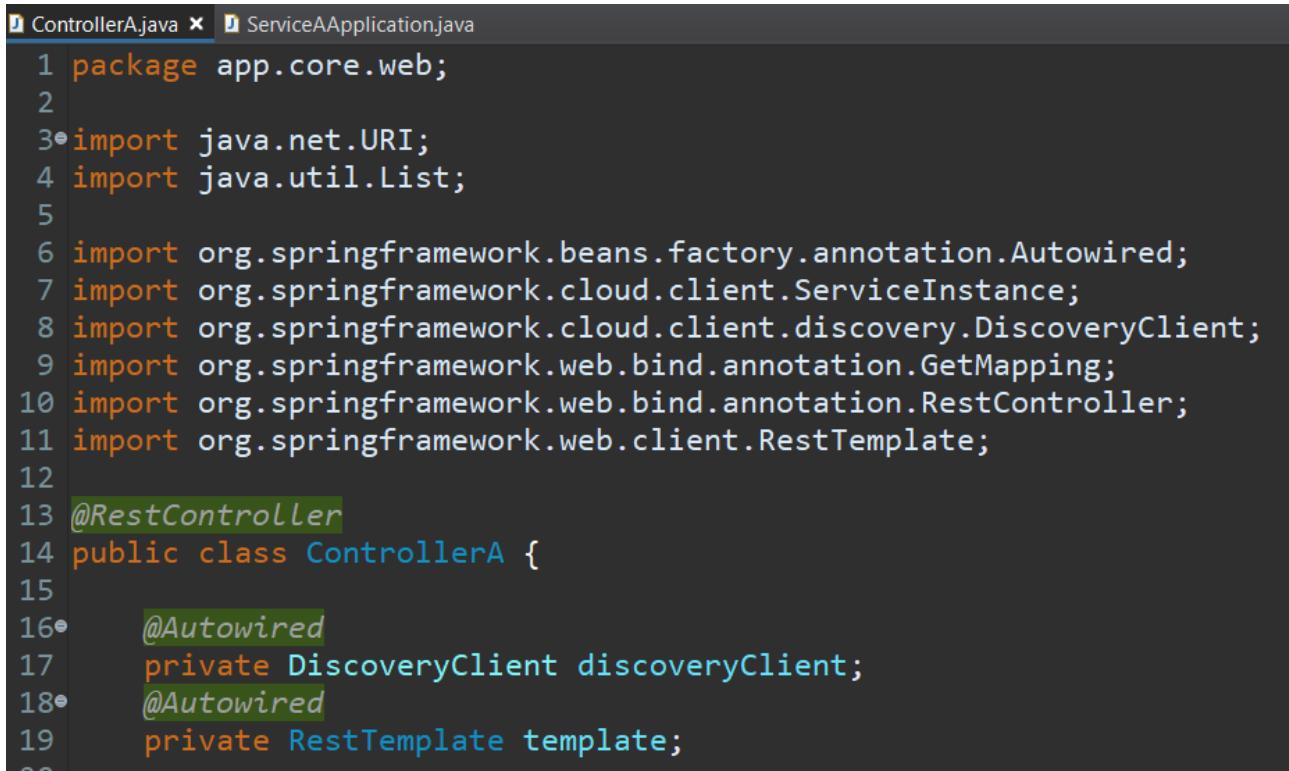
יש לנו discovery server או גם A וגם B יודעים מי רץ
עכשו נעשה ש A קורא ל B ואז נראה שיש לנו LB. אנחנו משתמש ב ribbon שקיים על A
.circuit breaking גם נרצה גם מאוחר יותר,

לעבודה:

נוסיף ל controller של A קריאה ל B באמצעות RestTemplate.
את המופע של RestTemplate נגדיר כ Bean בקובץ konfiguracija של spring על מנת שייהיה מנוהל (יהיו אלה יתרונות בהמשך)

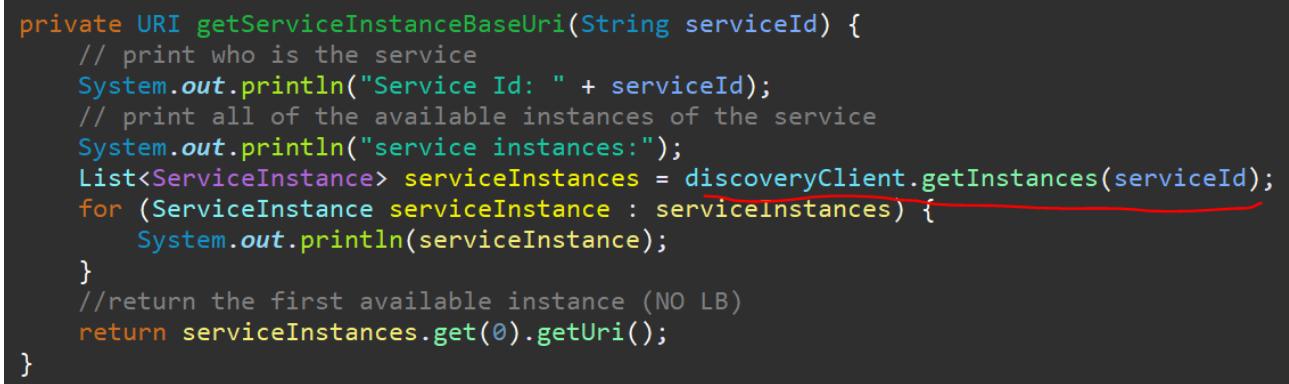
```
ServiceAApplication.java
1 package app.core;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.cloud.client.discovery.EnableDiscoveryClient;
6 import org.springframework.context.annotation.Bean;
7 import org.springframework.web.client.RestTemplate;
8
9 @SpringBootApplication
10 @EnableDiscoveryClient
11 public class ServiceAApplication {
12
13     public static void main(String[] args) {
14         SpringApplication.run(ServiceAApplication.class, args);
15     }
16
17     @Bean
18     public RestTemplate restTemplate() {
19         return new RestTemplate();
20     }
21
22 }
```

עכשו נשתמש controller מותך ב RestTemplate כדי לקרוא ל B. כדי לעשות זאת, נדרש DiscoveryClient כדיגלות בערכתו את הכתובות של המופעים השונים של B



```
1 package app.core.web;
2
3 import java.net.URI;
4 import java.util.List;
5
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.cloud.client.ServiceInstance;
8 import org.springframework.cloud.client.discovery.DiscoveryClient;
9 import org.springframework.web.bind.annotation.GetMapping;
10 import org.springframework.web.bind.annotation.RestController;
11 import org.springframework.web.client.RestTemplate;
12
13 @RestController
14 public class ControllerA {
15
16     @Autowired
17     private DiscoveryClient discoveryClient;
18     @Autowired
19     private RestTemplate template;
20 }
```

נוסף מתודה לקבלת URI הבסיס של אחד המופעים של שירות מסוים (B)



```
private URI getServiceInstanceBaseUri(String serviceId) {
    // print who is the service
    System.out.println("Service Id: " + serviceId);
    // print all of the available instances of the service
    System.out.println("service instances:");
    List<ServiceInstance> serviceInstances = discoveryClient.getInstances(serviceId);
    for (ServiceInstance serviceInstance : serviceInstances) {
        System.out.println(serviceInstance);
    }
    //return the first available instance (NO LB)
    return serviceInstances.get(0).getUri();
}
```

נערך את המетодה של ה controller כדי להוסיף את הקריאה ל B

```
@GetMapping("/service/a")
public String handleA() {
    try {
        // get the base URI of a B instance
        // http://ip:port/
        String serviceId = "service-b";
        URI baseUri = getServiceInstanceBaseUri(serviceId);
        String url = baseUri + "/service/b";
        System.out.println(url);
        // make the request to B
        String res = template.getForObject(url, String.class);
        return "Service A calling B: " + res;
    } catch (Exception e) {
        e.printStackTrace(System.out);
        return "error";
    }
}
```

נראה ונראה שהוא עובד אבל אין LB.
זה כל הזמן אותו מופע של B

נכניס LB לסייע

נזכיר לו לקוח B

```
@RestController
public class ControllerA {

    @Autowired
    private RestTemplate template;
    @Autowired
    private DiscoveryClient discoveryClient;
    @Autowired
    private LoadBalancerClient lbClient;
```

מוסיף עוד METHODה ל controller שתחזיר URI של מופע אחר בכל פעם:

```
private URI getServiceInstanceBaseUriLB(String serviceId) {
    ServiceInstance serviceInstance = lbClient.choose(serviceId);
    return serviceInstance.getUri();
}
```

נעדכן את הקריאה לקבלת ה URI

```
@GetMapping("/service/a")
public String handleA() {
    try {
        // get the base URI of a B instance
        // http://ip:port/
        String serviceId = "service-b";
        URI baseUri = getServiceInstanceBaseUrlLB(serviceId);
        String url = baseUri + "/service/b";   
        System.out.println(url);
        // make the request to B
        String res = template.getForObject(url, String.class);
        return "Service A calling B: " + res;
    } catch (Exception e) {
        e.printStackTrace(System.out);
        return "error";
    }
}
```

ונרץ. הפעם זה LB – כל פעם מופיע אחר של B

LB Aspect

כאמור, יש לנו את המופיע של RestTemplate שהגדכנו כ Bean בקובץ קונפיגורציה של spring.xml. ולכן הוא מנווה, אז נוכל גם להלביש עליו אספקט של LB.

האספקט הזה רץ הולך ל ribbon שואל אותו מי בתור ומפנה אליו את הבקשה הבאה.

```
ServiceAApplication.java
1 package com.example.serviceB;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.cloud.client.discovery.EnableDiscoveryClient;
6 import org.springframework.cloud.client.loadbalancer.LoadBalanced;
7 import org.springframework.context.annotation.Bean;
8 import org.springframework.web.client.RestTemplate;
9
10 @SpringBootApplication
11 @EnableDiscoveryClient
12 public class ServiceAApplication {
13
14     public static void main(String[] args) {
15         SpringApplication.run(ServiceAApplication.class, args);
16     }
17
18     @Bean
19     @LoadBalanced
20     public RestTemplate restTemplate() {
21         return new RestTemplate();
22     }
23
24 }
```

עכשו נשימוש בזה ב controller
לא צריך יותר את המתוודות שמחזירות URI למופע של B
במקום לציין IP ו PORT נשימוש בשם הלוגי של השירות:

```

1 package com.example.serviceA.web;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4
5 @RestController
6 public class ControllerA {
7
8     @Autowired
9     private RestTemplate template;
10
11     @GetMapping("/service/a")
12     public String handleA() {
13         String url = "http://service-b/service/b";
14         return "Service A calling B: " + template.getForObject(url, String.class);
15     }
16
17 }
18
19
20 }
21

```

במקום לציין IP ו PORT נשימוש בשם הלוגי (instance id) של השירות כפי שהגדרנו ב GIT

```

server:
  port: 0

spring:
  cloud:
    consul:
      discovery:
        instanceId: ${spring.application.name}:${random.value}
        health-check-interval: 5s
      config:
        management:

```

ששואב את החלק הראשון של השם מקובץ ה properties

```

1 spring.application.name=service-B
2 spring.config.import=configserver:http://localhost
3 #spring.config.import=optional:configserver:http://
4 server.port=7002

```

(lower case)
וכאמור הכל הופך ל lower case
אם הגענו ואמרנו קוראים לי service-b אז בתוך ה ribbon של כל מיקרו שירות שנרשם ב discovery יש map שבתוכו יש key ו values וזה ה host של כל אחד מהמופעים הקיימים. זה in memory ו לא ushims לו dump (הعبرת לזרקן טוחן ארוך - קובי) אז אם יש לנו שני מופעי B הדבר הזה יפנה אותנו לשתי כתובות אפשריות. ומשם נמשיך לנתיב של ה controller שהוא service/b

.LB זהו

Client Side Load Balancing

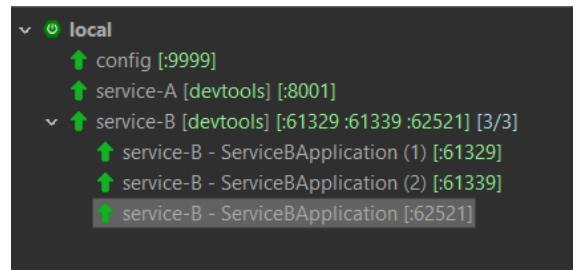
- For load-balancing multiple service-B instances should be running
 - Random ports by assigning zero value allows multiple instances on the same host
 - In order to provide each instance a unique name – edit service-B.yml on GIT:



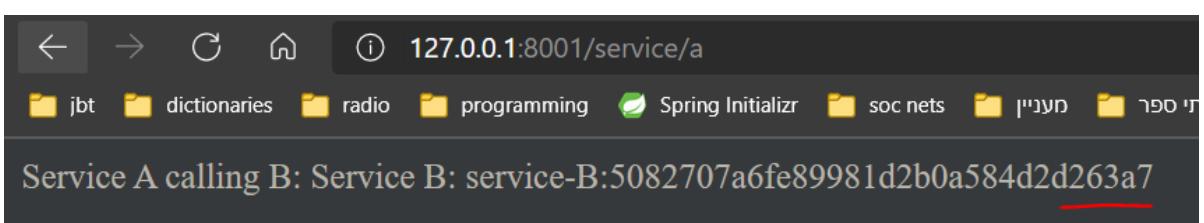
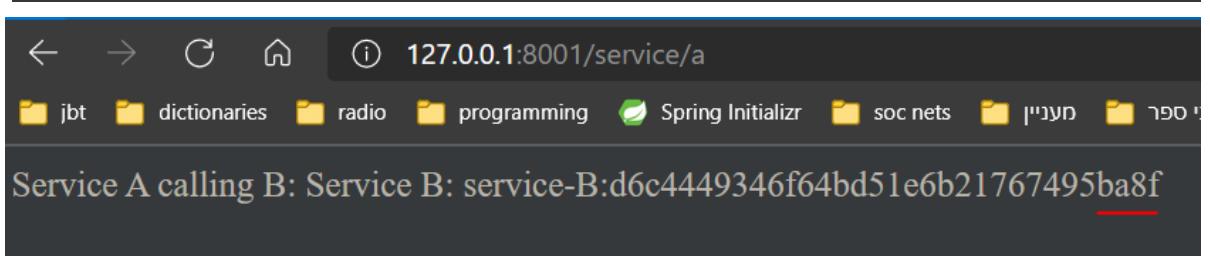
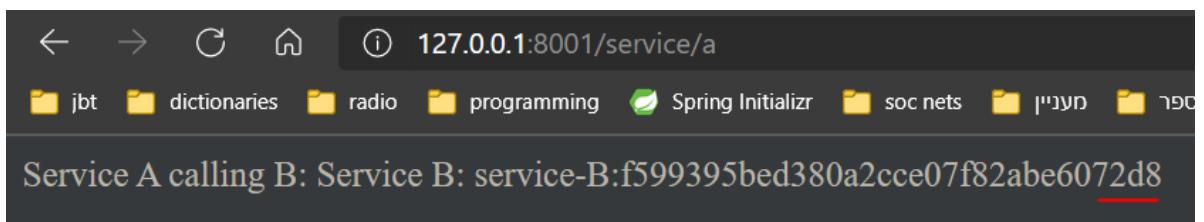
37

© All rights reserved to John Bryce Training LTD from Matrix group

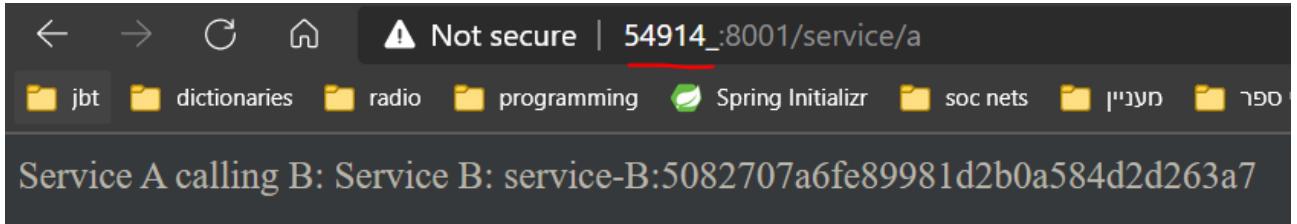
כדי לראות LB בפעולה נרים מופע A ו 3 מופעי B ונפנה לכתובת של השירות ב A



בפניה בדף ניתן לחת את ה IP את את ה domain name של DNS של consul נתן לנו:



אפשר כמובן לחת במקום הכתובת localhost IP 127.0.0.1 זה ה domain name



service-A

A screenshot of a web-based configuration interface for 'ServiceA'. At the top, there are tabs for 'Instances', 'Intentions', 'Routing', and 'Tags', with 'Instances' being the active tab. Below the tabs is a search bar with the placeholder 'Search' and a dropdown menu 'Search Across'. Further down are two status indicators: 'All service checks passing' and 'All node checks passing', both marked with green checkmarks. Underneath these is a list of instances, each with a name, status, and IP address. One instance, 'ServiceA' with IP '54914_8001', has its IP address highlighted with a red box.

עכשו מה שיפה שאפשר להרוג את אחד מмоפעי B ולראות שתוך חמש שניות ה ribbon מתעדכן ויודע כבר לא לשלוח אלינו למופע שאיננו. ב 5 שניות ראשונות כן ניפול עד שיתעדכן.

5 שניות interval זה יכול להיות 10 שניות בפועל או אפילו יותר משום ש 5 שניות הם של discovery ו עוד חמיש שניות בצד של השירות בקיצור יכול לקחת כמה שניות טובות.

Client Side Load Balancing



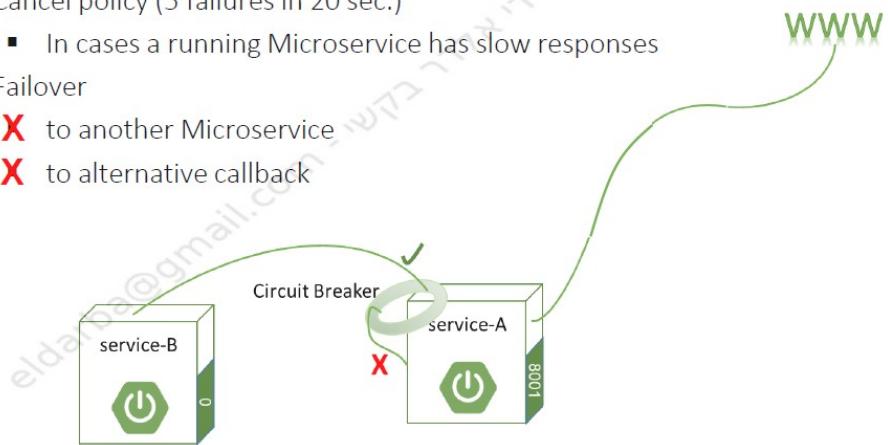
- Spring Cloud Ribbon Load Balancer & Circuit Breaker
 - Ribbon is for load balancing
 - Circuit breaker is for a case where there are no available servers on the Ribbon
- Gateway & client load balancing uses ribbon
- Any load balanced call may use circuit breaker as well

Circuit Breaker

Circuit Breaker – Hystrix

JOHN BRYCE
Leading in IT Education
a matrix company

- Hystrix implements Circuit breaker DP
 - Tracks requests
 - Cancel policy (5 failures in 20 sec.)
 - In cases a running Microservice has slow responses
 - Failover
 - X to another Microservice
 - X to alternative callback



40

© All rights reserved to John Bryce Training LTD from Matrix group

از אנחנו יודעים להגעה מ A ל B.
cut that's missing a circuit breaker to return to A if there are 0 failures of B. It's not clear what happens if there are failures.

המקרה שבו יש עיכוב כי הורדנו חלק מהמוסעים זה עדין לא .circuit breaker

בالمושך עוד נראה איך מתגברים על מקרה שבו LB חושב שיש ולכון לא מפעיל CB.

CB הוא רכיב נוסף חיוני. מה שמבצעו אותו זה LB. כשהוא LB עם אפס שרתים לעבוד איתם אז הוא CB מטיג את LB.

יש מקרה שלישי שבו שרת יורד או מתרסק בבדיקה כשלקוח פונה אליו. אז ה ribbon (רשימת השירותים) עדין לא התעדכן. הוא חושב שהשירות הזה חי. הוא רץ אליו אבל הוא כבר מת.פה יש קוצר כי CB לא הופעל כי יש עוד שרתים ובמקרה זה שתורו הגעת הטרסק איך שלחצנו.

.POM זה CB של Netflix. נוסיף אותו ל Hystrix

Circuit Breaker – Hystrix



- Client Microservice Hystrix
circuit breaker Maven
dependencies:

```
pom.xml
...
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-hystrix</artifactId>
</dependency>
...
...
```

כדי להוסיף hystrix יש לשנムך גרסה של הפרויקט A
לשם כך ראה פרק:
versions for hystrix

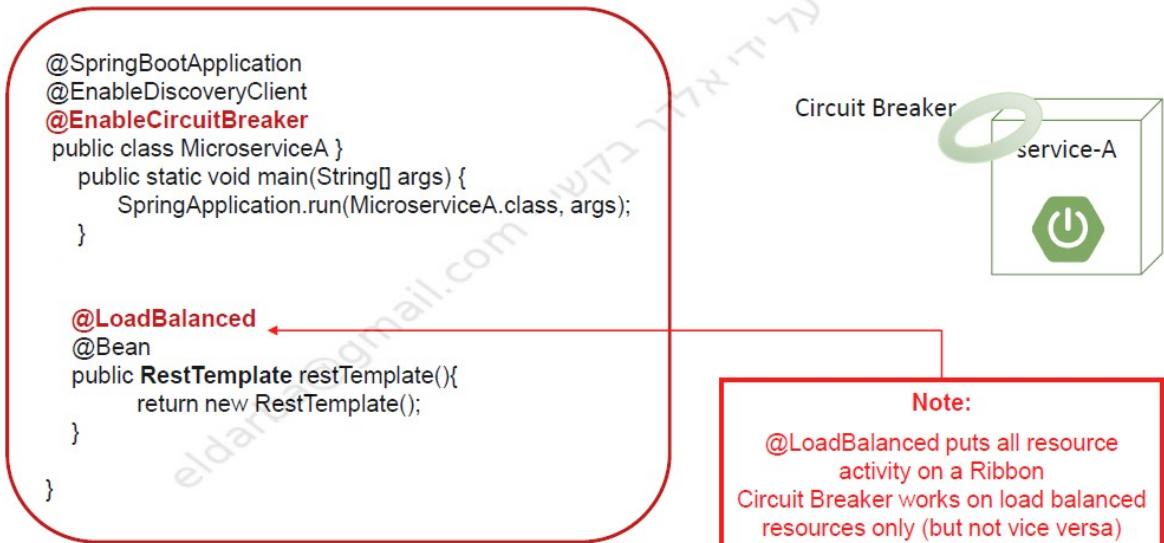
נוסיף hystrix לפרויקט של A כדי שבנוסף ל LB יהיה לנו גם CB

```
68
69     <dependency>
70         <groupId>org.springframework.cloud</groupId>
71         <artifactId>spring-cloud-starter-netflix-hystrix</artifactId>
72     </dependency>
```

נשים לב ש CB זה ממש כמו try-catch רק שהוא חוצה שפות חוצה מופעים.

Circuit Breaker – Hystrix

- Circuit Breaker is configured on client Microservice:



ازLB הוא LB ועשינו כאן EnableCB למעלה כך שהמנגנון הזה מתחילה הקוד אלטרנטיבי אם LB שלי מחזיק אפס available servers אז נכנס את השינוי בקונפיגורציה של A

A screenshot of a Java code editor showing the file `ServiceAApplication.java`. The code includes annotations `@SpringBootApplication`, `@EnableDiscoveryClient`, `@EnableCircuitBreaker`, and `@LoadBalanced`. The `@LoadBalanced` annotation is highlighted in green.

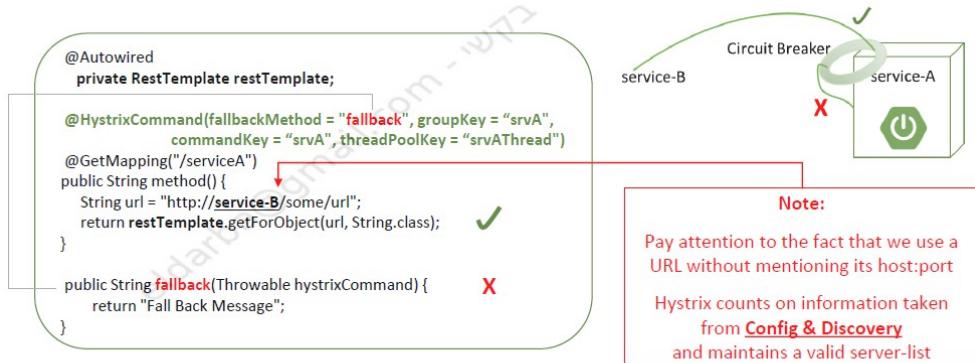
```
1 package com.example.serviceA;
2
3 import org.springframework.boot.SpringApplication;
4
5 @SpringBootApplication
6 @EnableDiscoveryClient
7 @EnableCircuitBreaker
8 public class ServiceAApplication {
9
10    public static void main(String[] args) {
11        SpringApplication.run(ServiceAApplication.class, args);
12    }
13
14    @LoadBalanced
15    @Bean
16    public RestTemplate restTemplate() {
17        return new RestTemplate();
18    }
19
20 }
```

לצורך בדיקה בשלבי הפיתוח אפשר להפעיל גם CB כדי לבדוק אם יש בעיות. אם זה נופל. ולהבין עד כמה ה production יציג. כਮובן לא ב

ועכשו השינוי ב controller .controller
איך מכניסים את הקוד האלטרנטיבי למקורה שאין שרתים זמינים לשירות שאחנו מוחפשים.

Circuit Breaker – Hystrix

- Using load-balanced resources:
 - @HystrixCommand – defines failover callback & meta data
 - URL is resolved by Gateway Server (“/service-B/”)



43

© All rights reserved to John Bryce Training LTD from Matrix group

ראשית, ניתן לראות שאנו פונים לשירות B עם שמו הלוגי, השם שהוא נרשם אצל config server והוא נרשם אצל ה CLOUD provider. ה Hystrix ישתמש בribbon discovery ובודק אם המופעים שנרשמו ב discovery והו אמורים לעמוד בדרישות ה Hystrix. ה Hystrix ינסה ליצור גישה ל-SERVICE B וесתכל אם ה SERVICE B מצליח לספק לנו עלייה ששמם service-b.service-A. אם לא, ה Hystrix ישתמש ב fallback method.

שמיים ribbon @HystrixCommand מעל למетодה. מגדירים מהי ה fallback method שתווסף במקרה שהריך – יש אפס שירותים מ service-b. המתוודה יcolla לשות rest template למקומות אחרים או לשות משלחו מקומי.

חייבת להחזיר את אותו type בבדיקה – כדי שיחזר אותו MIME type שהובטח ב business exception. אי אפשר להחזיר משהו אחר. HTTP לא מעוניין אותו מה קרה אצלנו בשרת. חשוב שהmethod תקבל Throwable כדי שנוכל לחזור את ה exception.

```
ControllerA.java x http://localhost:8001/service/a
1 package app.core.web;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.web.bind.annotation.GetMapping;
5 import org.springframework.web.bind.annotation.RestController;
6 import org.springframework.web.client.RestTemplate;
7
8 import com.netflix.hystrix.contrib.javanica.annotation.HystrixCommand;
9
10 @RestController
11 public class ControllerA {
12
13     @Autowired
14     private RestTemplate template;
15
16     @HystrixCommand(fallbackMethod = "handleAFallback")
17     @GetMapping("/service/a")
18     public String handleA() {
19         String url = "http://service-b/service/b";
20         return "Service A calling B: " + template.getForObject(url, String.class);
21     }
22
23     public String handleAFallback(Throwable t) {
24         return "service A fallback message: can't call service B. cause: " + t;
25     }
26
27 }
```

עכשו אם נוריד כל מופע B נקבל CB ואם נחזיר ייחזר לעבוד כרגע. במקרה שהmethod המקורית זורקת שגיאה או מופעל ה fallback ואנחנו מקבלים את הרפרנס לשגיאה דרך הparameter.

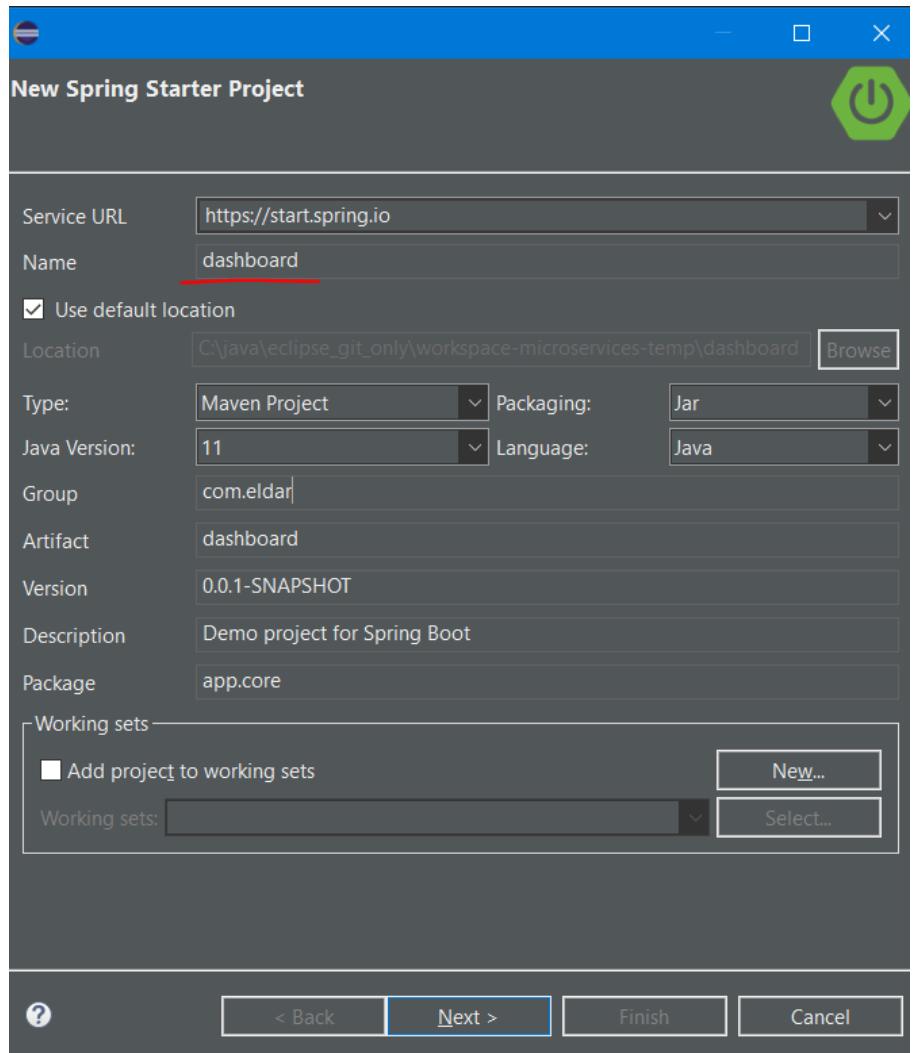
versions for hystrix

מגבלת גרסאות קיימת ב Hystrix של Netflix הדרוש לנו עברו:

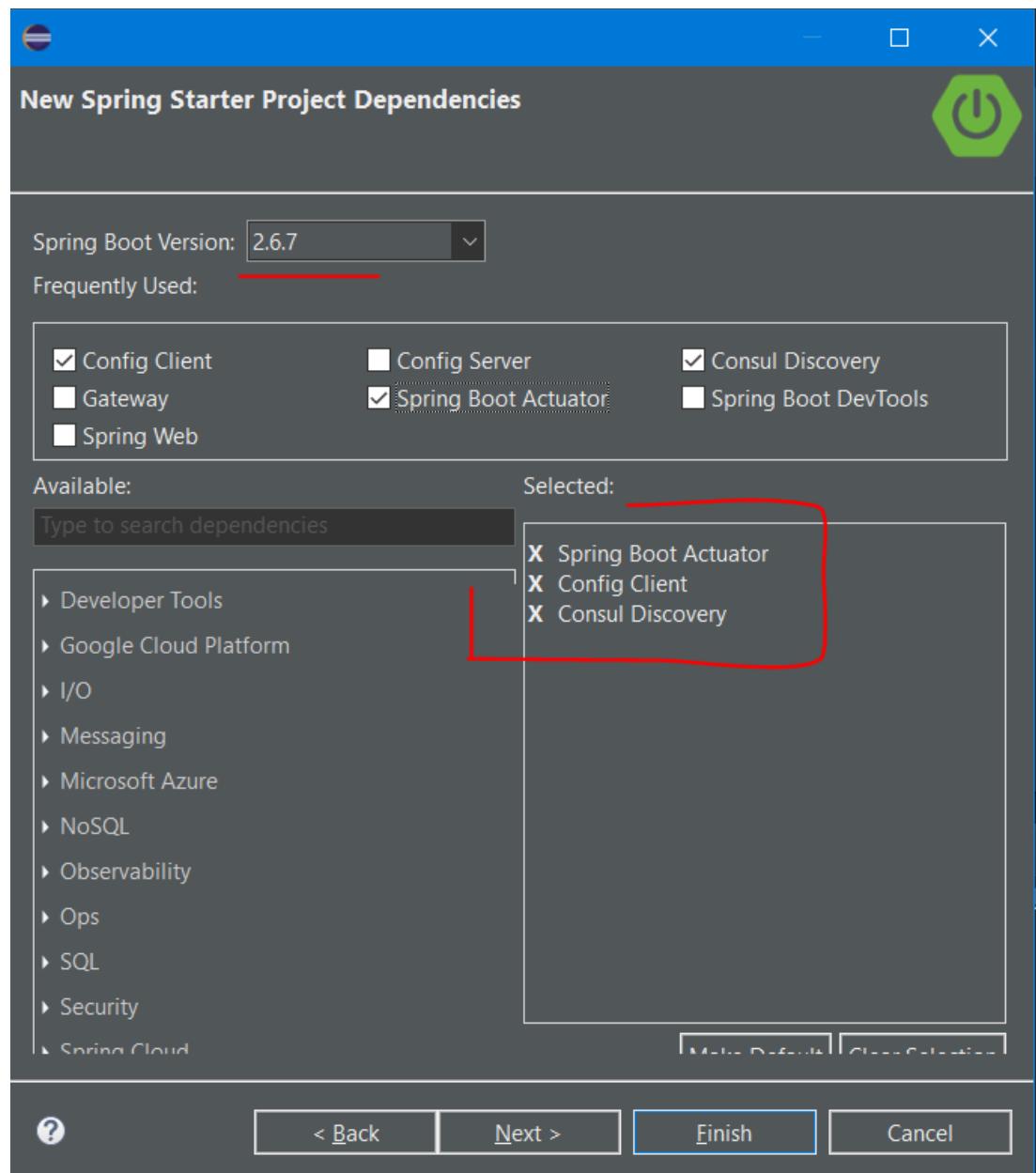
1. נחוץ בשירות – Circuit Breaking
2. נחוץ עבר שירות ה dashboard – Dashboard Monitoring

אז נפעיל כך ביצירת הפרויקטים שדורשים Hystrix

יצירת הפרויקט



קביעת התלויות



עדכון ה POM

שנמצא גרסאות כדי להתאים ל hystrix
 CB – אם רוצים hystrix –
 הוספה ידנית של התלוות – hystrix-dashboards – אם רוצים

```

1 <?xml version="1.0" encoding="UTF-8"?>
2<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001
3   <modelVersion>4.0.0</modelVersion>
4<parent>
5   <groupId>org.springframework.boot</groupId>
6   <artifactId>spring-boot-starter-parent</artifactId>
7   <!-- <version>2.6.7</version> -->
8   <version>2.3.10.RELEASE</version>
9   <relativePath /> <!-- lookup parent from repository -->
0</parent>
1<groupId>com.eldar</groupId>
2<artifactId>dashboard</artifactId>
3<version>0.0.1-SNAPSHOT</version>
4<name>dashboard</name>
5<description>Demo project for Spring Boot</description>
6<properties>
7   <java.version>11</java.version>
8   <!-- <spring-cloud.version>2021.0.2</spring-cloud.version> -->
9   <spring-cloud.version>Hoxton.SR4</spring-cloud.version>
0</properties>
1<dependencies>
2<dependency>
3   <groupId>org.springframework.cloud</groupId>
4   <artifactId>spring-cloud-starter-netflix-hystrix</artifactId>
5</dependency>
6<dependency>
7   <groupId>org.springframework.cloud</groupId>
8   <artifactId>spring-cloud-starter-netflix-hystrix-dashboard</artifactId>
9</dependency>

```

<version>2.3.10.RELEASE</version>

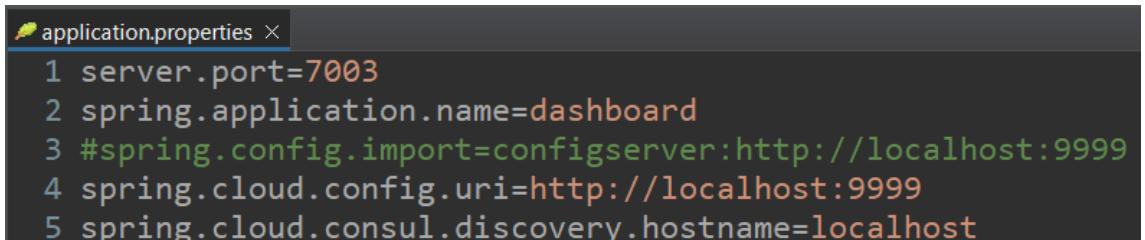
<spring-cloud.version>Hoxton.SR4</spring-cloud.version>

```

<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-netflix-hystrix</artifactId>
</dependency>

```

עדכון קובץ ה properties



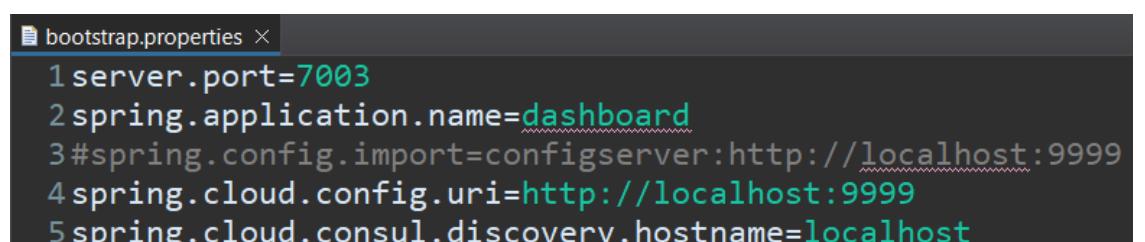
```

application.properties ×
1 server.port=7003
2 spring.application.name=dashboard
3 #spring.config.import=configserver:http://localhost:9999
4 spring.cloud.config.uri=http://localhost:9999
5 spring.cloud.consul.discovery.hostname=localhost

```

spring.cloud.config.uri=http://localhost:9999

יש לשנות את שם הקובץ ל bootstrap.properties



```

bootstrap.properties ×
1 server.port=7003
2 spring.application.name=dashboard
3 #spring.config.import=configserver:http://localhost:9999
4 spring.cloud.config.uri=http://localhost:9999
5 spring.cloud.consul.discovery.hostname=localhost

```

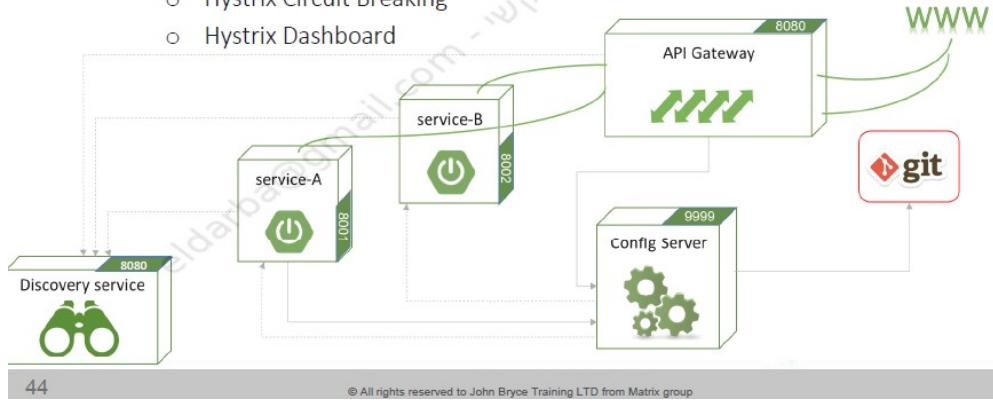
API Gateway

הלקוח לא צריך לדעת שהרצינו את A על פורט 8001. ובכלל גם A אמור להיות כמו B (מרובה מופעים) לכן אנחנו צריכים gateway.

API Gateway – Spring API Gateway

JOHN BRUCE
Leading in IT Education
a matrix company

- Spring Cloud provides its own API-Gateway
 - Is reactive and comes with embedded Netty by default
 - Easy to configure
 - Except from routing, also supports
 - Forwarding via Ribbon LB
 - Hystrix Circuit Breaking
 - Hystrix Dashboard



לקוחות לא יגיעו ישירות ל A ול B. הם גם לא יודעים מה ה port שלהם כי שניהם אמורים להיות orchestrated gateway.

ל gateway יהיה IP קבוע.

חוץ מזה שהוא מנטב את הבקשות למיקרו שירות הוא יודע לתת לנו:

- תמיכה ב ribbon כדי לעשות LB
- תמיכה ב CB.

אם B נפל ו A קורא ל B כי עדין לא התעדכן אז A חוטף את ה exception. אבל A לא מפעיל CB כי ribbon שלו חי בשרת או הוא מעיף את ה exception ואז ה gateway קולט שהבקשה נכשלה ומפעיל bullet proof gateway. אם כן CB שלו. אם כן gateway הדריך לעשות את המערכת

איך עושים את זה. צריך לבנות פרויקט נוסף עם כמה

API Gateway – Spring API Gateway

JOHN BRYCE
Leading in IT Education
a matrix company

- Supports the following routing patterns:
 - / - relative local path
 - http:// or https:// – absolute address
 - lb:// - using load-balancer (ribbon) patterns

- Maven dependencies:

```
pom.xml
...
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-gateway</artifactId>
</dependency>...
```



לעיל היבוא של gateway – מגיע כבר עם Netty משומש שהוא ריאקטיבי

```
: You already have RibbonLoadBalancerClient on you
: You have RibbonLoadBalancerClient on your classp
: Initializing ExecutorService 'catalogWatchTaskSc
: Netty started on port(s): 8080
: Registering service with consul: NewService{id='
: Started GatewayApplication in 8.099 seconds (JVM
: Fetching config from server at : http://localhost:8080
: Located environment: name=Gateway, profiles=[def
```

API Gateway – Spring API Gateway

- Adding Spring API Gateway Microservice
 - Set API Gateway to work with Discovery service:



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

& set configuration accordingly

API Gateway – Spring API Gateway

- More Maven configuration
 - In order to fetch configuration from Config-Server add:

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-config</artifactId>
</dependency>
```
 - In order to support Hystrix Circuit Breaker & Dashboard add:

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-hystrix</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```



צריך פה כמה :dependencies

• gateway – זה פשוט gateway שעבוד כבר עם spring. נוסיף לו:
• console discovery (ולכן צריך גם actuator)

שוהלך ומושך את הקונפיגורציה מ git שתגידי לו רוץ לגבעה ותעשה שלום ל discovery (או צרי)
(consul discovery)

hystrix כדי שנוכל ללחט ולעבד עם CB (ומאוחר יותר גם כדי לנטר את המיקרו סרвис הזה).
אנחנו רוצים ש gateway הזה הוא זה שיפעל את CB במקומות האחרון אם אייכשו משחו קרה
והצליח להתגלגל עד ל client. זה המקום האחרון לעצור ולתפוס את זה עם CB.

actuator •

אם כן זהו המיקרו סרבר השלישי שאנו חובה כתובים ויש לו כבר קונפיגורציה ב GIT.
ניתן לראות שהוא צריך להתגלוות ולאחר מכן יש עוד סימן של discovery שהריגל של קונסול.
אחר מכן יש שם מלא קונפיגורציה שנראה מה היא אומרת בהמשך.

```
spring:
  cloud:
    gateway:
      discovery:
        locator:
          enabled: true
      routes:
        - id: serviceA
          uri: lb://service-A
          predicates:
            - Path=/srva/**
          filters:
            - RewritePath=/srva/(?<segment>.*), /$\\{segment}
            - name: Hystrix
              args:
                name: monitor-srvA
                fallbackUri: forward:/fallback/serviceA
        - id: ServiceB
          uri: lb://service-B
          predicates:
            - Path=/srvb/**
          filters:
            - RewritePath=/srvb/(?<segment>.*), /$\\{segment}
            - name: Hystrix
              args:
                name: monitor-srvB
                fallbackUri: forward:/fallback/serviceB
        - id: MQActivityService
          uri: lb://activity
          predicates:
            - Path=/act/**
          filters:
            - RewritePath=/act/(?<segment>.*), /$\\{segment}
      consul:
        discovery:
          instanceId: Gateway
          health-check-interval: 5s
      config:
```

בשלב הבא ניגש למחלקה קונפיגורציה וניתן ל hystrix enable ול discovery בשביל CB ובשביל המוניטורינג שעוד לא הגנו אליו ולא כתבנו כדי שה gateway יתגלה ויראה בקונסול.

API Gateway – Spring API Gateway

JOHN BRYCE
Leading in IT Education
a matrix company

- Update configuration main
 - Enable discovery client and Hystrix dashboard (later)



```
@SpringBootApplication
@EnableHystrix
@EnableDiscoveryClient
public class APIGatewayServiceApplication {

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

איזה URL נופל – איזה CB להפעיל.

API Gateway – Spring API Gateway

- Add Circuit-Breaker to API Gateway service
 - Optionally, add fallback endpoints in your Gateway service
 - API-Gateway can be configured to use
 - Fallback on the Gateway server (itself)
 - Fallback on other endpoint using URIs & Ribbon LB



```

@RestController
public class FallbackController {
    @GetMapping("fallback/serviceA")
    public String fallbackA() {
        return "API Gateway Service-A Fallback";
    }
    @GetMapping("fallback/serviceB")
    public String fallbackB() {
        return "API Gateway Service-B Fallback";
    }
}

```

ניתן URL Patern בשביל fallback לשרת A ולשרת B

זה אומר שאם פנינו דרך ה gateway ל A אז יש כמה אפשרויות:

1. A לא זמין – יופעל ה CB לעיל שהוגדר עבור פניה כושלת ל A

2. A זמין אבל כשהוא קורא ל B אז B לא זמין – שוב, יופעל ה CB לעיל שהוגדר עבור פניה כושלת

ל A

3. A זמין ו B זמין – לא יופעל CB אלא מקבל response רגיל

זה הופך אותנו ל bullet proof

אבל איך אומרים ל gateway שבמקרה של כישלון בפניה ל A אז צריך להפעיל את ה fallback שהגדכנו ב gateway לעיל. אז זה נעשה באמצעות הקונפיגורציה שלו ב GIT

API Gateway – Spring API Gateway

- Request configuration from Config-server
- Optionally, configure the following in your bootstrap.yml:

```
bootstrap.yml
spring:
  application:
    name: api-gateway
  cloud:
    config:
      uri: http://localhost:9999
```



50 © All rights reserved to John Bryce Training LTD from Matrix group
נותנים שם – בעצם אומרים לו רוץ ל config server תגיד שקוראים לך api-gateway ותקבל את הקונפיגורציה שלך. אז הוא מקבל מ GIT את כל מה שמופיע בקובץ שראינו ב GIT

זה מה שהוא מקבל מ GIT

API Gateway – Spring API Gateway

- Configuring discovery and monitor
 - api-gateway.yml eureka client & hystrix configuration:

```
api-gateway.yml
management:
  endpoints:
    web:
      exposure:
        include: hystrix.stream, info, health
```



זה נראה קצת אחרת:

כי כאן עבדו עם אבל אנחנו עובדים עם Consul (ראה קובץ ב GIT)

API Gateway – Spring API Gateway

JOHN BRYCE
Leading in IT Education
a matrix company

- Configuring discovery and monitor
 - api-gateway.yml eureka client & hystrix configuration:



51

© All rights reserved to John Bryce Training LTD from Matrix group

از עבר על הקובץ קונפיגורציה שב GIT ונראה מה ה gateway מקבל משם discovery

דבר ראשון לכת להתגלות ב discovery

דבר שני, תחת management (שייך ל actuator) כתוב לו מה לגנות וכוכי.

Spring Boot Actuator: Production-ready Features

כפי שניתן לראות אם הולכים לlianק לעיל של actuator החשיפה של info ושל health קיימת גם כך כבירות מחדל

```
consul:
discovery:
instanceId: Gateway
health-check-interval: 5s
config:
management:
enabled: true
management:
security:
enabled: false
endpoints:
web:
exposure:
include: hystrix.stream, info, health
```

עד כאן הדברים הרגילים אבל יש בהמשך קונפיגורציה ספציפית ל gateway gateway זו כבר ספציפי ל spring.cloud.gateway

API Gateway – Spring API Gateway

JOHN BRYCE
Leading in IT Education
a matrix company

- Configuring routing & fallbacks
 - api-gateway.yml routing and fallback URLs configuration:

```
api-gateway.yml

spring:
  cloud:
    gateway:
      discovery:
        locator:
          enabled: true
      routes:
        - id: serviceA
          uri: lb://service-A
          predicates:
            - Path=/aService/**
          filters:
            - RewritePath=/ aService/(?<segment>.*), ${segment}
            - name: Hystrix
              args:
                name: monitor-serviceA
                fallbackUri: forward:/fallback/serviceA
```



- Same can be configured to serviceB

52

© All rights reserved to John Bryce Training LTD from Matrix group

קודם כל תתגלה אחר כך נוסיף את היכולת של gateway ליצור routes המבוססים על שירותים שנרשמו כ discovery locator. כדי לאפשר זאת:

```
discovery:
  locator:
    enabled: true
```

8.2 DiscoveryClient Route Definition Locator

The Gateway can be configured to create routes based on services registered with a `DiscoveryClient` compatible service registry.

To enable this, set `spring.cloud.gateway.discovery.locator.enabled=true` and make sure a `DiscoveryClient` implementation is on the classpath and enabled (such as Netflix Eureka, Consul or Zookeeper).

דבר שני זה routes. זה מתחילה רשימה של המסלולים שאפשר לעבור עליהם.
<https://cloud.spring.io/spring-cloud-gateway/reference/html/#gateway-request-predicates-factories>

במקרה שלנו הוגדרו 3 נתיבים.

2 הראשונים זה הנתיבים ל A ול B
השלישי לא שיק ולא כולל בקורס – זה reactive

```
routes:
- id: serviceA
  uri: lb://service-A
  predicates:
  - Path=/srva/**
  filters:
  - RewritePath=/srva/(?<segment>.*), /${segment}
  - name: Hystrix
    args:
      name: monitor-srvA
      fallbackUri: forward:/fallback/serviceA
- id: ServiceB
  uri: lb://service-B
  predicates:
  - Path=/srvb/**
  filters:
  - RewritePath=/srvb/(?<segment>.*), /${segment}
  - name: Hystrix
    args:
      name: monitor-srvB
      fallbackUri: forward:/fallback/serviceB
- id: MQActivityService
  uri: lb://activity
  predicates:
  - Path=/act/**
  filters:
  - RewritePath=/act/(?<segment>.*), /${segment}
```

Reactive

לדוגמה נתמך על ה route שהוגדר בתמונה מלמטה:

```
apiGateway.yaml  
routes:  
- id: serviceA  
uri: lb://service-A  
predicates:  
- Path=/aService/**  
filters:  
- RewritePath=/aService/(?<segment>.*), /$\\{segment}  
- name: Hystrix  
args:  
    name: monitor-serviceA  
    fallbackUri: forward:/fallback/serviceA
```

מה המשמעות:

אם מישו קורא ל'ם path
/aService/**

תחליף את זה ל: lb זה (load balancer) אם היינו כתבים http במקום lb אז היינו צריכים לתת ip:port
lb://service-A

ב חלק של RewritePath אנחנו אומרים לו להשאיר את כל מה שכתוב אחר כך (החלק של ה **) מבלי
לחזור כלום. ככלומר מה שיתקבל בסוף זה החלפה של זה:

/aService/**

לזה

lb://service-A/**

בקיצור להחליף רק את aService ב service-A ולהשאיר את כל מה שארה

CB

ולא רק זה. ב filter שמננו לו גם Hystrix (שהה CB) ואמרנו לו monitor-serviceA
הולך ל

זה בעצם ה URL של ה controller שהגדכנו: fallback/serviceA

```
@RestController  
public class FallbackController {  
    @GetMapping("fallback/serviceA")  
    public String fallbackA() {  
        return "API Gateway Service-A Fallback";  
    }  
    @GetMapping("fallback/serviceB")  
    public String fallbackB() {  
        return "API Gateway Service-B Fallback";  
    }  
}
```

כלומר בעצם מה שכתוב פה זה ככה:

אם תעבירו לי בקשה URL שיש עלייה

/aService/service/a

אני אחליף את aService ב service-A וזה השם של ה load balancer וכאן אתם עכשו על ה /service-A/service/a

ואם ה ribbon יראה שאין כלום אז הוא ייפול על המתוודה הזאת לעיל.

```
routes:  
  - id: serviceA  
    uri: lb://service-A  
    predicates:  
      - Path=/srva/**  
    filters:  
      - RewritePath=/srva/(?<segment>.*), /$\\{segment}  
      - name: Hystrix  
    args:  
      name: monitor-srvA  
      fallbackUri: forward:/fallback/serviceA
```

=====

- id: serviceA

uri: lb://service-A

מעביר ל load balancer של service-A (שייה באוטוית קטנות כי כה זה ב GIT).

=====

- Path=/srva/**

ה path יהיה srva – ככה מגיעים ל service-A

=====

fallbackUri: forward:/fallback/serviceA

ה fallback יהיה forward/serviceA כמו שראינו במצגת
יכלנו פה לתת משהו עם LB, למשל:

fallbackUri:lb://service-c

(צריךแคת ניסוי וטעיה כדי לוודא שהוא אכן התחבר)
או

fallbackUri:http://.../...

בנייה הפרויקט Gateway

עכשו נבנה את זה

The screenshot shows the Spring Initializr web application interface. On the left, there are sections for 'Project' (selected 'Maven Project'), 'Language' (selected 'Java'), and 'Spring Boot' (version 2.3.10, highlighted with a red box). The 'Project Metadata' section includes fields for Group (com.example), Artifact (gateway), Name (gateway), Description (Demo project for Spring Boot), Package name (com.example.gateway), Packaging (Jar), and Java version (11). On the right, the 'Dependencies' section lists several components: Config Client (Spring Cloud Config), Consul Discovery (Spring Cloud Discovery), Spring Boot Actuator (OPS), Hystrix [Maintenance] (Spring Cloud Circuit Breaker), and Gateway (Spring Cloud Routing). Each dependency has a brief description below it.

זה כדי ש config client יוריד את הקונפיגורציה מ GIT
זה כדי להציג שלום לקונסול ושהחרי זה היא תנטר אותנו
CB עוזר לנטר אותנו ועובד גם ל actuator
CB זה hystrix – זה מה שאנחנו עושים gateway

גרסאות ב POM

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd"
  modelVersion="4.0.0">
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.3.10.RELEASE</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>com.eldar</groupId>
  <artifactId>gateway</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>gateway</name>
  <description>Demo project for Spring Boot</description>
  <properties>
    <java.version>11</java.version>
    <spring-cloud.version>Hoxton.SR11</spring-cloud.version>
  </properties>
```

לעומת מה שעשיתי קודם עבור A ו B. אז עדכנתי אותו לגרסה החדשה יותר כמו למעלה זה עובד אותו
דבר רק עם אזהרות על reflective

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd"
  modelVersion="4.0.0">
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.3.0.RELEASE</version>
    <relativePath /> <!-- lookup parent from repository -->
  </parent>
  <groupId>com.eldar</groupId>
  <artifactId>service-A</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>service-A</name>
  <description>Demo project for Spring Boot</description>
  <properties>
    <java.version>11</java.version>
    <spring-cloud.version>Hoxton.SR4</spring-cloud.version>
  </properties>
```

שימוש לב שלא הוספנו .spring-web

API Gateway – Spring API Gateway

- Update configuration main
 - Enable discovery client and Hystrix dashboard (later)



```
@SpringBootApplication
@EnableHystrix
@EnableDiscoveryClient
public class APIGatewayServiceApplication {

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

```
GatewayApplication.java x
1 package com.example.gateway;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.cloud.client.discovery.EnableDiscoveryClient;
6 import org.springframework.cloud.netflix.hystrix.EnableHystrix;
7
8 @SpringBootApplication
9 @EnableHystrix // for Hystrix circuit breakers and dashboard
10 @EnableDiscoveryClient
11 public class GatewayApplication {
12
13     public static void main(String[] args) {
14         SpringApplication.run(GatewayApplication.class, args);
15     }
16
17 }
```

עכשו ניצור controller עם שני endpoints בשביל תגובה fallback למצבים שבהם ה gateway מנסה לגשת ל A או ל B

```
FallbackController.java x
1 package com.example.gateway.web;
2
3 import org.springframework.web.bind.annotation.GetMapping;
4
5 @RestController
6 public class FallbackController {
7
8     @GetMapping("/fallback/serviceA")
9     public String fallbackA() {
10         return "Fateway: FALBACK message - cannot call service A";
11     }
12
13
14     @GetMapping("/fallback/serviceB")
15     public String fallbackB() {
16         return "Fateway: FALBACK message - cannot call service B";
17     }
18
19 }
```

את הנתיבים ל fallbackים צריך לקחת מה yml שב GIT

```
gateway:
  discovery:
    locator:
      enabled: true
  routes:
    - id: serviceA
      uri: lb://service-A
      predicates:
        - Path=/srva/**
      filters:
        - RewritePath=/srva/(?<segment>.*), /$\\{segment}
        - name: Hystrix
      args:
        name: monitor-srvA
      fallbackUri: forward:/fallback/serviceA
```

כדי ש יקבל את כל הקונפיגרציות שלו נעדכן את קובץ ה properties בהתאם – פשוטו געתייק מאחד השירות ונסנה לפורט 8080 (שזה למעשה ה default או אפשר לוותר)

```
bootstrap.properties
1spring.application.name=Gateway
2spring.cloud.config.uri=http://localhost:9999
3server.port=8080
```

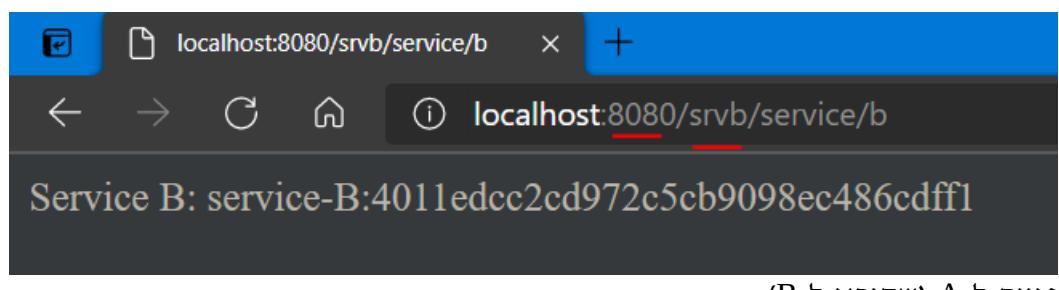
שם האפליקציה תואם ל yml שב GIT

```
consul:
  discovery:
    instanceId: Gateway
    health-check-interval: 5s
  config:
```

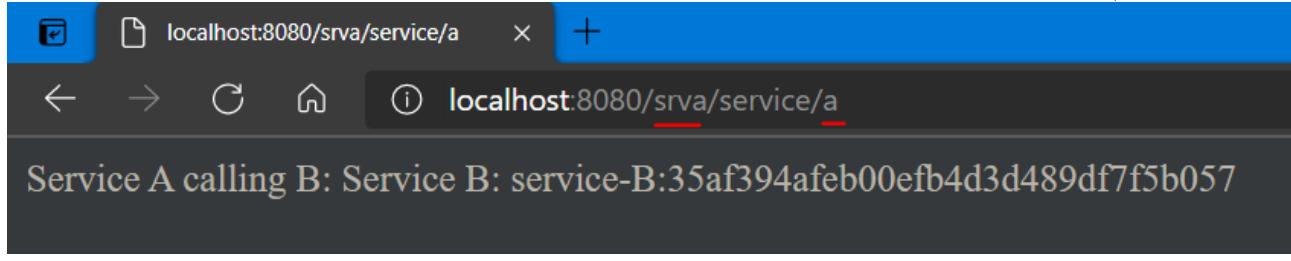
כלומר לך ל TagID שאתה gateway והוא יdag לך.

עכשו נרים הcola ונפנה ל gateway כדי לפנות ל A ול B

פנינה ל B

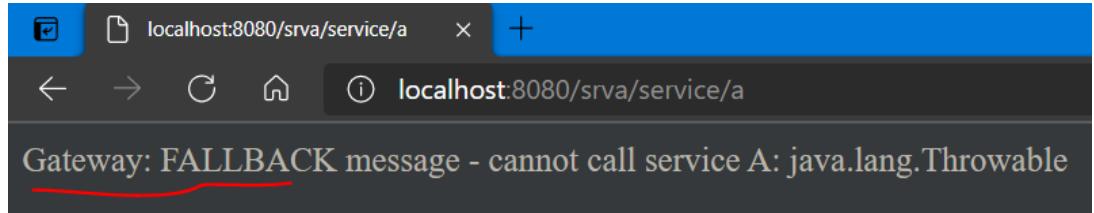


פנינה ל A (ש庫רא ל B)



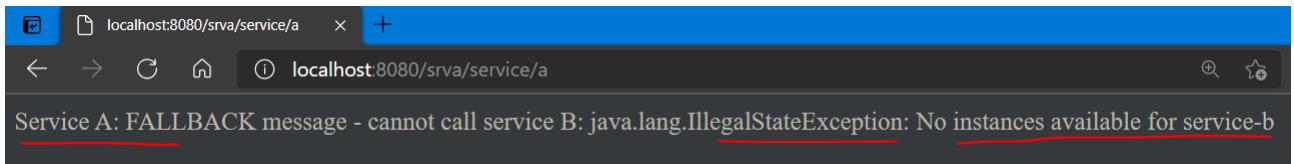
ואם נרפרש נראה גם שיש לנו LB על B

עכשו נראה fallback על ידי זה שנחרוג את כל מופעי B וננסה לפנות שוב ל A
בשניות הראשונות נראה את זה:



ה gateway תפס אותו. כי A חי בסרט. הוא ניסה לקרוא ל B ונפל מבלי להפעיל CB כי עדיין לא מבין אז A. אז A למשהו התרסק על exception של RestTemplae.getForObject וזה אילץ את gateway להפעיל את ה fallback שלו.

אבל אחרי כמה שניות A כבר לא יהיה הסרט וכשהוא יידע שאין B אז סרвис A יטריג את ה CB שלו.
בשניות שנייה ננסה ל B באמצעות RestTemplate עוד לפני שייפול על Exception.



cashsmies על ה CB על ה הופך את המערכת ל bullet proof וזה מה שצורך לעשות

כאשר ביטלתי CB של gateway והרגתי את B קיבלתי בשניות הראשונות נפילה

This application has no configured error view, so you are seeing this as a fallback.

Sun Apr 25 13:04:07 IDT 2021
[3c8601bf] There was an unexpected error (type=Gateway Timeout, status=504).
Response took longer than configured timeout
org.springframework.cloud.gateway.support.TimeoutException
 Suppressed: reactor.core.publisher.FluxOnAssembly\$OnAssemblyException:
Error has been observed at the following site(s):
 |_ checkpoint → org.springframework.cloud.gateway.filter.WeightCalculatorWebFilter [DefaultWebFilterChain]
 |_ checkpoint → org.springframework.boot.actuate.metrics.web.reactive.server.MetricsWebFilter [DefaultWebFilterChain]
 |_ checkpoint → HTTP GET "/srva/service/a" [ExceptionHandlingWebHandler]

Stack trace:

כלומר, gateway קרא ל A ואז A ניסה לקרוא ל B ונפל (קיבל timeout) מבלתי להטrig את CB שלו כי הוא חושב ש B קיים ו CB פועל רק כשהאין מופעים. אז מה שה gateway קיבל זה שגיאה אבל לא יכול היה למכת ל fallback שלו ביטלתי אותו.

אחרי כמה שניות, אותה פעולה בדיקוק. הפעם A כבר התעורר על עצמו והבין שאין B אז באמת הטריג את CB שלו כמו שצריך. אבל הנפילה הזאת של כמה שניות היא לא טובה. וכך אנחנו נותנים gateway לכך שניהה bullet proof על timeout נגיע ל fallback שהוגדרו ב עבור אותו ה route

Service A: FALBACK message - cannot call service B: java.lang.IllegalStateException: No instances available for service-b

כדי לראות בבירור מה קורה שם הוסףי קצר הדפסות:

ב A

```
@RestController
public class ControllerA {

    @Autowired
    private RestTemplate template;

    @HystrixCommand(fallbackMethod = "fallback")
    @GetMapping("/service/a")
    public String handleA() {
        String url = "http://service-b/service/b";

        System.out.println("ControllerA - /service/a attempt to call B");
        try {
            return "Service A calling B: " + template.getForObject(url, String.class);
        } catch (Exception e) {
            System.out.println(e);
            throw e;
        }
    }

    public String fallback(Throwable th) {
        System.out.println("ControllerA - /service/a fallback");
        return "Service A: FALLBACK message - cannot call service B: " + th;
    }
}
```

גמ ב gateway של controller

```
@RestController
public class FallbackController {

    @GetMapping("/fallback/serviceA")
    public String fallbackA(Throwable th) {
        System.out.println("GATEWAY fallback A");
        return "Gateway: FALLBACK message - cannot call service A: " + th;
    }

    @GetMapping("/fallback/serviceB")
    public String fallbackB(Throwable th) {
        System.out.println("GATEWAY fallback B");
        return "Gateway: FALLBACK message - cannot call service B: " + th;
    }
}
```


המשר – Gateway

התמקדות ב regular expression שעושה את הכתיבה מחדש של ה URI

API Gateway – Spring API Gateway

JOHN BRYCE
Leading in IT Education
a matrix company

- Focusing on routing & fallbacks
 - api-gateway.yml routing configuration:

api-gateway.yml

```
spring:  
cloud:  
gateway:  
...  
routes:  
- id: serviceA  
uri: lb://service-A  
predicates:  
- Path=/aService/**  
filters:  
- RewritePath=/aService/(?<segment>.*), /${segment}  
...
```

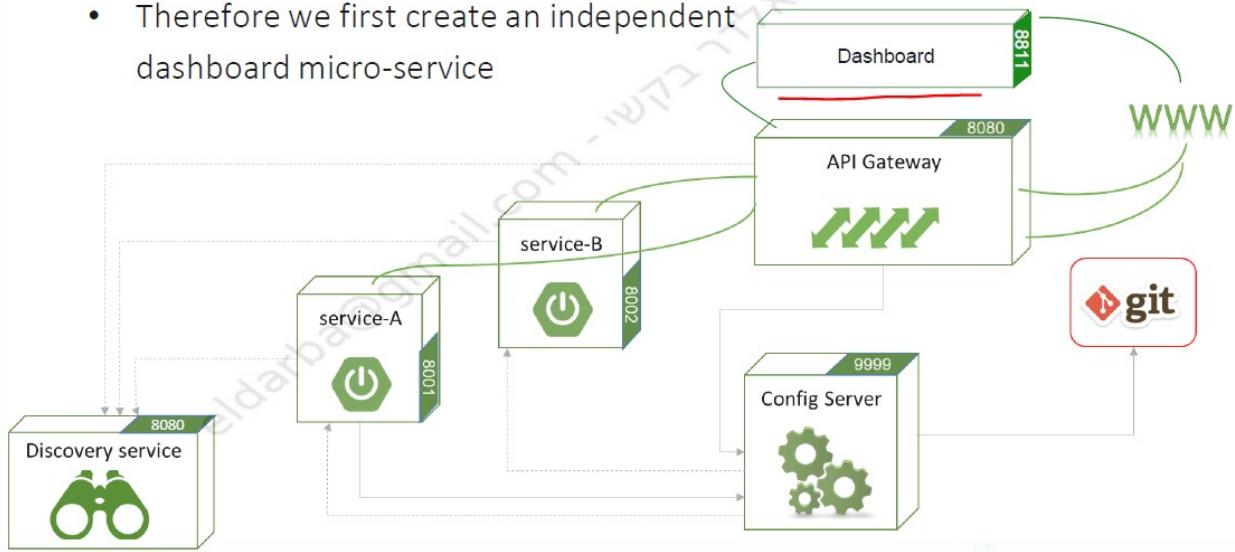
- Calling Path **http://localhost:8080/aService/...** is forwarded with load-balancer logical service URI: **http://service-A/...**
- **lb://service-A/** - causes Ribbon LB to forward **http://...** can be used for a direct address

Here we make sure **/aService/** is removed from the forwarded URI



Hystrix Dashboard

- Hystrix Dashboard is build on MVC
 - Spring APIGateway is reactive (supports Flux/Mono)
 - So we can't include Hystrix Dashboard as part of Gateway build
 - Therefore we first create an independent dashboard micro-service



זהו מיקרו סרвис שיאפשר לאדמיניסטרטורים לראות את ה Hits במערכת.
ברזולוציה נמוכה נשים אותו על ה gateway ונוכל לנטר כניסה עד יציאה. ברזולוציה גבוהה יותר יוכל להציג אותו על מיקרו סרвис ספציפי

Hystrix Dashboard

dependencies



- Adding Dashboard Microservice
 - Create a new project with the following dependencies

```
pom.xml
...
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-hystrix-dashboard</artifactId>
</dependency>...
```

Dashboard

8811

- It may use Client-Configuration in order to load configuration from GIT
- It may use Consul client to get discovered and monitored
- Set service port in application/bootstrap.yml

```
dashboard-service.yml
server:
  port: 8811
```

hystrix dashboard של נטפליקס וcmbwn שהוא צריך actuator כי כל מה שהוא עושה זה בדיקות דופק, והוא מזרים מידע בין שירותים – בודקת את ה health events.

נרים אותו על פורט 8811

בגלל שגם הוא מיקרו שירות אז גם הוא שואב קונפיגורציה מ GIT

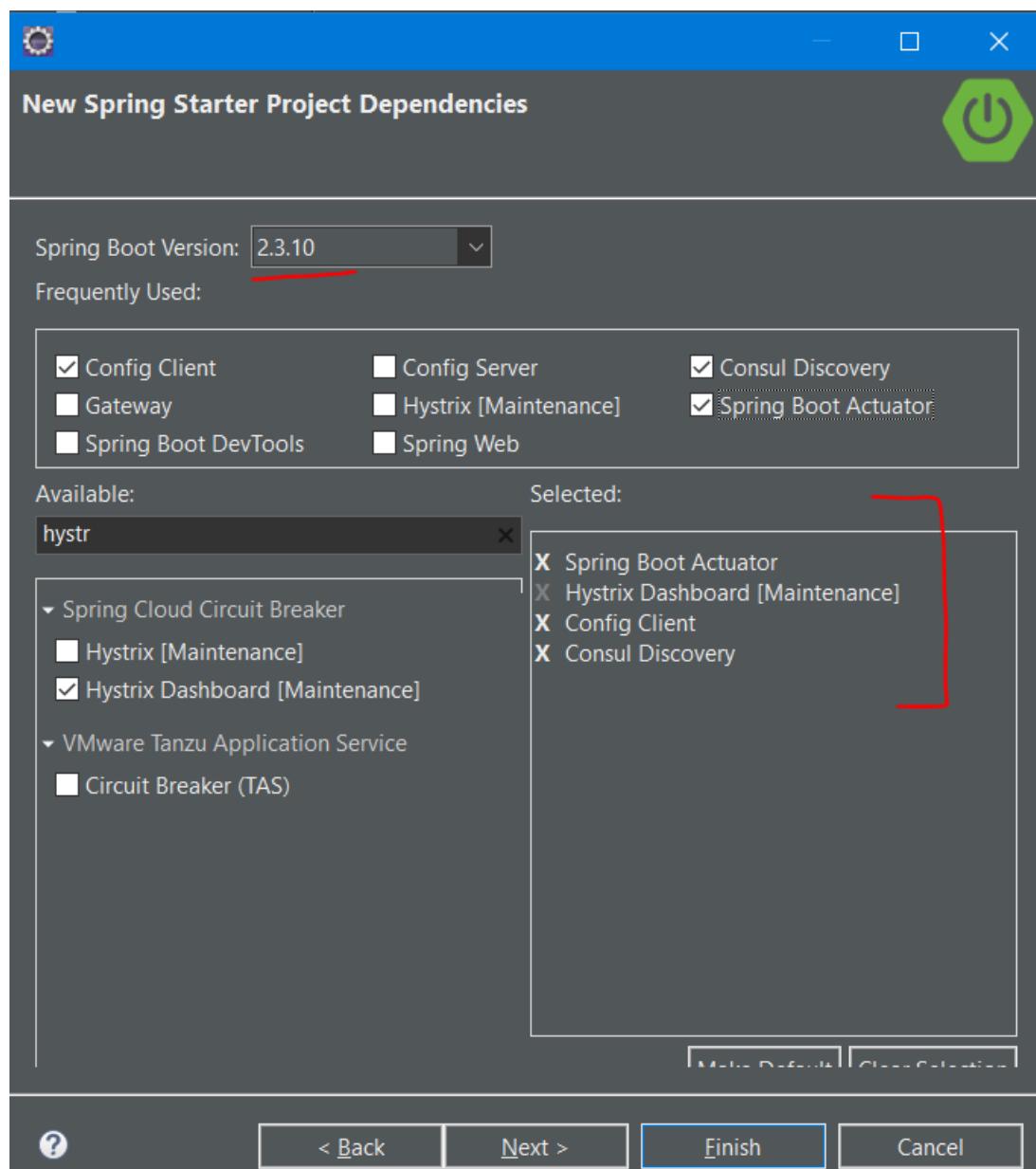
כל מיקרו שירות שנרצה לנטר נעדכן properties לכך ונשים לו :gateway במחלקת קונפיגורציה של EnableHystrix

```
/  
8 @SpringBootApplication  
9 @EnableHystrix // for Hystrix circuit breakers and dashboard  
10 @EnableDiscoveryClient  
11 public class GatewayApplication {  
12  
13     public static void main(String[] args) {  
14         SpringApplication.run(GatewayApplication.class, args);  
15     }  
16 }
```

עדכון ה properties שלו:
זה המאפיינים של gateway מ GIT

```
        enabled: true  
management:  
    security:  
        enabled: false  
endpoints:  
    web:  
        exposure:  
            include: hystrix.stream, info, health
```

יצירת הפרויקט



יש לו כבר קונפיגורציה ב GIT

master [cloud / dashboard.yml](#)

eldarba Add files via upload

1 contributor

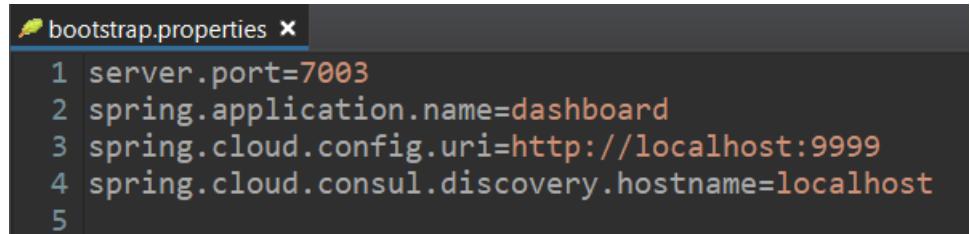
15 lines (14 sloc) | 237 Bytes

```
1 server:
2   port: 8811
3
4   spring:
5     cloud:
6       consul:
7         discovery:
8           instanceId: dashboard
9           health-check-interval: 5s
10      config:
11        management:
12          enabled: true
13      management:
14        security:
15          enabled: false
```

יש לעדכן POM לגרישה מתאימה ולהוסיף:

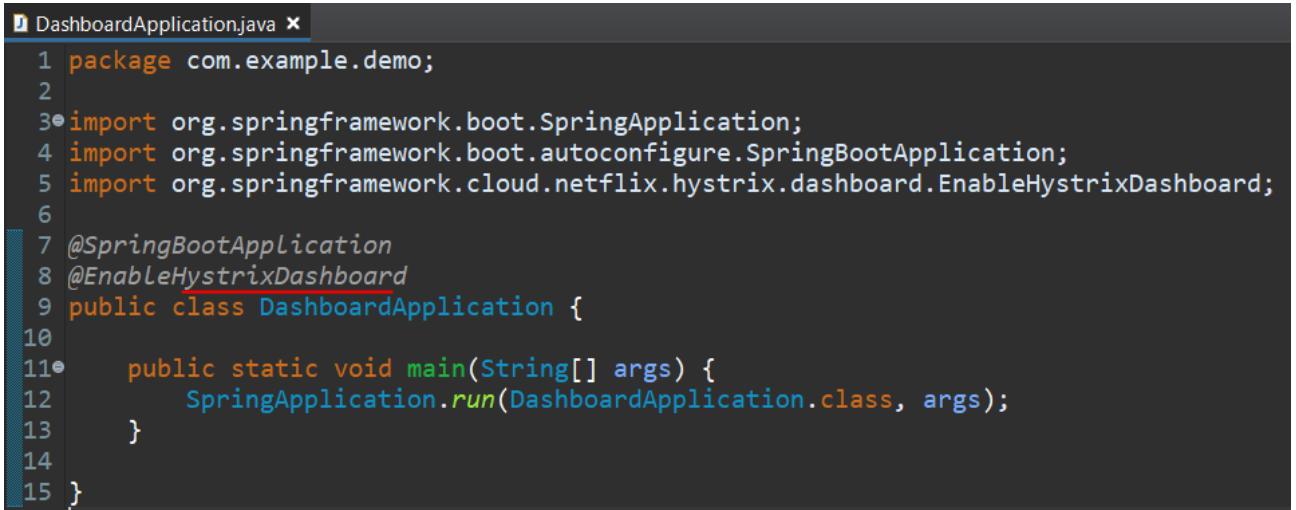
```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-hystrix-dashboard</artifactId>
</dependency>
```

עדכון קובץ properties



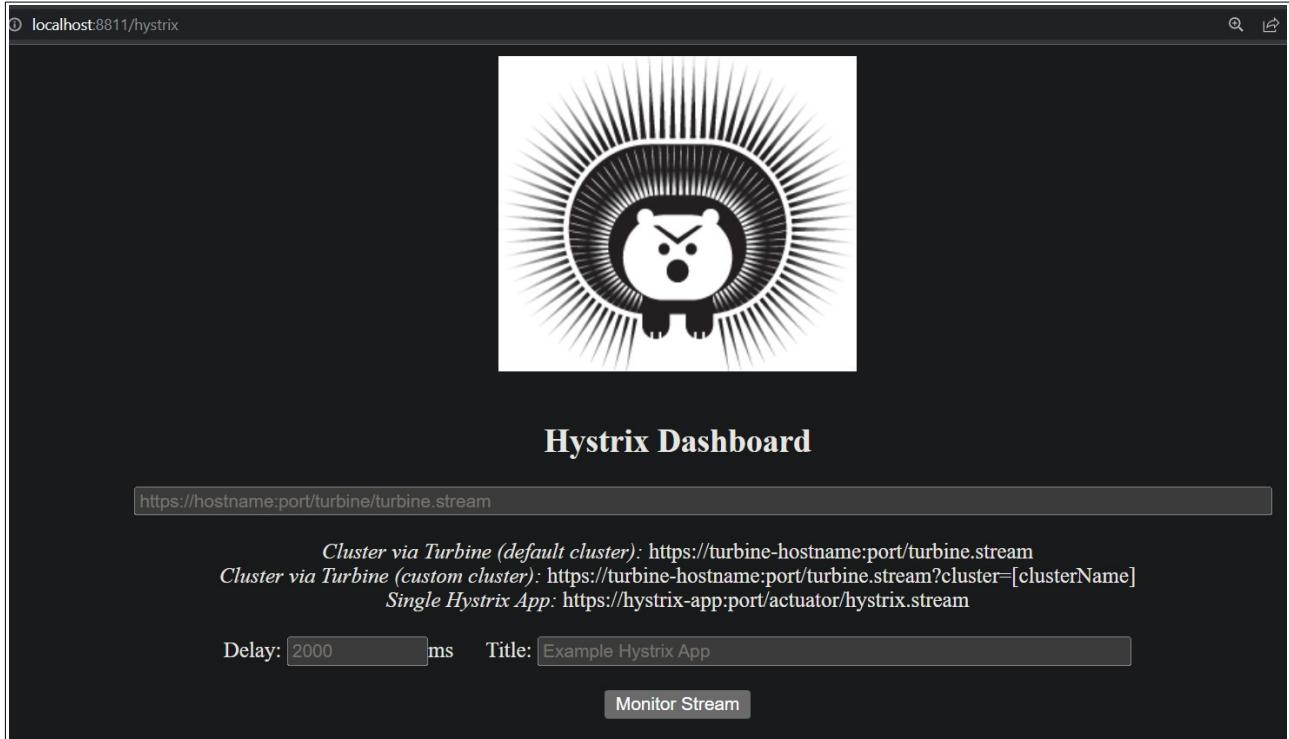
```
bootstrap.properties
1 server.port=7003
2 spring.application.name=dashboard
3 spring.cloud.config.uri=http://localhost:9999
4 spring.cloud.consul.discovery.hostname=localhost
5
```

קונפיגורציה

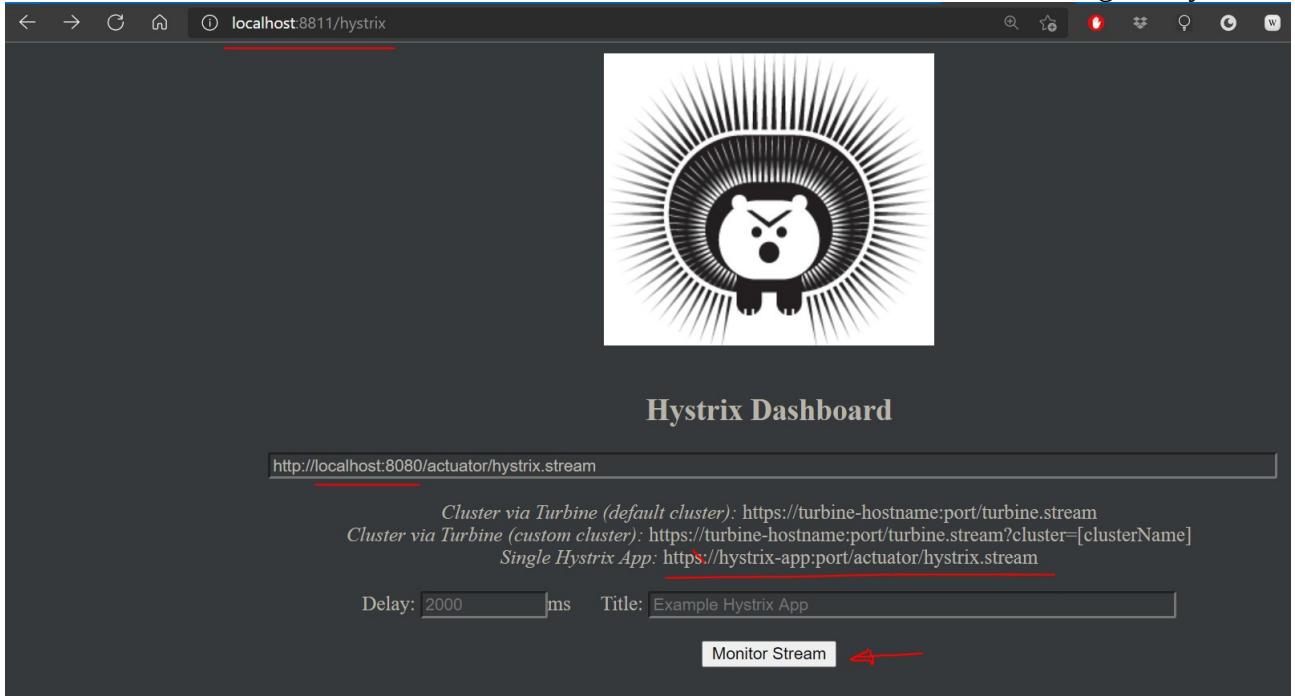


```
DashboardApplication.java
1 package com.example.demo;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.cloud.netflix.hystrix.dashboard.EnableHystrixDashboard;
6
7 @SpringBootApplication
8 @EnableHystrixDashboard
9 public class DashboardApplication {
10
11     public static void main(String[] args) {
12         SpringApplication.run(DashboardApplication.class, args);
13     }
14
15 }
```

ועכשיו הדרבן יהיה זמין ב port GIT קלומר 8811

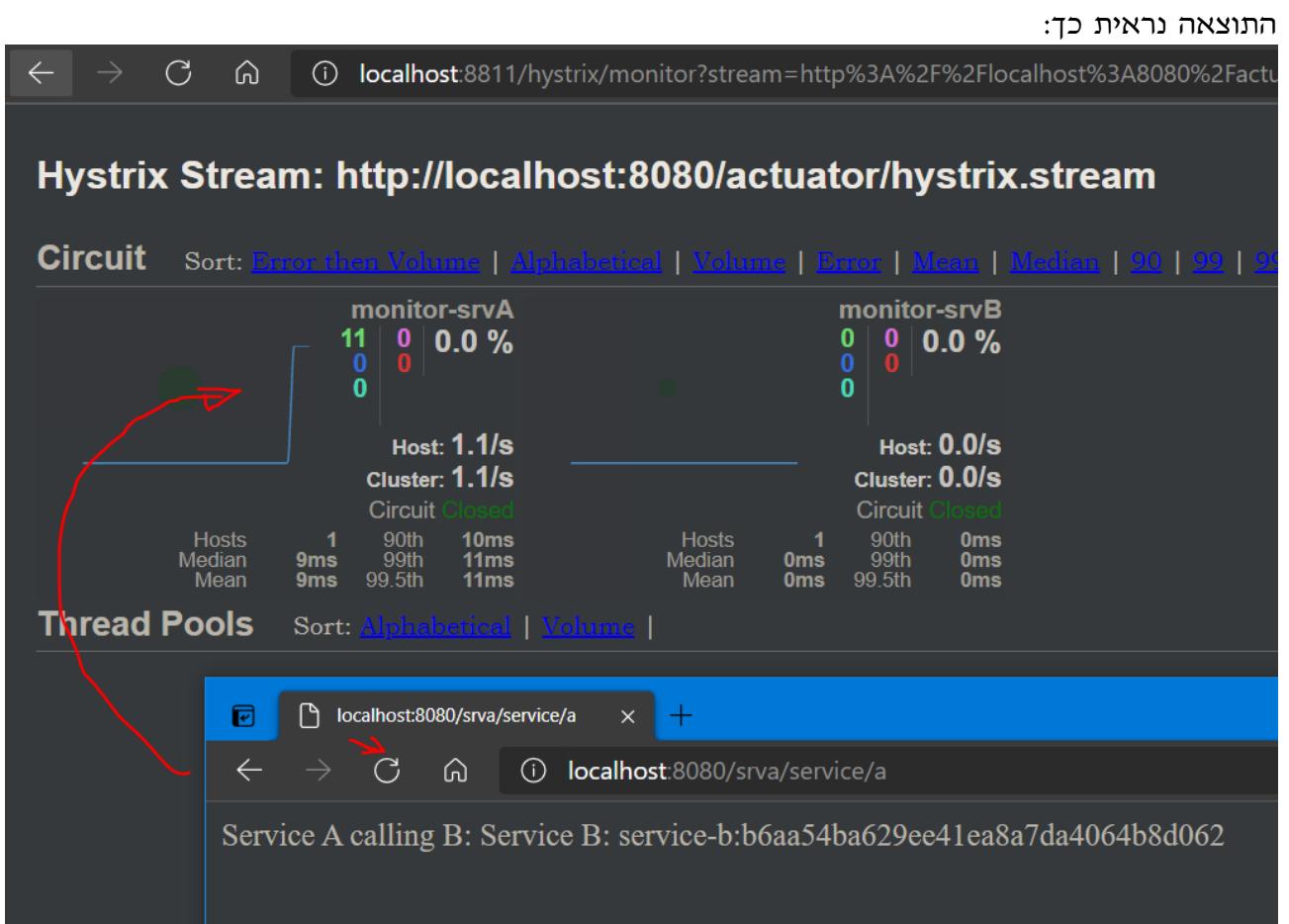


מה עושים עם זה
 עם זה אפשר עכשו לנטר את המיקרו סרבריס ששםנו עליהם enable hystrix ונתנו להם את ה properties המתאימים.
 קלומר gateway



The screenshot shows the Hystrix Dashboard interface. At the top is the Hystrix logo, a white bear with a shocked expression inside a sunburst. Below it is the title "Hystrix Dashboard". A search bar contains the URL "http://localhost:8080/actuator/hystrix.stream". Underneath are three lines of text: "Cluster via Turbine (default cluster): https://turbine-hostname:port/turbine.stream", "Cluster via Turbine (custom cluster): https://turbine-hostname:port/turbine.stream?cluster=[clusterName]", and "Single Hystrix App: https://hystrix-app:port/actuator/hystrix.stream". There are input fields for "Delay: 2000 ms" and "Title: Example Hystrix App", and a "Monitor Stream" button with a red arrow pointing to it.

התוצאה נראה כז:



The screenshot shows the Hystrix Stream dashboard at the URL "http://localhost:8811/hystrix/monitor?stream=http%3A%2F%2Flocalhost%3A8080%2Factuator%2Fhystrix.stream". The title is "Hystrix Stream: http://localhost:8080/actuator/hystrix.stream". It displays two sections: "Circuit" and "Thread Pools".
Circuit:

	monitor-srvA	monitor-srvB
Hosts	11 0 0.0 %	0 0 0.0 %
Cluster	1.1/s	0.0/s
Circuit	Closed	Closed
Hosts	1	1
Median	9ms	90th 0ms
Mean	9ms	99th 0ms
	11ms	99.5th 11ms

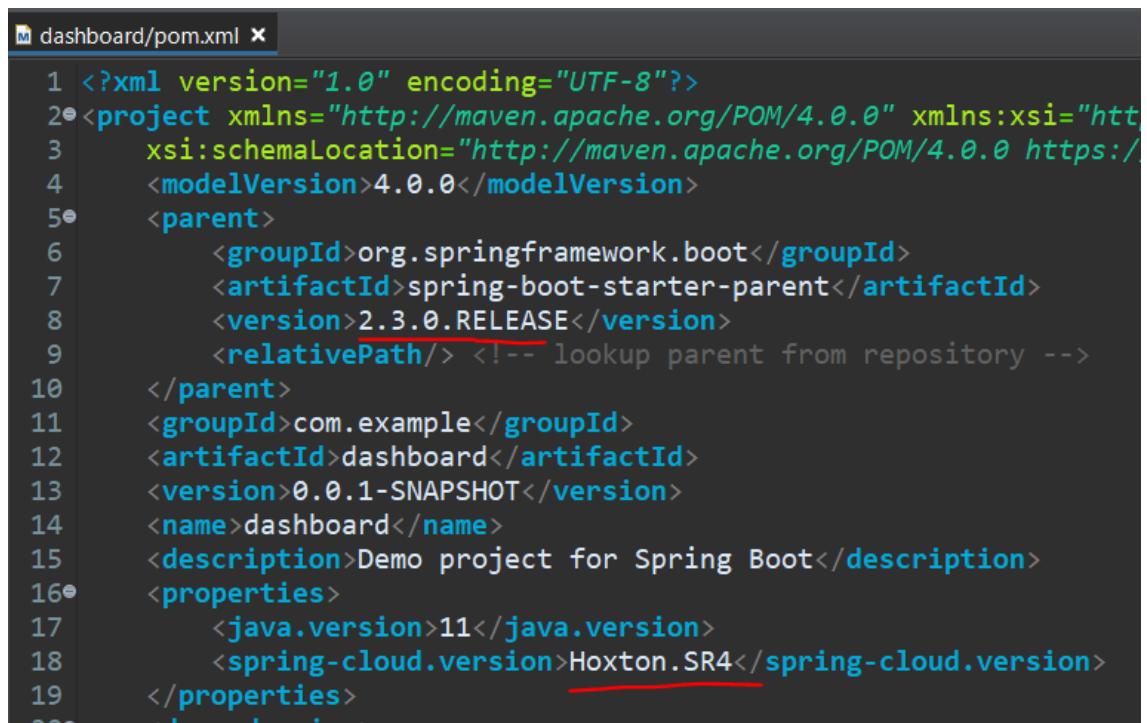
Thread Pools:

	srva	srvb
Hosts	localhost:8080/srva/service/a	localhost:8080/srvb/service/b
Median	9ms	9ms
Mean	9ms	9ms

A red arrow points from the "Circuit" section to the browser's address bar, which shows "localhost:8080/srva/service/a". Another red arrow points from the "Thread Pools" section to the browser's address bar, which shows "localhost:8080/srvb/service/b".

כדי שיוציאו שני הגרפים יש לנקת נתיבים srva, srvb

אם לא עובד אז לעדכן POM על מנת לשנמך את dashboard לגרסה:



The screenshot shows a code editor window with the title "dashboard/pom.xml". The content of the file is as follows:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
3   <modelVersion>4.0.0</modelVersion>
4   <parent>
5     <groupId>org.springframework.boot</groupId>
6     <artifactId>spring-boot-starter-parent</artifactId>
7     <version>2.3.0.RELEASE</version>
8     <relativePath/> <!-- lookup parent from repository -->
9   </parent>
10  <groupId>com.example</groupId>
11  <artifactId>dashboard</artifactId>
12  <version>0.0.1-SNAPSHOT</version>
13  <name>dashboard</name>
14  <description>Demo project for Spring Boot</description>
15  <properties>
16    <java.version>11</java.version>
17    <spring-cloud.version>Hoxton.SR4</spring-cloud.version>
18  </properties>
19  <!-- ... -->
```

אילו רצינו לנטר את A למשל ישרות אז גם עבورو היינו מעדכנים ומחלקת קונפיגורציה:

The screenshot shows two tabs in a code editor: 'bootstrap.properties' and 'ServiceAApplication.java'. The 'bootstrap.properties' tab contains configuration properties for Service A, including port, application name, and discovery settings. The 'ServiceAApplication.java' tab contains the Java code for the application, which includes annotations for Spring Boot, discovery client, circuit breaker, and Hystrix.

```
bootstrap.properties
1 server.port=7001
2 spring.application.name=service-a
3
4 # spring.config.import=configserver:http://localhost:9999
5 # use this option if you want the instance to start with local configuration
6 #spring.config.import=optional:configserver:http://localhost:9999
7 # older syntax for config server uri
8 spring.cloud.config.uri=http://localhost:9999
9
10 spring.cloud.consul.discovery.hostname=localhost
11
12 # enable hystrix dashboard from local config
13 management.endpoints.web.exposure.include=hystrix.stream, info, health
14
```

```
ServiceAApplication.java
1 package app.core;
2
3 import org.springframework.boot.SpringApplication;
4
5 @SpringBootApplication
6 @EnableDiscoveryClient
7 @EnableCircuitBreaker
8 @EnableHystrix
9
10 public class ServiceAApplication {
11 }
```

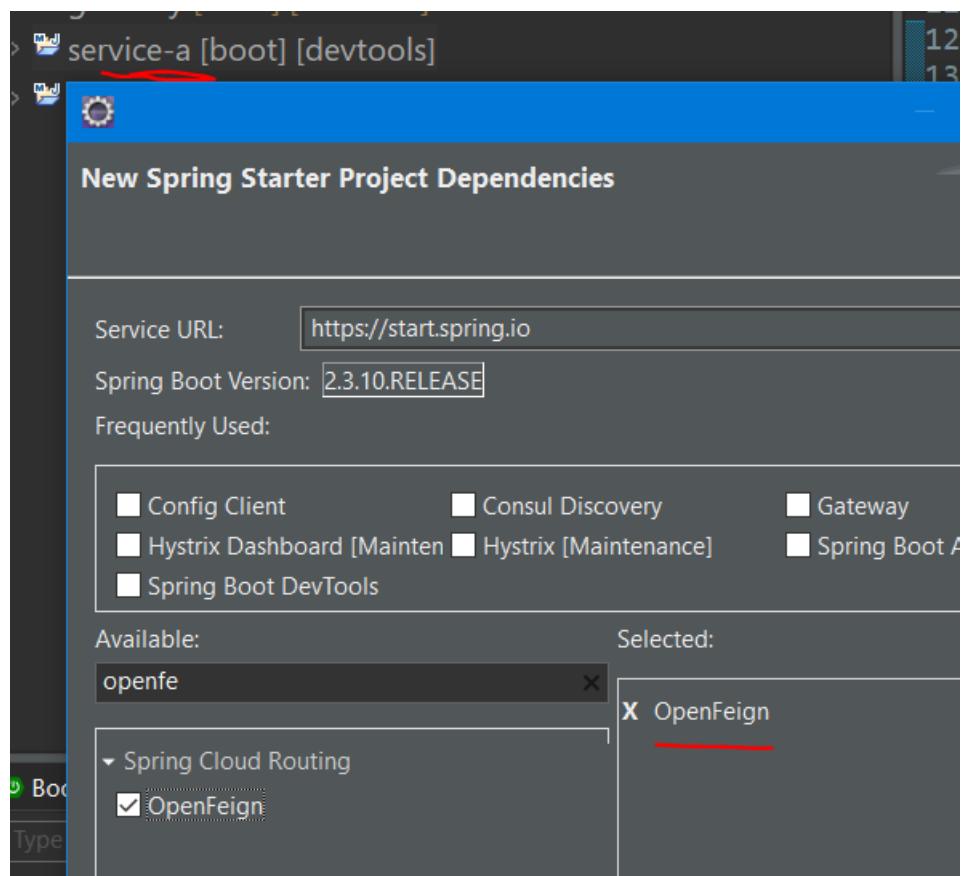
The screenshot shows the same code editor interface as above, but the tabs are swapped. Now 'ServiceAApplication.java' is the active tab, and 'bootstrap.properties' is the tab shown in the preview pane. The Java code remains the same, defining the application's main class and its dependencies.

```
bootstrap.properties
1 server.port=7001
2 spring.application.name=service-a
3
4 # spring.config.import=configserver:http://localhost:9999
5 # use this option if you want the instance to start with local configuration
6 #spring.config.import=optional:configserver:http://localhost:9999
7 # older syntax for config server uri
8 spring.cloud.config.uri=http://localhost:9999
9
10 spring.cloud.consul.discovery.hostname=localhost
11
12 # enable hystrix dashboard from local config
13 management.endpoints.web.exposure.include=hystrix.stream, info, health
14
```

```
ServiceAApplication.java
1 package app.core;
2
3 import org.springframework.boot.SpringApplication;
4
5 @SpringBootApplication
6 @EnableDiscoveryClient
7 @EnableCircuitBreaker
8 @EnableHystrix
9
10 public class ServiceAApplication {
11 }
```

Feign Clients

נוסיף ל OpenFeign



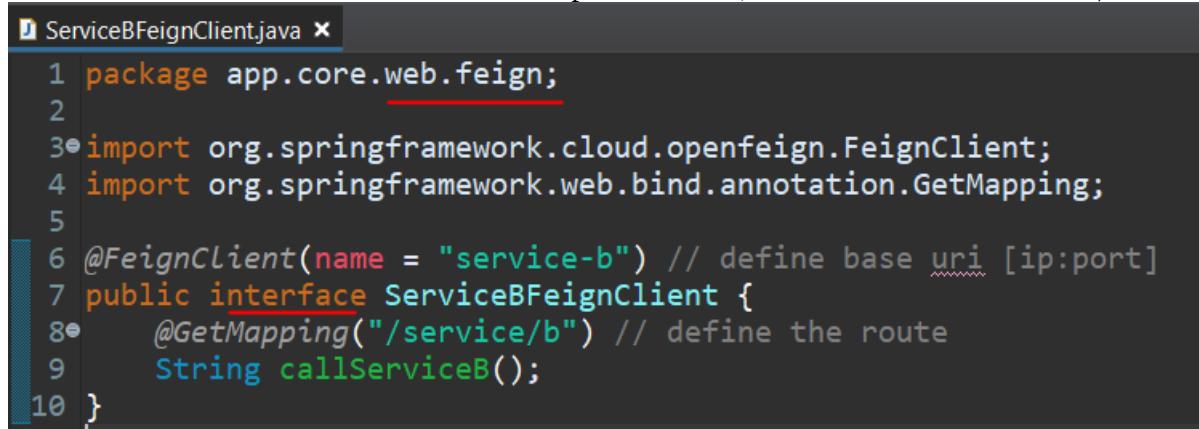
Declarative REST Client. OpenFeign creates a dynamic implementation of an interface decorated with JAX-RS or Spring MVC annotations.

References

[Spring Boot Reference Doc](#)

Feign זה לקוחות דקלרטיבי של שירותים רשת. כך קל יותר לכתוב לקוחות לשירותים רשת. כל מה שצריך לעשות הוא לכתוב ממשק ולהוסיף לו אנטווציות "יעודיות".
בשימושים כולל בתוכו Spring Cloud LoadBalancer כדי לספק לקוחות HTTP עם LB כמשתמשים ב-Feign.

עכשו נכתב ממשק שייצג لكוח של השירות B.
 נסיף לממשק אנותציה שתגדיר מיהו השירות אליו פונים – להגדרת ה .base uri
 כתוב מトודה שתייצג קריאה ל endpoint מסוים בשירות – עם אנותציה להגדרת ה route
 ספרינט תנת לנו מימוש שלו שיבצע את קריאתה rest template בшибילנו



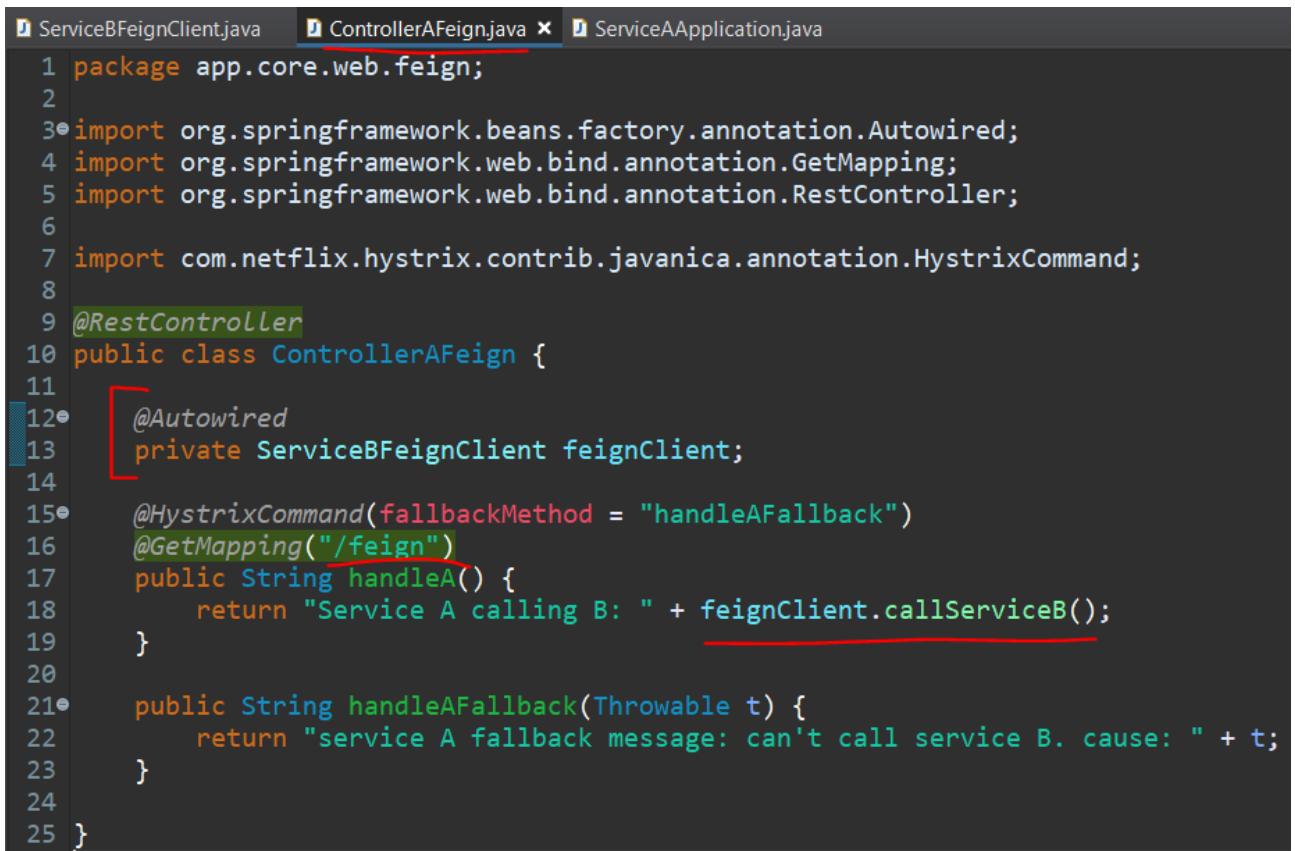
```

1 package app.core.web.feign;
2
3 import org.springframework.cloud.openfeign.FeignClient;
4 import org.springframework.web.bind.annotation.GetMapping;
5
6 @FeignClient(name = "service-b") // define base uri [ip:port]
7 public interface ServiceBFeignClient {
8     @GetMapping("/service/b") // define the route
9     String callServiceB();
10 }

```

ונקבל מימוש של لكוח שפונה ל route המבוקש בשירות B

נסכפל את ה controller אבל הפעם נבצע את הקריאה באמצעות הממשק שהגדכנו

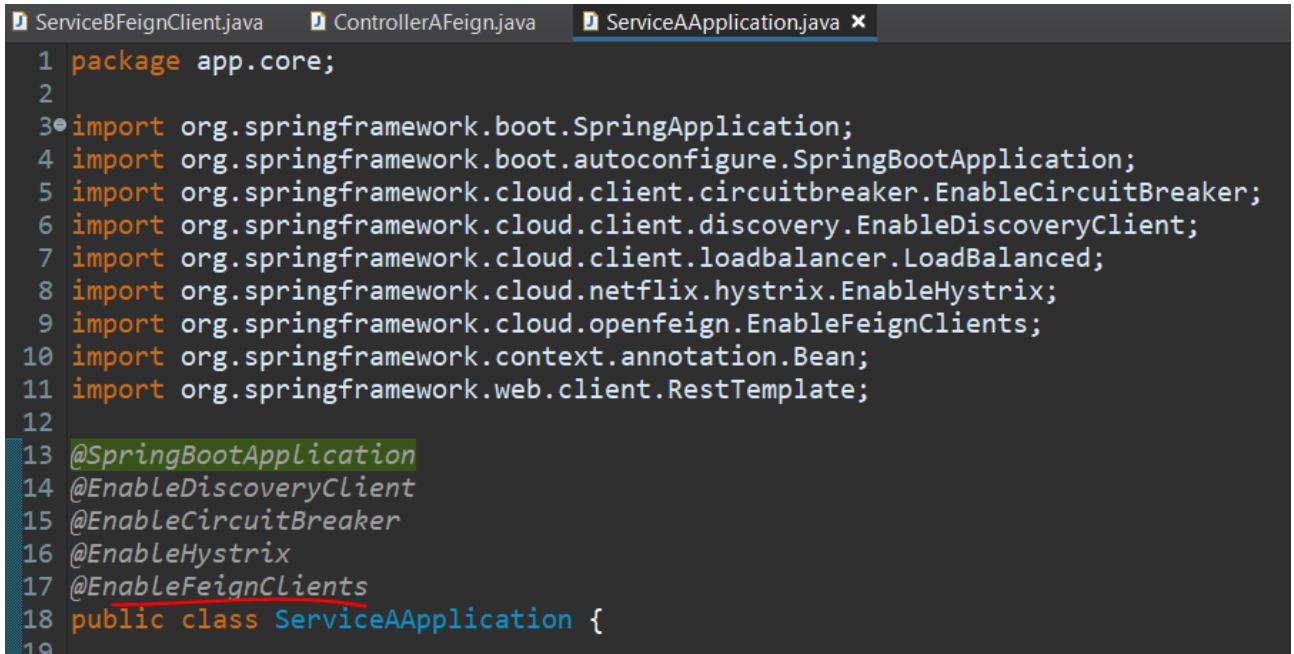


```

1 package app.core.web.feign;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.web.bind.annotation.GetMapping;
5 import org.springframework.web.bind.annotation.RestController;
6
7 import com.netflix.hystrix.contrib.javanica.annotation.HystrixCommand;
8
9 @RestController
10 public class ControllerAFeign {
11
12     @Autowired
13     private ServiceBFeignClient feignClient;
14
15     @HystrixCommand(fallbackMethod = "handleAFallback")
16     @GetMapping("/feign")
17     public String handleA() {
18         return "Service A calling B: " + feignClient.callServiceB();
19     }
20
21     public String handleAFallback(Throwable t) {
22         return "service A fallback message: can't call service B. cause: " + t;
23     }
24 }

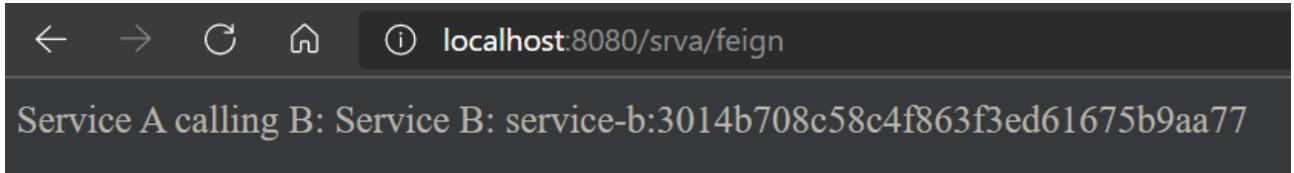
```

ולא לשכוח קונפיגורציה (אחרת זה לא עולה בכלל)



```
1 package app.core;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.cloud.client.circuitbreaker.EnableCircuitBreaker;
6 import org.springframework.cloud.client.discovery.EnableDiscoveryClient;
7 import org.springframework.cloud.client.loadbalancer.LoadBalanced;
8 import org.springframework.cloud.netflix.hystrix.EnableHystrix;
9 import org.springframework.cloud.openfeign.EnableFeignClients;
10 import org.springframework.context.annotation.Bean;
11 import org.springframework.web.client.RestTemplate;
12
13 @SpringBootApplication
14 @EnableDiscoveryClient
15 @EnableCircuitBreaker
16 @EnableHystrix
17 @EnableFeignClients
18 public class ServiceAApplication {
19 }
```

עכשו נפנה ל היזה כרגע דרך gateway שידוע לנוות את A לפי התחלתית srva של אחריה יוכל לציין כל endpoint שהגדנו שם.



להשוואה נפתח את שני controllers ונראה איך בחדש חסכנו את הצורך ב RestTemplate ו שימושו במתודות שלה.