Research & Development Team

PnP Solution Co., Ltd.

Microservice For Management

www.pnpsw.com

sommai.k@gmail.com

081-754-4663

Lineid : sommaik

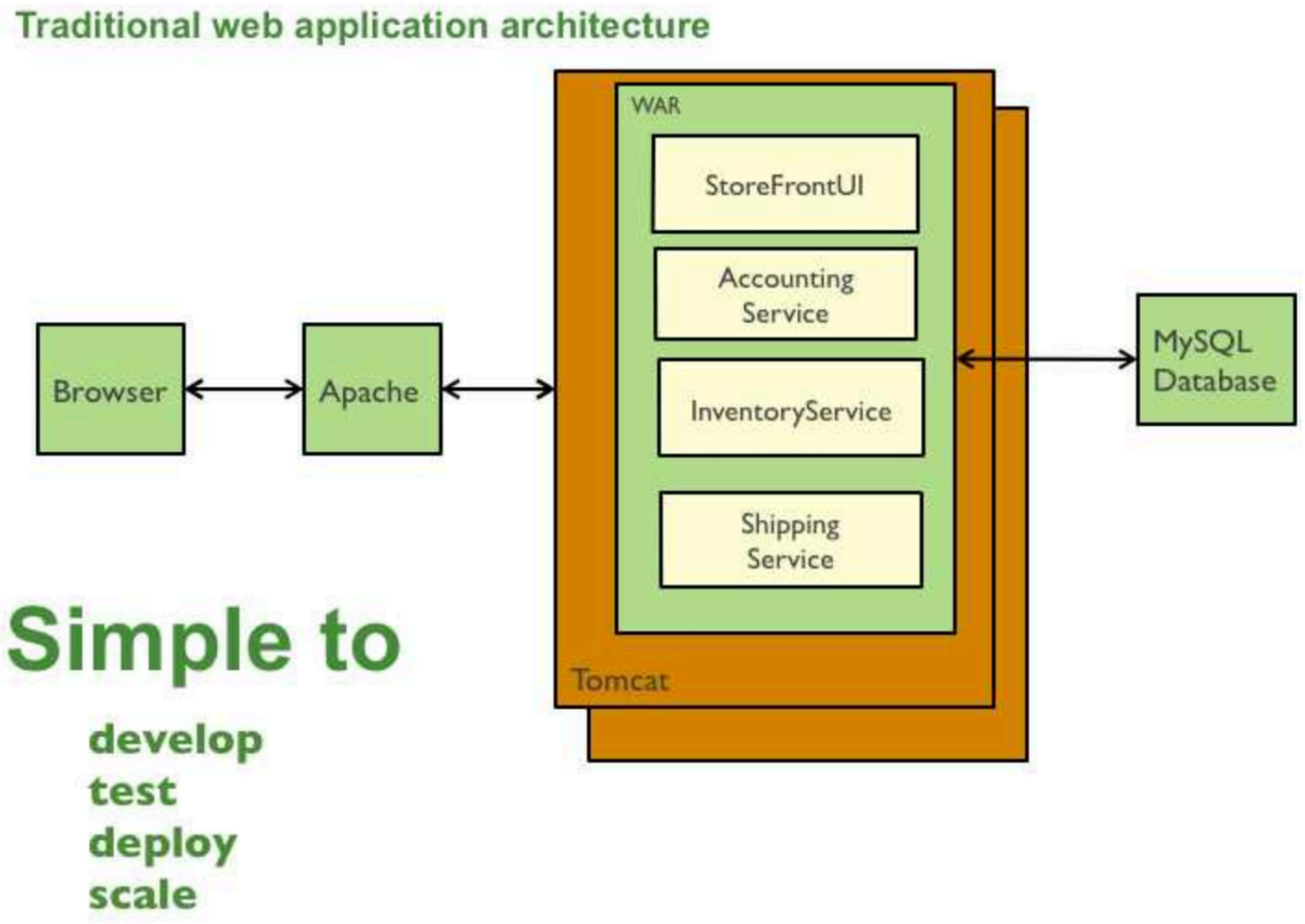
Medium: @sommaikrangpanich

IMC Institute

Introduction to

MICROSERVICE ARCHITECTURE

Monolithic Architecture



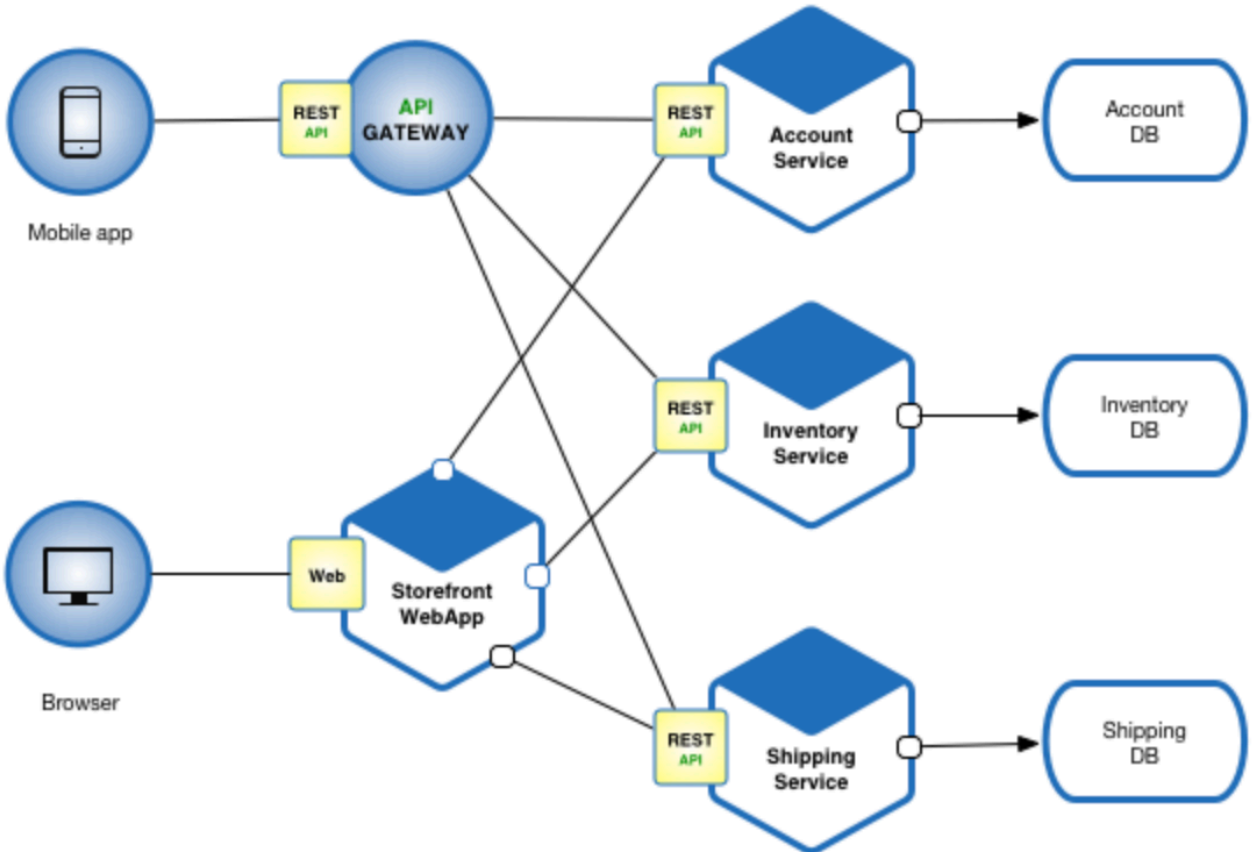
Benefits

- Simple to develop - the goal of current development tools and IDEs is to support the development of monolithic applications
- Simple to deploy - you simply need to deploy the WAR file (or directory hierarchy) on the appropriate runtime
- Simple to scale - you can scale the application by running multiple copies of the application behind a load balancer

Drawbacks

- The large monolithic code base intimidates developers, especially ones who are new to the team.
- Overloaded IDE
- Overloaded web container
- Continuous deployment is difficult
- Scaling the application can be difficult
- Obstacle to scaling development
- Requires a long-term commitment to a technology stack

Microservice Architecture



Benefits

- Enables the continuous delivery and deployment of large, complex applications.
  - Improved maintainability - each service is relatively small and so is easier to understand and change
  - Better testability - services are smaller and faster to test
  - Better deployability - services can be deployed independently
  - It enables you to organize the development effort around multiple, autonomous teams. Each (so called two pizza) team owns and is responsible for one or more services. Each team can develop, test, deploy and scale their services independently of all of the other teams.

Benefits #2

- Each microservice is relatively small:
  - Easier for a developer to understand
  - The IDE is faster making developers more productive
  - The application starts faster, which makes developers more productive, and speeds up deployments
- Improved fault isolation. For example, if there is a memory leak in one service then only that service will be affected. The other services will continue to handle requests. In comparison, one misbehaving component of a monolithic architecture can bring down the entire system.
- Eliminates any long-term commitment to a technology stack. When developing a new service you can pick a new technology stack. Similarly, when making major changes to an existing service you can rewrite it using a new technology stack.

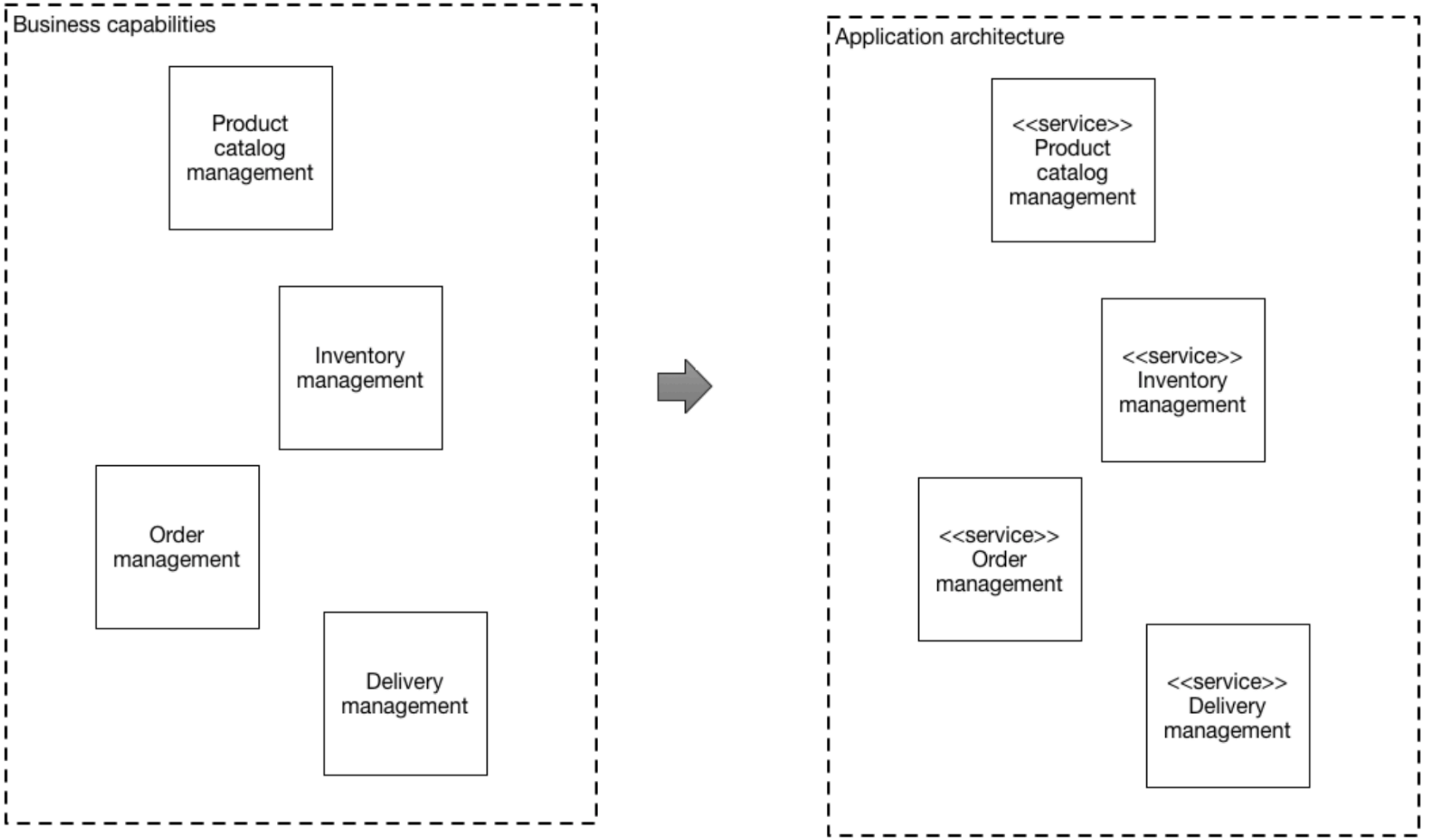
Drawbacks

- Developers must deal with the additional complexity of creating a distributed system:
  - Developers must implement the inter-service communication mechanism and deal with partial failure
  - Implementing requests that span multiple services is more difficult
  - Testing the interactions between services is more difficult
  - Implementing requests that span multiple services requires careful coordination between the teams
  - Developer tools/IDEs are oriented on building monolithic applications and don't provide explicit support for developing distributed applications.
- Deployment complexity. In production, there is also the operational complexity of deploying and managing a system comprised of many different services.

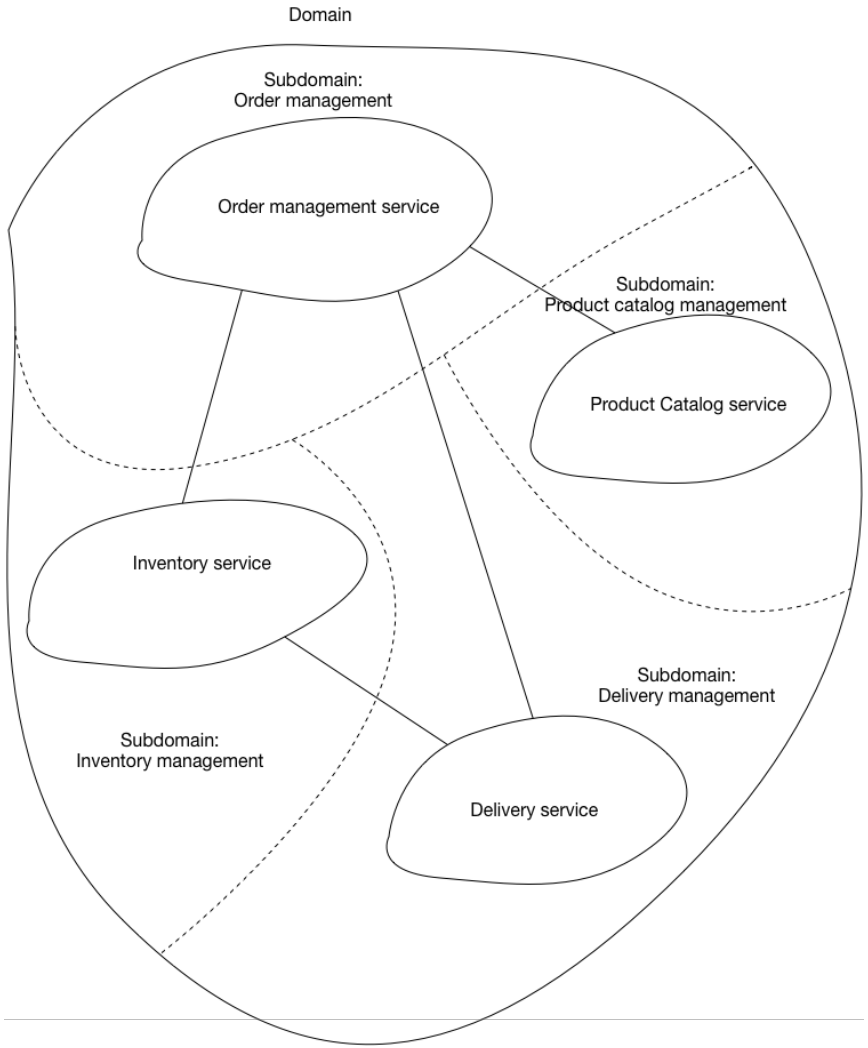
How to decompose

- Decompose by business capability and define services corresponding to business capabilities.
- Decompose by domain-driven design subdomain.
- Decompose by verb or use case and define services that are responsible for particular actions. e.g. a Shipping Service that's responsible for shipping complete orders.
- Decompose by by nouns or resources by defining a service that is responsible for all operations on entities/resources of a given type. e.g. an Account Service that is responsible for managing user accounts.

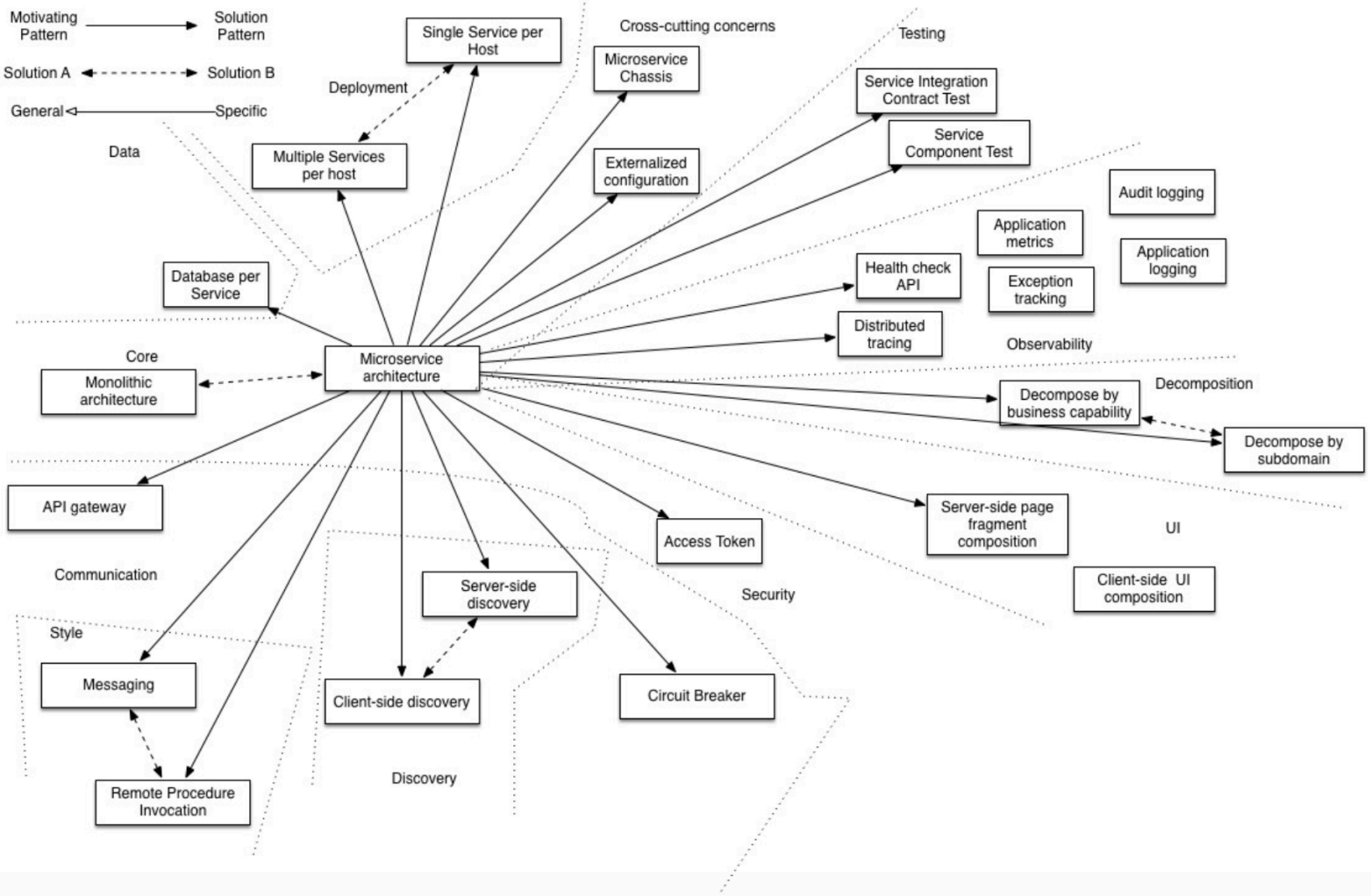
Decompose by business capability



Decompose by domain-driven design subdomain



Related patterns



Demo Microservice with

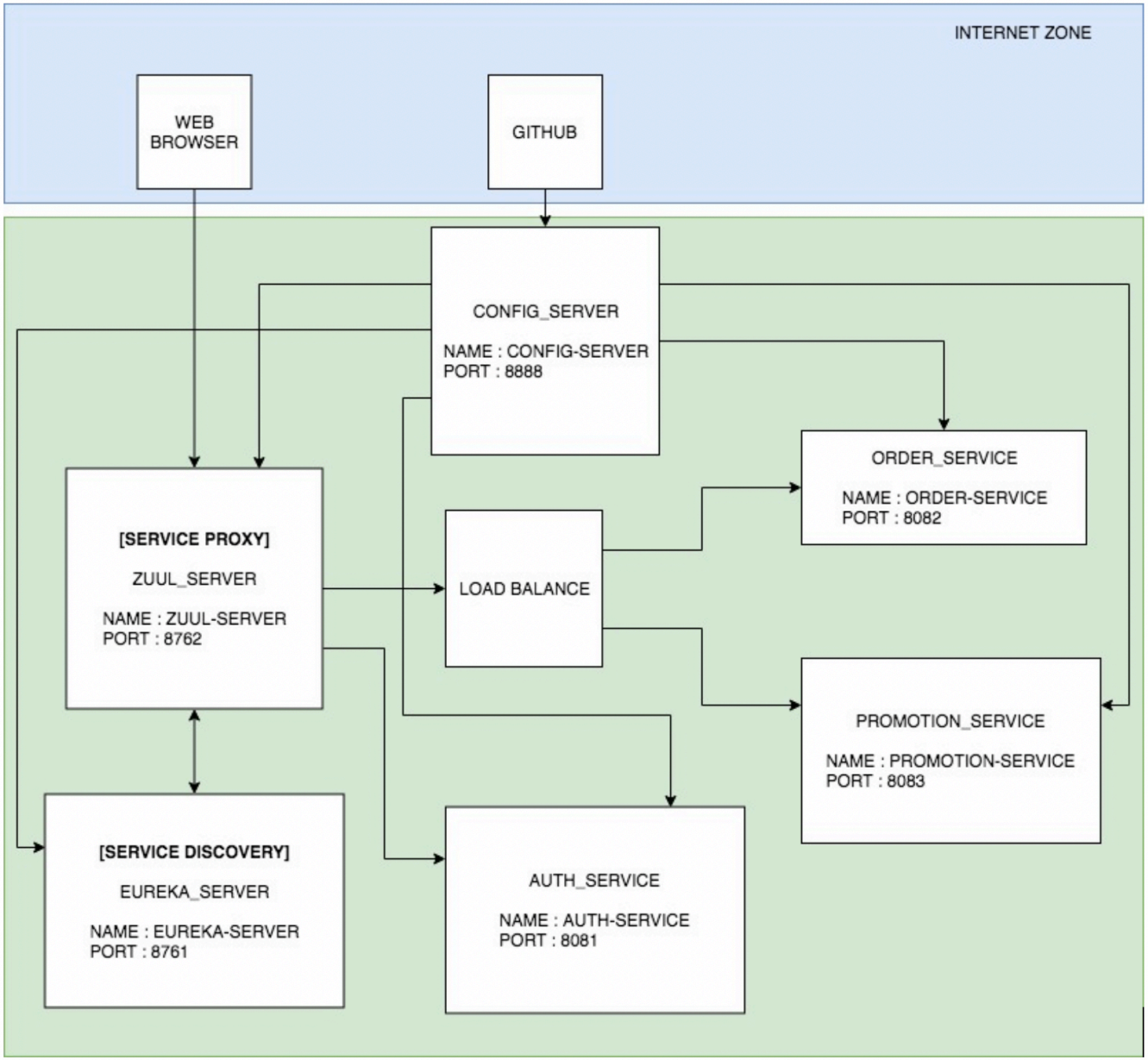
SPRING BOOT

Introduction

- Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that you can "just run".
- We take an opinionated view of the Spring platform and third-party libraries so you can get started with minimum fuss.
- Most Spring Boot applications need very little Spring configuration.

Features

- Create stand-alone Spring applications
- Embed Tomcat, Jetty or Undertow directly (no need to deploy WAR files)
- Provide opinionated 'starter' dependencies to simplify your build configuration
- Automatically configure Spring and 3rd party libraries whenever possible
- Provide production-ready features such as metrics, health checks and externalized configuration
- Absolutely no code generation and no requirement for XML configuration


INTERNET ZONE
WEB BROWSER
GITHUB
CONFIG_SERVER
NAME : CONFIG-SERVER
PORT : 8888
ORDER_SERVICE
NAME : ORDER-SERVICE
PORT : 8082
[SERVICE PROXY]
ZUUL_SERVER
NAME : ZUUL-SERVER
PORT : 8762
LOAD BALANCE
PROMOTION_SERVICE
NAME : PROMOTION-SERVICE
PORT : 8083
[SERVICE DISCOVERY]
EUREKA_SERVER
NAME : EUREKA-SERVER
PORT : 8761
AUTH_SERVICE
NAME : AUTH-SERVICE
PORT : 8081

Rest API with

SPRING REST

REST API

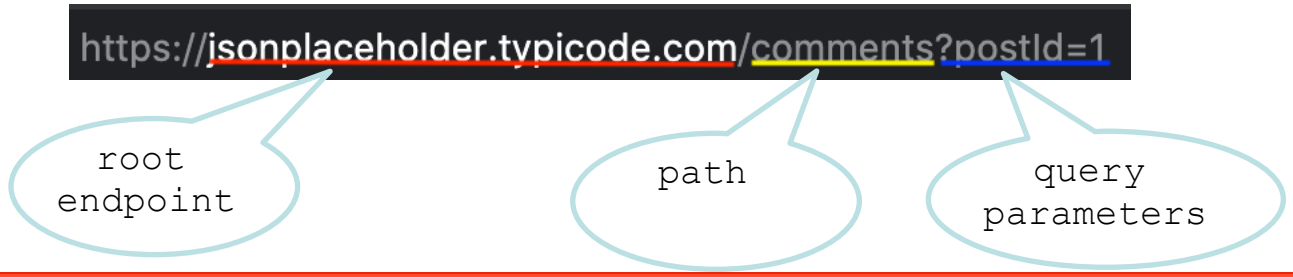
- An API is an application programming interface.
- It is a set of rules that allow programs to talk to each other.
- The developer creates the API on the server and allows the client to talk to it.
- REST determines how the API looks like.
- It stands for “Representational State Transfer”.
- It is a set of rules that developers follow when they create their API.
- One of these rules states that you should be able to get a piece of data (called a resource) when you link to a specific URL.
- Each URL is called a **request** while the data sent back to you is called a **response**.

The Anatomy Of A Request

- **The endpoint (or route)** is the url you request for.
- **The method** is the type of request you send to the server.
- **The headers** are used to provide information to both the client and server.
- **The data (or body)** contains information you want to be sent to the server. This option is only used with POST, PUT, PATCH requests.

The endpoint

- The **root-endpoint** is the starting point of the API you're requesting from.
- The **path** determines the resource you're requesting for.
- The **query parameters** give you the option to modify your request with key-value pairs.
- They always begin with a question mark (?).
- Each parameter pair is then separated with an ampersand (&).



The method

- GET
- POST
- PUT and PATCH
- DELETE
- These methods provide meaning for the request you're making. They are used to perform four possible actions: Create, Read, Update and Delete (CRUD).

GET

- This request is used to get a resource from a server.
- If you perform a `GET` request, the server looks for the data you requested and sends it back to you.
- In other words, a `GET` request performs a `READ` operation.
- This is the default request method.

POST

- This request is used to create a new resource on a server.
- If you perform a `POST` request, the server creates a new entry in the database and tells you whether the creation is successful.
- In other words, a `POST` request performs an `CREATE` operation.

PUT and PATCH

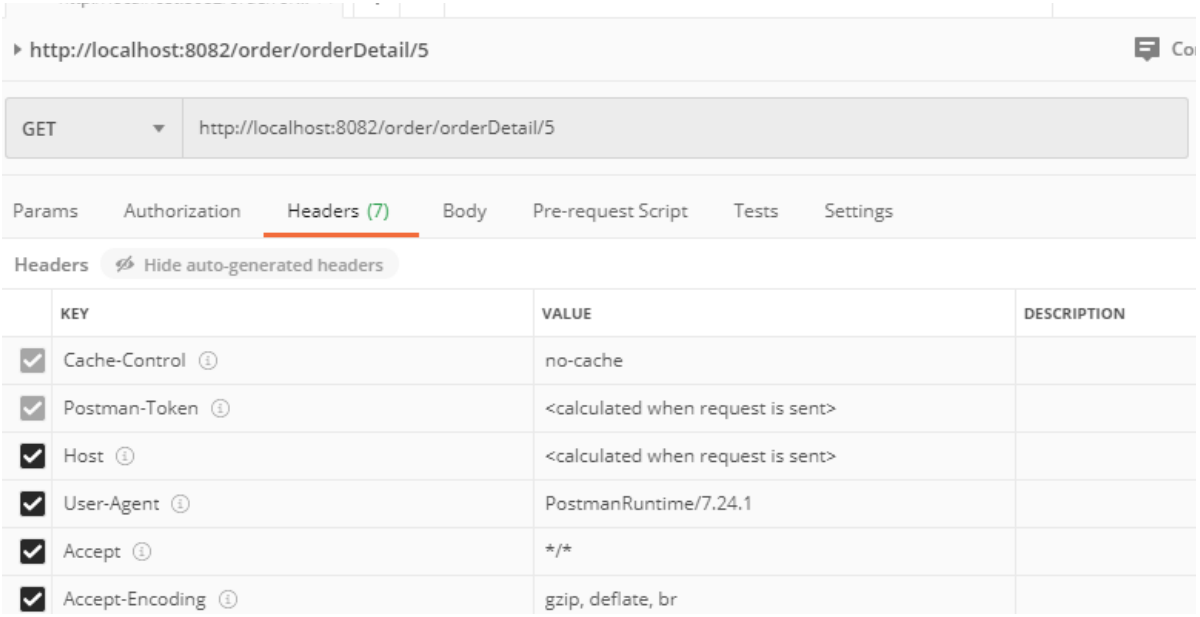
- These two requests are used to update a resource on a server.
- If you perform a `PUT` or `PATCH` request, the server updates an entry in the database and tells you whether the update is successful.
- In other words, a `PUT` or `PATCH` request performs an `UPDATE` operation.

DELETE

- This request is used to delete a resource from a server.
- If you perform a `DELETE` request, the server deletes an entry in the database and tells you whether the deletion is successful.
- In other words, a `DELETE` request performs a `DELETE` operation.

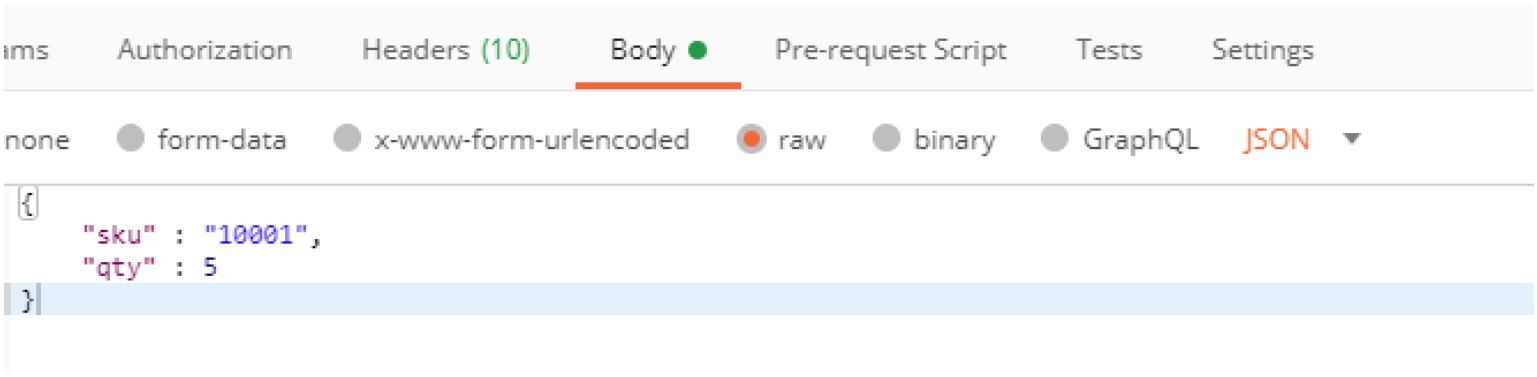
The Headers

- Headers are used to provide information to both the client and server.
- It can be used for many purposes, such as authentication and providing information about the body content.
- **HTTP Headers** are property-value pairs that are separated by a colon.



The Data

- The data (sometimes called “body” or “message”) contains information you want to be sent to the server.
- This option is only used with POST, PUT, PATCH requests.



Data repository with

SPRING DATA JPA

Repositories

- The central interface in the Spring Data repository abstraction is Repository.
- It takes the domain class to manage as well as the ID type of the domain class as type arguments.
- This interface acts primarily as a marker interface to capture the types to work with and to help you to discover interfaces that extend this one.
- The CrudRepository provides sophisticated CRUD functionality for the entity class that is being managed.

Query Creation

- The query builder mechanism built into Spring Data repository infrastructure is useful for building constraining queries over entities of the repository.
- The mechanism strips the prefixes find…By, read…By, query…By, count…By, and get…By from the method and starts parsing the rest of it.
- The introducing clause can contain further expressions, such as aDistinctto set a distinct flag on the query to be created.
- However, the firstBy acts as delimiter to indicate the start of the actual criteria.
- At a very basic level, you can define conditions on entity properties and concatenate them with And andOr.

Limiting Query Results

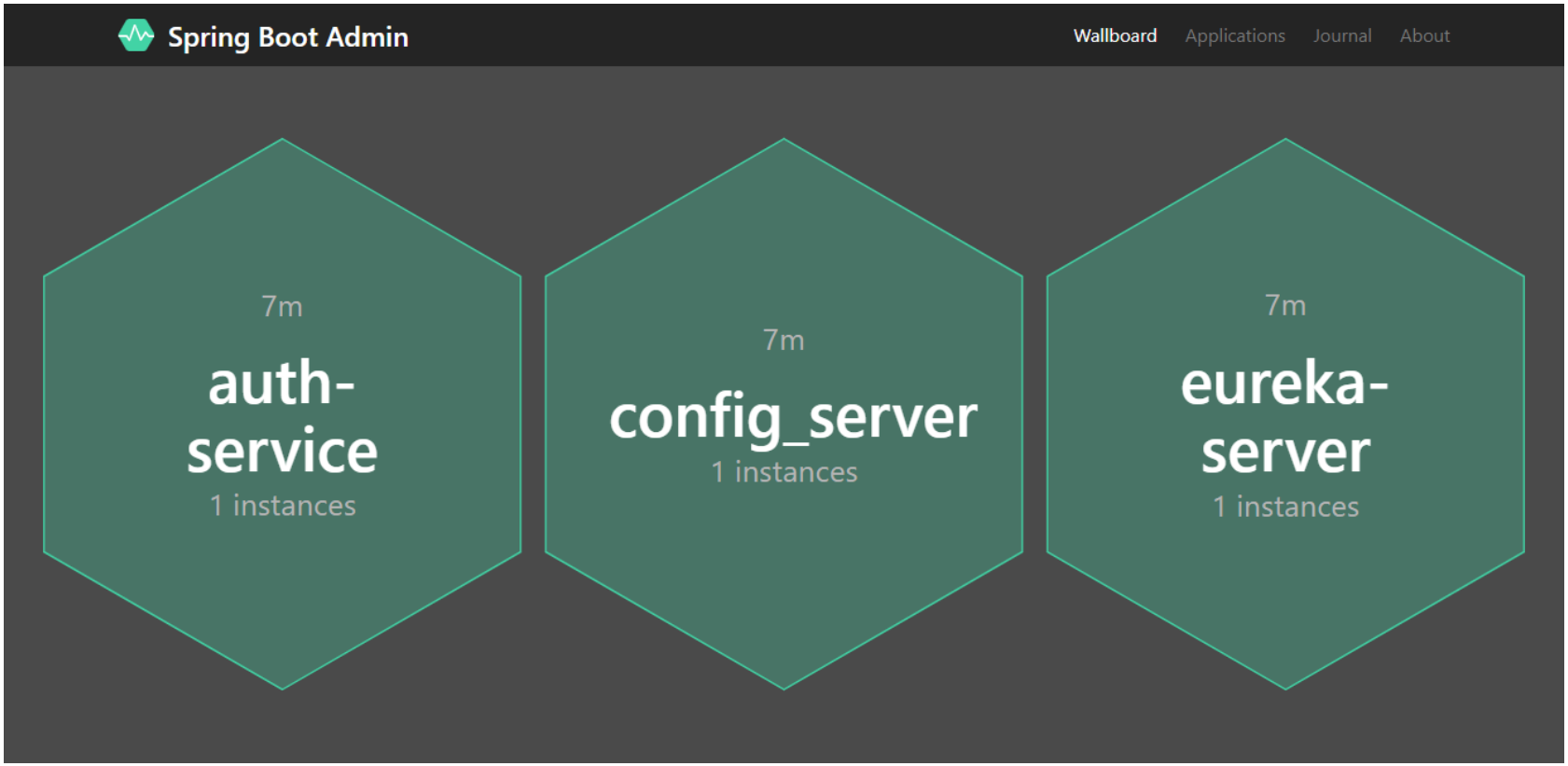
```
User findFirstByOrderByLastnameAsc();

User findTopByOrderByAgeDesc();

Page<User> queryFirst10ByLastname(String lastname, Pageable pageable);

Slice<User> findTop3ByLastname(String lastname, Pageable pageable);

List<User> findFirst10ByLastname(String lastname, Sort sort);

List<User> findTop10ByLastname(String lastname, Pageable pageable);
```

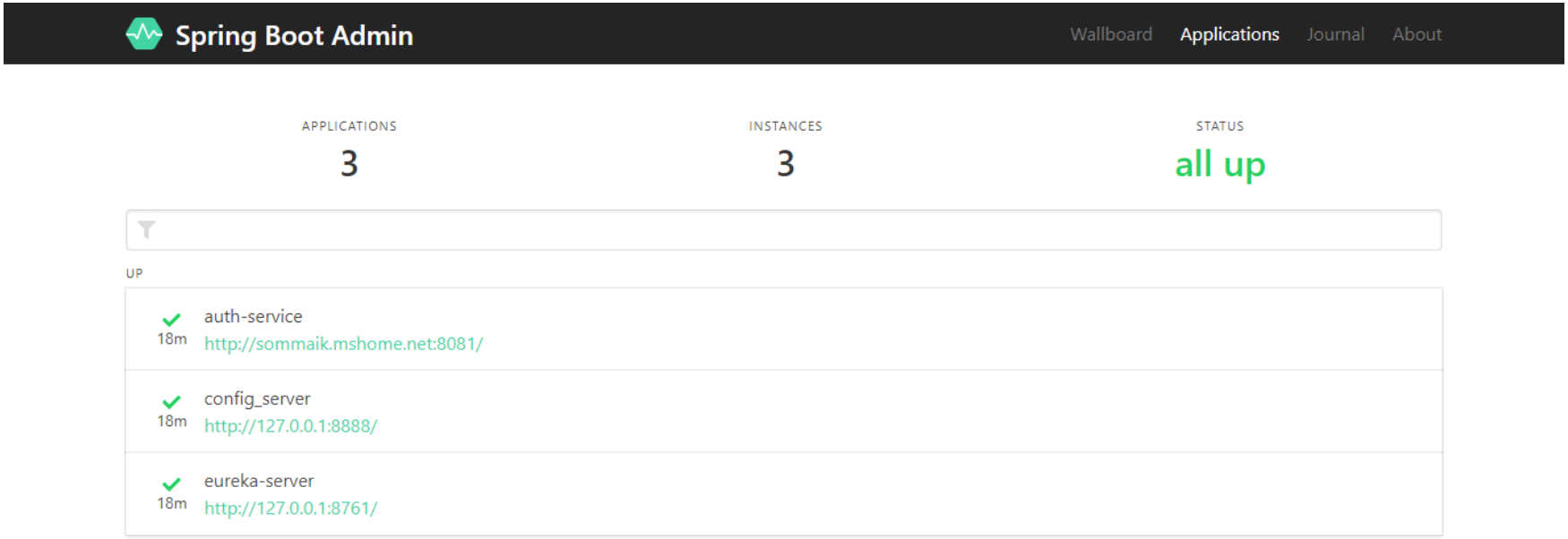
service admin with

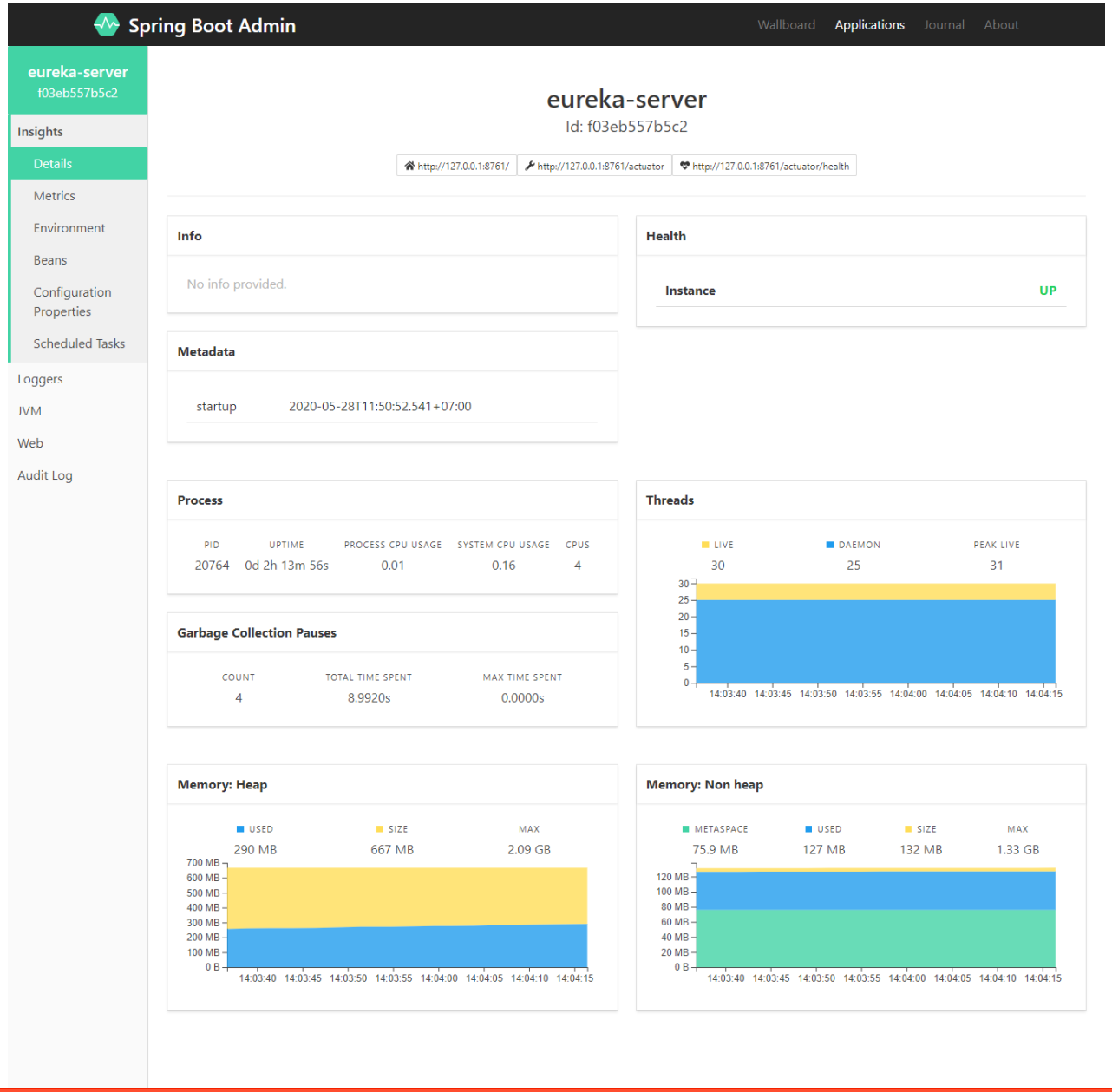
SPRING BOOT ADMIN

Actuator

- Actuator is mainly used to expose operational information about the running application– health, metrics, info, dump, env, etc. It uses HTTP endpoints or JMX beans to enable us to interact with it.
- Once this dependency is on the classpath several endpoints are available for us out of the box. As with most Spring modules, we can easily configure or extend it in many ways

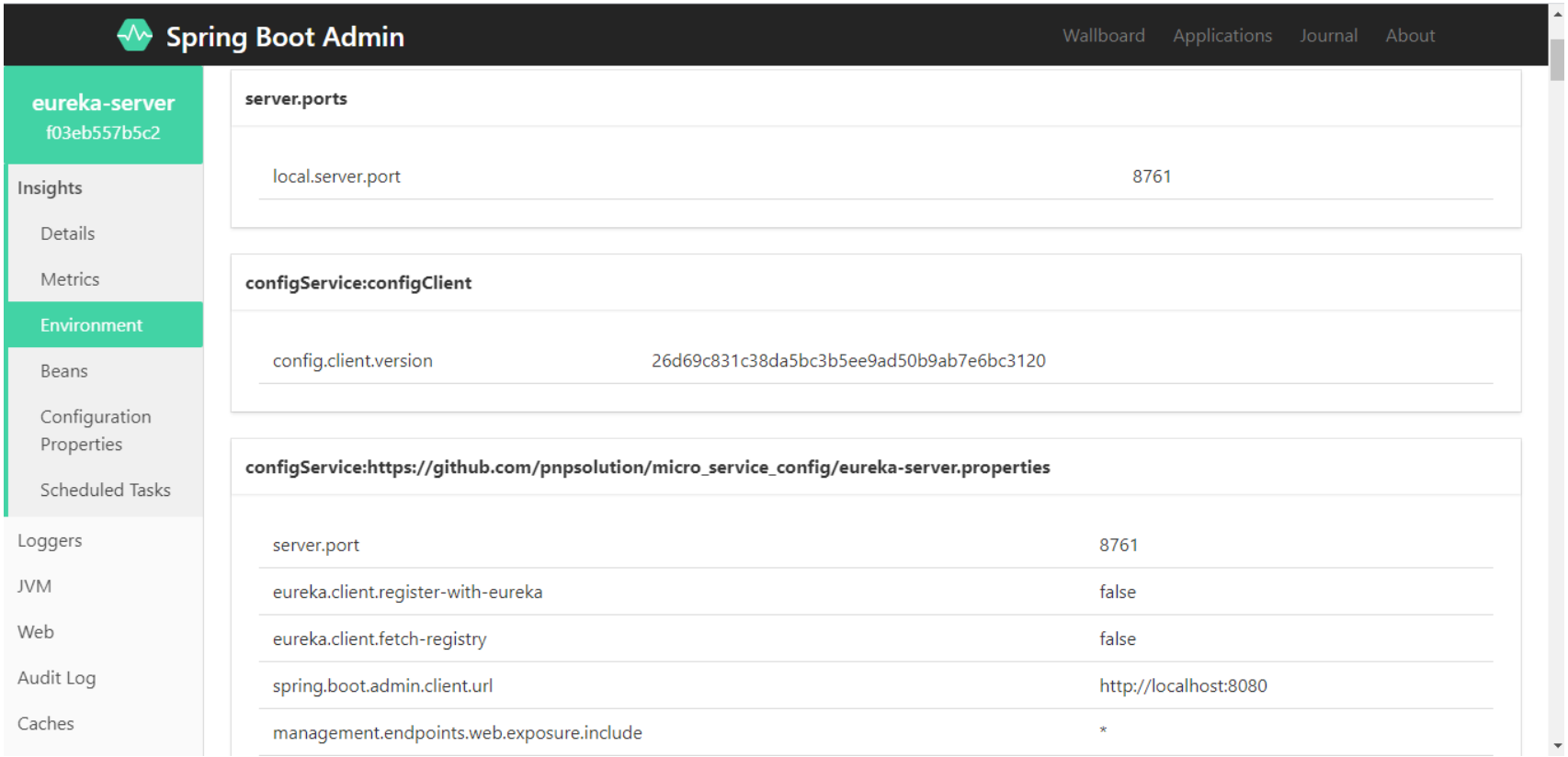

Spring Boot Admin
Wallboard
Applications
Journal
About
7m
auth-service
1 instances
7m
config_server
1 instances
7m
eureka-server
1 instances

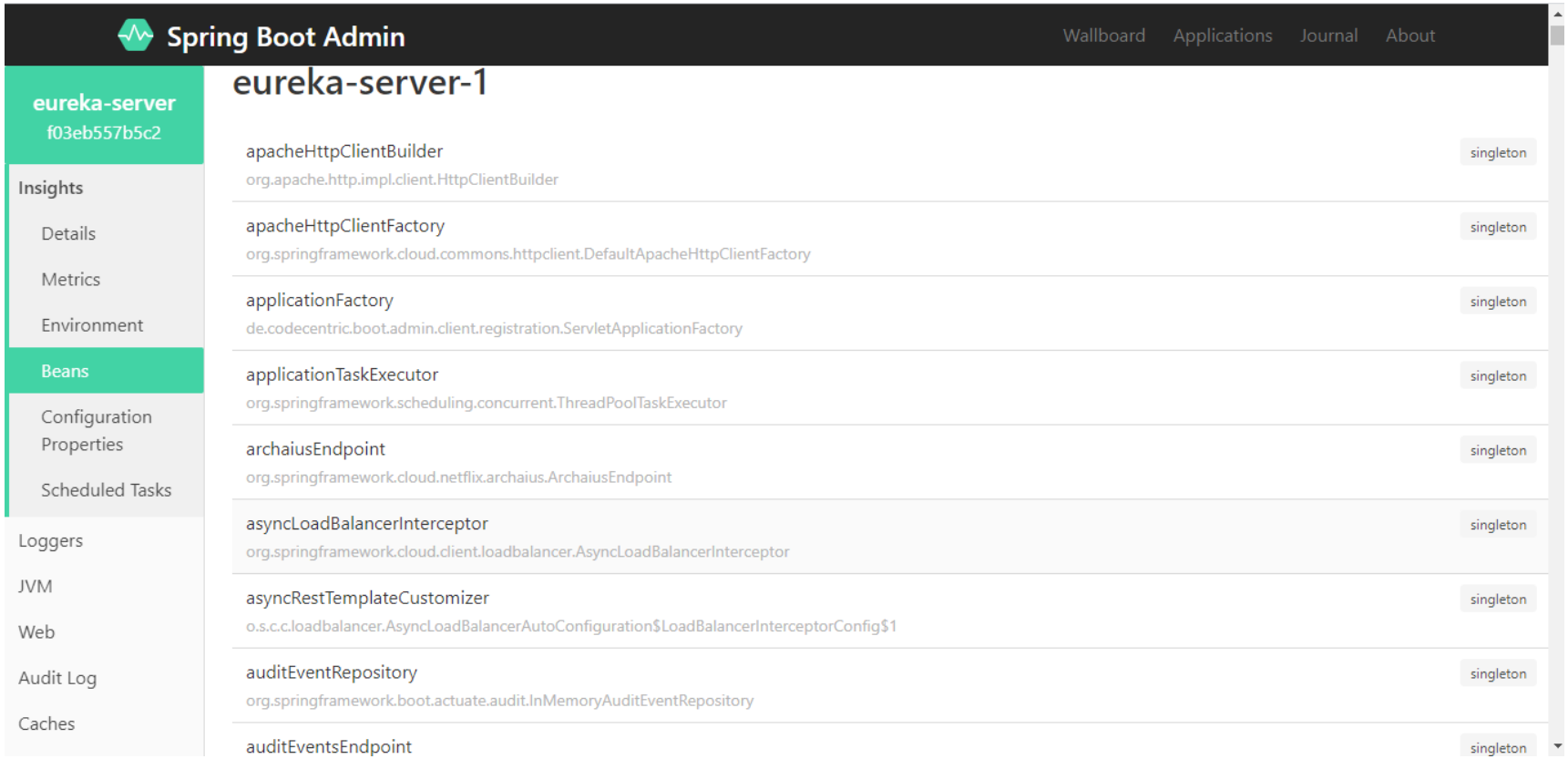

Spring Boot Admin
Wallboard
Applications
Journal
About
APPLICATIONS
3
INSTANCES
3
STATUS
all up
UP
auth-service
18m
http://sommaik.mshome.net:8081/
config_server
18m
http://127.0.0.1:8888/
eureka-server
18m
http://127.0.0.1:8761/

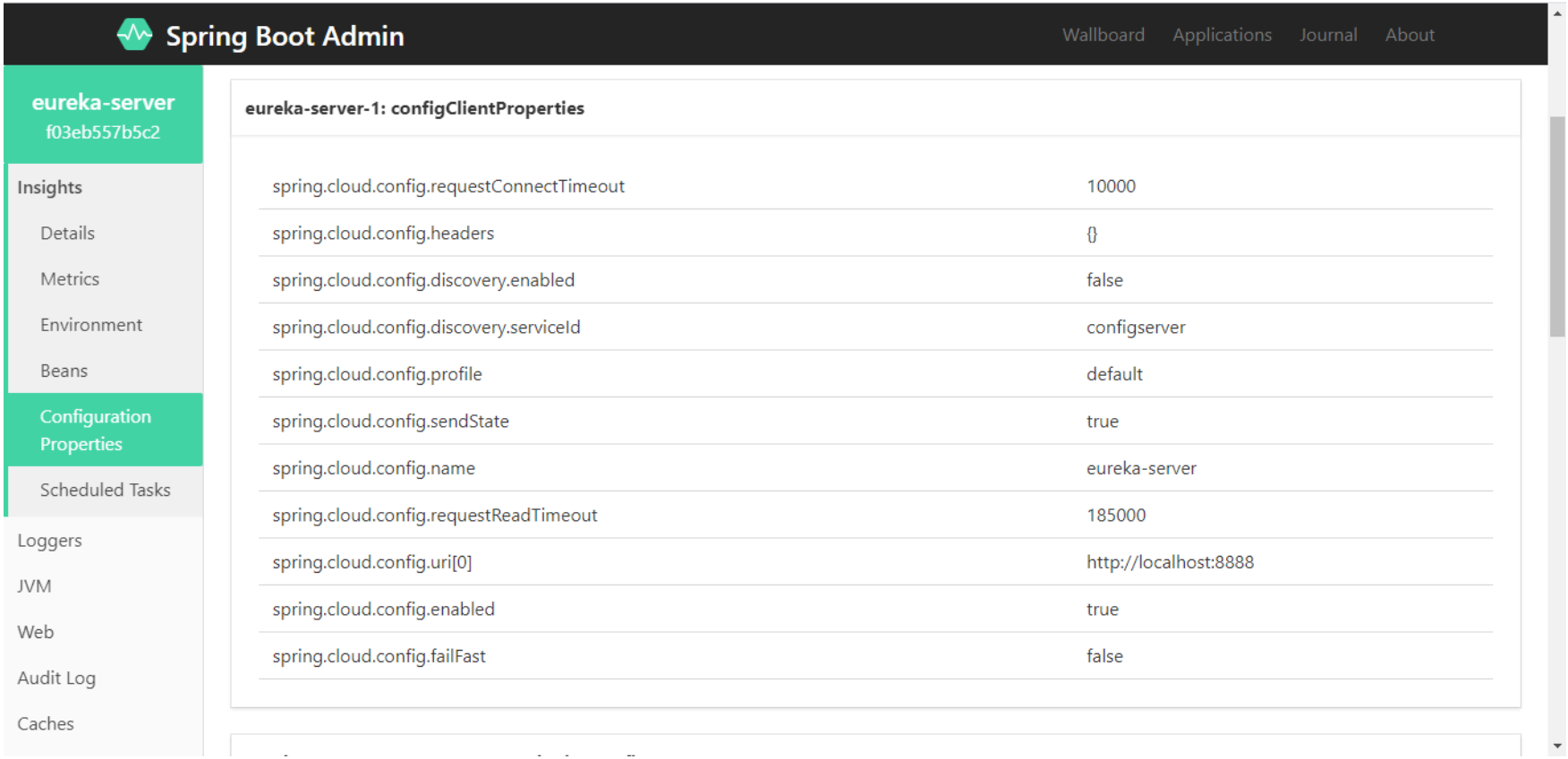

Spring Boot Admin
Wallboard
Applications
Journal
About
eureka-server
f03eb557b5c2
Insights
Details
Metrics
Environment
Beans
Configuration Properties
Scheduled Tasks
Loggers
JVM
Web
Audit Log
eureka-server
Id: f03eb557b5c2
http://127.0.0.1:8761/
http://127.0.0.1:8761/actuator
http://127.0.0.1:8761/actuator/health
Info
No info provided.
Health
Instance
UP
Metadata
startup
2020-05-28T11:50:52.541+07:00
Process
PID
UPTIME
PROCESS CPU USAGE
SYSTEM CPU USAGE
CPUS
20764
0d 2h 13m 56s
0.01
0.16
4
Threads
LIVE
DAEMON
PEAK LIVE
30
25
31
Garbage Collection Pauses
COUNT
TOTAL TIME SPENT
MAX TIME SPENT
4
8.9920s
0.0000s
Memory: Heap
USED
SIZE
MAX
290 MB
667 MB
2.09 GB
Memory: Non heap
METASPACE
USED
SIZE
MAX
75.9 MB
127 MB
132 MB
1.33 GB

Event Journal

Application	Instance	Time	Event
config_server	549fd52bc1a8	05/28/2020 13:47:33.369	ENDPOINTS_DETECTED
config_server	549fd52bc1a8	05/28/2020 13:47:33.071	STATUS_CHANGED(UP)
config_server	549fd52bc1a8	05/28/2020 13:47:29.899	REGISTERED
eureka-server	f03eb557b5c2	05/28/2020 13:47:27.629	ENDPOINTS_DETECTED
auth-service	5aed4f762772	05/28/2020 13:47:27.603	INFO_CHANGED
auth-service	5aed4f762772	05/28/2020 13:47:27.591	ENDPOINTS_DETECTED
eureka-server	f03eb557b5c2	05/28/2020 13:47:27.521	STATUS_CHANGED(UP)
auth-service	5aed4f762772	05/28/2020 13:47:27.510	STATUS_CHANGED(UP)
eureka-server	f03eb557b5c2	05/28/2020 13:47:27.278	REGISTERED
auth-service	5aed4f762772	05/28/2020 13:47:27.252	REGISTERED


Spring Boot Admin
Wallboard
Applications
Journal
About
eureka-server
f03eb557b5c2
Insights
Details
Metrics
Environment
Beans
Configuration Properties
Scheduled Tasks
Loggers
JVM
Web
Audit Log
Caches
server.ports
local.server.port
8761
configService:configClient
config.client.version
26d69c831c38da5bc3b5ee9ad50b9ab7e6bc3120
configService:https://github.com/pnpsolution/micro_service_config/eureka-server.properties
server.port
8761
eureka.client.register-with-eureka
false
eureka.client.fetch-registry
false
spring.boot.admin.client.url
http://localhost:8080
management.endpoints.web.exposure.include
*


Spring Boot Admin
Wallboard
Applications
Journal
About
eureka-server
f03eb557b5c2
Insights
Details
Metrics
Environment
Beans
Configuration Properties
Scheduled Tasks
Loggers
JVM
Web
Audit Log
Caches
eureka-server-1
apacheHttpClientBuilder
org.apache.http.impl.client.HttpClientBuilder
singleton
apacheHttpClientFactory
org.springframework.cloud.commons.httpclient.DefaultApacheHttpClientFactory
singleton
applicationFactory
de.codecentric.boot.admin.client.registration.ServletApplicationFactory
singleton
applicationTaskExecutor
org.springframework.scheduling.concurrent.ThreadPoolTaskExecutor
singleton
archaiusEndpoint
org.springframework.cloud.netflix.archaius.ArchaiusEndpoint
singleton
asyncLoadBalancerInterceptor
org.springframework.cloud.client.loadbalancer.AsyncLoadBalancerInterceptor
singleton
asyncRestTemplateCustomizer
o.s.c.c.loadbalancer.AsyncLoadBalancerAutoConfiguration$LoadBalancerInterceptorConfig$1
singleton
auditEventRepository
org.springframework.boot.actuate.audit.InMemoryAuditEventRepository
singleton
auditEventsEndpoint
singleton


Spring Boot Admin
Wallboard
Applications
Journal
About
eureka-server
f03eb557b5c2
Insights
Details
Metrics
Environment
Beans
Configuration Properties
Scheduled Tasks
Loggers
JVM
Web
Audit Log
Caches
eureka-server-1: configClientProperties
spring.cloud.config.requestConnectTimeout 10000
spring.cloud.config.headers {}
spring.cloud.config.discovery.enabled false
spring.cloud.config.discovery.serviceId configserver
spring.cloud.config.profile default
spring.cloud.config.sendState true
spring.cloud.config.name eureka-server
spring.cloud.config.requestReadTimeout 185000
spring.cloud.config.uri[0] http://localhost:8888
spring.cloud.config.enabled true
spring.cloud.config.failFast false

centralize config with

CONFIG SERVER

Spring Cloud Configuration Server

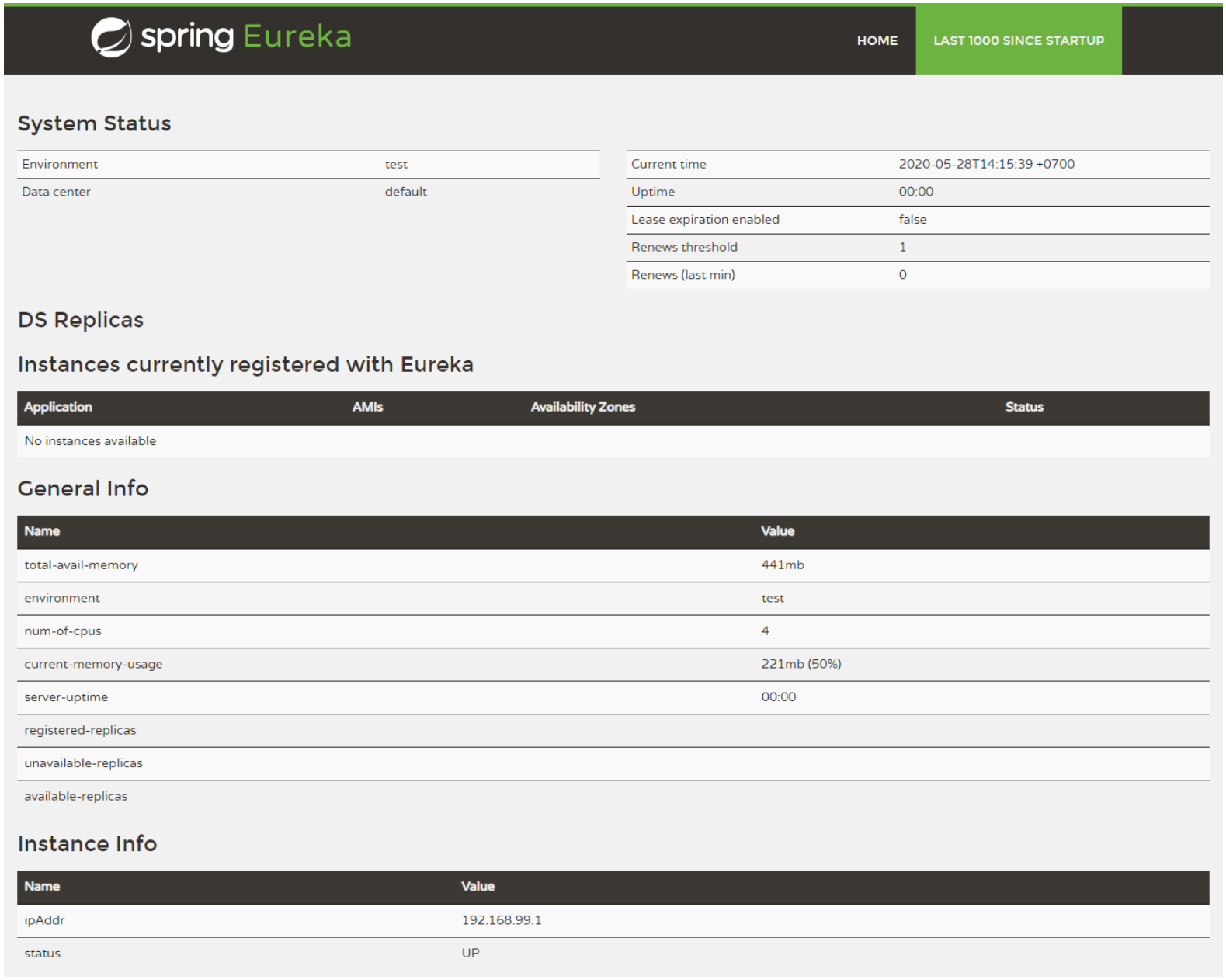
- Spring Cloud Configuration Server is a centralized application that manages all the application related configuration properties.
- In this chapter, you will learn in detail about how to create Spring Cloud Configuration server.

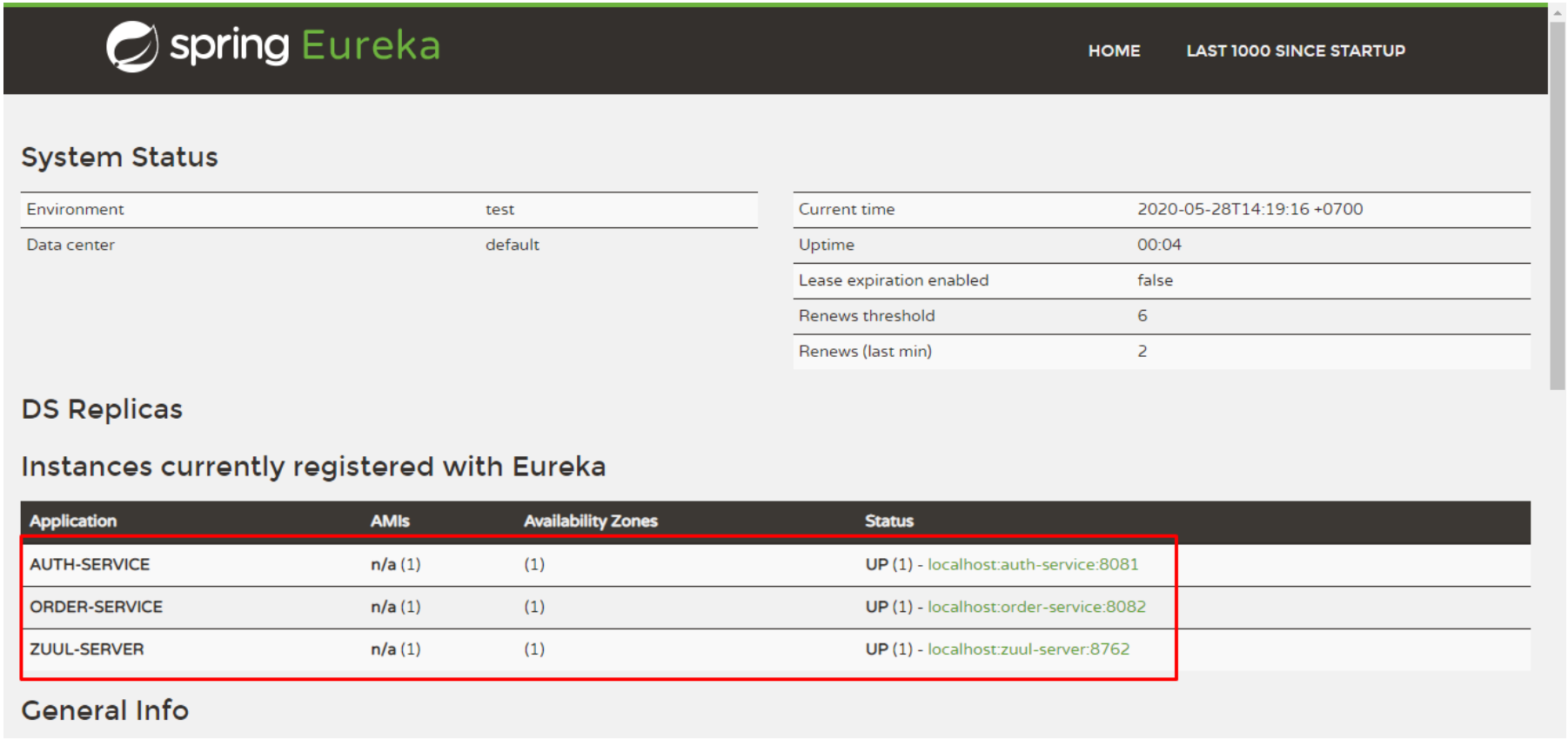
Microservice with

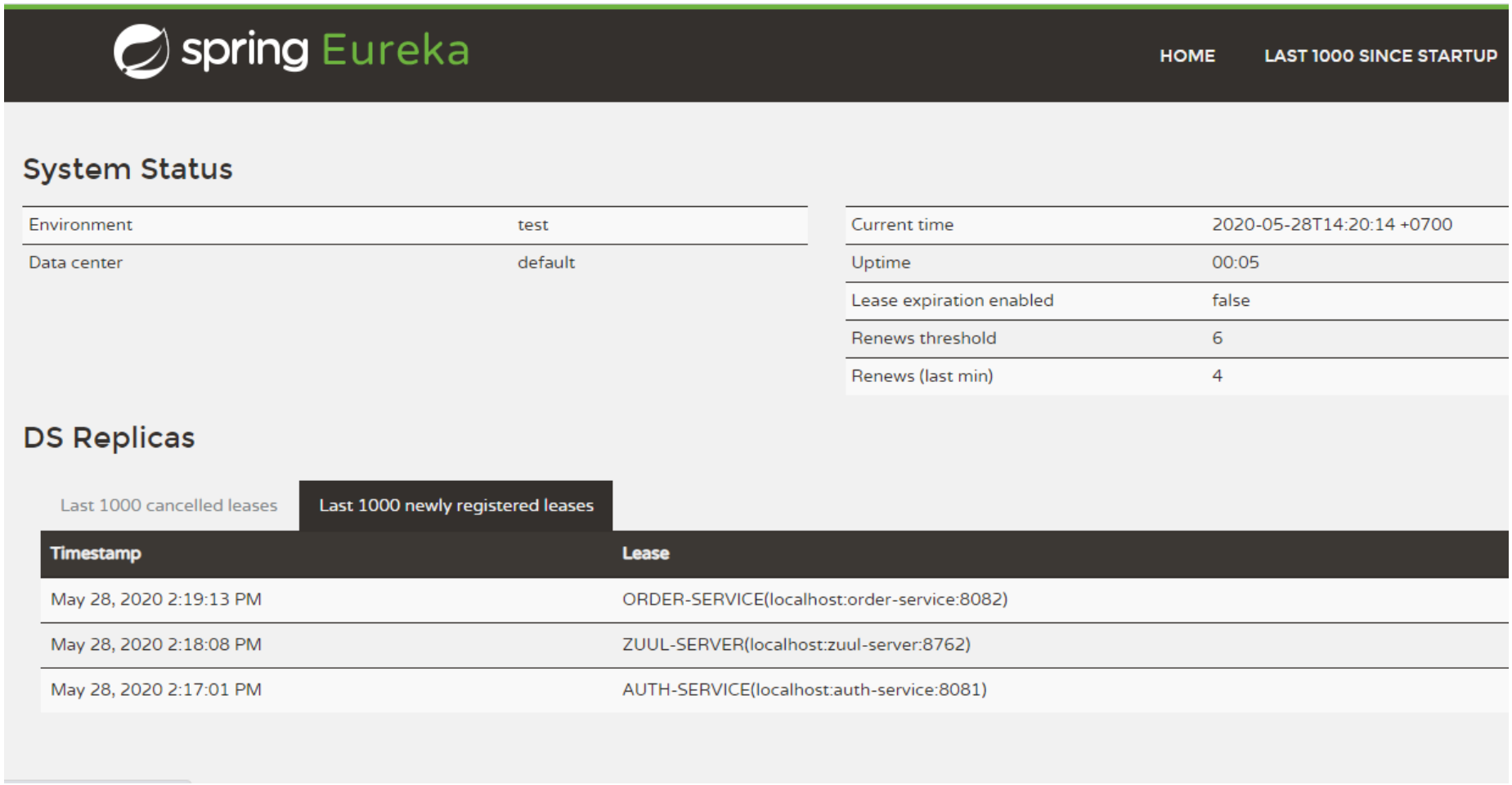
EUREKA SERVER

Eureka Server

- Eureka Server is an application that holds the information about all client-service applications.
- Every Micro service will register into the Eureka server and Eureka server knows all the client applications running on each port and IP address.
- Eureka Server is also known as Discovery Server.
- In this chapter, we will learn in detail about How to build a Eureka server.


spring Eureka
HOME
LAST 1000 SINCE STARTUP
System Status
Environment test
Data center default
Current time 2020-05-28T14:15:39 +0700
Uptime 00:00
Lease expiration enabled false
Renews threshold 1
Renews (last min) 0
DS Replicas
Instances currently registered with Eureka
Application AMIs Availability Zones Status
No instances available
General Info
Name Value
total-avail-memory 441mb
environment test
num-of-cpus 4
current-memory-usage 221mb (50%)
server-uptime 00:00
registered-replicas
unavailable-replicas
available-replicas
Instance Info
Name Value
ipAddr 192.168.99.1
status UP


spring Eureka
HOME
LAST 1000 SINCE STARTUP
System Status
Environment test
Data center default
Current time 2020-05-28T14:19:16 +0700
Uptime 00:04
Lease expiration enabled false
Renews threshold 6
Renews (last min) 2
DS Replicas
Instances currently registered with Eureka
Application AMIs Availability Zones Status
AUTH-SERVICE n/a (1) (1) UP (1) - localhost:auth-service:8081
ORDER-SERVICE n/a (1) (1) UP (1) - localhost:order-service:8082
ZUUL-SERVER n/a (1) (1) UP (1) - localhost:zuul-server:8762
General Info


spring Eureka
HOME
LAST 1000 SINCE STARTUP
System Status
Environment
test
Data center
default
Current time
2020-05-28T14:20:14 +0700
Uptime
00:05
Lease expiration enabled
false
Renews threshold
6
Renews (last min)
4
DS Replicas
Last 1000 cancelled leases
Last 1000 newly registered leases
Timestamp
Lease
May 28, 2020 2:19:13 PM
ORDER-SERVICE(localhost:order-service:8082)
May 28, 2020 2:18:08 PM
ZUUL-SERVER(localhost:zuul-server:8762)
May 28, 2020 2:17:01 PM
AUTH-SERVICE(localhost:auth-service:8081)

Cloud Routing with

ZUUL

Zuul

- Zuul Server is a gateway application that handles all the requests and does the dynamic routing of microservice applications.
- The Zuul Server is also known as Edge Server.
- For Example, **/api/user** is mapped to the user service and /api/products is mapped to the product service and Zuul Server dynamically routes the requests to the respective backend application.

security with

SPRING SECURITY

JWT

- JSON Web Token (JWT) is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object.
- This information can be verified and trusted because it is digitally signed.
- JWTs can be signed using a secret (with the **HMAC** algorithm) or a public/private key pair using **RSA** or **ECDSA**.
- Although JWTs can be encrypted to also provide secrecy between parties, we will focus onsignedtokens.
- Signed tokens can verify theintegrityof the claims contained within it, while encrypted tokenshidethose claims from other parties.
- When tokens are signed using public/private key pairs, the signature also certifies that only the party holding the private key is the one that signed it.

Circuit Breaker with

HYSTRIX

Hystrix

- Hystrix is a library from Netflix.
- Hystrix isolates the points of access between the services, stops cascading failures across them and provides the fallback options.
- For example, when you are calling a 3rdparty application, it takes more time to send the response.
- So at that time, the control goes to the fallback method and returns the custom response to your application.

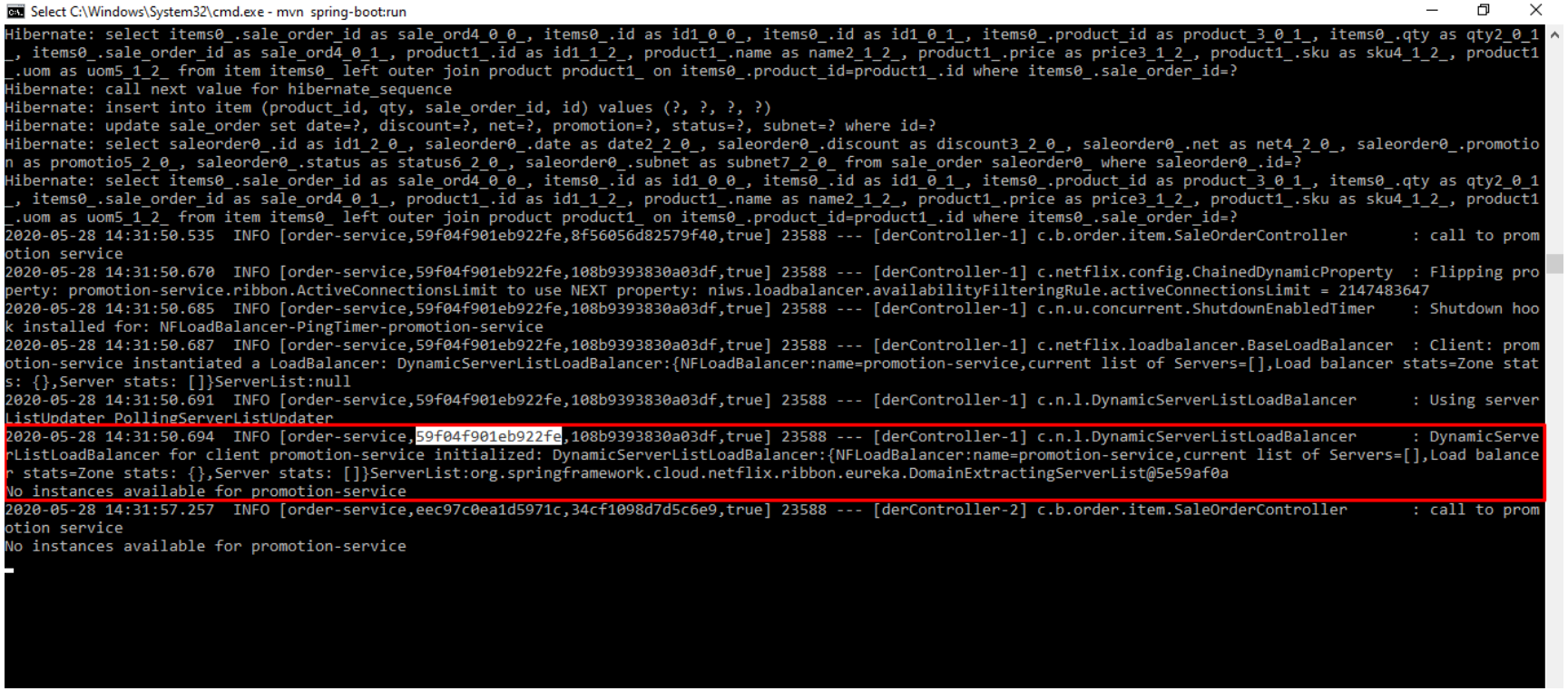
Log Tracing with

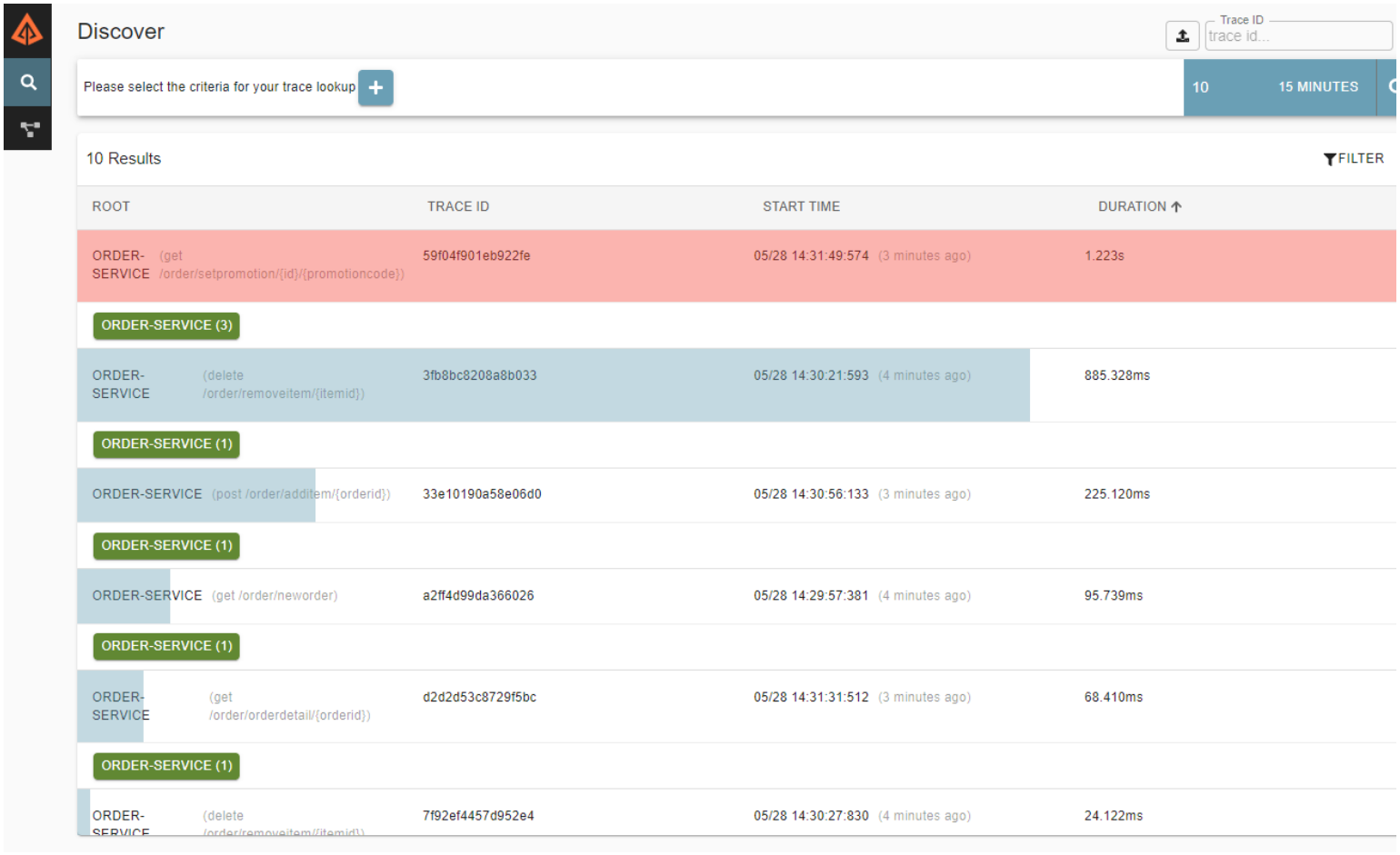
SLEUTH

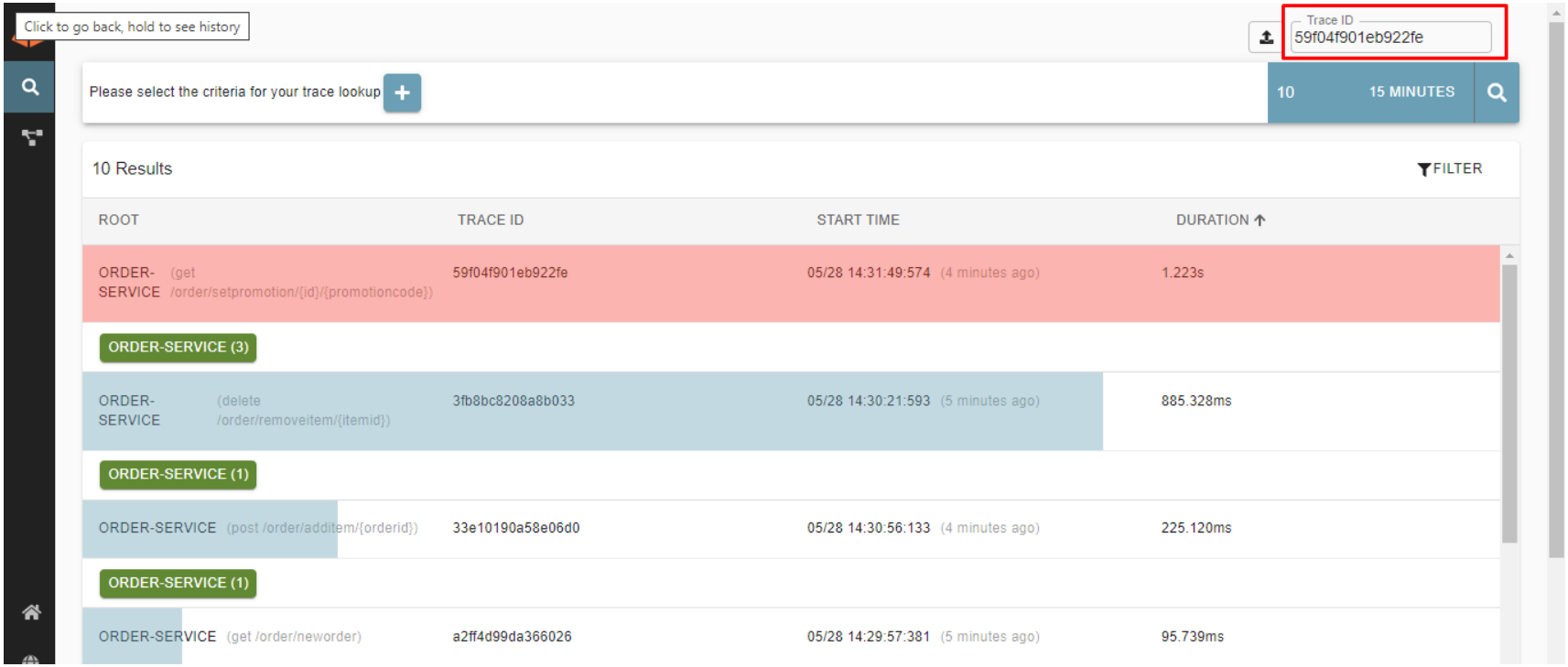
trace and span ids

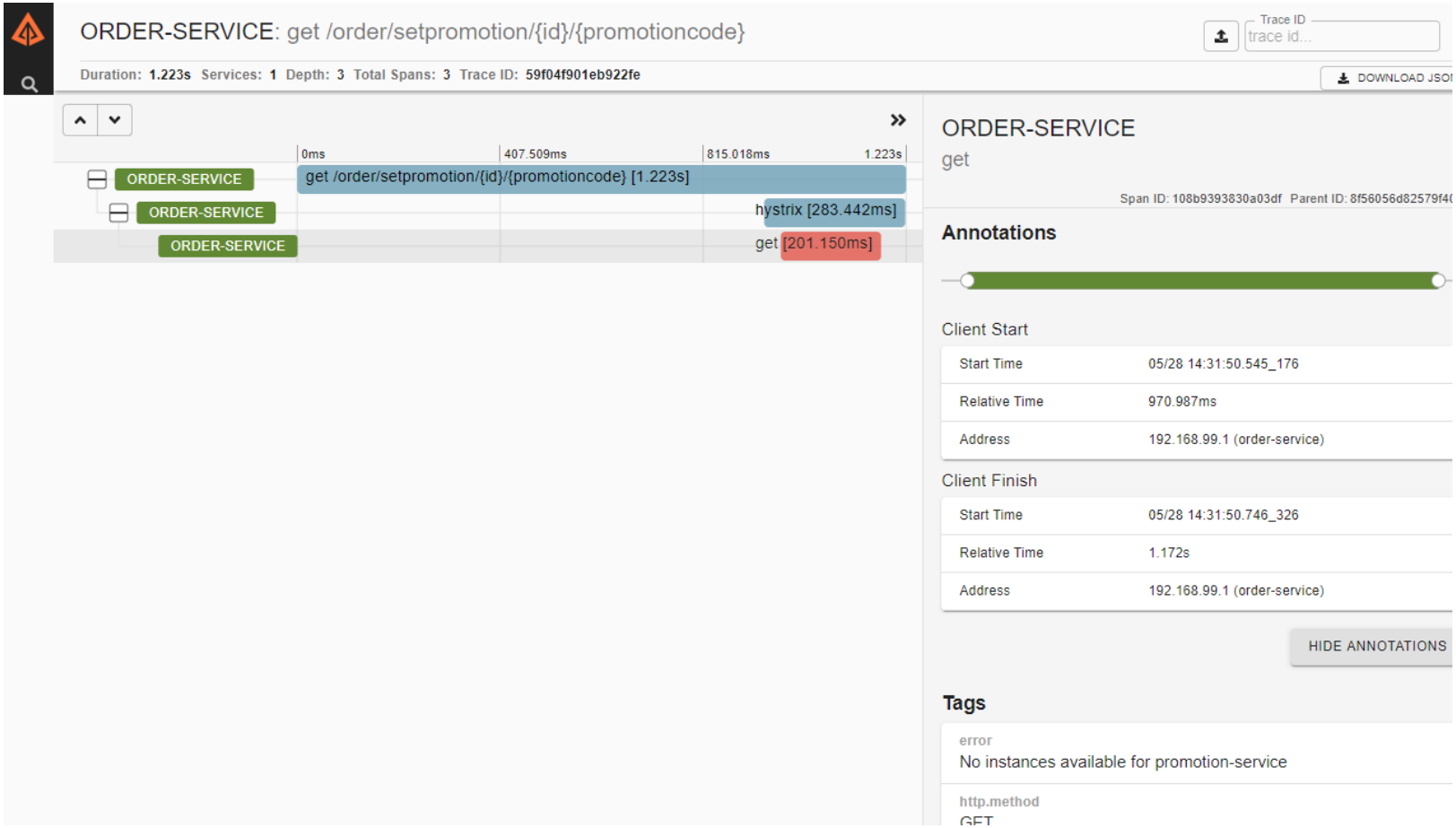
distributed tracing system with

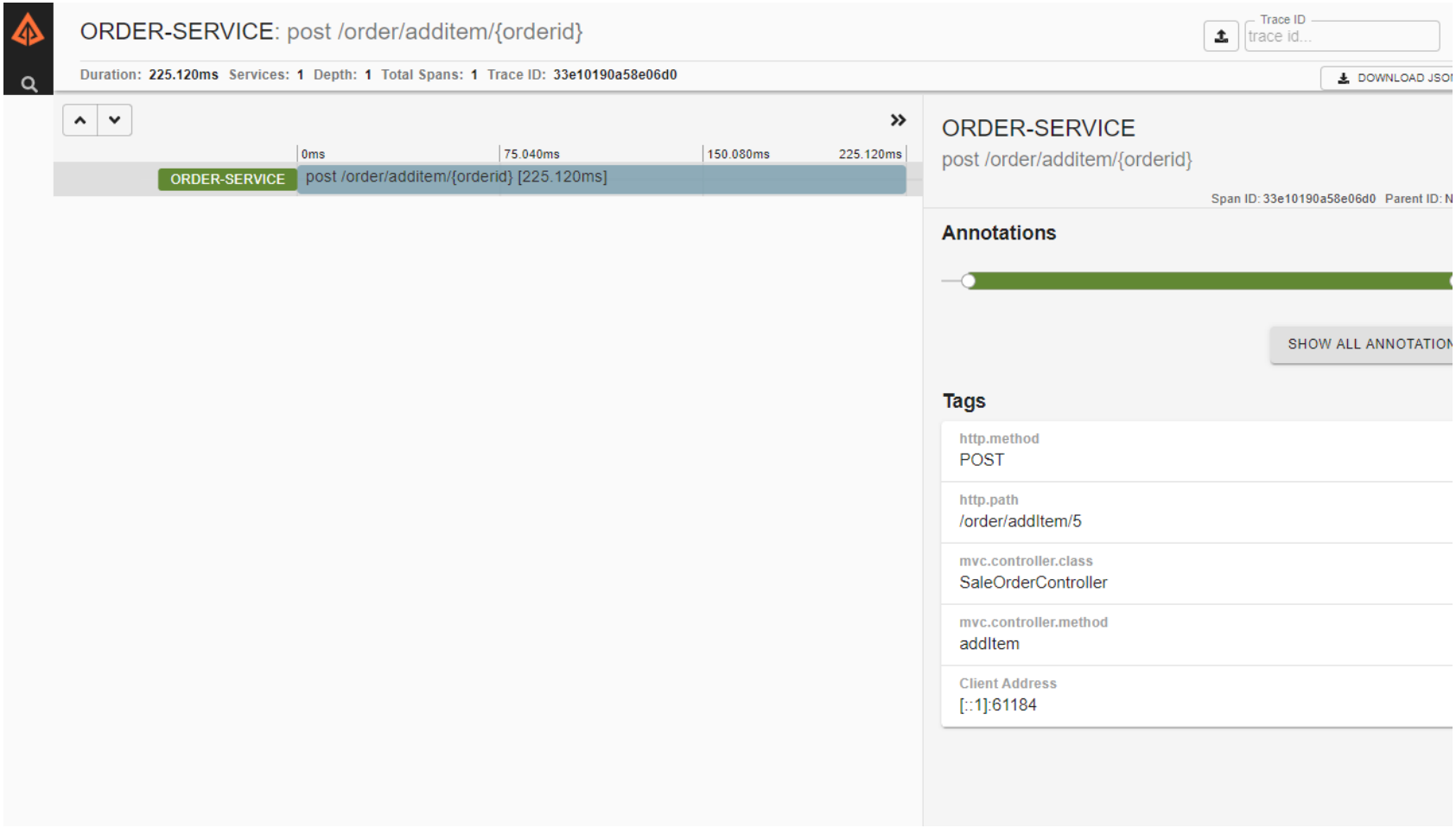
ZIPKIN


Select C:\Windows\System32\cmd.exe - mvn spring-boot:run
Hibernate: select items0_.sale_order_id as sale_ord4_0_0_, items0_.id as id1_0_0_, items0_.id as id1_0_1_, items0_.product_id as product_3_0_1_, items0_.qty as qty2_0_1_, items0_.sale_order_id as sale_ord4_0_1_, product1_.id as id1_1_2_, product1_.name as name2_1_2_, product1_.price as price3_1_2_, product1_.sku as sku4_1_2_, product1_.uom as uom5_1_2_ from item items0_ left outer join product product1_ on items0_.product_id=product1_.id where items0_.sale_order_id=?
Hibernate: call next value for hibernate_sequence
Hibernate: insert into item (product_id, qty, sale_order_id, id) values (?, ?, ?, ?)
Hibernate: update sale_order set date=?, discount=?, net=?, promotion=?, status=?, subnet=? where id=?
Hibernate: select saleorder0_.id as id1_2_0_, saleorder0_.date as date2_2_0_, saleorder0_.discount as discount3_2_0_, saleorder0_.net as net4_2_0_, saleorder0_.promotion as promotio5_2_0_, saleorder0_.status as status6_2_0_, saleorder0_.subnet as subnet7_2_0_ from sale_order saleorder0_ where saleorder0_.id=?
Hibernate: select items0_.sale_order_id as sale_ord4_0_0_, items0_.id as id1_0_0_, items0_.id as id1_0_1_, items0_.product_id as product_3_0_1_, items0_.qty as qty2_0_1_, items0_.sale_order_id as sale_ord4_0_1_, product1_.id as id1_1_2_, product1_.name as name2_1_2_, product1_.price as price3_1_2_, product1_.sku as sku4_1_2_, product1_.uom as uom5_1_2_ from item items0_ left outer join product product1_ on items0_.product_id=product1_.id where items0_.sale_order_id=?
2020-05-28 14:31:50.535 INFO [order-service,59f04f901eb922fe,8f56056d82579f40,true] 23588 --- [derController-1] c.b.order.item.SaleOrderController : call to promotion service
2020-05-28 14:31:50.670 INFO [order-service,59f04f901eb922fe,108b9393830a03df,true] 23588 --- [derController-1] c.netflix.config.ChainedDynamicProperty : Flipping property: promotion-service.ribbon.ActiveConnectionsLimit to use NEXT property: niws.loadbalancer.availabilityFilteringRule.activeConnectionsLimit = 2147483647
2020-05-28 14:31:50.685 INFO [order-service,59f04f901eb922fe,108b9393830a03df,true] 23588 --- [derController-1] c.n.u.concurrent.ShutdownEnabledTimer : Shutdown hook installed for: NFLoadBalancer-PingTimer-promotion-service
2020-05-28 14:31:50.687 INFO [order-service,59f04f901eb922fe,108b9393830a03df,true] 23588 --- [derController-1] c.netflix.loadbalancer.BaseLoadBalancer : Client: promotion-service instantiated a LoadBalancer: DynamicServerListLoadBalancer:{NFLoadBalancer:name=promotion-service,current list of Servers=[],Load balancer stats=Zone stats: {},Server stats: []}ServerList:null
2020-05-28 14:31:50.691 INFO [order-service,59f04f901eb922fe,108b9393830a03df,true] 23588 --- [derController-1] c.n.l.DynamicServerListLoadBalancer : Using serverListUpdater PollingServerListUpdater
2020-05-28 14:31:50.694 INFO [order-service,59f04f901eb922fe,108b9393830a03df,true] 23588 --- [derController-1] c.n.l.DynamicServerListLoadBalancer : DynamicServerListLoadBalancer for client promotion-service initialized: DynamicServerListLoadBalancer:{NFLoadBalancer:name=promotion-service,current list of Servers=[],Load balancer stats=Zone stats: {},Server stats: []}ServerList:org.springframework.cloud.netflix.ribbon.eureka.DomainExtractingServerList@5e59af0a
No instances available for promotion-service
2020-05-28 14:31:57.257 INFO [order-service,eec97c0ea1d5971c,34cf1098d7d5c6e9,true] 23588 --- [derController-2] c.b.order.item.SaleOrderController : call to promotion service
No instances available for promotion-service


Discover
Trace ID
trace id...
Please select the criteria for your trace lookup
10
15 MINUTES
10 Results
FILTER
ROOT
TRACE ID
START TIME
DURATION
ORDER-SERVICE (get /order/setpromotion/{id}/{promotioncode})
59f04f901eb922fe
05/28 14:31:49:574 (3 minutes ago)
1.223s
ORDER-SERVICE (3)
ORDER-SERVICE (delete /order/removeitem/{itemid})
3fb8bc8208a8b033
05/28 14:30:21:593 (4 minutes ago)
885.328ms
ORDER-SERVICE (1)
ORDER-SERVICE (post /order/additem/{orderid})
33e10190a58e06d0
05/28 14:30:56:133 (3 minutes ago)
225.120ms
ORDER-SERVICE (1)
ORDER-SERVICE (get /order/neworder)
a2ff4d99da366026
05/28 14:29:57:381 (4 minutes ago)
95.739ms
ORDER-SERVICE (1)
ORDER-SERVICE (get /order/orderdetail/{orderid})
d2d2d53c8729f5bc
05/28 14:31:31:512 (3 minutes ago)
68.410ms
ORDER-SERVICE (1)
ORDER-SERVICE (delete
7f92ef4457d952e4
05/28 14:30:27:830 (4 minutes ago)
24.122ms


Click to go back, hold to see history
Trace ID
59f04f901eb922fe
Please select the criteria for your trace lookup
10
15 MINUTES
10 Results
FILTER
ROOT
TRACE ID
START TIME
DURATION
ORDER-SERVICE (get /order/setpromotion/{id}/{promotioncode})
59f04f901eb922fe
05/28 14:31:49:574 (4 minutes ago)
1.223s
ORDER-SERVICE (3)
ORDER-SERVICE (delete /order/removeitem/{itemid})
3fb8bc8208a8b033
05/28 14:30:21:593 (5 minutes ago)
885.328ms
ORDER-SERVICE (1)
ORDER-SERVICE (post /order/additem/{orderid})
33e10190a58e06d0
05/28 14:30:56:133 (4 minutes ago)
225.120ms
ORDER-SERVICE (1)
ORDER-SERVICE (get /order/neworder)
a2ff4d99da366026
05/28 14:29:57:381 (5 minutes ago)
95.739ms


ORDER-SERVICE: get /order/setpromotion/{id}/{promotioncode}
Trace ID
trace id...
Duration: 1.223s Services: 1 Depth: 3 Total Spans: 3 Trace ID: 59f04f901eb922fe
DOWNLOAD JSO
0ms
407.509ms
815.018ms
1.223s
ORDER-SERVICE
get /order/setpromotion/{id}/{promotioncode} [1.223s]
ORDER-SERVICE
hystrix [283.442ms]
ORDER-SERVICE
get [201.150ms]
ORDER-SERVICE
get
Span ID: 108b9393830a03df Parent ID: 8f56056d82579f4
Annotations
Client Start
Start Time 05/28 14:31:50.545_176
Relative Time 970.987ms
Address 192.168.99.1 (order-service)
Client Finish
Start Time 05/28 14:31:50.746_326
Relative Time 1.172s
Address 192.168.99.1 (order-service)
HIDE ANNOTATIONS
Tags
error
No instances available for promotion-service
http.method
GET


ORDER-SERVICE: post /order/additem/{orderid}
Trace ID
trace id...
Duration: 225.120ms Services: 1 Depth: 1 Total Spans: 1 Trace ID: 33e10190a58e06d0
DOWNLOAD JSON
0ms
75.040ms
150.080ms
225.120ms
ORDER-SERVICE
post /order/additem/{orderid} [225.120ms]
ORDER-SERVICE
post /order/additem/{orderid}
Span ID: 33e10190a58e06d0 Parent ID: N
Annotations
SHOW ALL ANNOTATION
Tags
http.method
POST
http.path
/order/addItem/5
mvc.controller.class
SaleOrderController
mvc.controller.method
addItem
Client Address
[::1]:61184