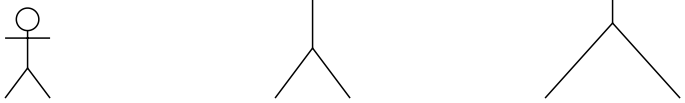
Terminologies

1. Scaling (Scale-up, Scale-out, Scale-in, Vertical Scaling, Horizontal Scaling)

1. Memory (RAM)
2. CPU
3. Disc/Storage
4. NIC (Network Interface Card)

Anthony is a celebrity

Vertical Scaling
(Scale up)



Boxer
10 $/hour
beat < 3 guys

i3, 4GB, 250GB

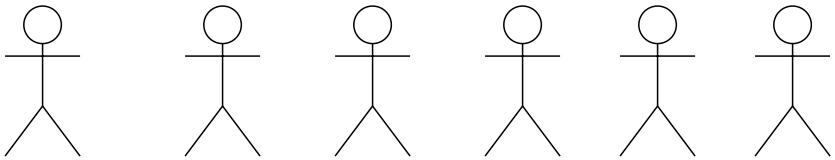
Mr Rock
30 $/hour
beat < 6 guys

i5, 8GB, 500GB

John Rambo
100 $/hour
beat < 20 guys

i7, 16GB, 1TB

Horizontal Scaling
(Scale out/in)



Mr Rock
30 $/hour
beat < 6 guys

Mr Rock
30 $/hour
beat < 6 guys

Mr Rock
30 $/hour
beat < 6 guys

Mr Rock
30 $/hour
beat < 6 guys

Mr Rock
30 $/hour
beat < 6 guys

Mr Rock
30 $/hour
beat < 6 guys

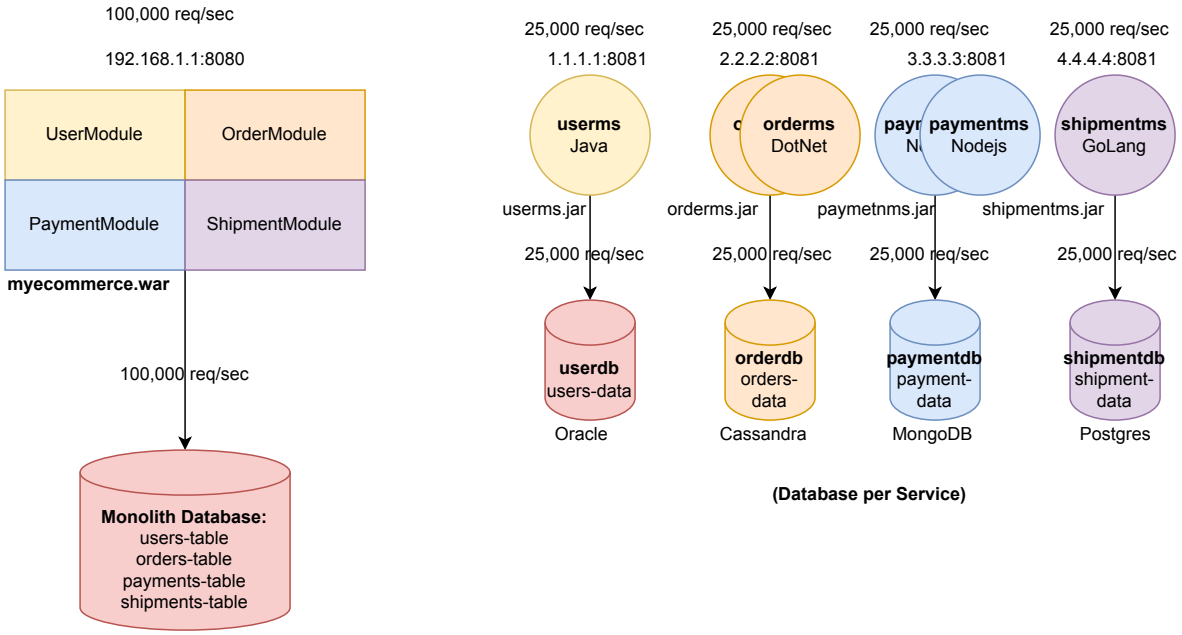
Black Friday/Diwali Season

One Application x 5 instances = 5 instances

100 microservices x 5 instances = 500 instances

Myecommerce Monolith Application

Microservices Application
(Collection of distributed standalone miniature applications)



(Database per Service)

1. Single Code Repo
2. Single Deployable File
3. Single Database eg. Oracle
4. Single Language eg Java

1. userms Code Repo
2. userms Deployable File
3. userms Database eg. Oracle
4. userms Language eg Java

1. orderms Code Repo
2. orderms Deployable File
3. orderms Database eg. Oracle
4. orderms Language eg Java

Pros of Microservices:

1. Cherry-pick scaling
2. Agility-1: Development is fast
3. Agility-2: Build is fast
4. Agility-3: Testing is fast
5. Agility-4: CI/CD is fast
6. Aglity-5: Release is fast
7. Resiliency
8. Distributed Service Load
9. Distributed Database Load
10. Technology Hetrogenity
11. Database Hetrogenity
12. Seucirty (Segregation)

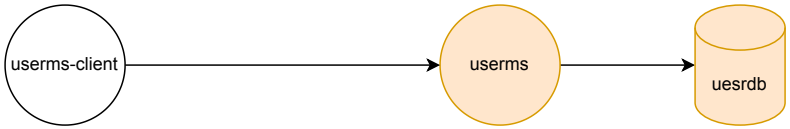
Cons of Microservices:

1. Latency between Microservices calls
2. Distributed Database (Aggregation/TxManagement)
3. Complexity in manageing Nodes (services + DB)
4. Cost (Infra + Resources)

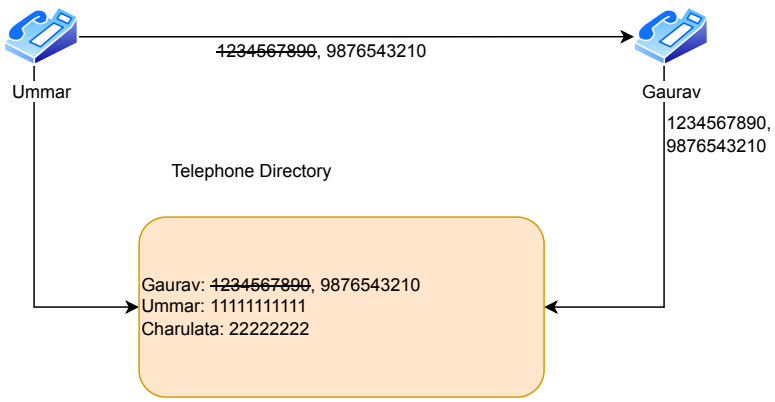
2-Pizza Team: Team size should be small for microserivces

ACID:

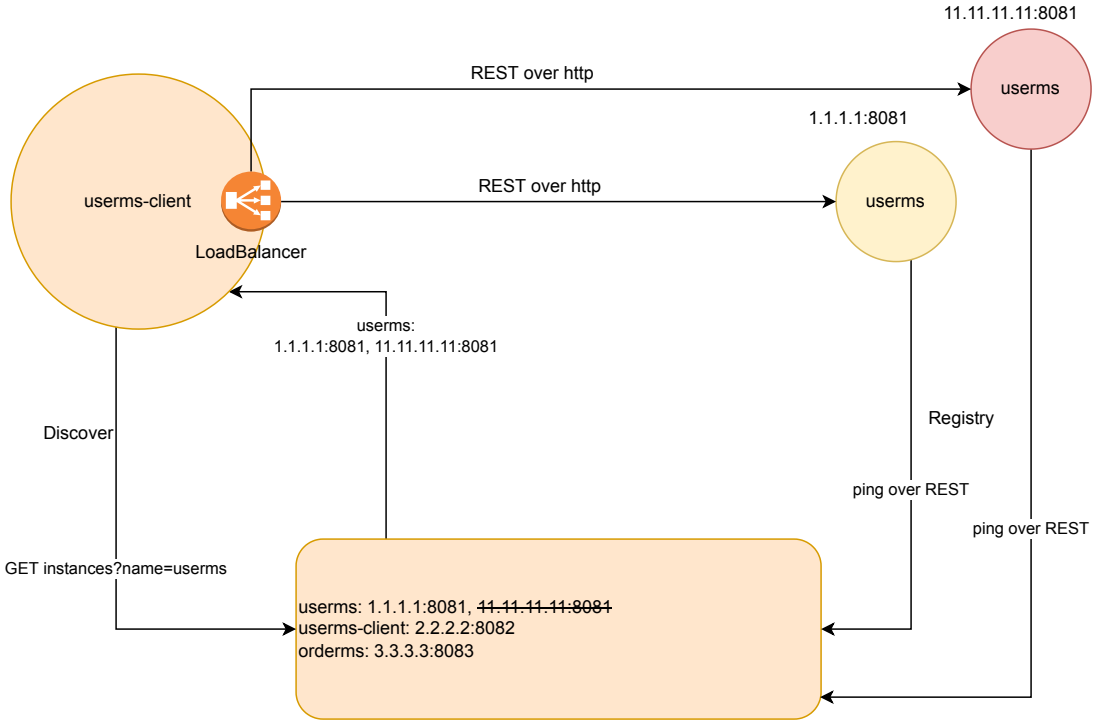
Microservice-to-Microservice Communication



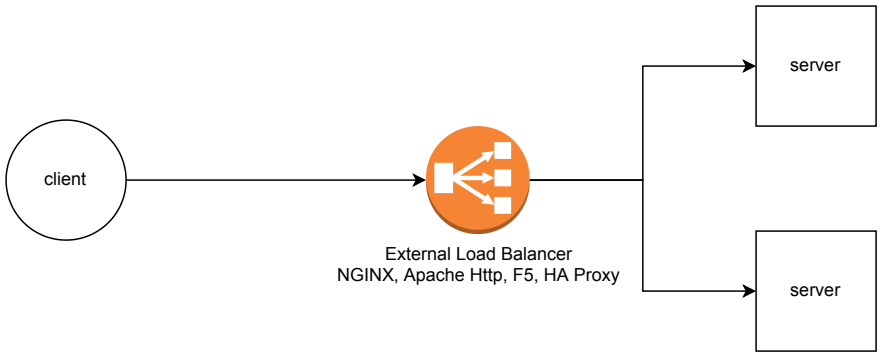
Service Discovery/Registry

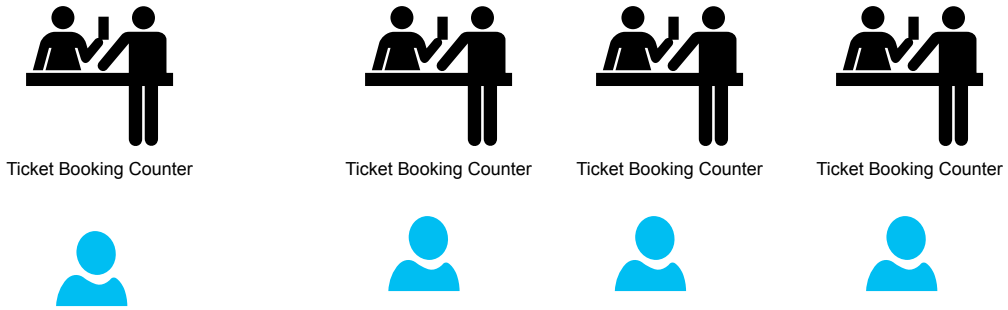


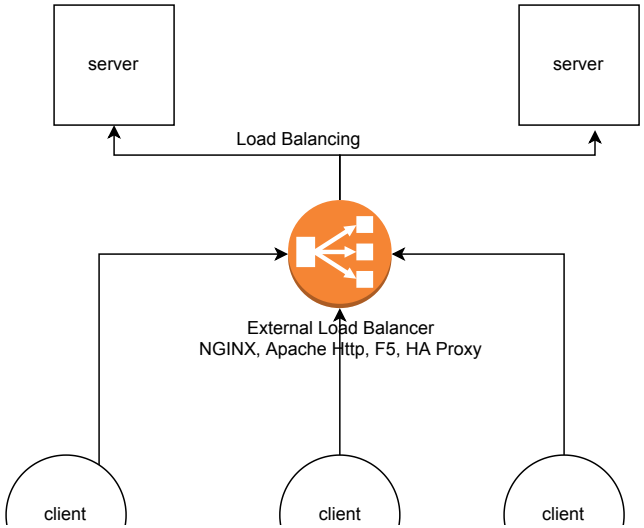
Design Patterns:
Service Registry -> Netflix Eureka
Service Discovery -> Netflix Eureka
Client-side Load Balancing -> Netflix Ribbon, Spring Cloud Load Balancer



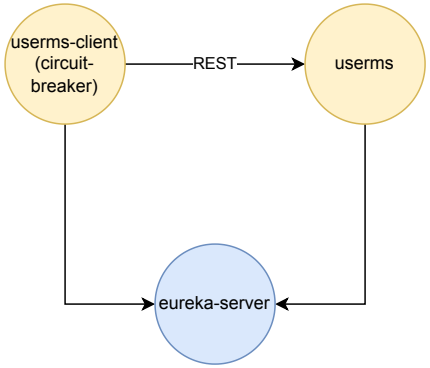
eureka-server: Service Registry/Discovery


client
server
server
External Load Balancer
NGINX, Apache Http, F5, HA Proxy


Ticket Booking Counter
Ticket Booking Counter
Ticket Booking Counter
Ticket Booking Counter

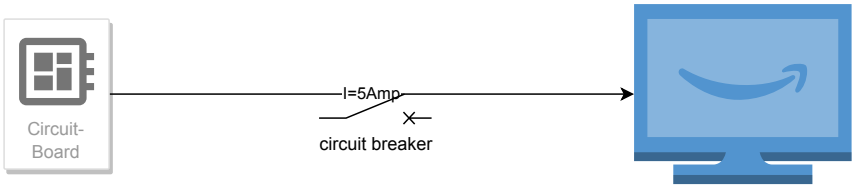

server
server
Load Balancing
External Load Balancer
NGINX, Apache Http, F5, HA Proxy
client
client
client

Microservices



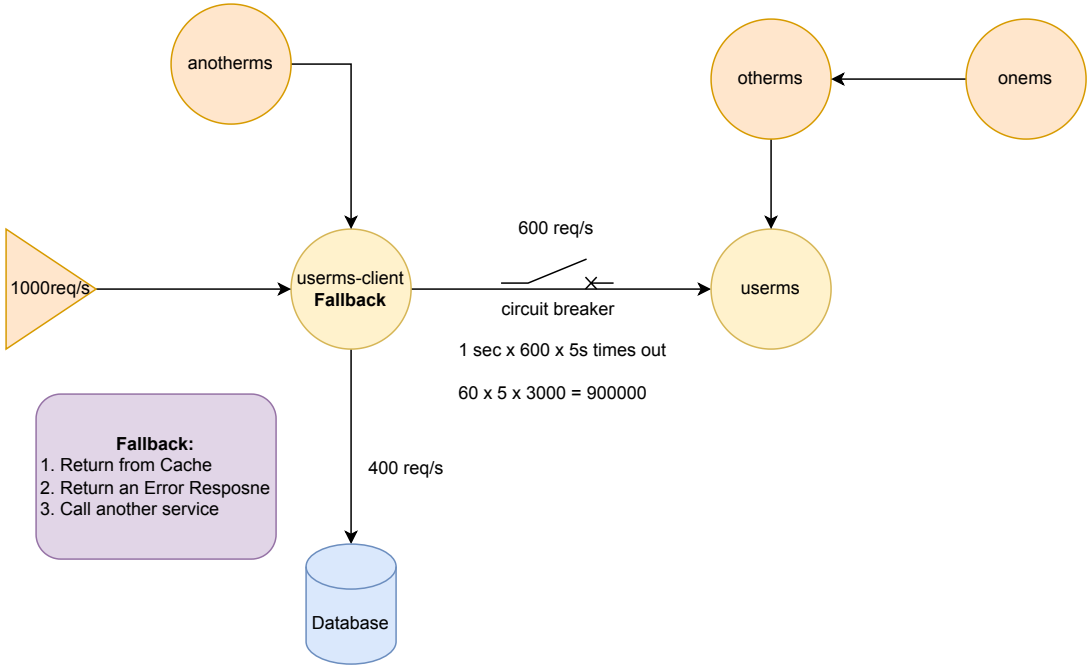
Fuse is a circtuit breaker
MCB

Circuite Breaker

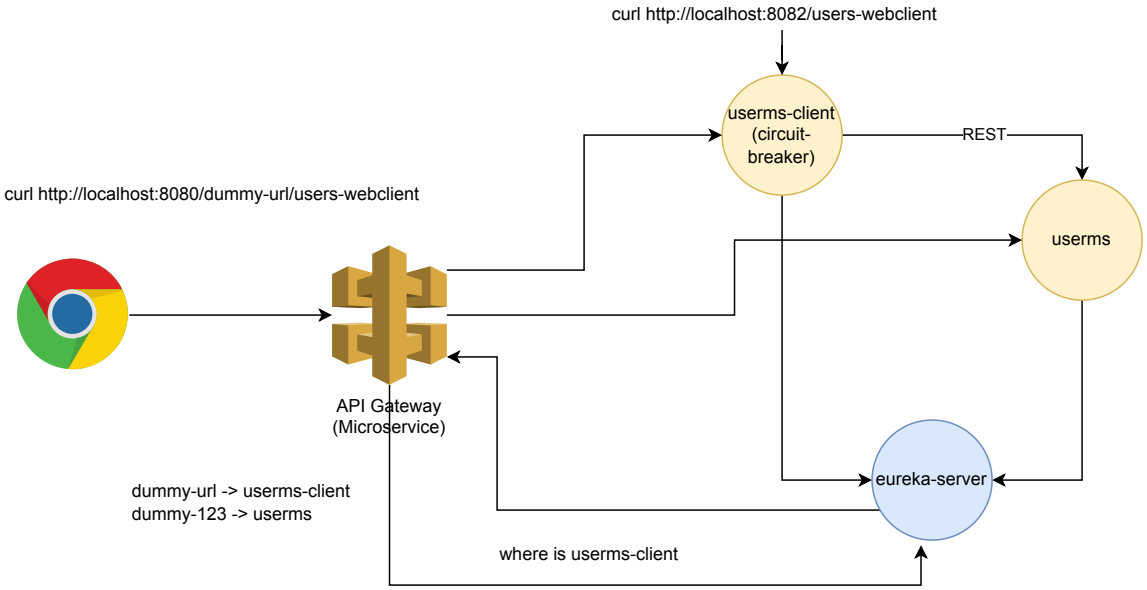


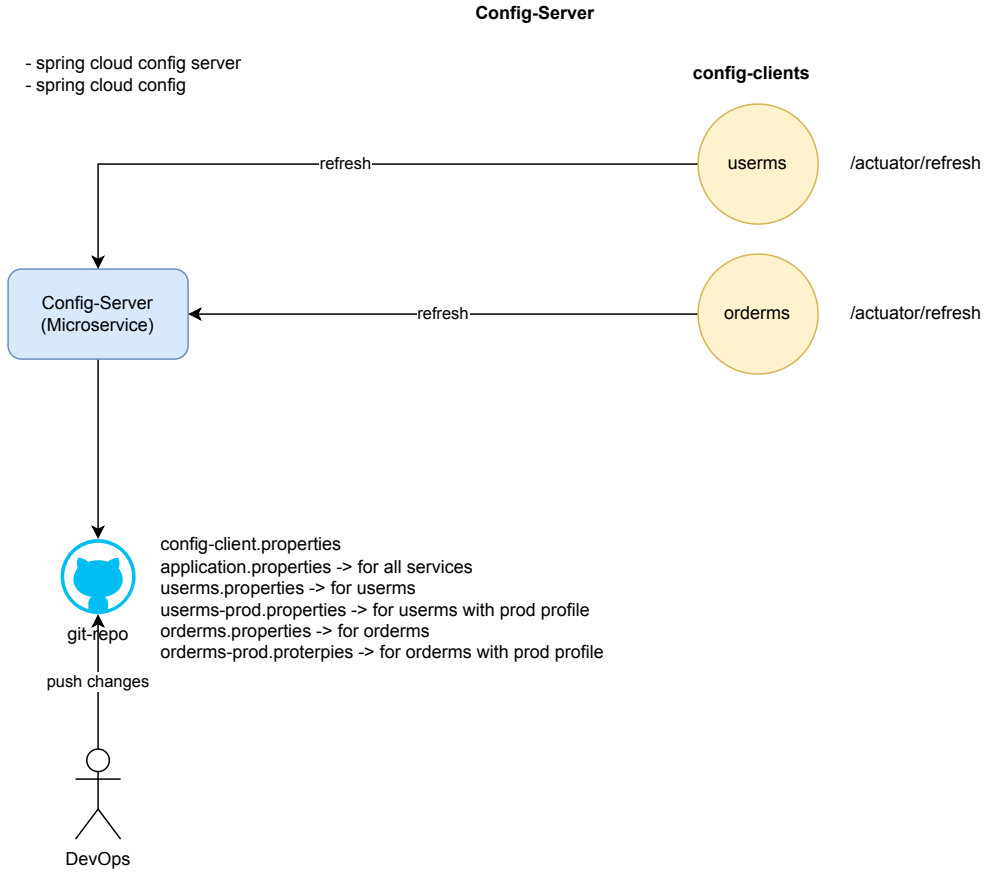
V=110V -> 220V
R=22ohm
i=5Amp -> 10Amp

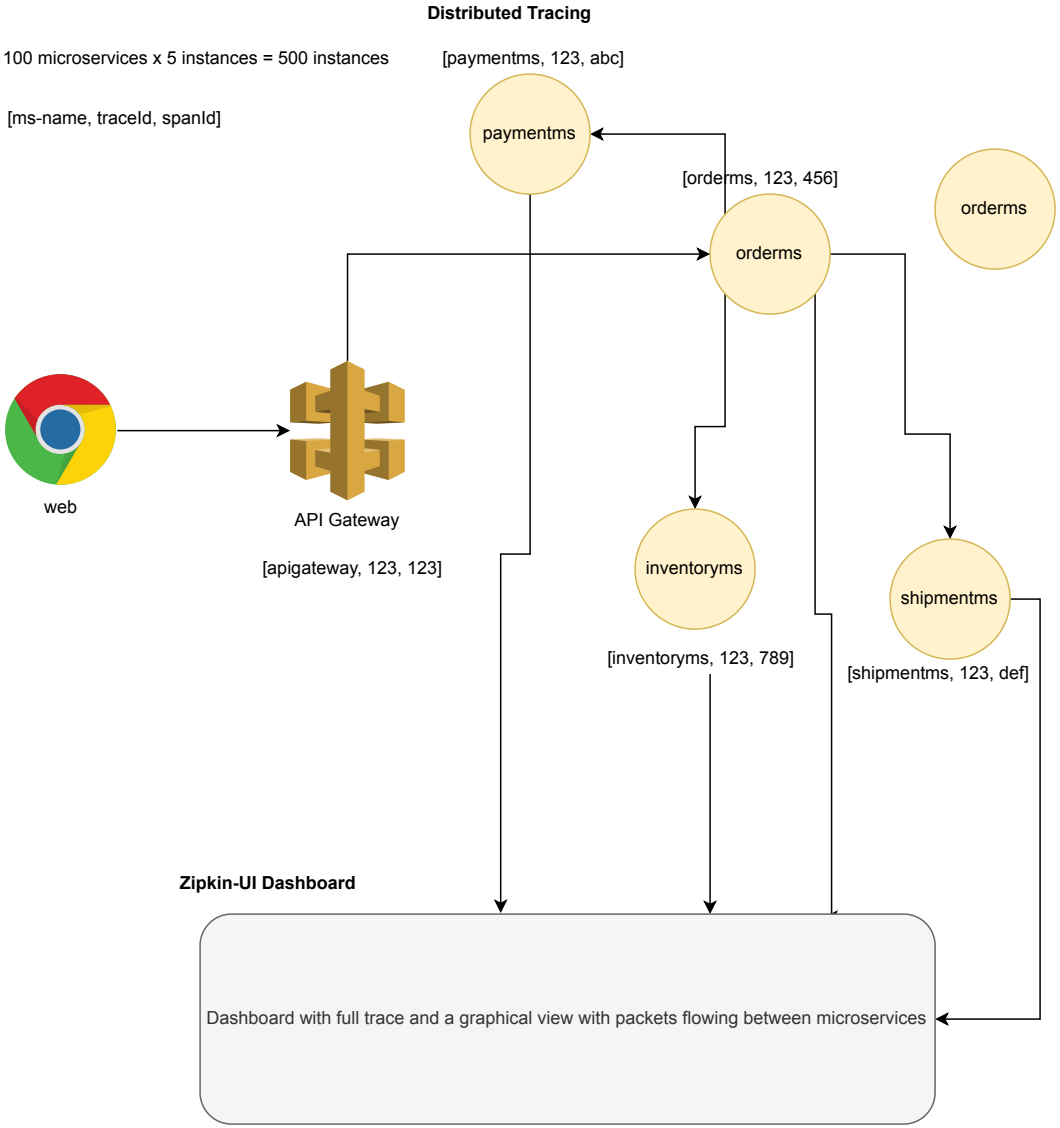
Circuit Breaker
(Resiliency)



API Gateway


curl http://localhost:8082/users-webclient
userms-client (circuit-breaker)
REST
curl http://localhost:8080/dummy-url/users-webclient
userms
API Gateway (Microservice)
eureka-server
dummy-url -> userms-client
dummy-123 -> userms
where is userms-client


Config-Server
- spring cloud config server
- spring cloud config
config-clients
refresh
userms
/actuator/refresh
Config-Server
(Microservice)
refresh
orderms
/actuator/refresh
config-client.properties
application.properties -> for all services
userms.properties -> for userms
userms-prod.properties -> for userms with prod profile
orderms.properties -> for orderms
orderms-prod.proterpies -> for orderms with prod profile
git-repo
push changes
DevOps


Distributed Tracing
100 microservices x 5 instances = 500 instances
[ms-name, traceId, spanId]
[paymentms, 123, abc]
paymentms
[orderms, 123, 456]
orderms
orderms
web
API Gateway
[apigateway, 123, 123]
inventoryms
[inventoryms, 123, 789]
shipmentms
[shipmentms, 123, def]
Zipkin-UI Dashboard
Dashboard with full trace and a graphical view with packets flowing between microservices