



Microservices & Cloud Computing – Part 2

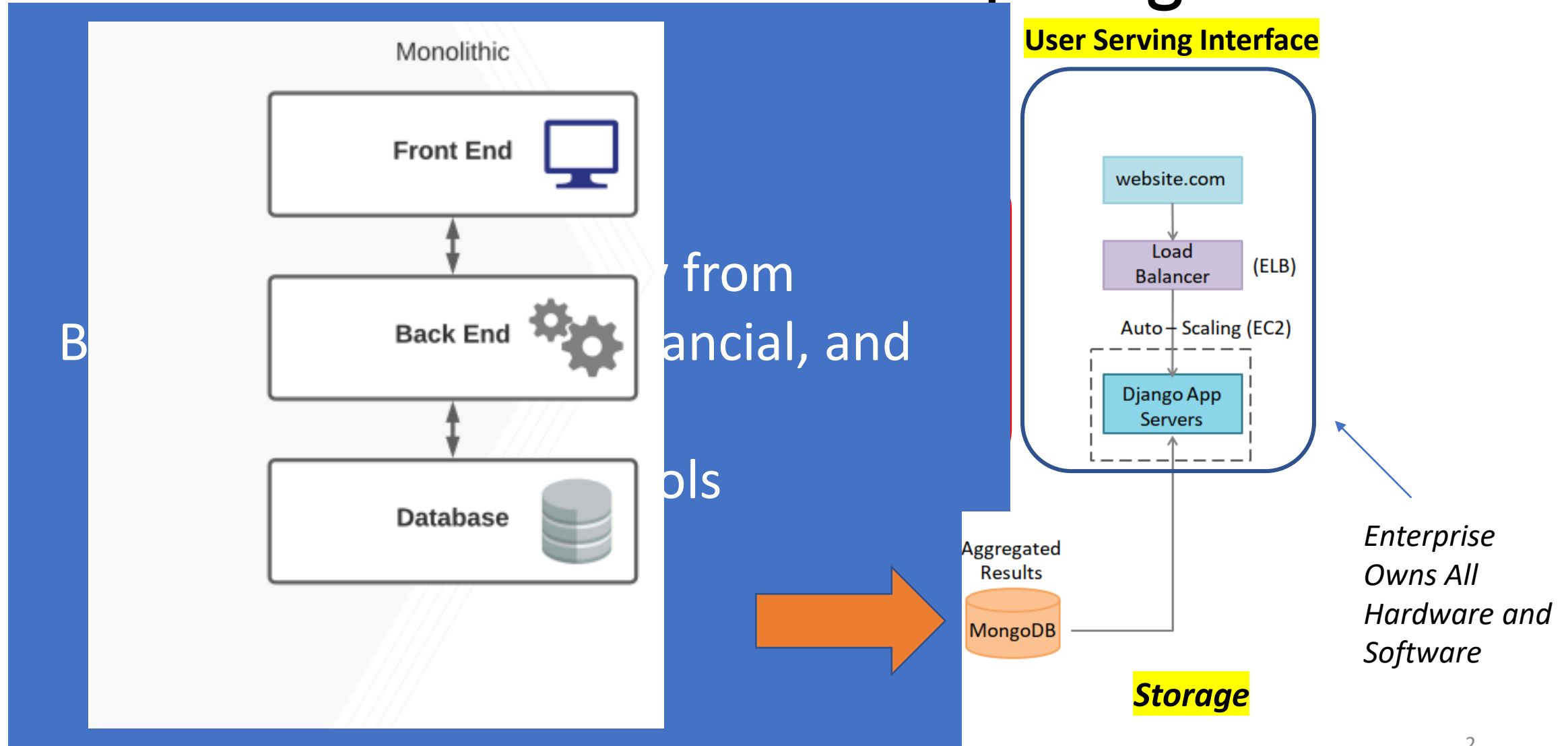
ECE 4150

Vijay Madisetti

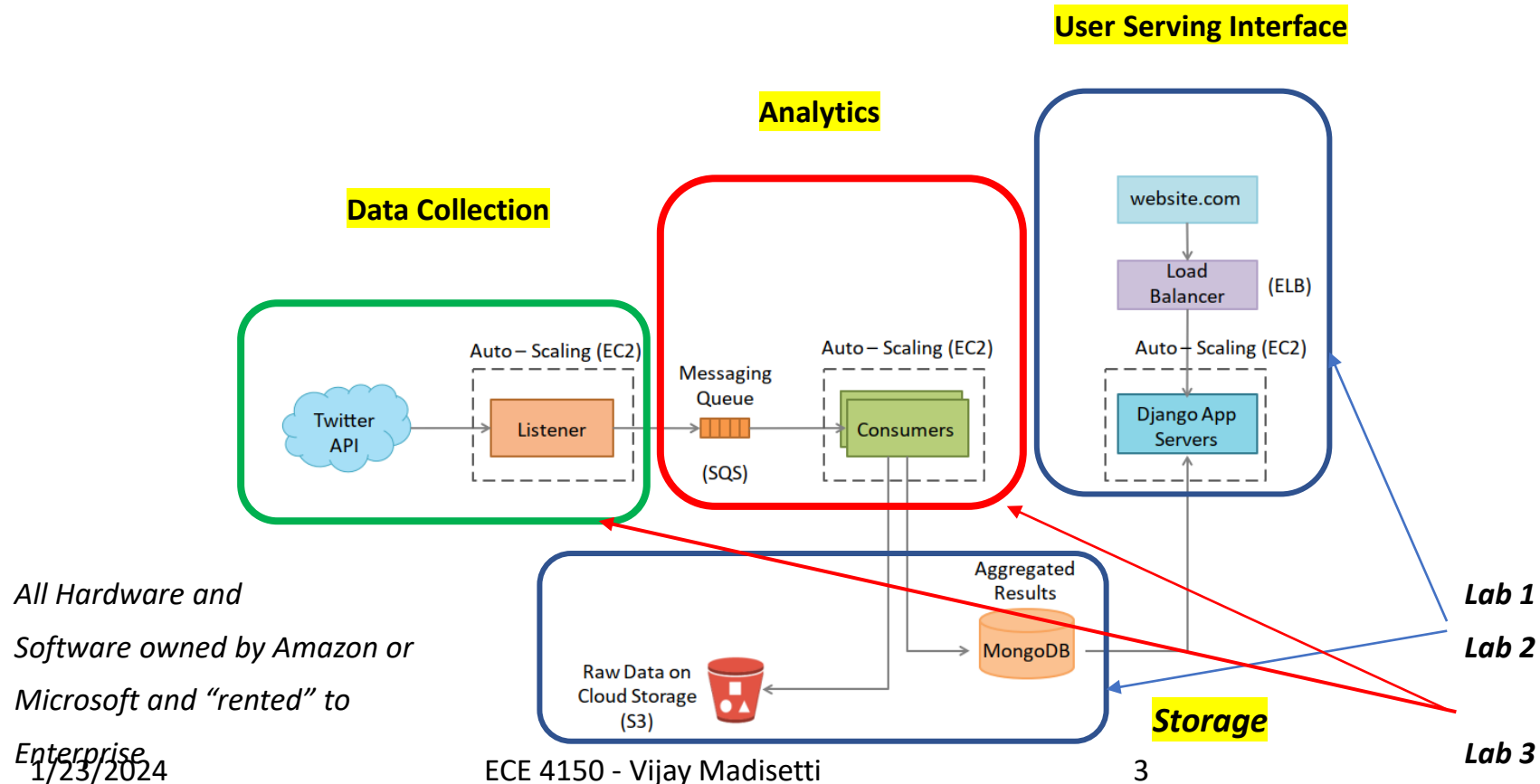
Spring 2024

Georgia Tech

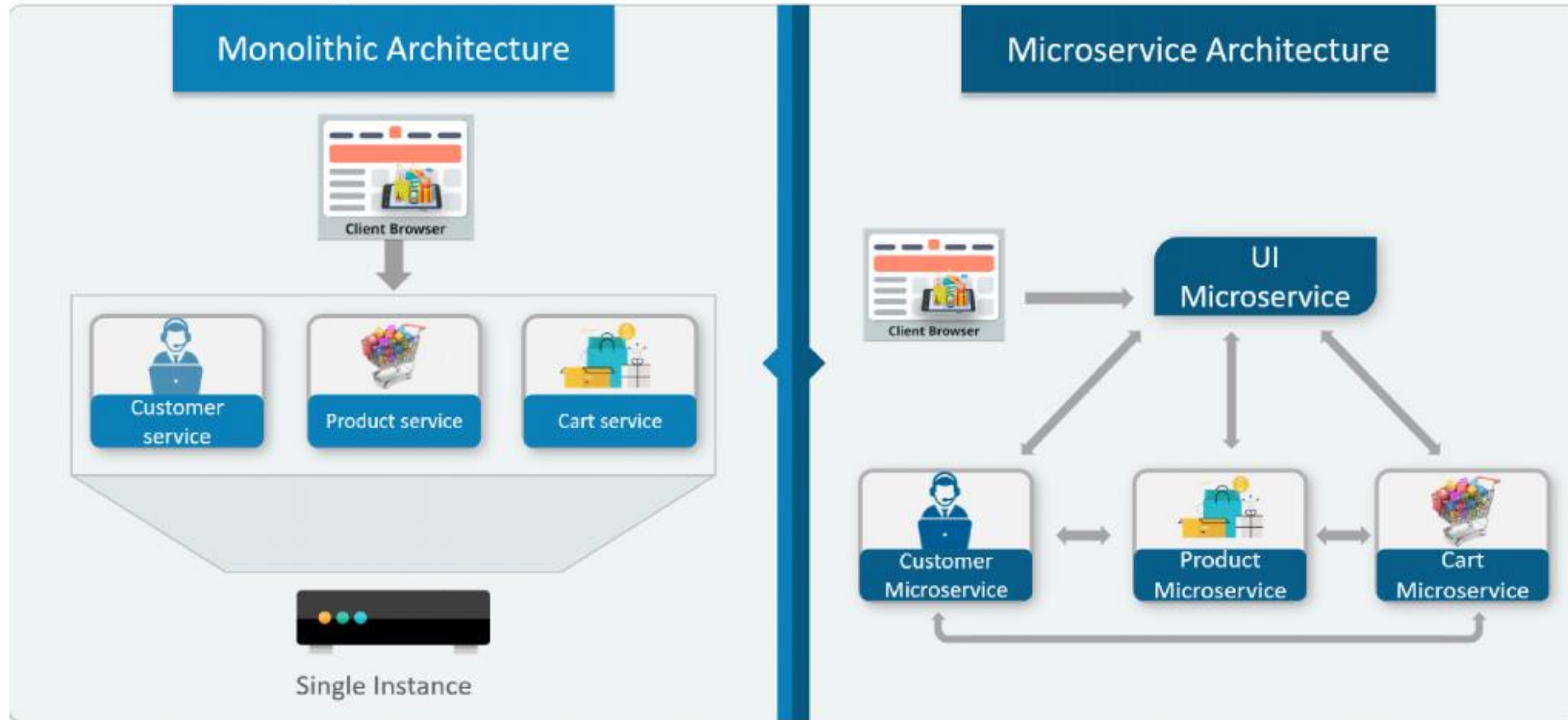
How Enterprise Applications Were *Monolithic* Prior to Cloud Computing



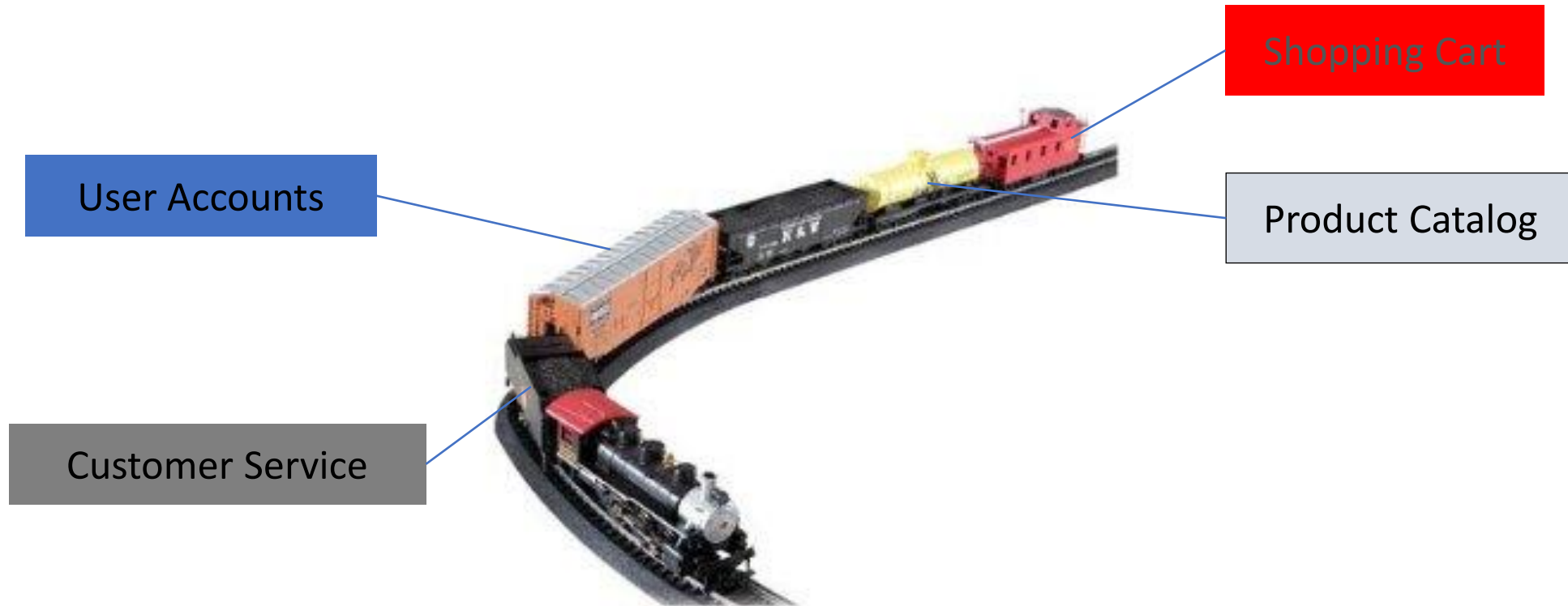
Evolution to Cloud Architectures



Monolithic versus Microservices Approach



Evolution of a Monolithic App



MicroServices – separate single purpose services



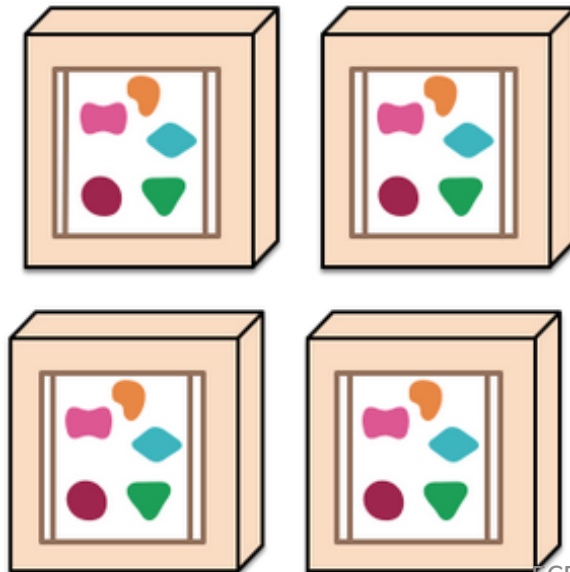
The Scaling Cube in Action via Microservices

Which is why we need HDFS, Storm, Zookeeper, Leader Election, Queues, Brokers, Load Balancers, etc., etc., - Distributed Computing & Data

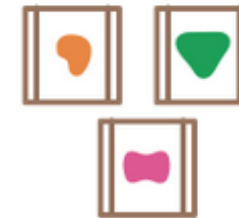
A monolithic application puts all its functionality into a single process...



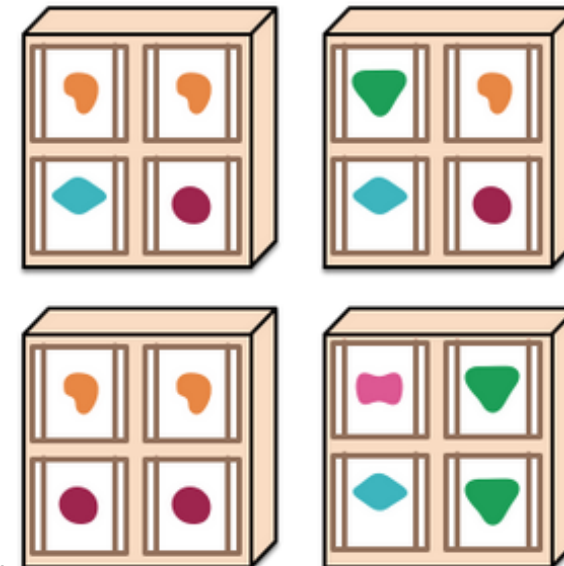
... and scales by replicating the monolith on multiple servers



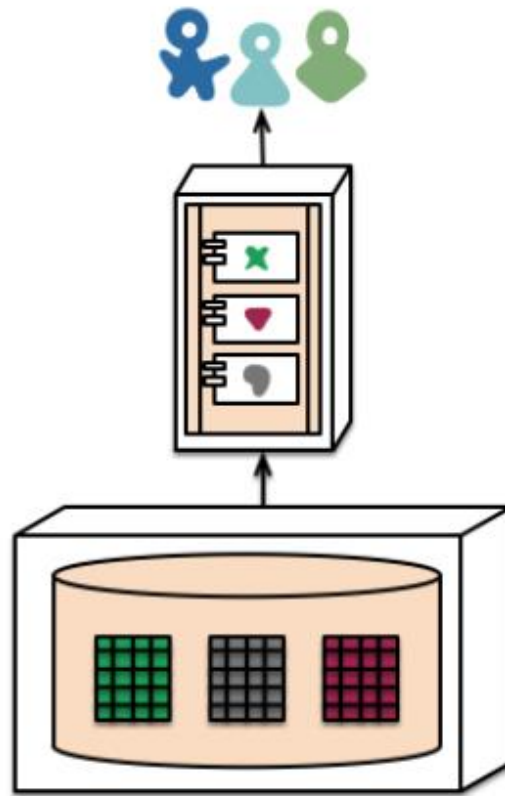
A microservices architecture puts each element of functionality into a separate service...



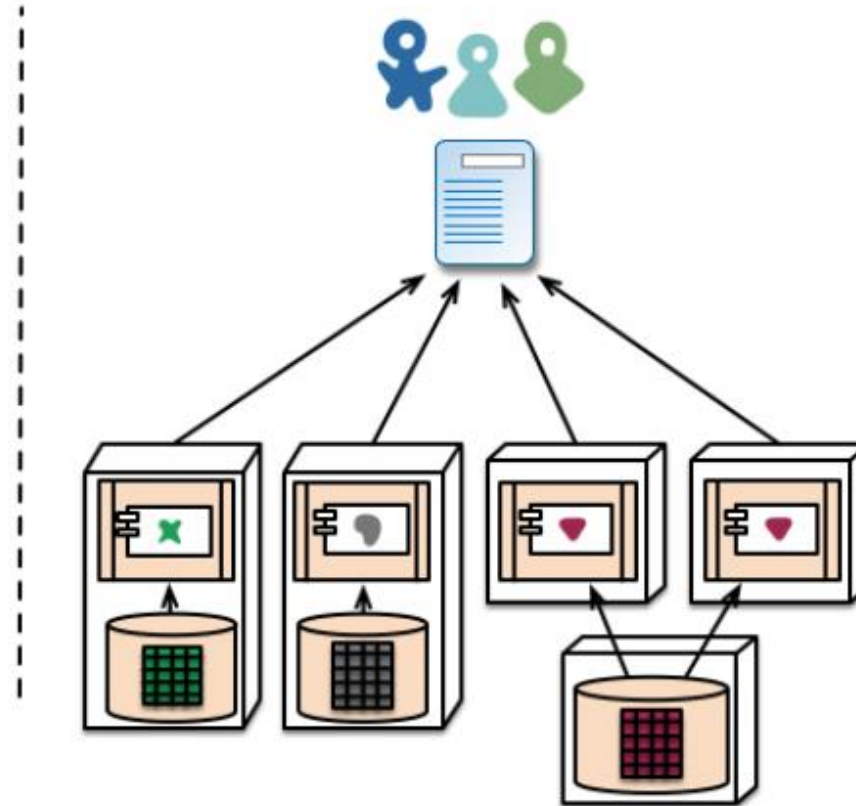
... and scales by distributing these services across servers, replicating as needed.



Replication and Partitioning

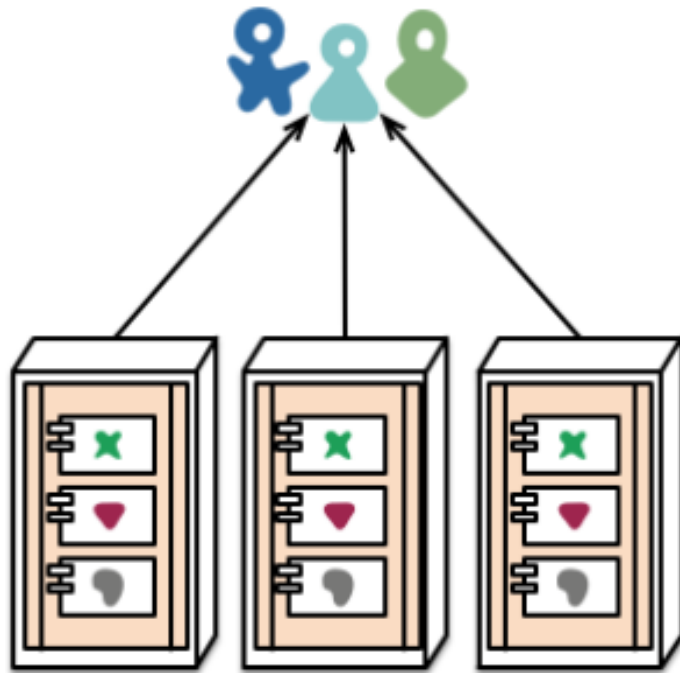


monolith - single database

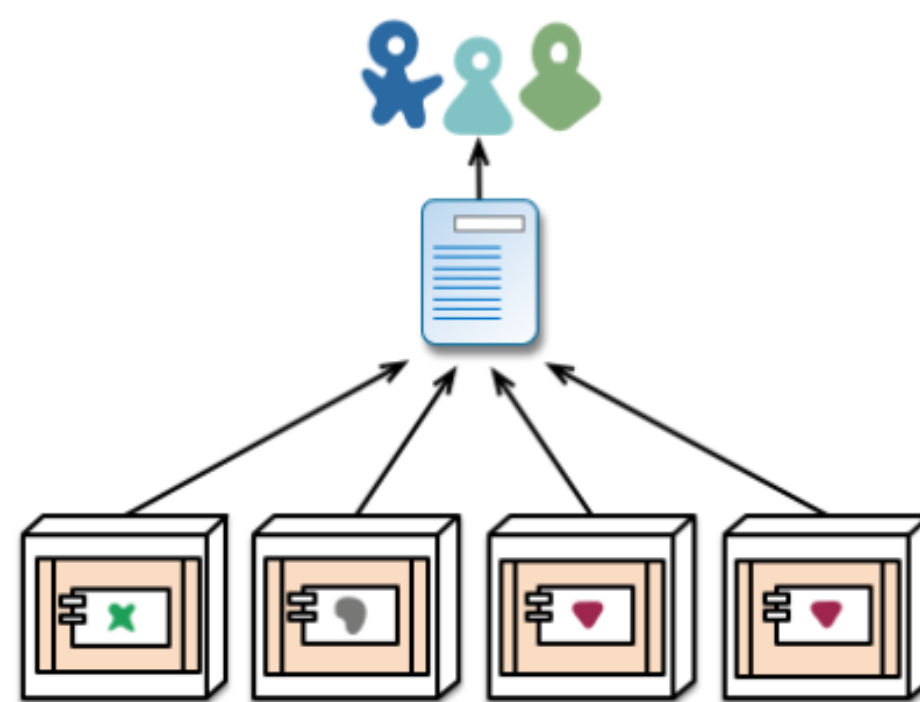


microservices - application databases

Replication and Partitioning



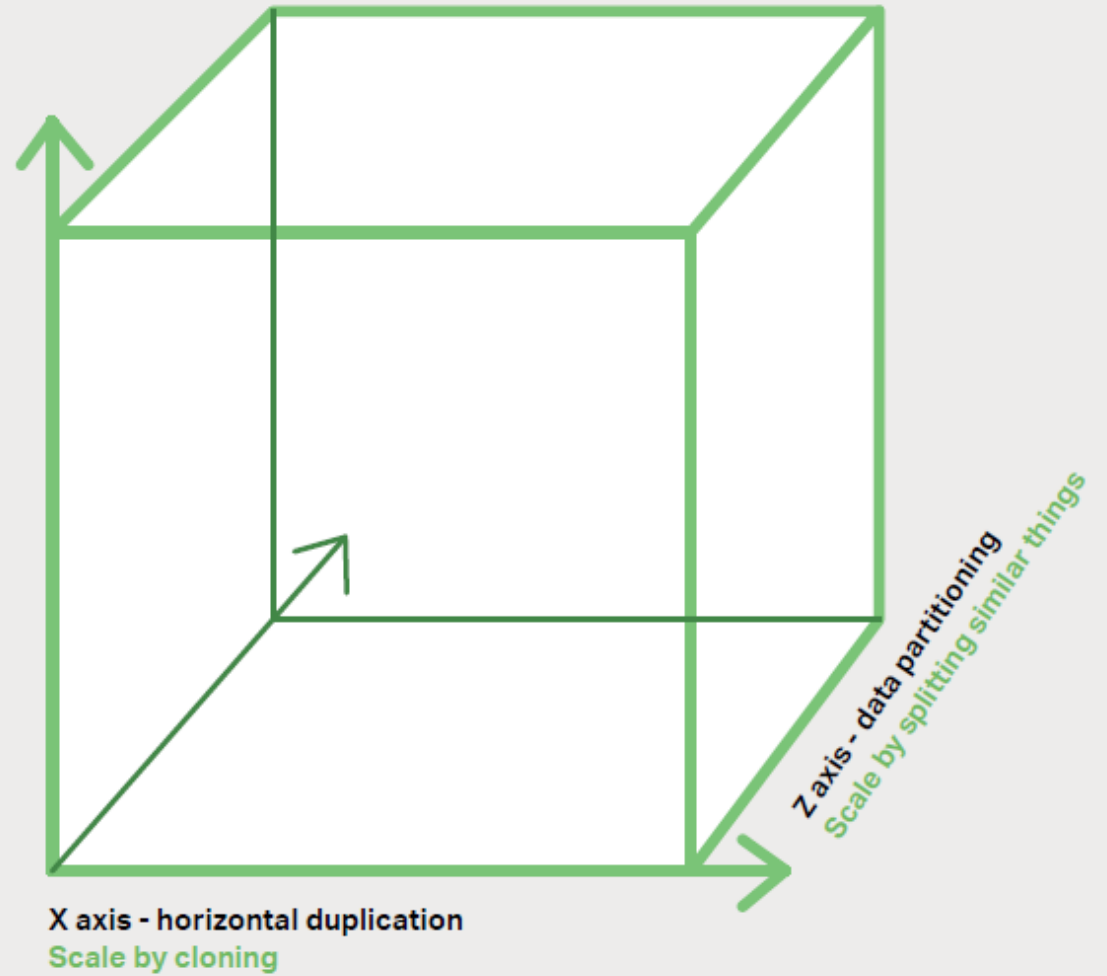
monolith - multiple modules in the same process



microservices - modules running in different processes

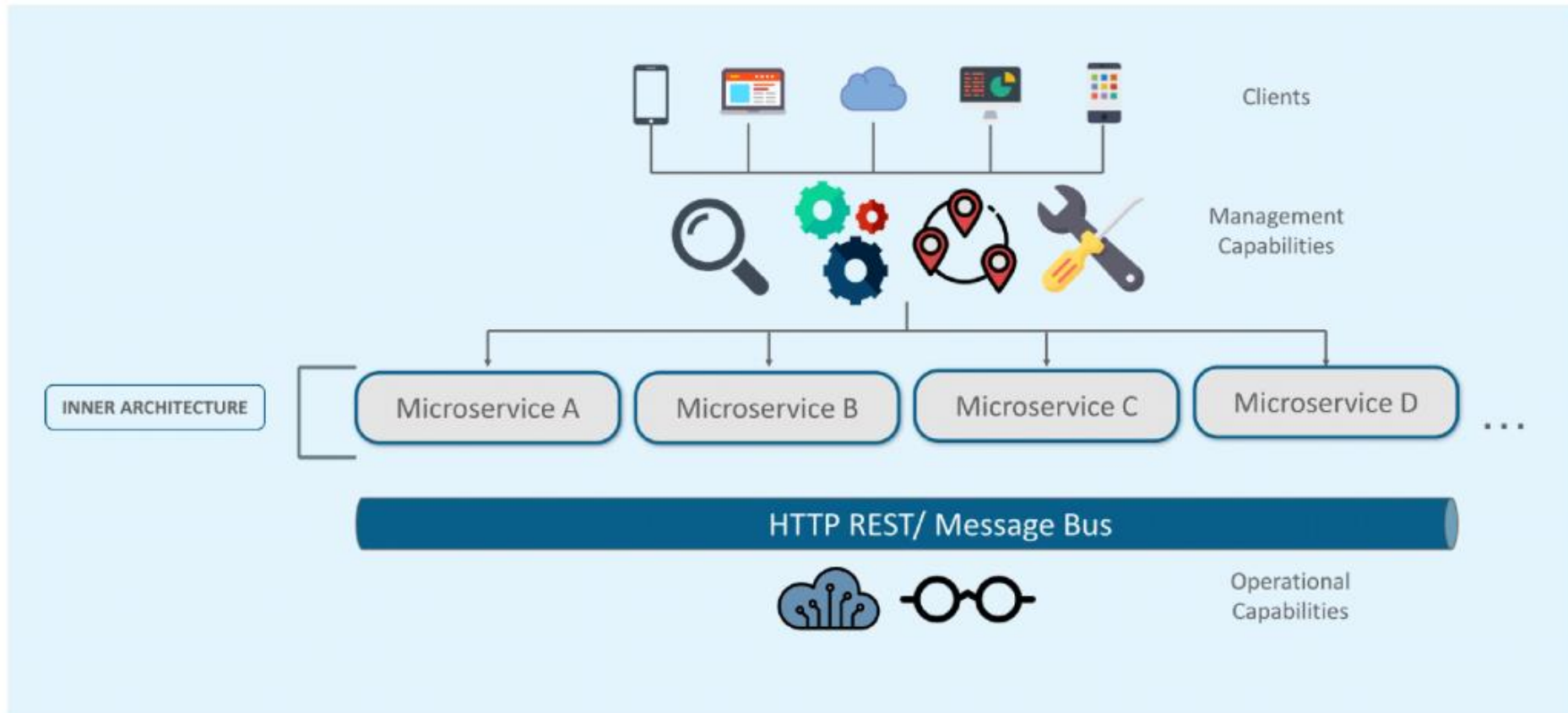
The Scaling Cube

Y axis -
functional
decomposition
Scale by splitting
different things



Microservices-based decomposition

Architecture of Microservices

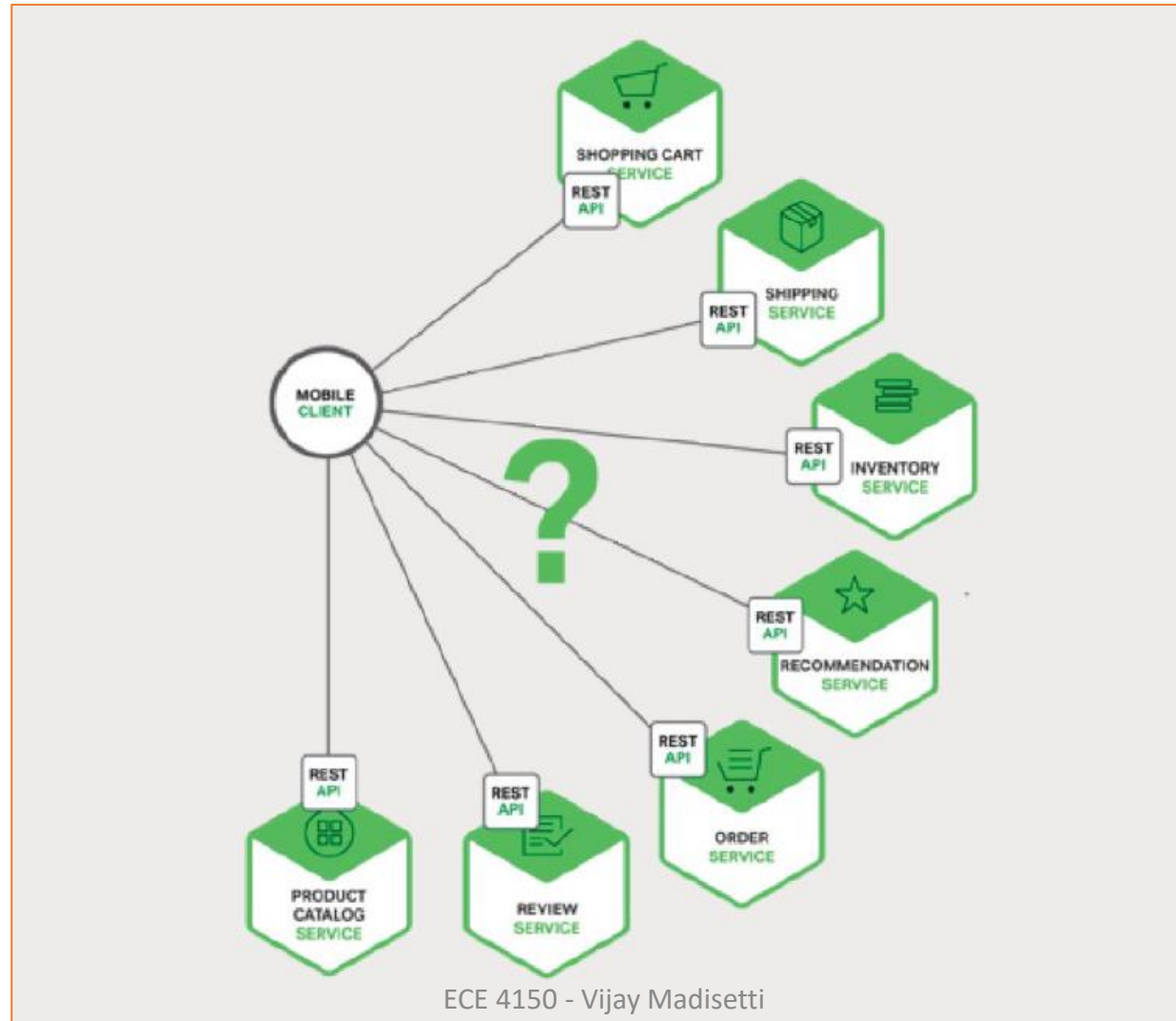




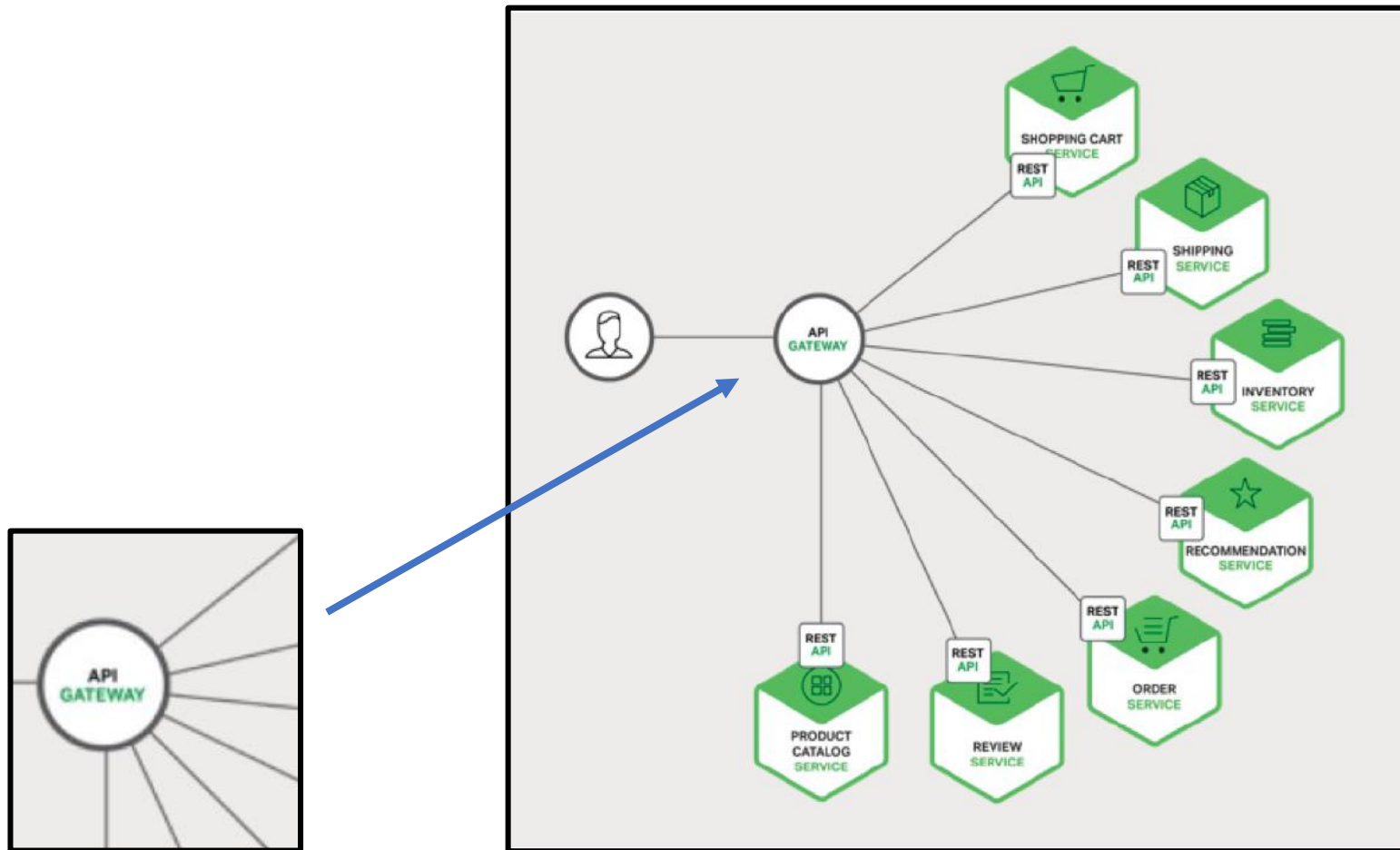
API Gateways

Important to Cloud Deployment

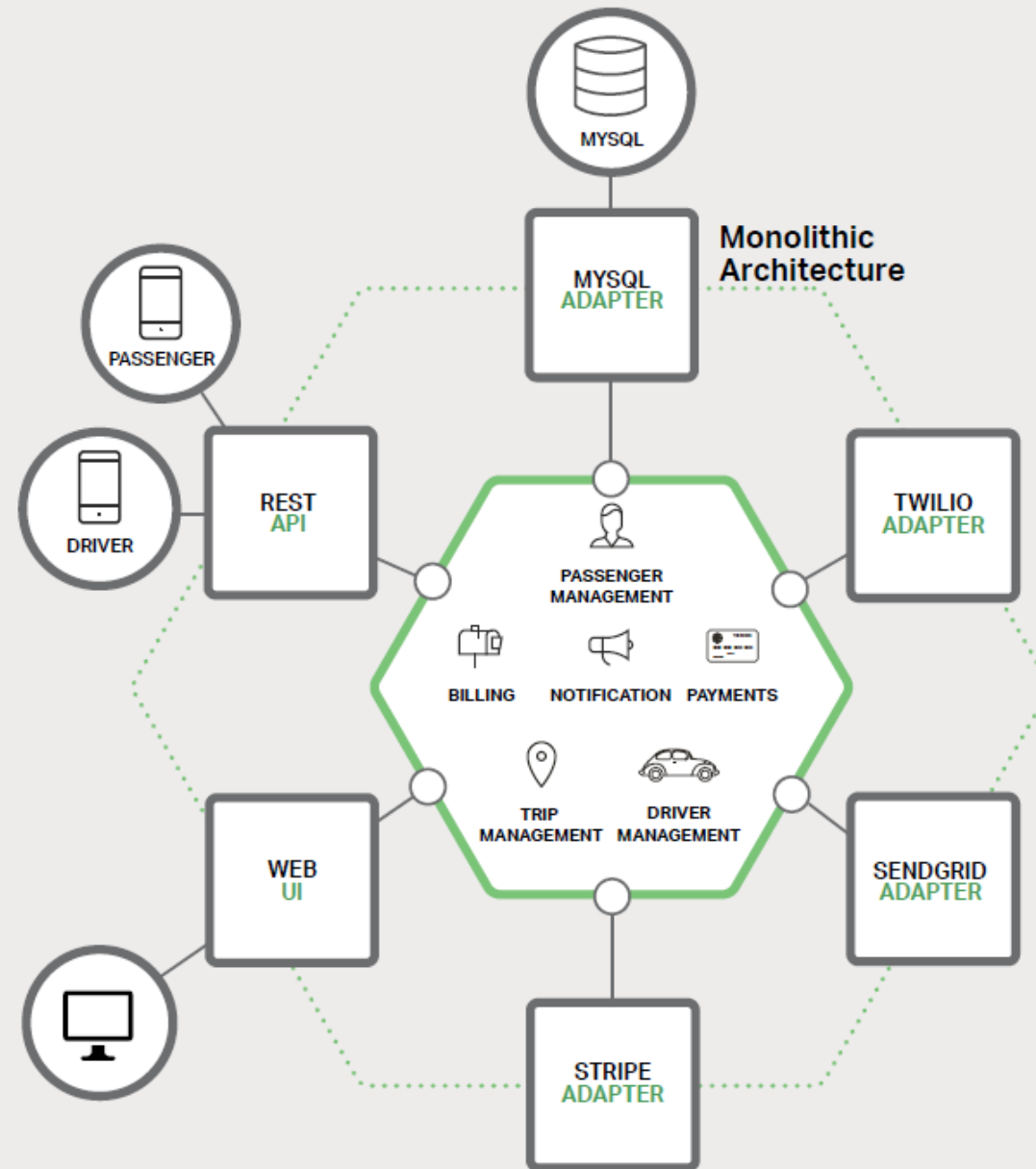
How to have single mobile UI interface with multiple “microservices” ? Option 1



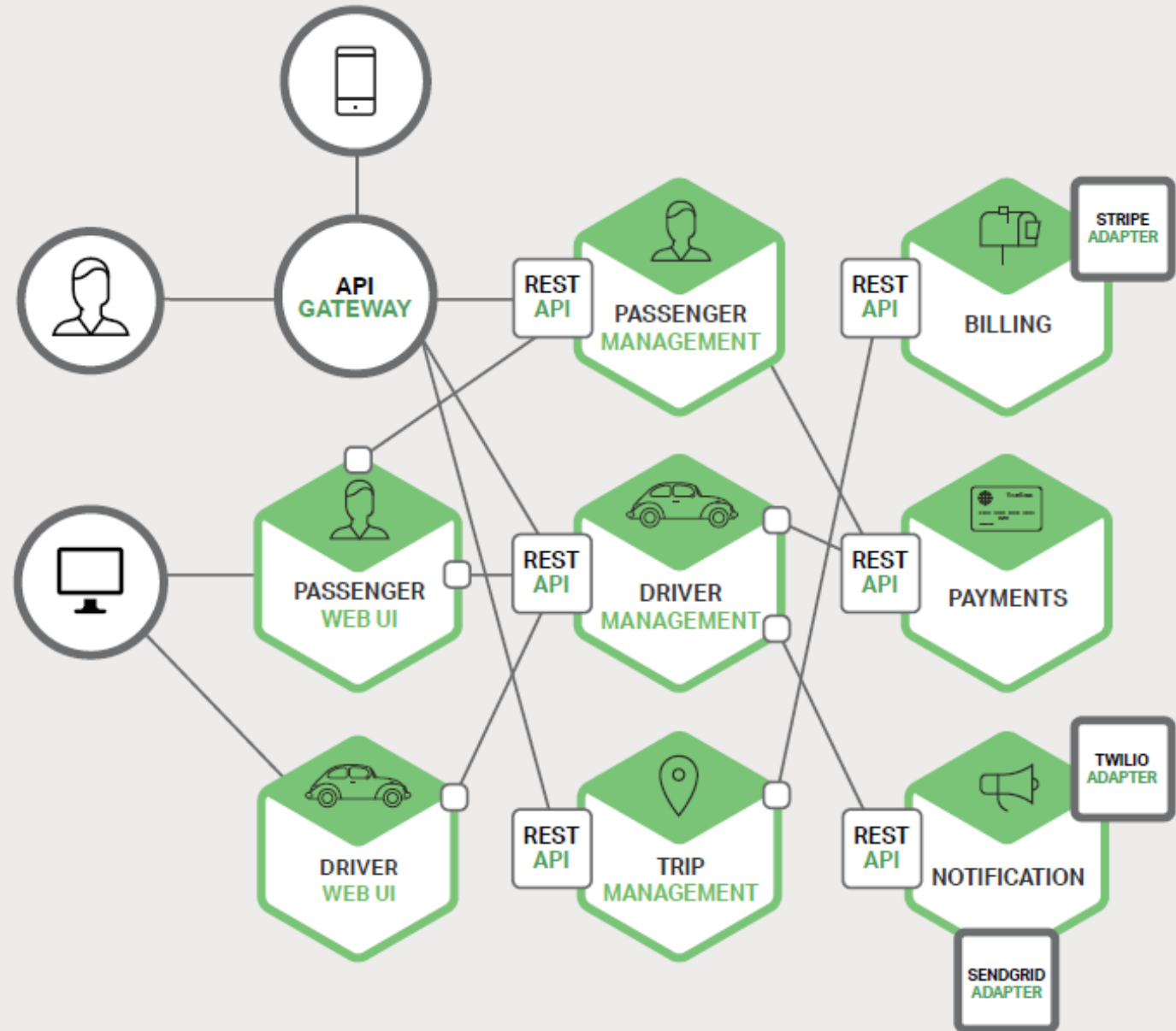
How to have the mobile UI interface with multiple microservices ? Option 2

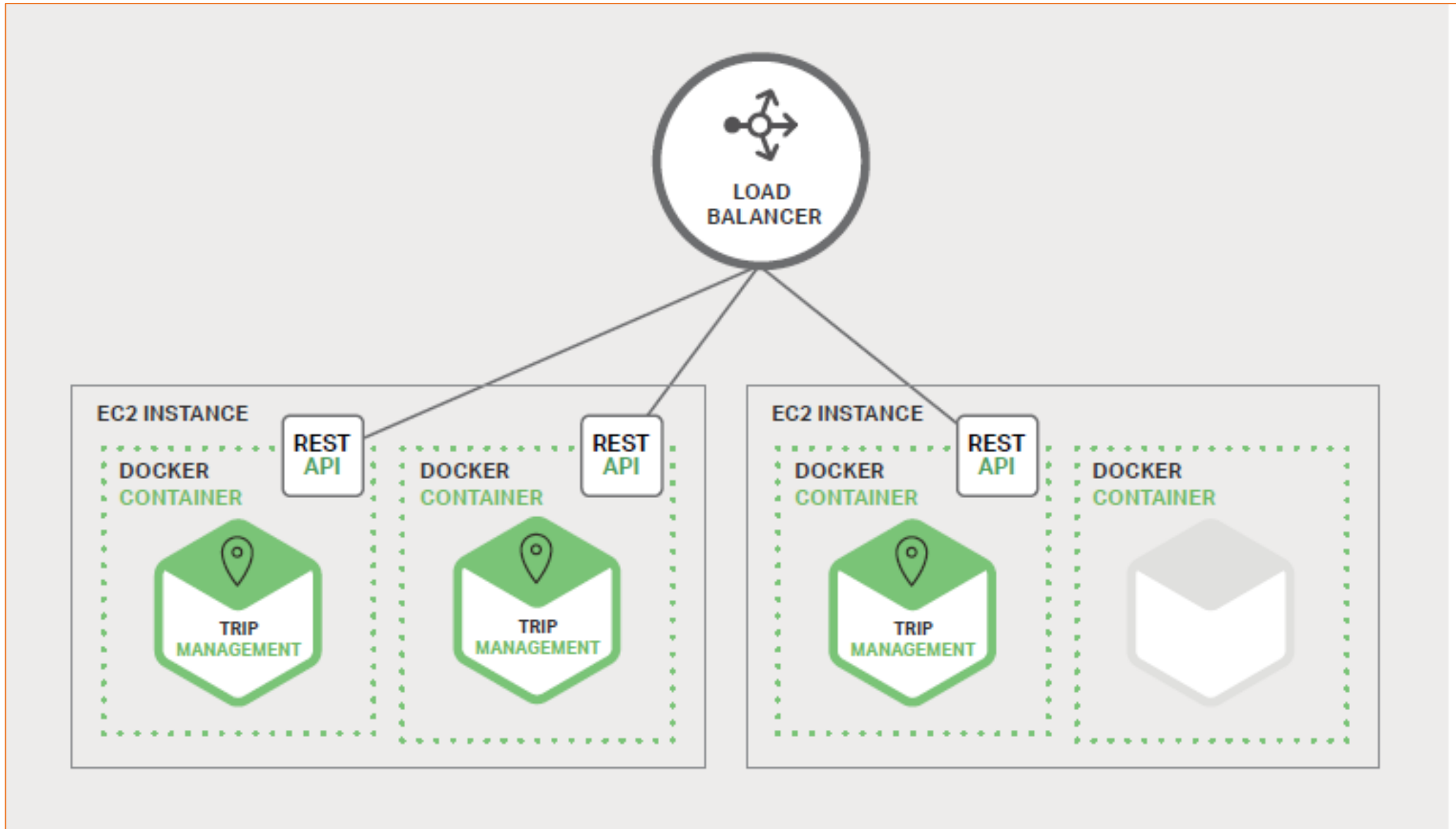


Uber-like Application - Analysis

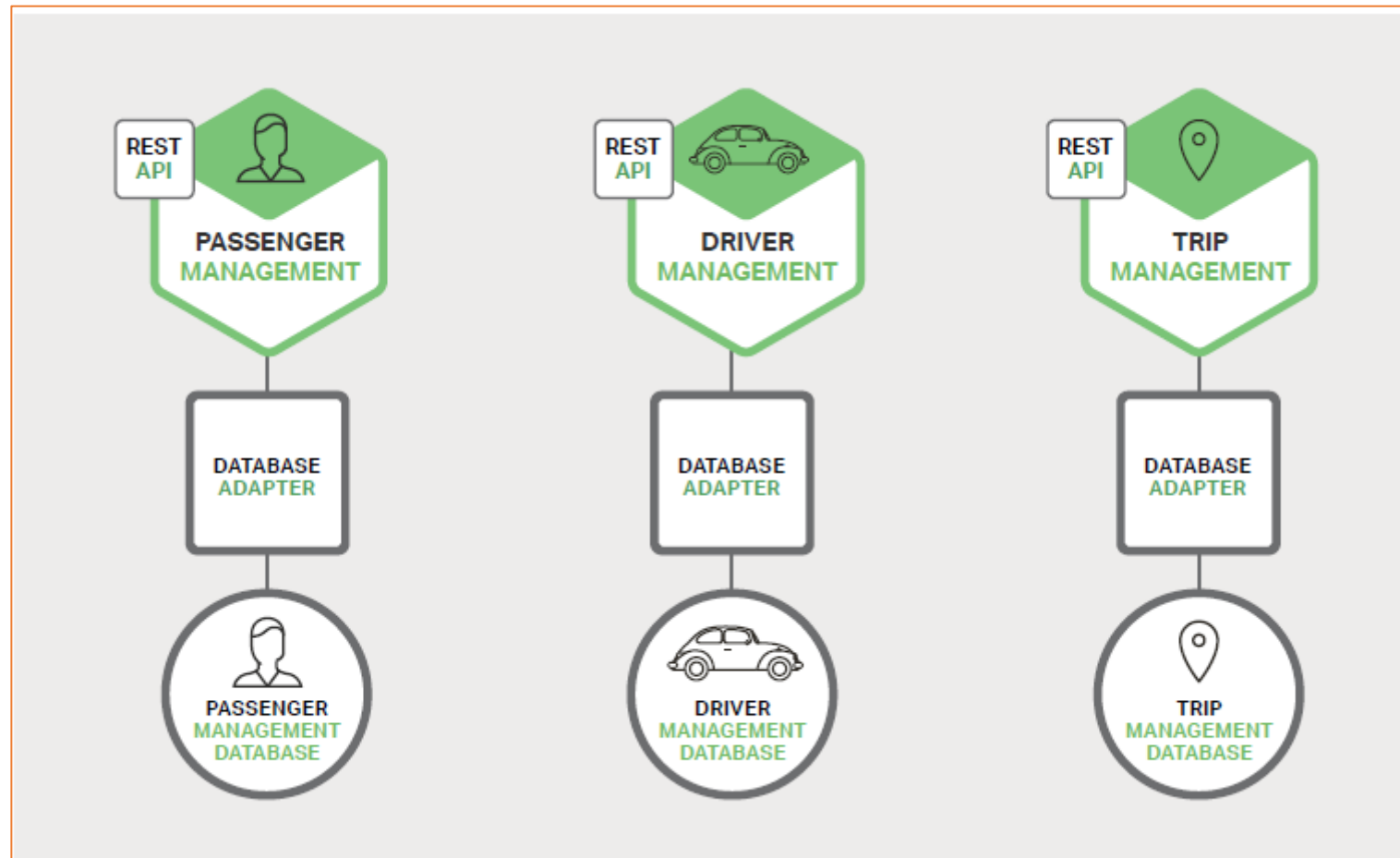


Microservices Architecture of Uber-like System





Database Architecture



Amazon.com implements hundreds of microservices

1. ORDER HISTORY
2. REVIEWS
3. BASIC PRODUCT INFO
4. RECOMMENDATION
5. INVENTORY
6. SHIPPING

NEW & INTERESTING FINDS ON AMAZON EXPLORE

Shop **cyber monday** deals

amazon prime Books madisetti

Departments Your Pickup Location Browsing History Vijay's Amazon.com Cyber Monday Deals Week Gift Cards

Books Advanced Search New Releases NEW! Amazon Charts Best Sellers & More The New York Times® Best Sellers Children's Books Textbooks Textbook Rentals Sell Us Your Books Best Books of the Month

KINDLE HOLIDAY DEALS Save on all things Kindle Shop now

Back to search results for "madisetti"

You purchased this item on July 3, 2017. View this order

Look inside

Blockchain Applications
A Hands-On Approach

Arshdeep Bahga • Vijay Madisetti

See all 2 images

Blockchain Applications: A Hands-On Approach

2017
by Arshdeep Bahga (Author), Vijay Madisetti (Author)
★★★★☆ 12 customer reviews

See all 2 formats and editions

Format	Price
Hardcover	\$59.44
Paperback	\$49.79

7 Used from \$43.69 21 New from \$56.58 1 Used from \$50.04 2 New from \$49.79

In the US, the services sector provides employment to about 100 million, while the manufacturing sector provides employment to about 20 million. These sectors are highly automated, and driven by sophisticated business processes forming an integral part of the digital economy. While the applications themselves may be distributed over the Internet in time and space, the core business, regulatory, and financial aspects of the digital economy are still centralized, with the need for centralized agencies (such as banks, customs authorities, and tax agencies) to authenticate and settle payments and transactions. These centralized services often are manual, difficult to automate, and represent a bottleneck to facilitating a frictionless digital economy. The next revolutionary step in the services and manufacturing

Report incorrect product information.

Get \$5 off book purchases over \$20 Learn more

Buy New \$49.79
Qty: 1 List Price: \$60.00
Save: \$10.21 (17%)

FREE Shipping for Prime members

In Stock.

Ships from and sold by Amazon.com. Gift-wrap available.

Add to Cart

Turn on 1-Click ordering for this browser

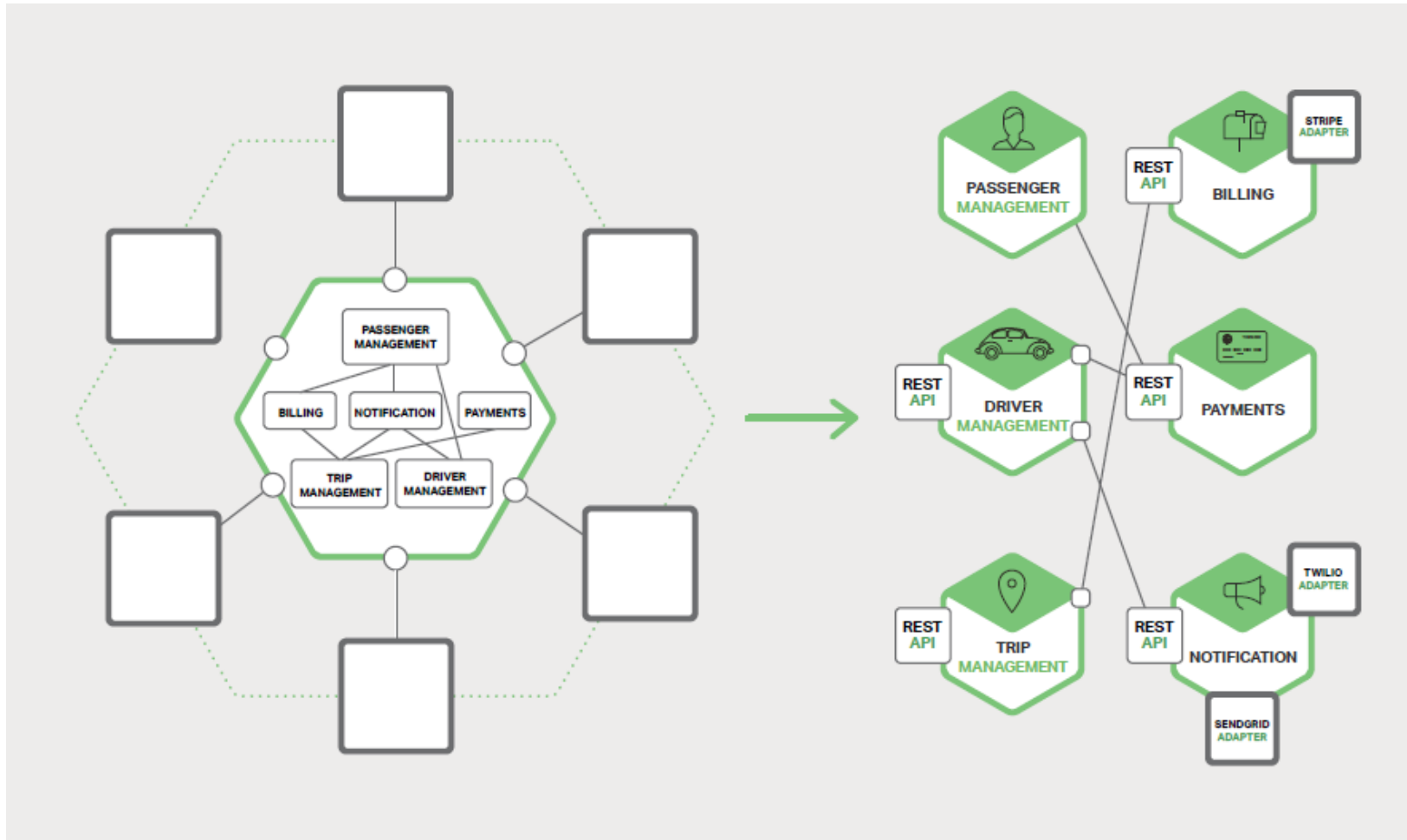
Want it Wednesday, Nov. 29? Order within 4 hrs 53 mins and choose One-Day Pickup at checkout. Details

Ship to:
Amazon@georgiatec- Atlanta - 30308

Add to List

Add to your Dash Buttons

How do Microservices Interact with Each Other ?

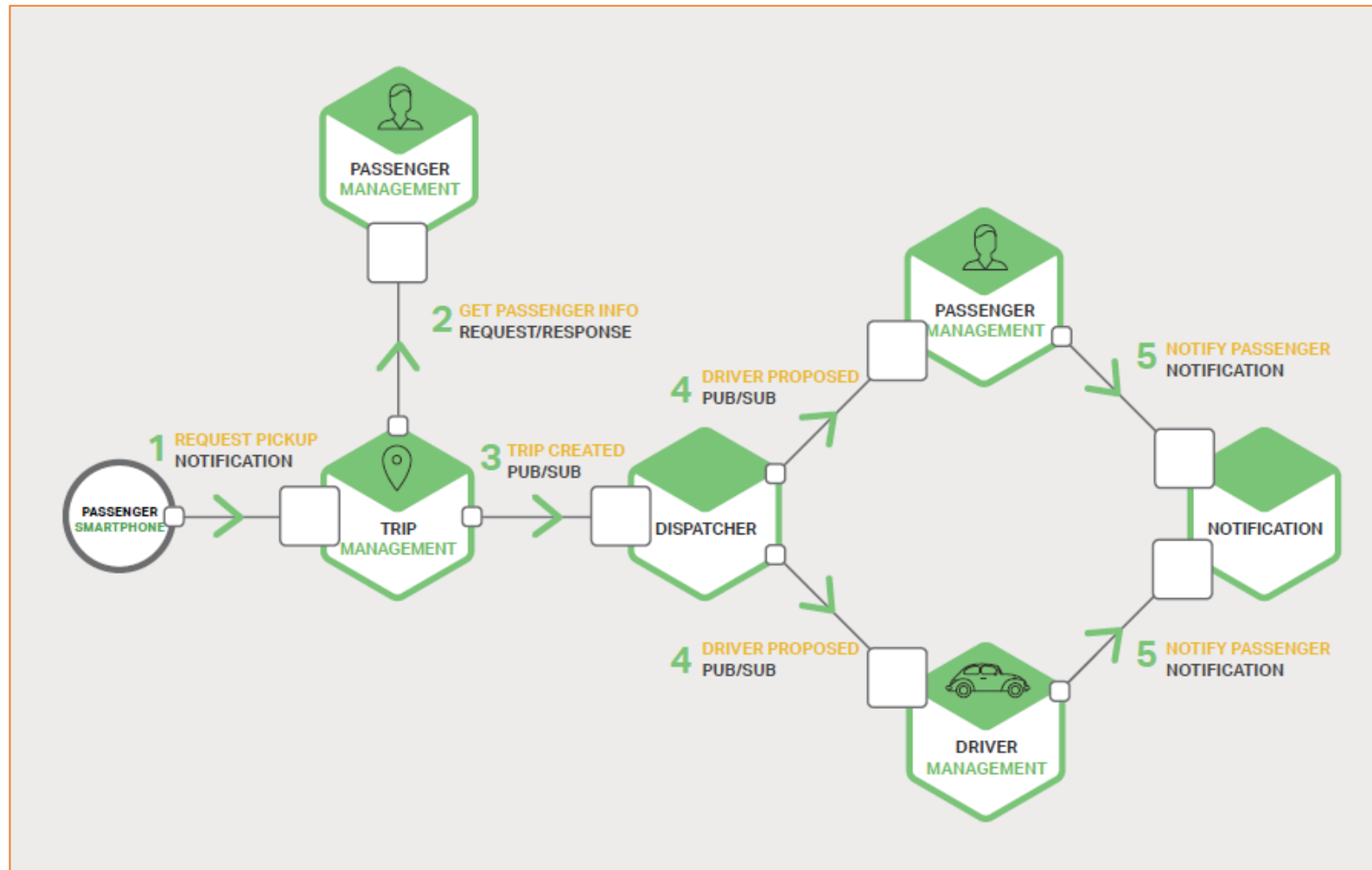


Microservice Interaction Styles

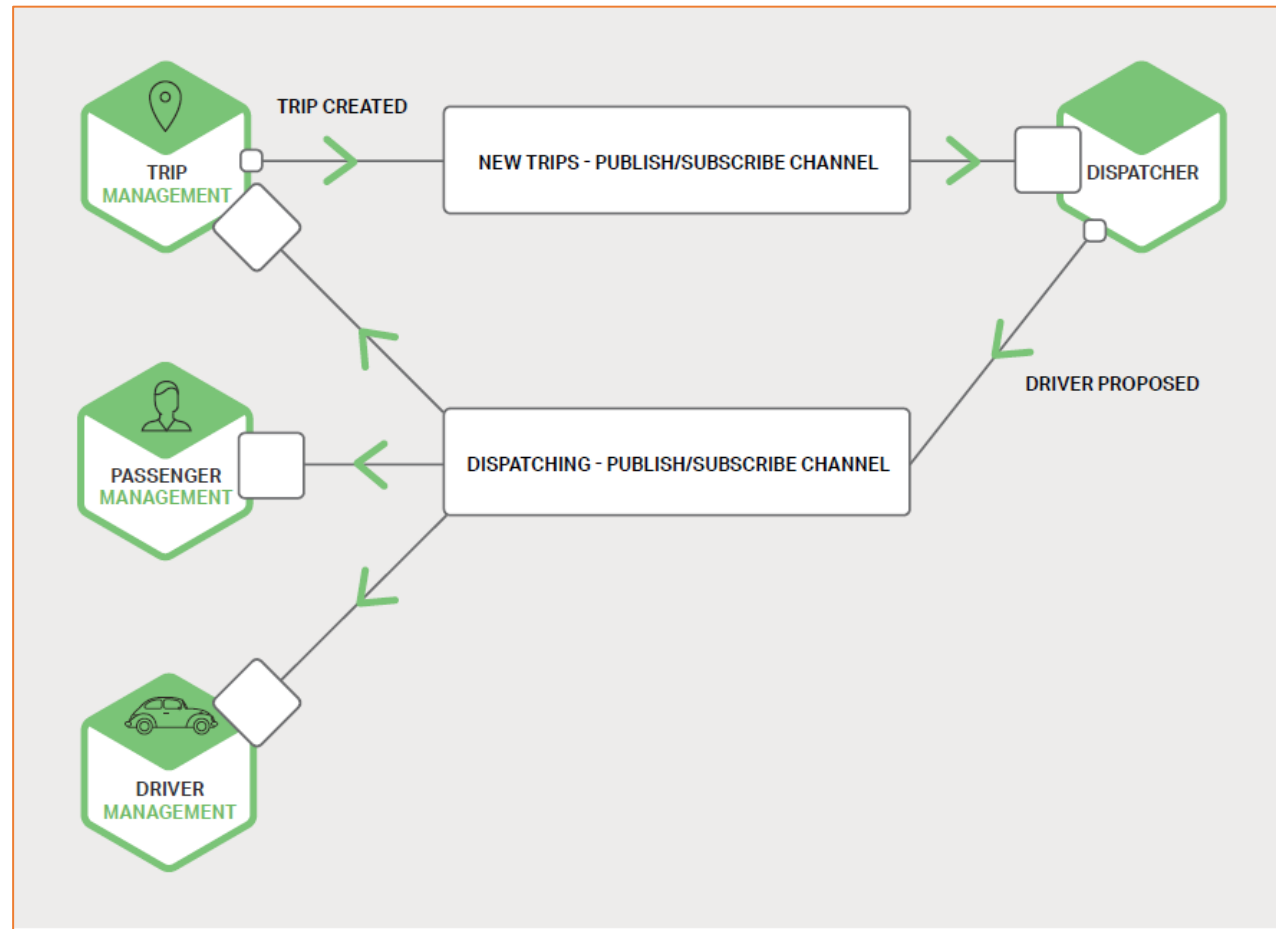
	ONE-TO-ONE	ONE-TO-MANY
SYNCHRONOUS	Request/response	—
ASYNCHRONOUS	Notification	Publish/subscribe
	Request/async response	Publish/async responses

Note: REST can be asynchronous. There is no reason a client need wait for a response
In REST, it can do other things as well. It is a programming choice. A HTTP web server is usually Request/Response

Applying Interaction Styles to Microservices Architecture of Uber-like App

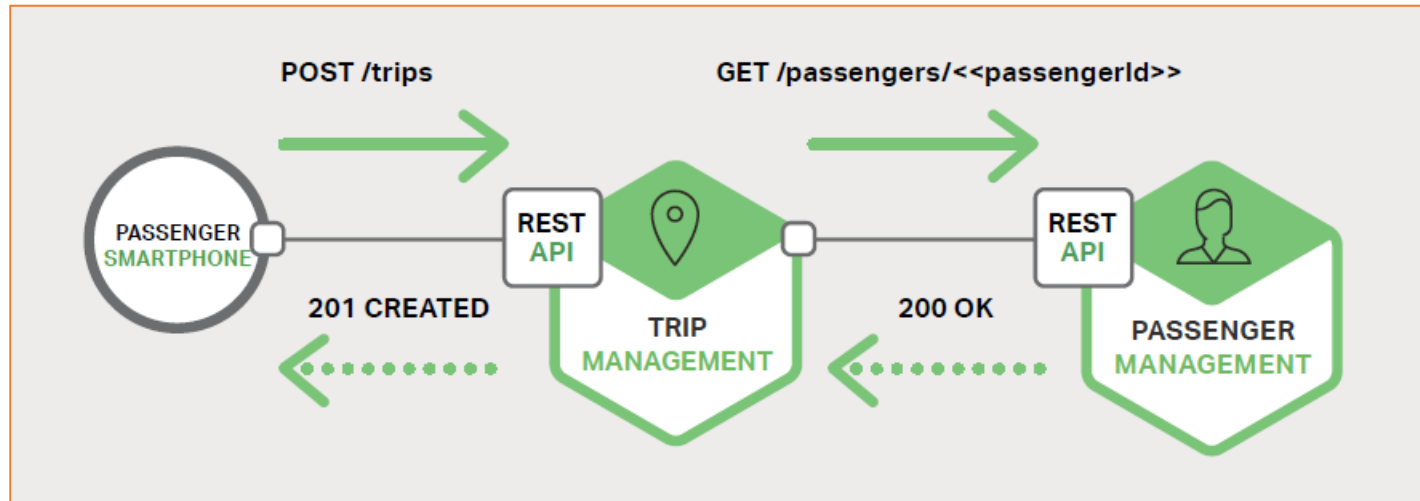


Messaging between Microservices



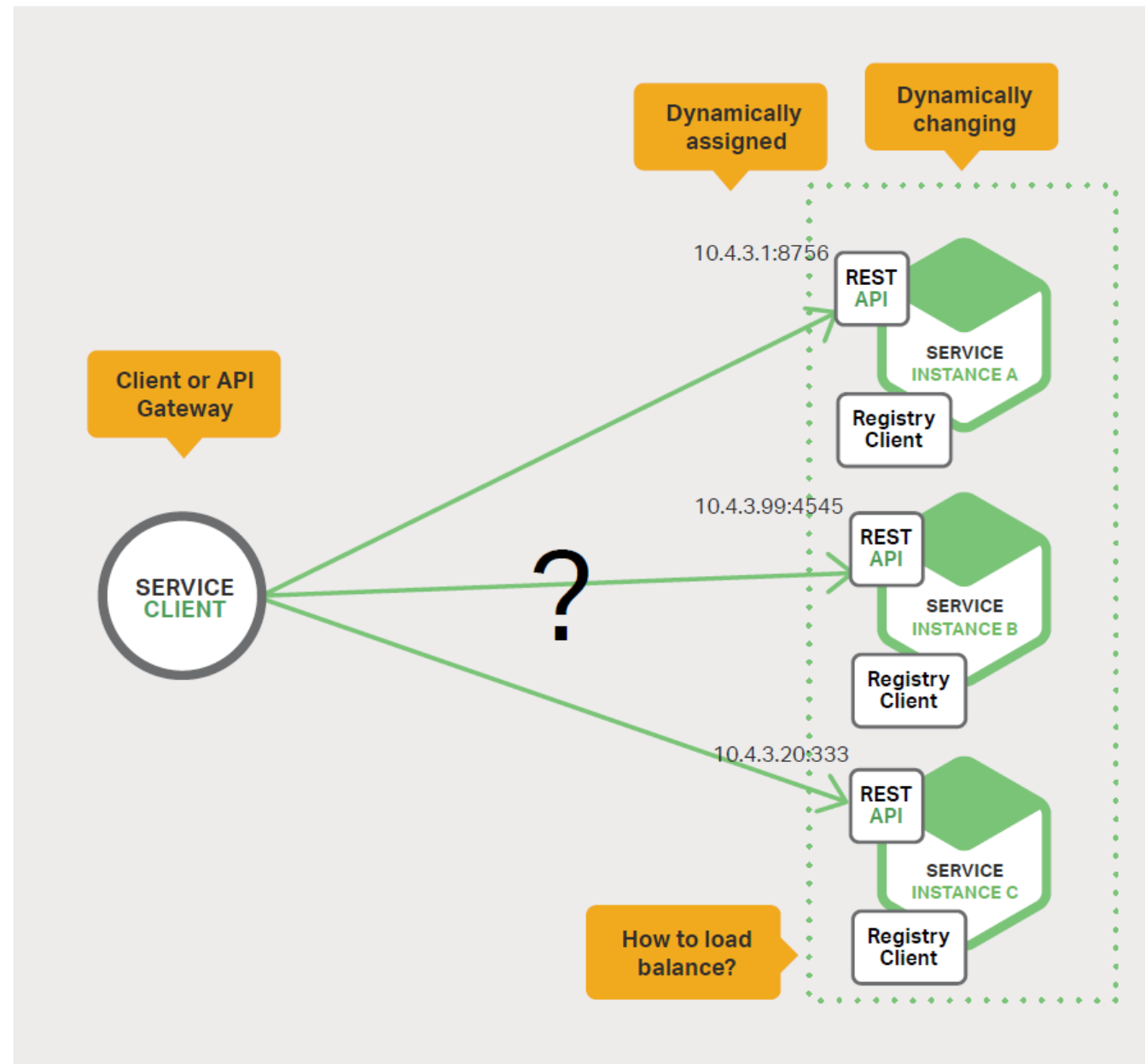
We already covered REST APIs in the class & labs

Note: Alternative to REST is the Apache Thrift API

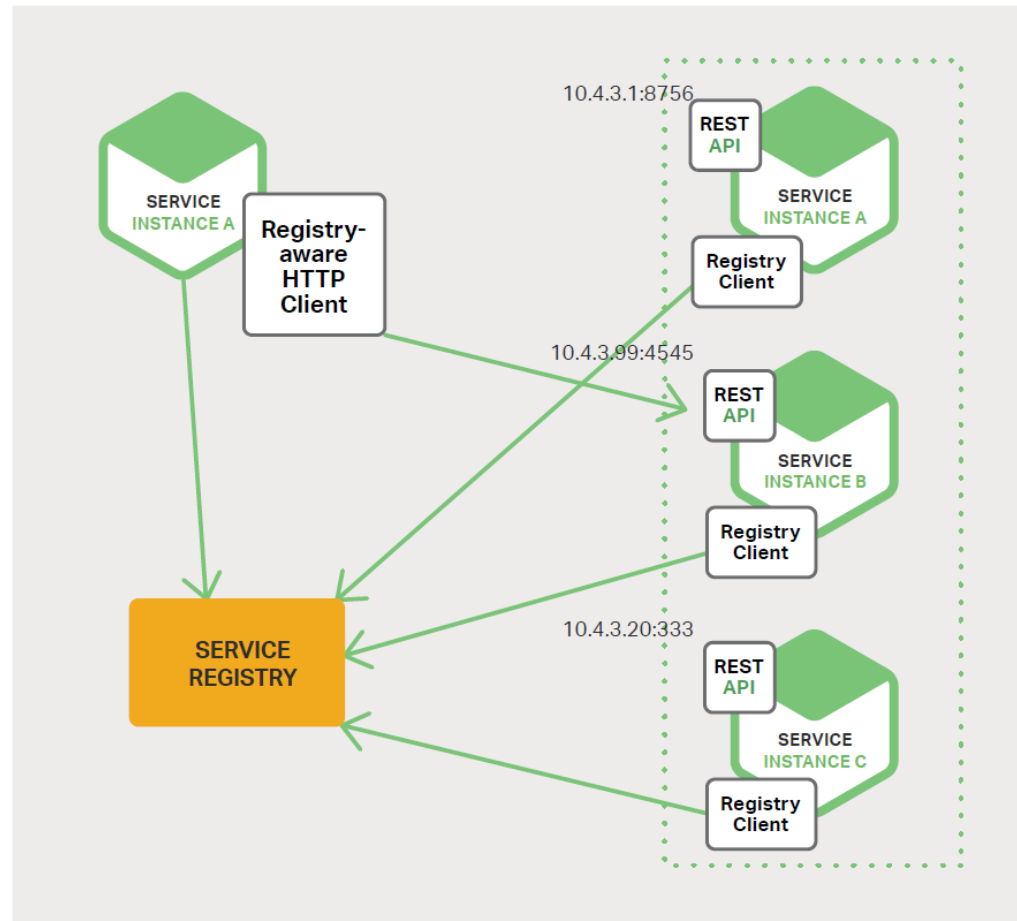


"REST provides a set of architectural constraints that, when applied as a whole, emphasizes scalability of component interactions, generality of interfaces, independent deployment of components, and intermediary components to reduce interaction latency, enforce security, and encapsulate legacy systems."
—Roy Fielding, *Architectural Styles and the Design of Network-based Software Architectures*

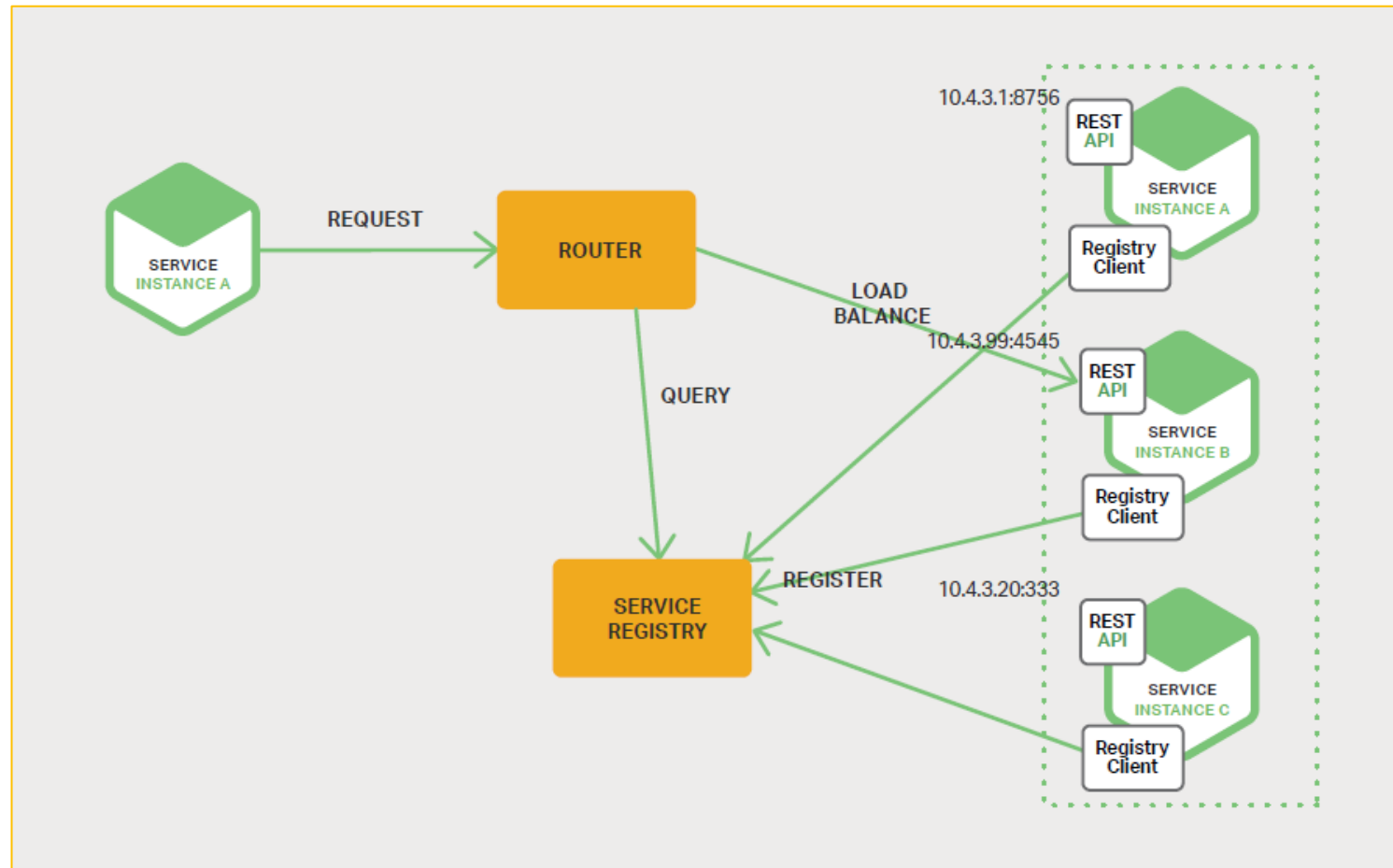
Service Discovery is Needed for Dynamic Environments



Option 1 – Client has to use a Service Registry



Option 2 – Server has to use a Service Registry





Netflix Microservices Architecture

Netflix Scale

- ~ 1/5 to 1/3 of the peak Internet traffic a day
- ~209M subscribers
- ~2+ Billion Edge API Requests/Day – 6 billions hours of streaming content each month.
- >500 MicroServices
- ~30 Engineering Teams (owning many microservices)



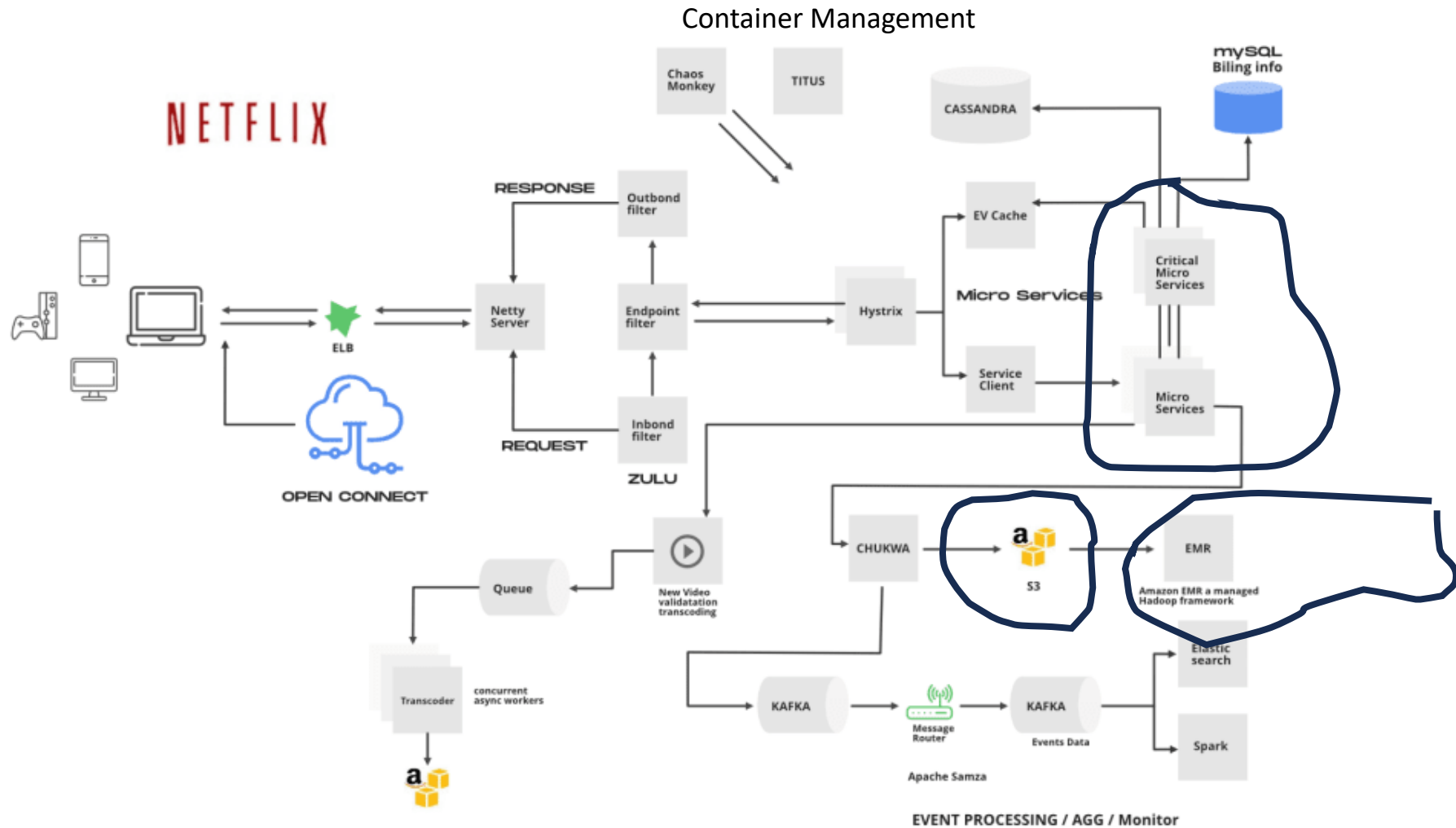
Netflix - Evolution

- Old DataCenter (2008)
- Everything in one WebApp (.war)

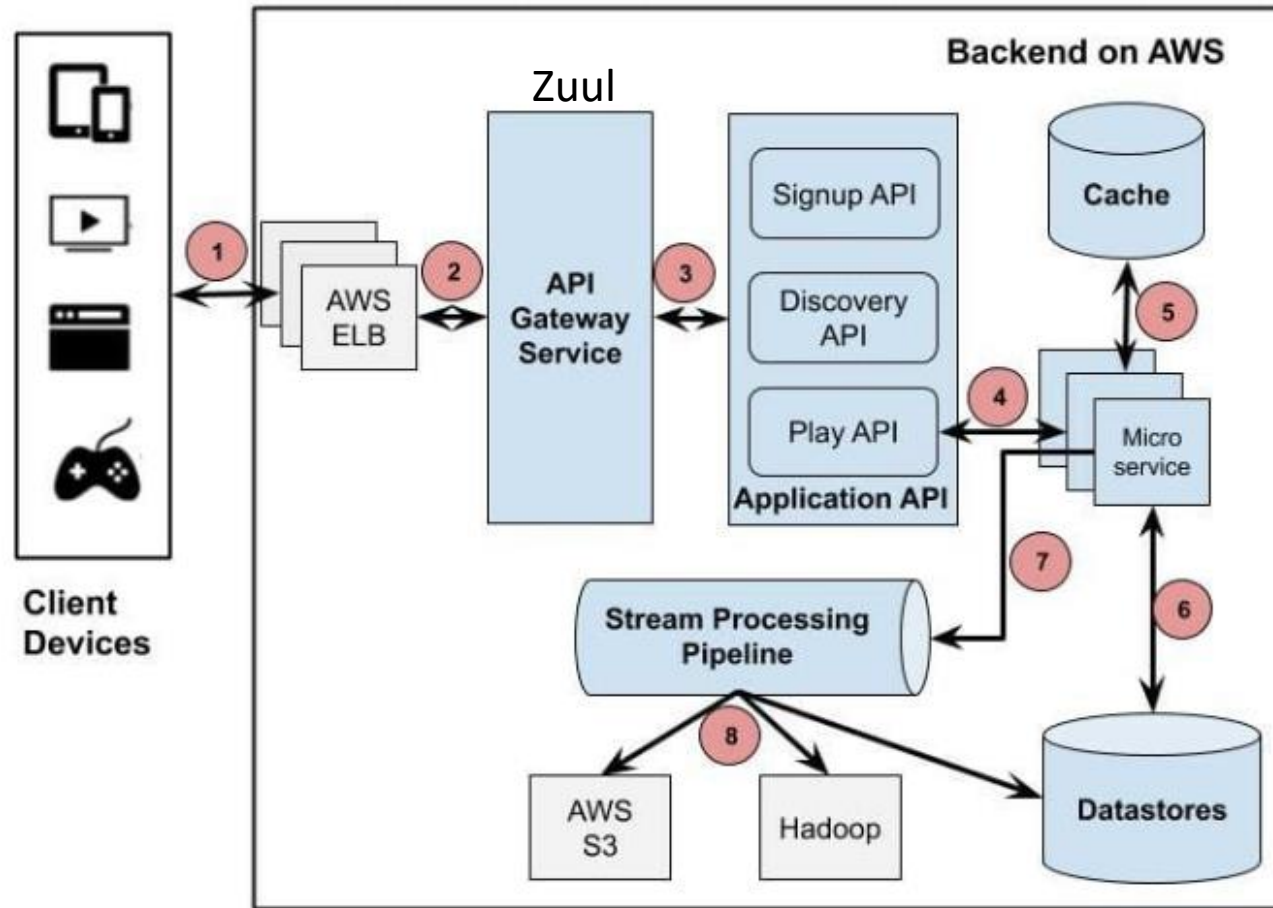
- AWS Cloud (~2010)
- 100s of Fine Grained Services



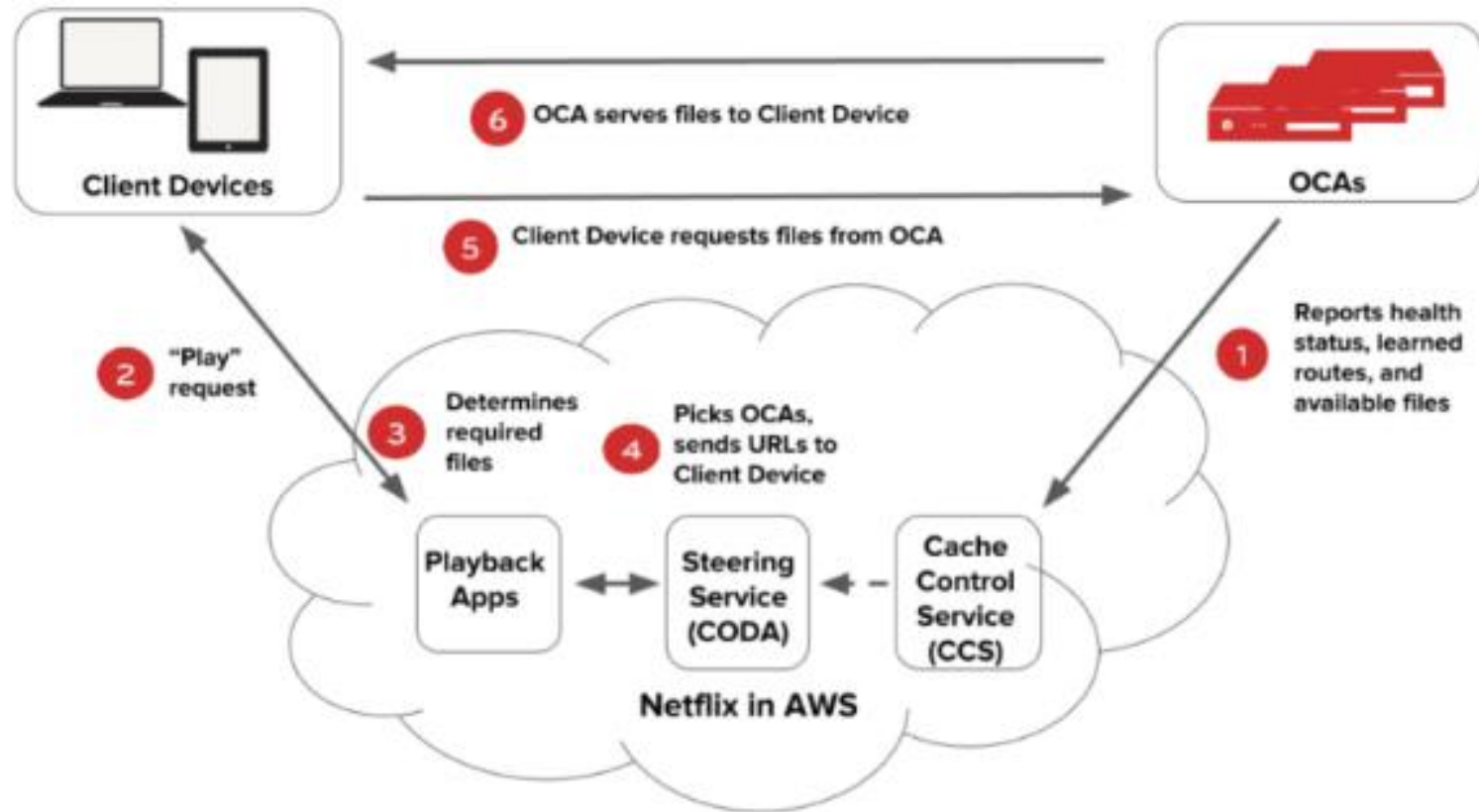
Netflix Cloud Architecture

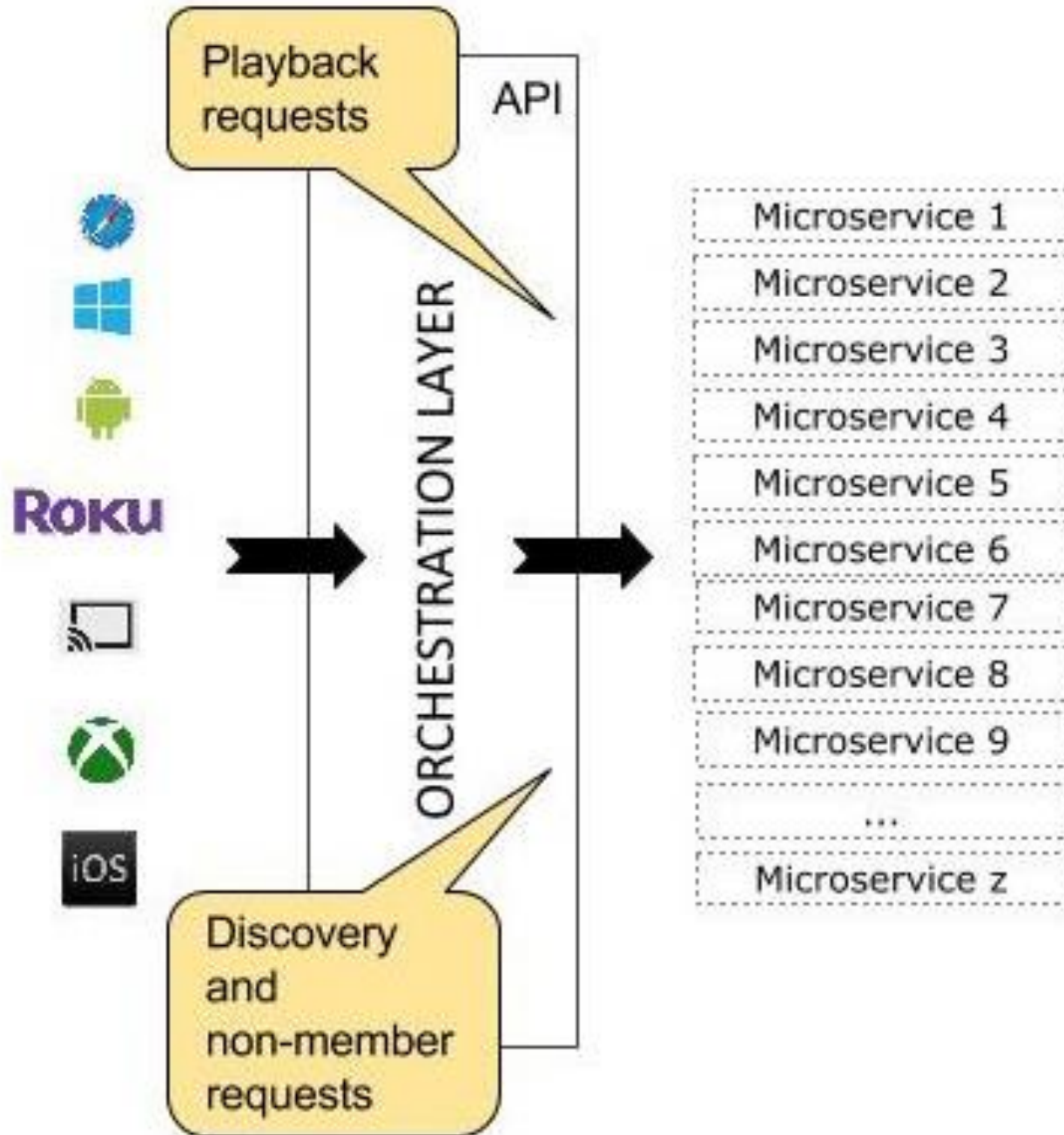


Slightly Simplified Netflix Architecture on AWS Cloud



Netflix OCA Architecture on AWS



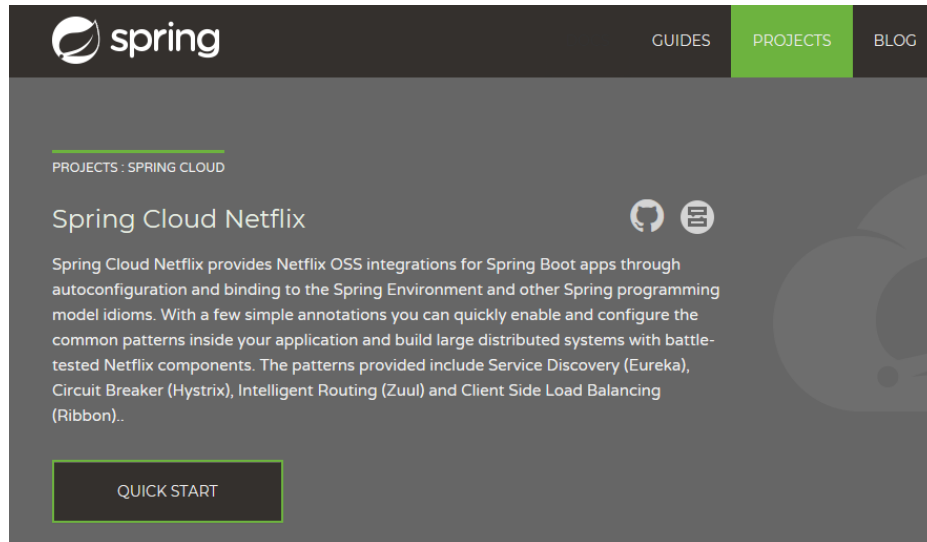


Netflix Microservices Architecture

Netflix Offers API Gateway with Service Registry - Eureka

<https://cloud.spring.io/spring-cloud-netflix/>

<https://spring.io/guides>

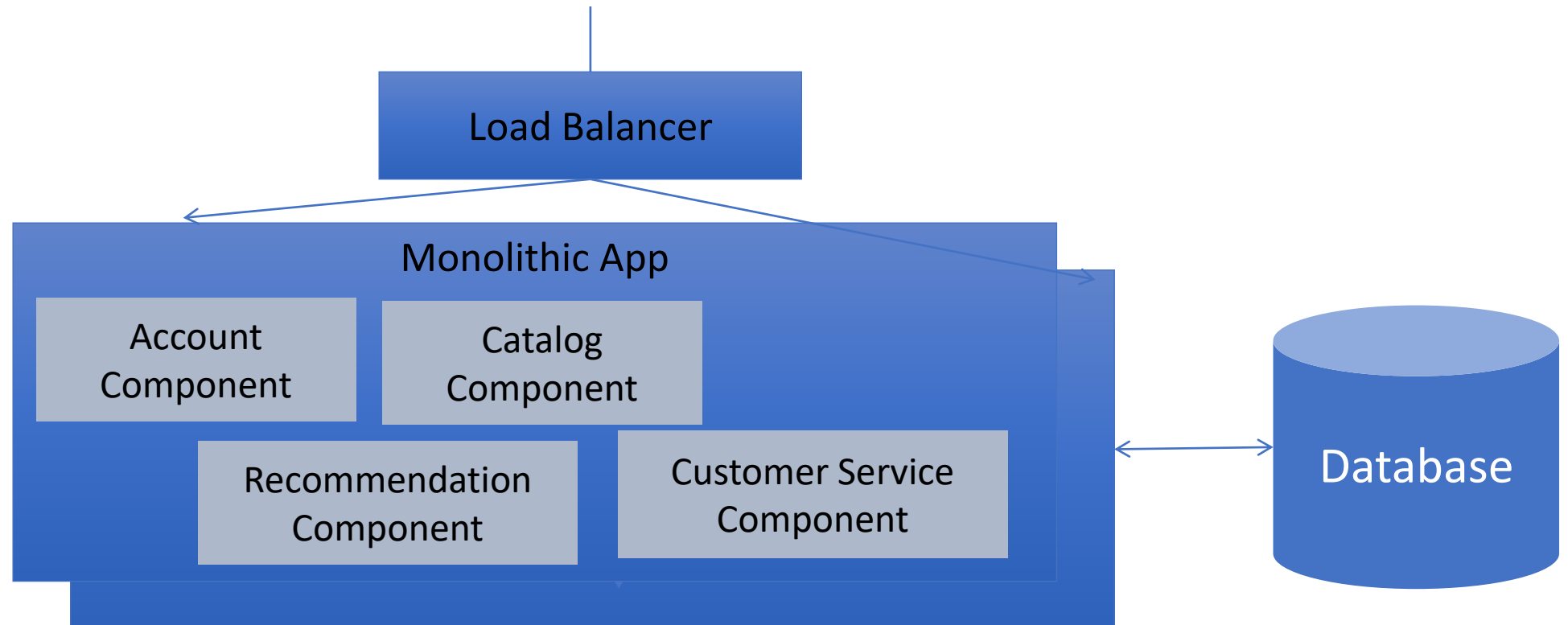


Features

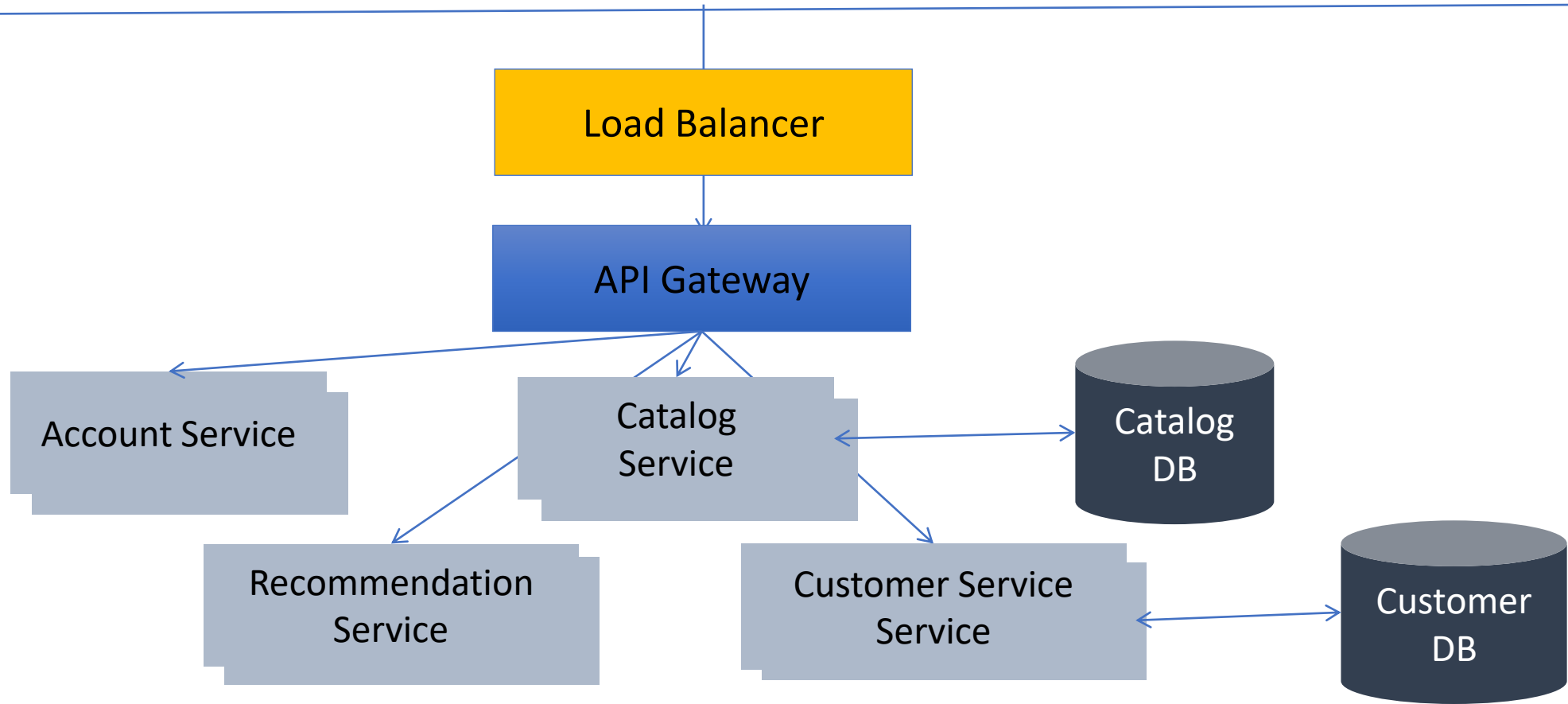
Spring Cloud Netflix features:

- Service Discovery: Eureka instances can be registered and clients can discover the instances using Spring-managed beans
- Service Discovery: an embedded Eureka server can be created with declarative Java configuration
- Circuit Breaker: Hystrix clients can be built with a simple annotation-driven method decorator
- Circuit Breaker: embedded Hystrix dashboard with declarative Java configuration
- Declarative REST Client: Feign creates a dynamic implementation of an interface decorated with JAX-RS or Spring MVC annotations
- Client Side Load Balancer: Ribbon
- External Configuration: a bridge from the Spring Environment to Archaius (enables native configuration of Netflix components using Spring Boot conventions)
- Router and Filter: automatic registration of Zuul filters, and a simple convention over configuration approach to reverse proxy creation

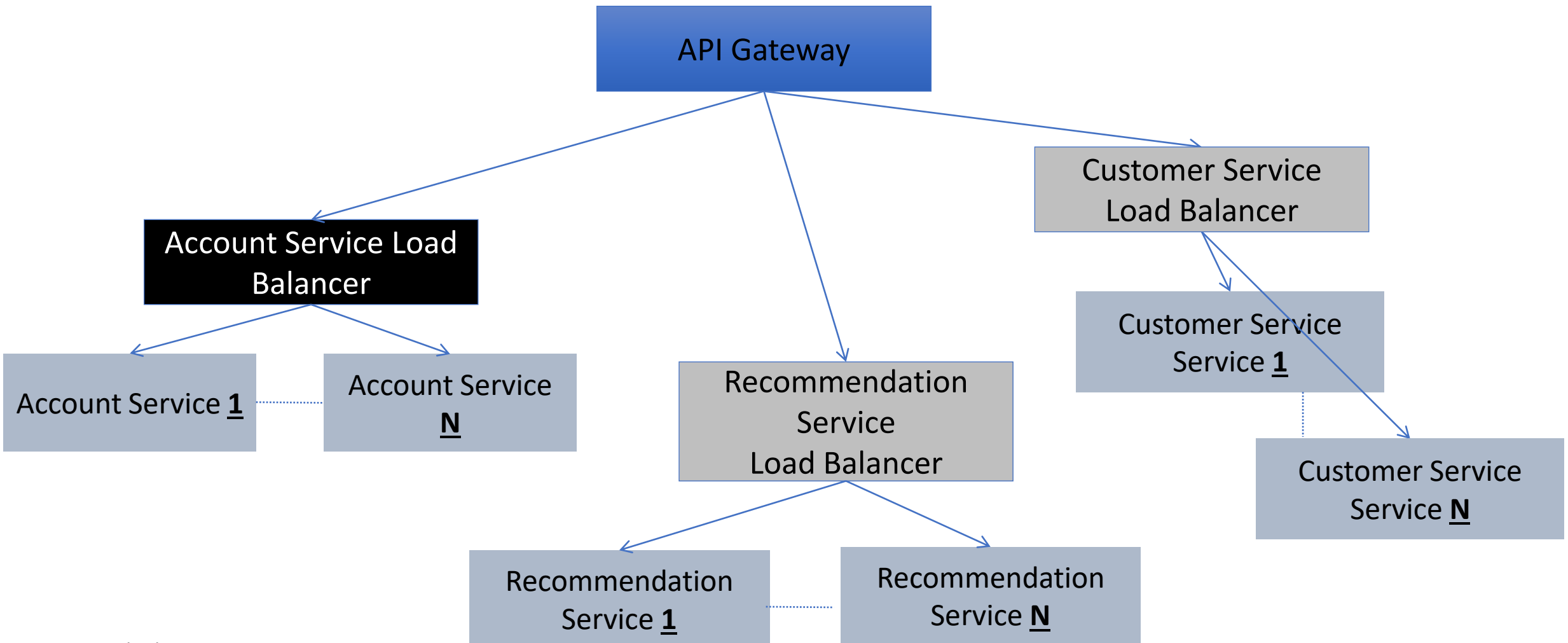
Monolithic Architecture Netflix (prior 2010)



Microservices Architecture (2010+) on AWS

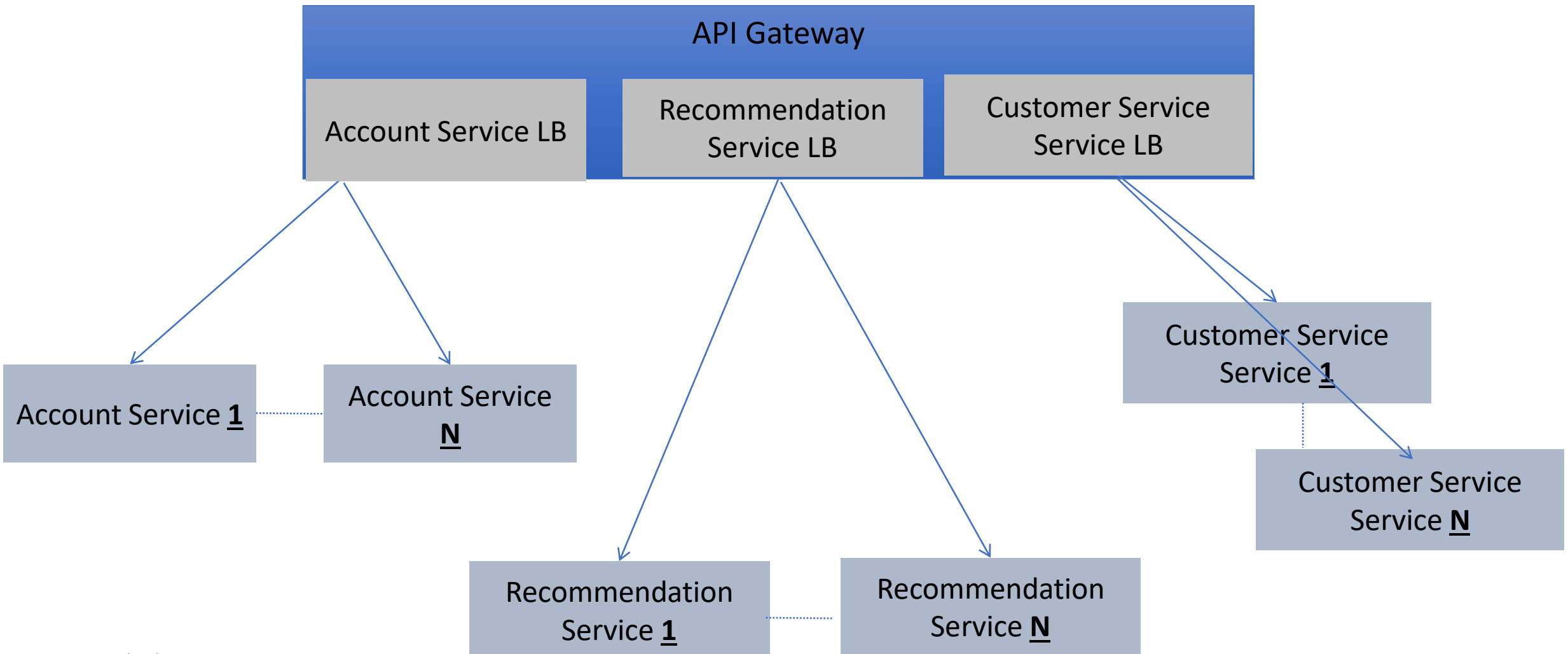


Central (Proxy) Loadbalancer



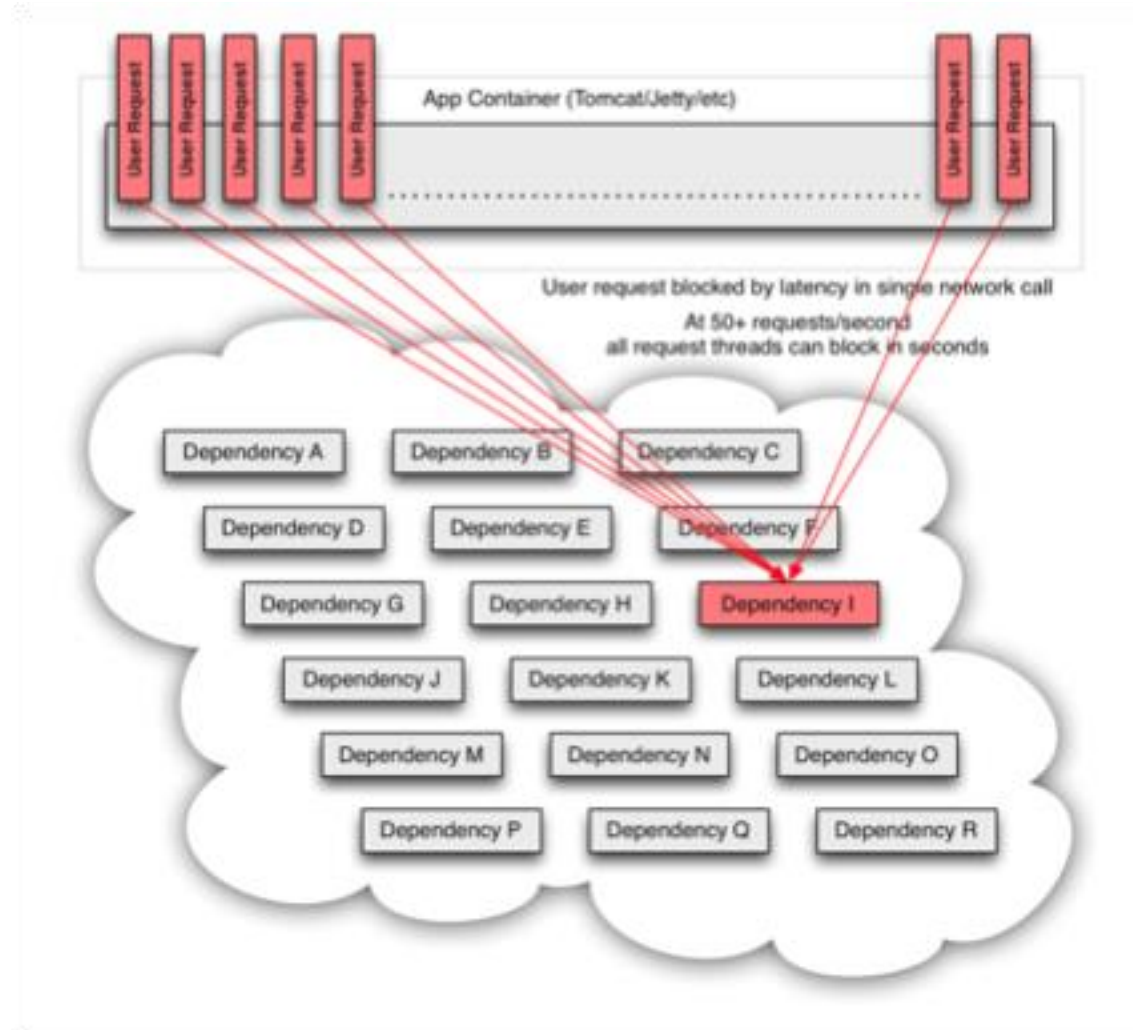
Client Loadbalancer

NETFLIX

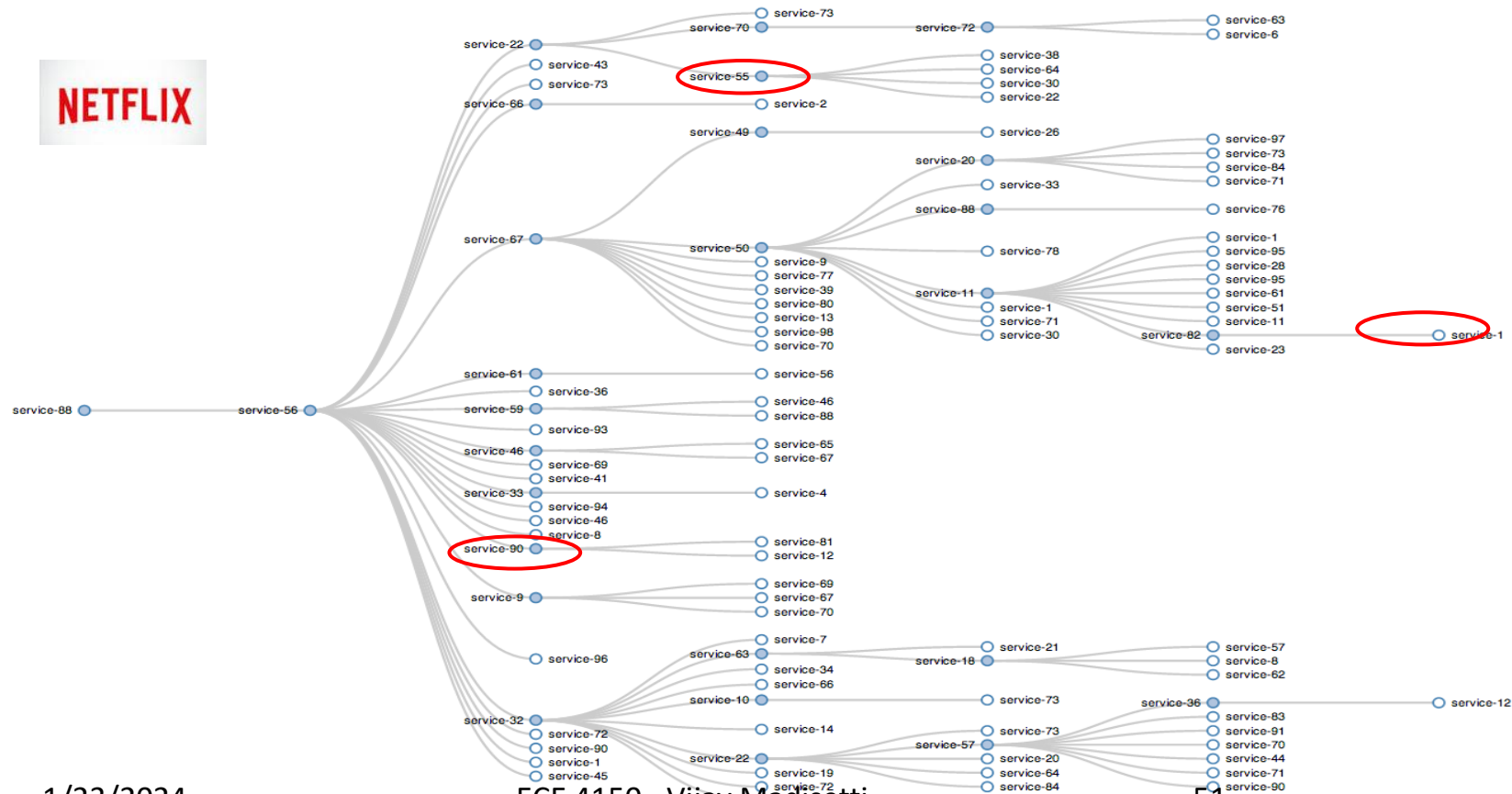


NETFLIX

Single
Microservice
Can Be A
Bottleneck

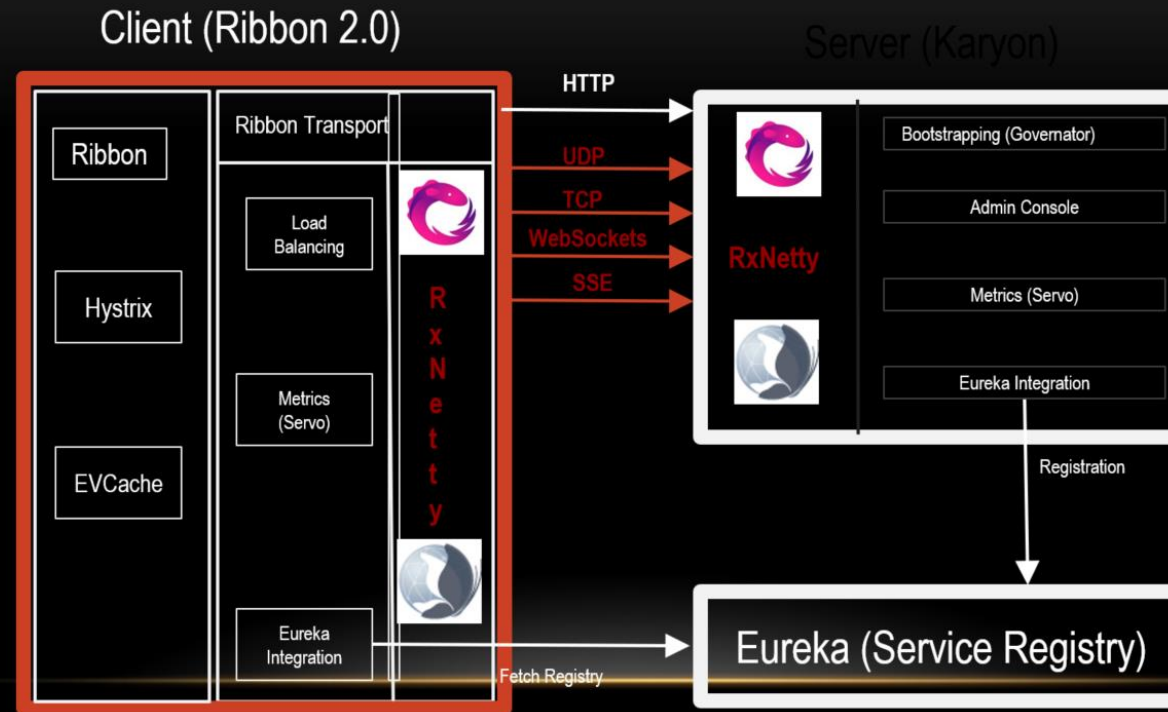


Test Resiliency – to dependencies



A
Completely
Reactive
Architecture

Netflix IPC Stack (2.0)



A Completely Reactive Architecture

NetflixOSS Instance Libraries

Initialization

- Baked AMI – Tomcat, Apache, your code
- Governor – Guice based dependency injection
- Archaius – dynamic configuration properties client
- Eureka - service registration client

Service Requests

- Karyon - Base Server for inbound requests
- RxJava – Reactive pattern
- Hystrix/Turbine – dependencies and real-time status
- Ribbon and Feign - REST Clients for outbound calls

Data Access

- Astyanax – Cassandra client and pattern library
- Evcache – Zone aware Memcached client
- Curator – Zookeeper patterns
- Denominator – DNS routing abstraction

Logging

- Blitz4j – non-blocking logging
- Servo – metrics export for autoscaling
- Atlas – high volume instrumentation



- Eureka – for Service Registry/Discovery
- Karyon – for Server (Reactive or threaded/servlet container based)
- Ribbon – for IPC Client
And Fault Tolerant Smart LoadBalancer
- Hystrix – for Fault Tolerance and Resiliency
- Archaius – for distributed/dynamic Properties
- Servo – unified Feature rich Metrics/Insight
- EVCache – for distributed cache
- Curator/Exhibitor – for zookeeper based operations



Summary

- Microservices are a newer model of cloud application development that have at least two flavors – serverless and container-based.
- Netflix uses microservices (container-based) and McDonald's uses microservices (Lambda services or serverless)
- Lab 1 shows microservices using serverless model, while Lab 5 will focus on microservices using container model. Lab 2 will focus on an intermediate step towards the container model using Flask-based Microservices development.