Introduction

Big Data Systems

Dr. Rubi Boim

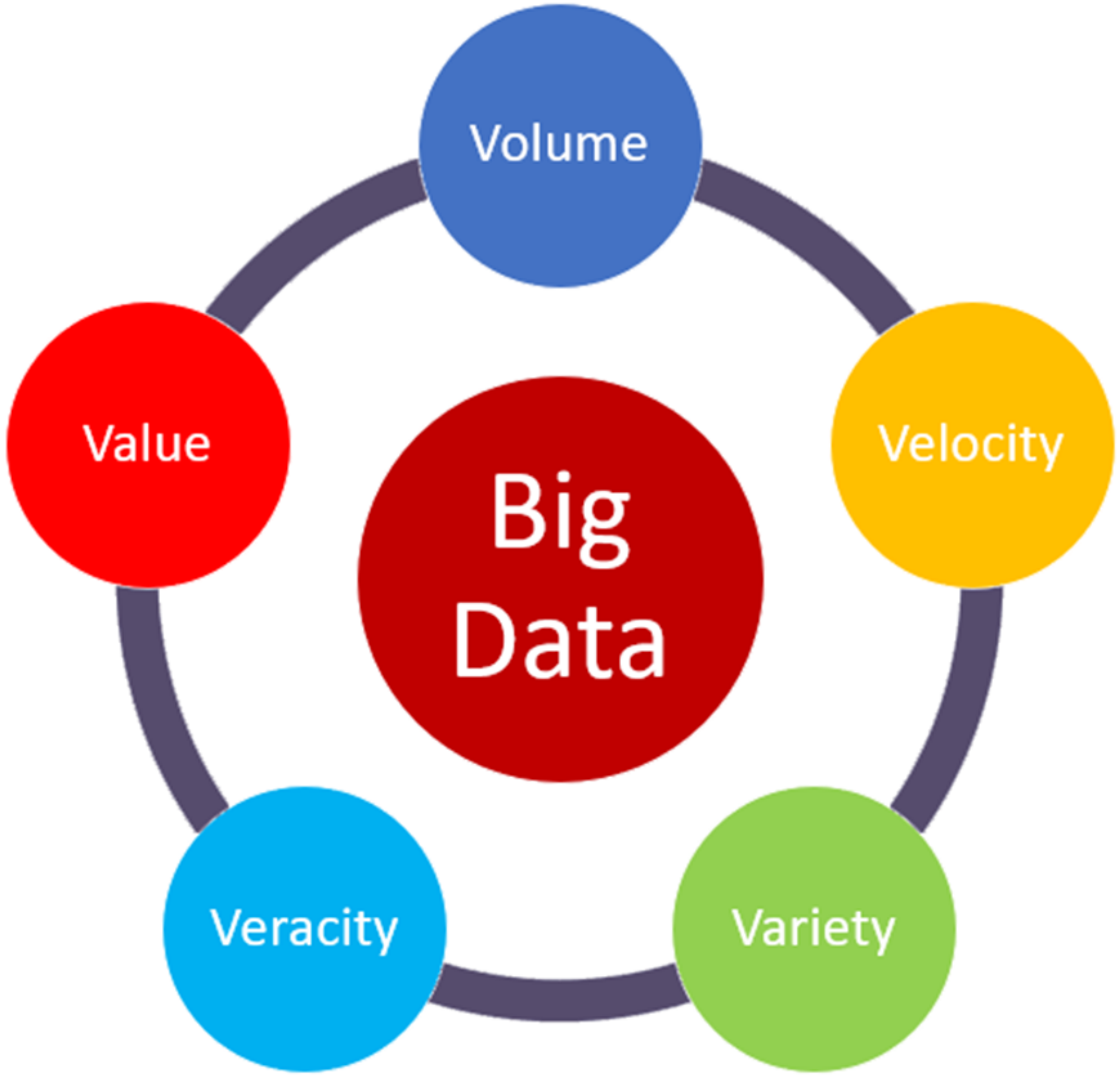
Agenda for today

- 5 V’s of Big Data
- Cloud computing
- Highly available / highly Scalable
- Managed vs Unmanaged services

When data is Big Data?

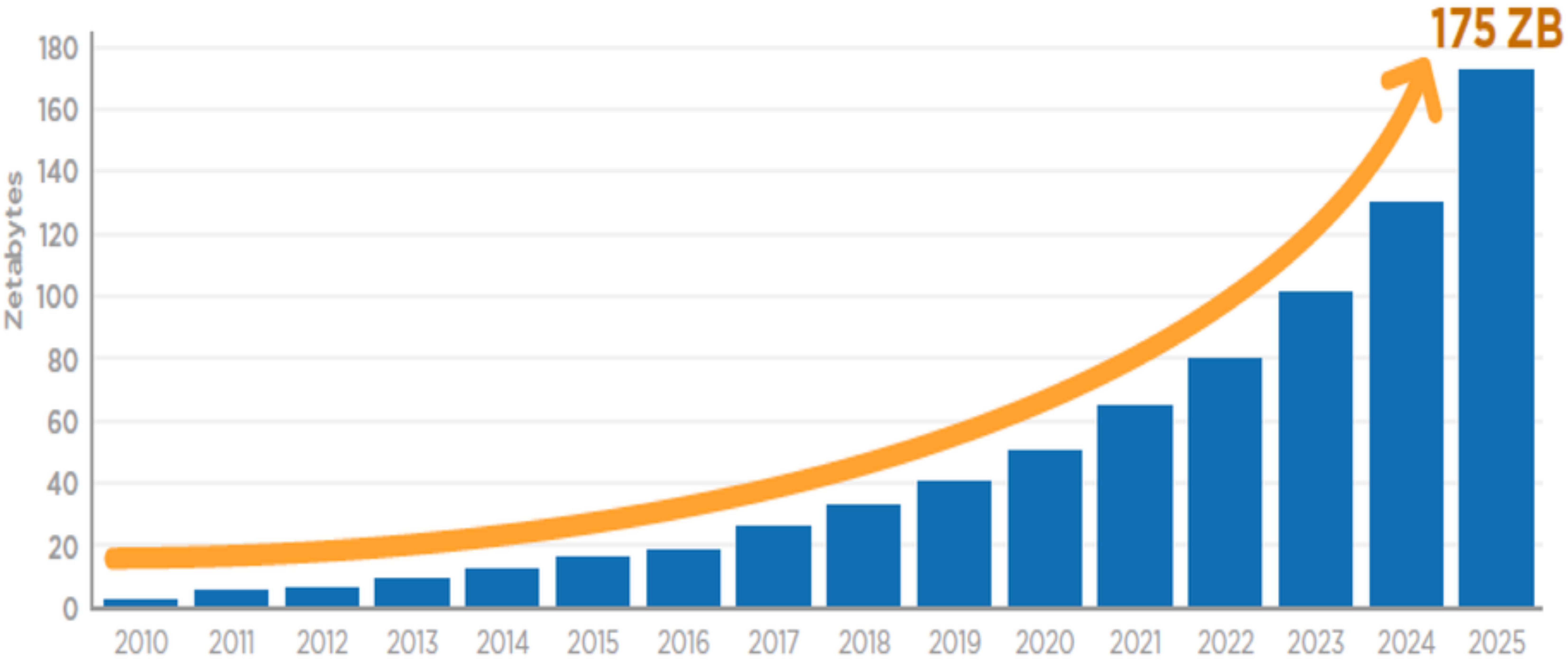
5 V’s of Big Data

- Volume
- Velocity
- Variety
- Veracity
- Value

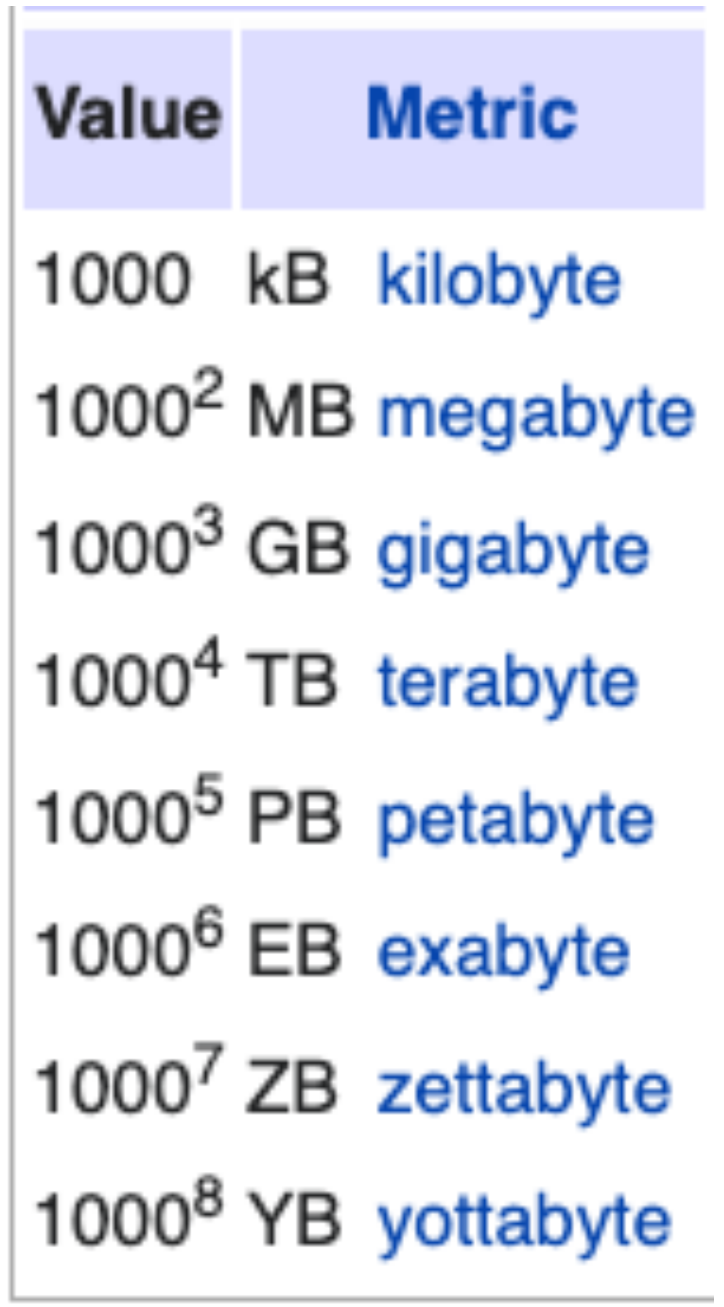


Volume

- Data is rapidly increasing
  (due to cloud computing, mobile and more)



Source: Data Age 2025, sponsored by Seagate with data from IDC Global DataSphere, Nov 2018



Value	Metric	
1000	kB	kilobyte
1000^2	MB	megabyte
1000^3	GB	gigabyte
1000^4	TB	terabyte
1000^5	PB	petabyte
1000^6	EB	exabyte
1000^7	ZB	zettabyte
1000^8	YB	yottabyte

Volume

- Data is rapidly increasing
  (due to cloud computing, mobile and more)

As of 2020, WhatsApp users send over 100 billion messages each day

Velocity

The speed at which data is generated

- Frequency of data generation (write)
  everything is measured
- Frequency of data processing (read)
  real time experience

Variety

- Structured data
  info, transactions…

- Semi structured data
  logs, sensor data…

- Unstructured data
  images, video, audio…

Veracity

The truthfulness or reliability of the data

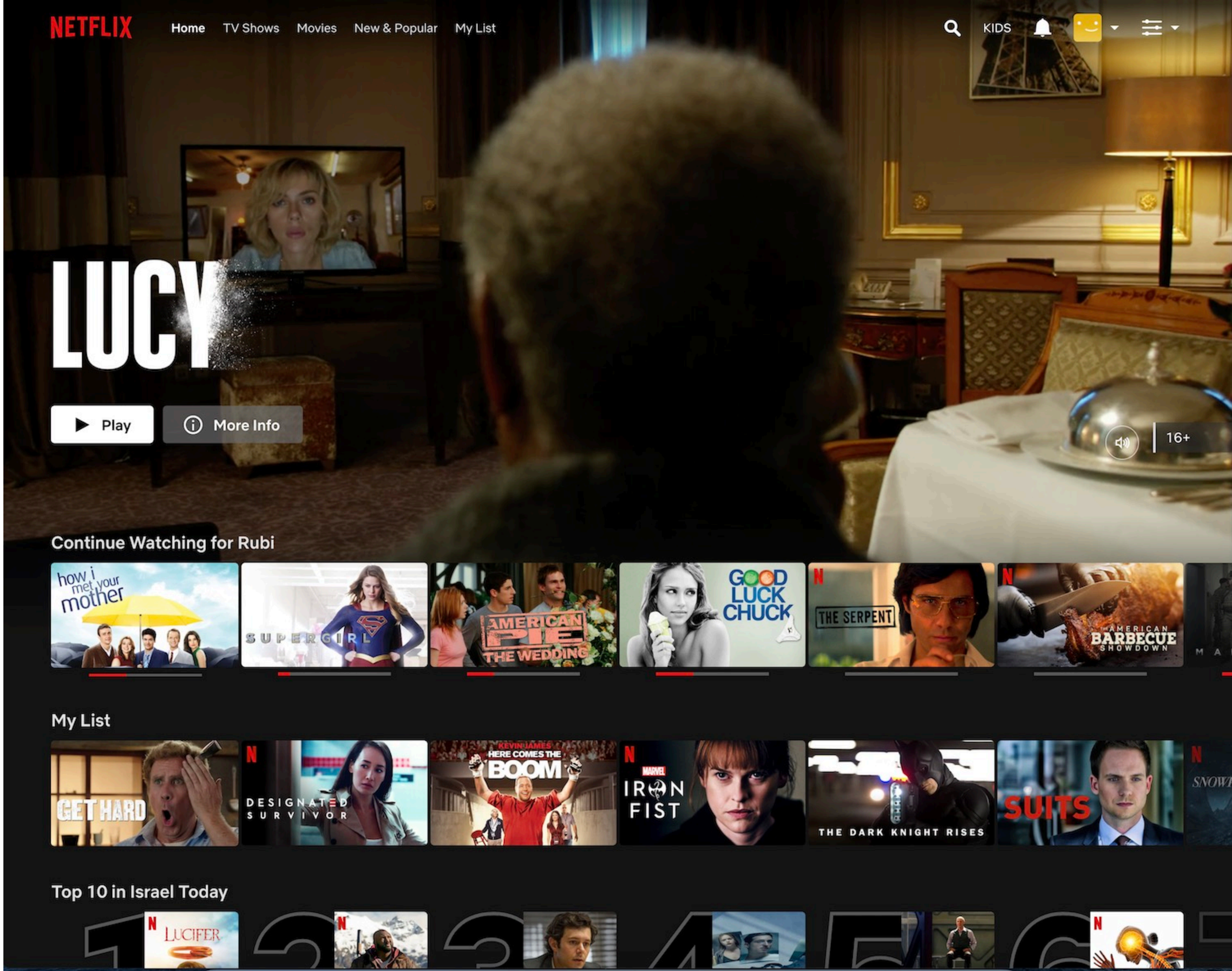
- data quality of captured data can vary greatly
  - bias
  - abnormalities
  - inconsistencies
  - duplication

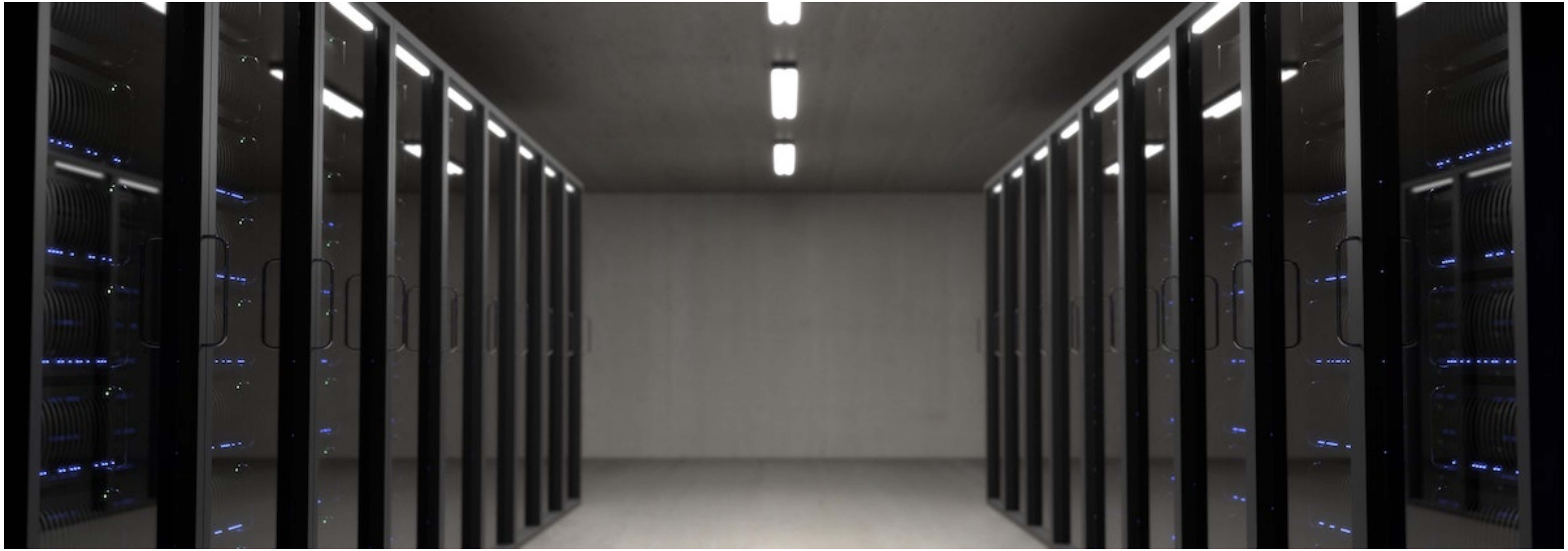
Value

The final result.

- which questions were answered
- hidden insights (machine learning)
- collecting data without use is, well, useless

- Volume
- Velocity
- Variety
- Veracity
- Value

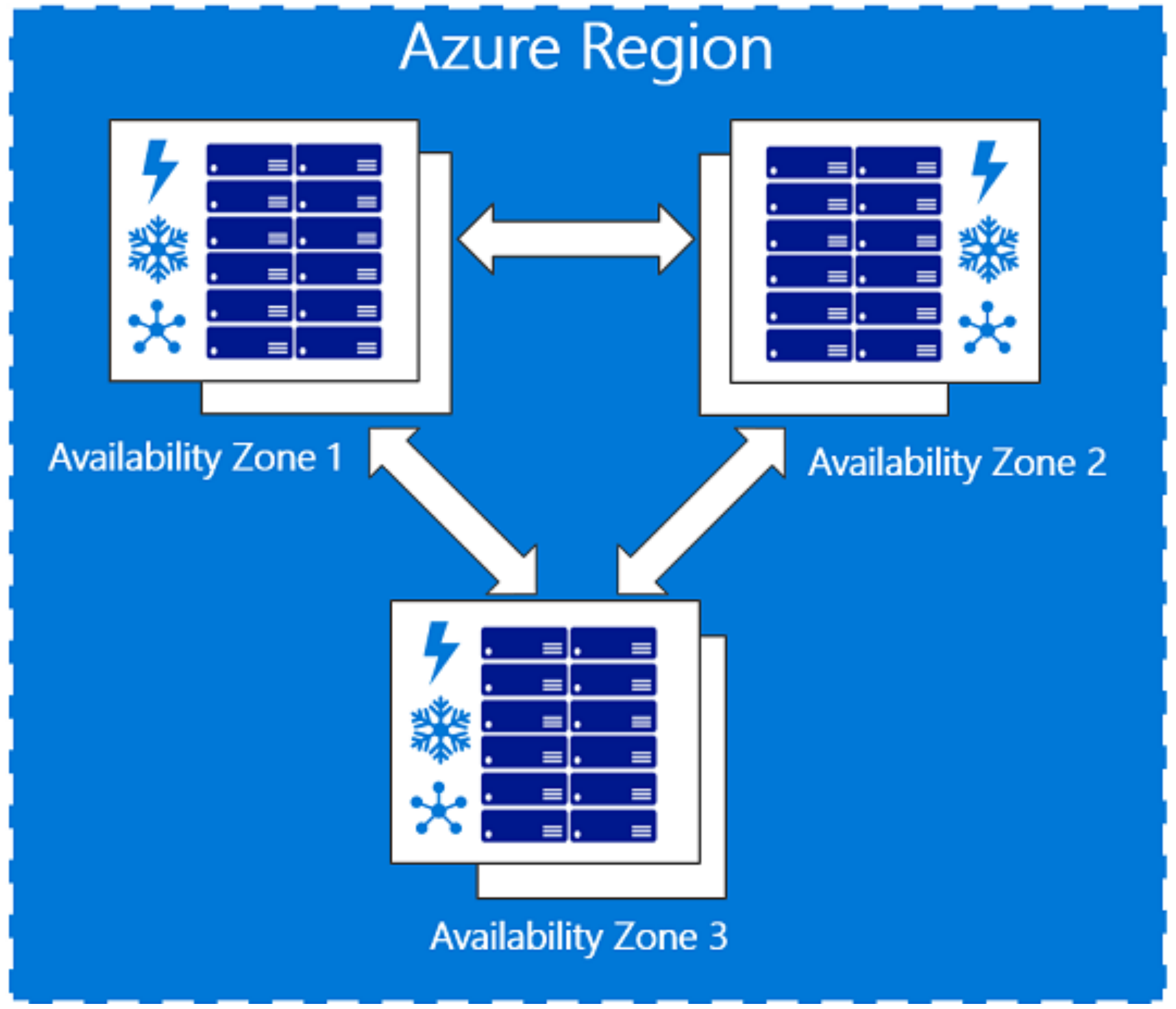




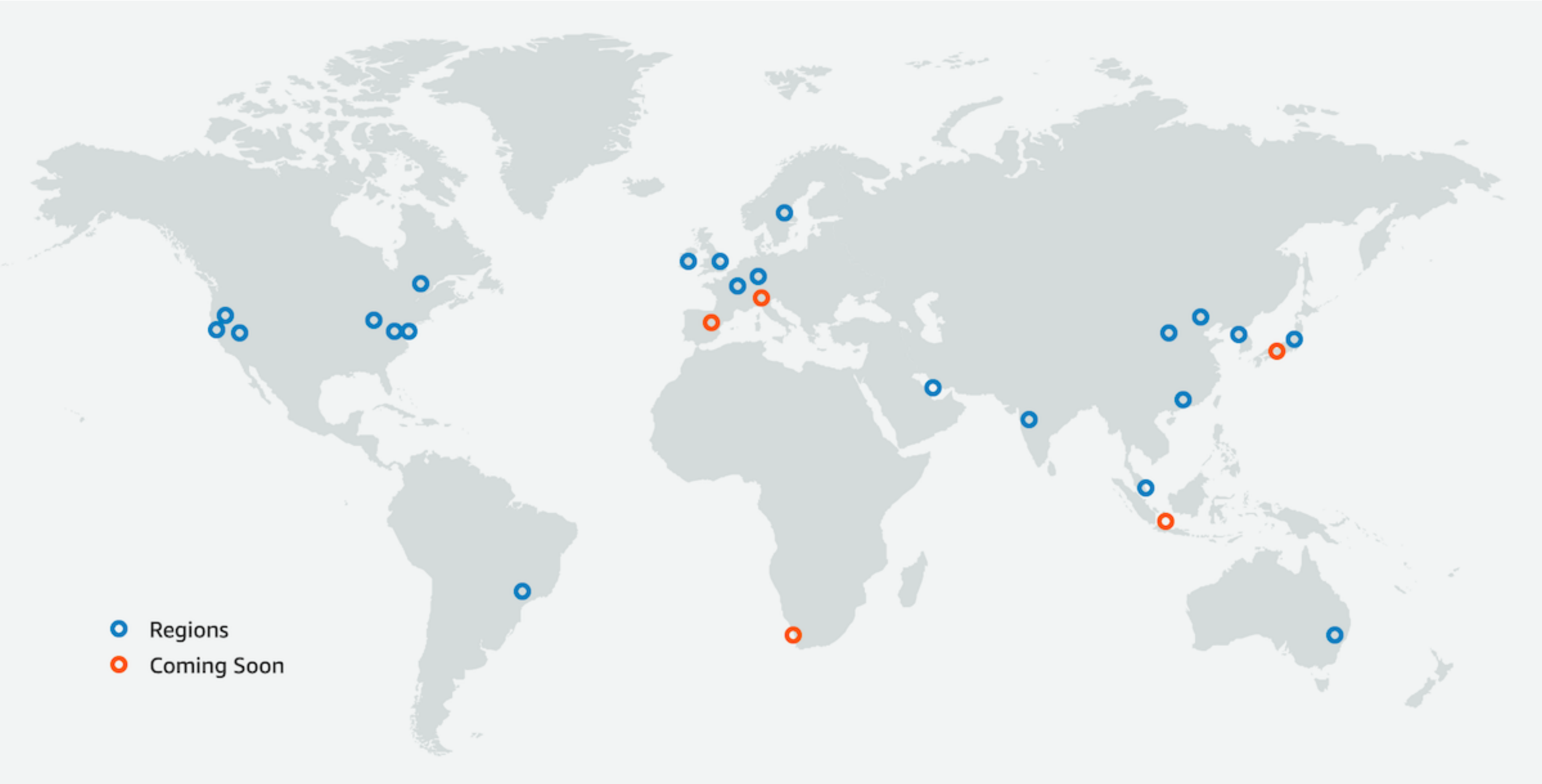
Cloud computing

Region / AZ / EL

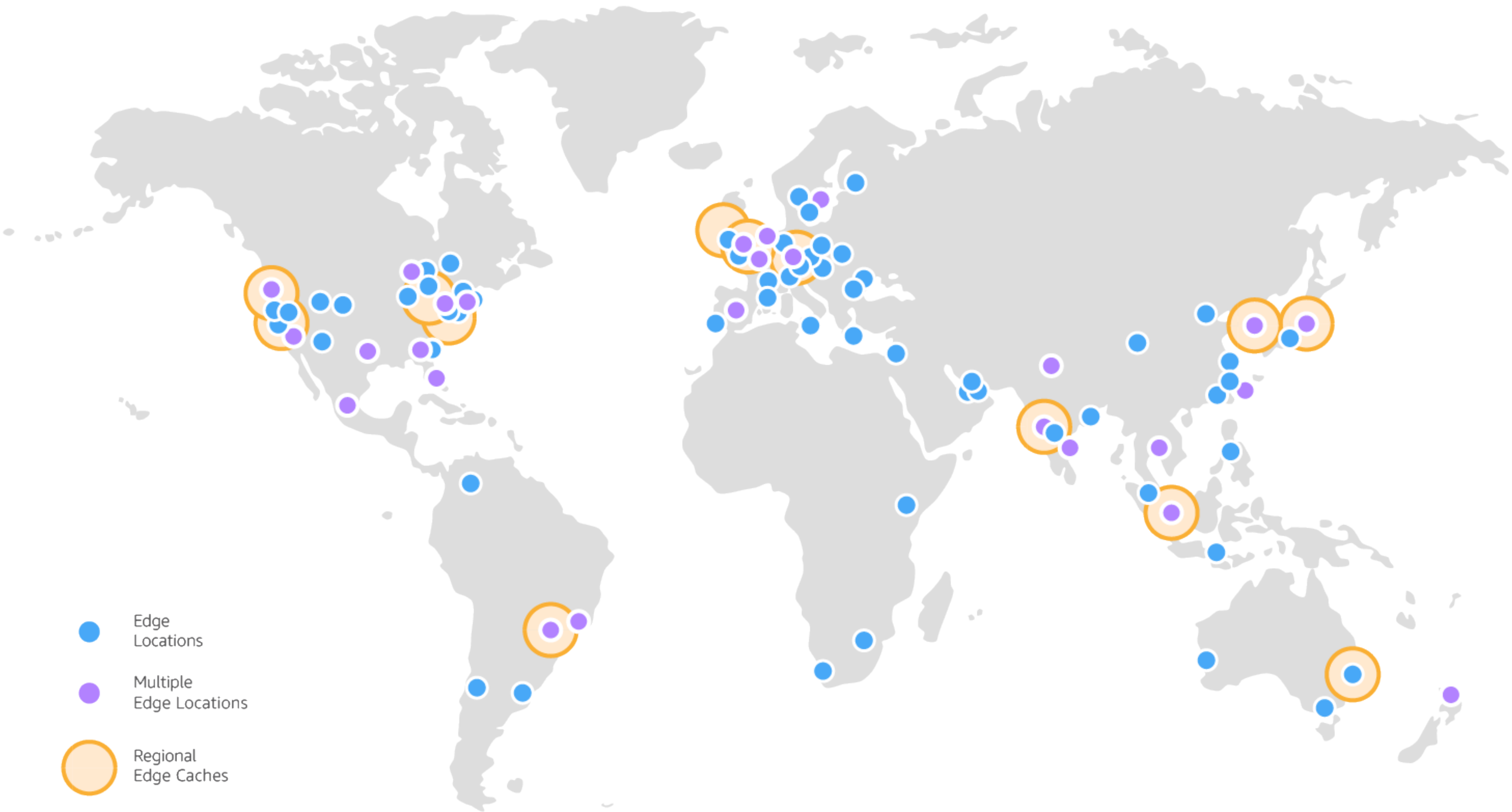
- Region
  Cluster of data centers in a physical location
- Availability Zone
  a discrete data center with redundant power, networking, and connectivity in a Region
- Edge Location
  access to the network with limited services (usually CDN)



- (Names may vary between cloud providers)



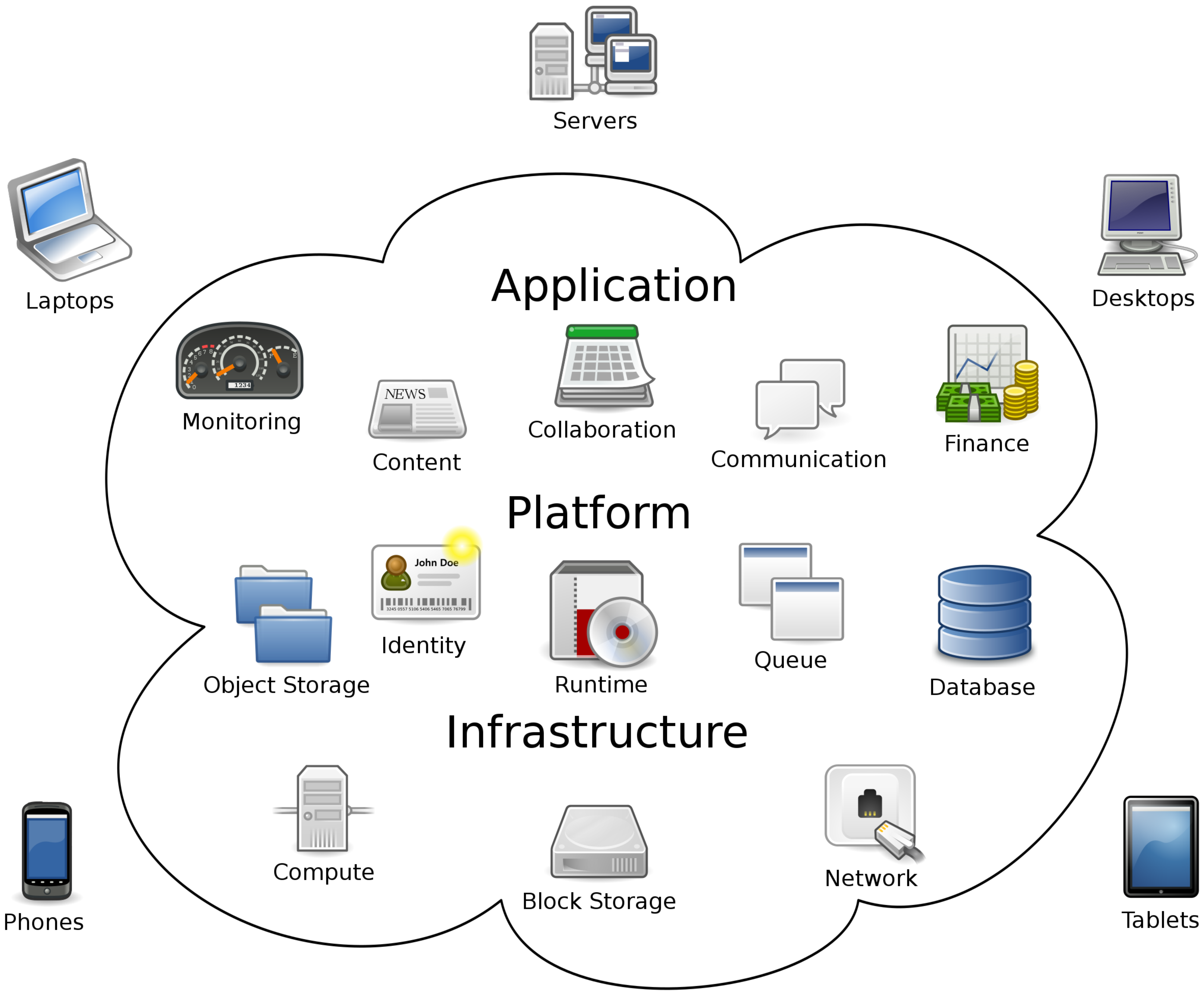
AWS regions (march 2020)



AWS edge locations (march 2020)

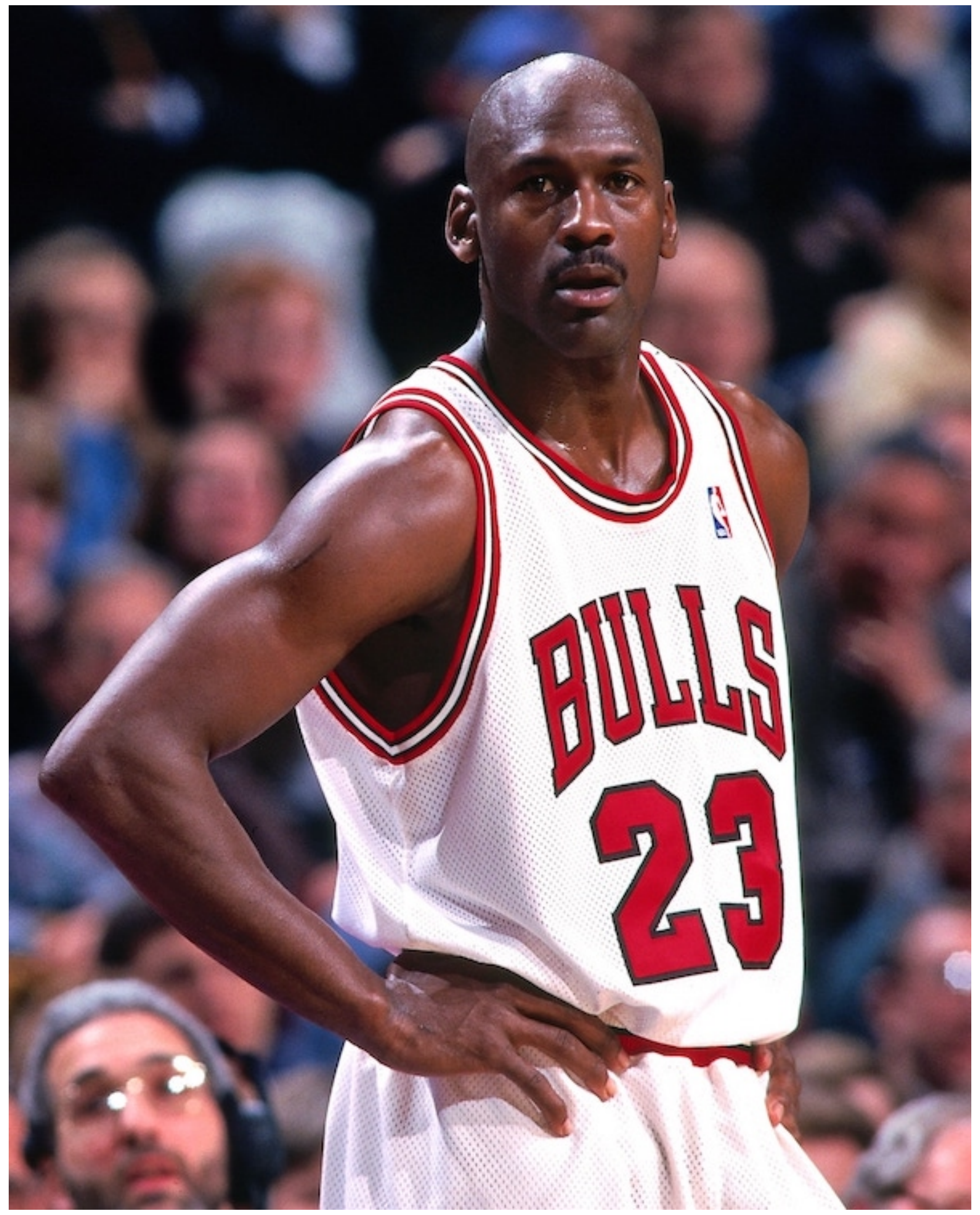
Cloud computing

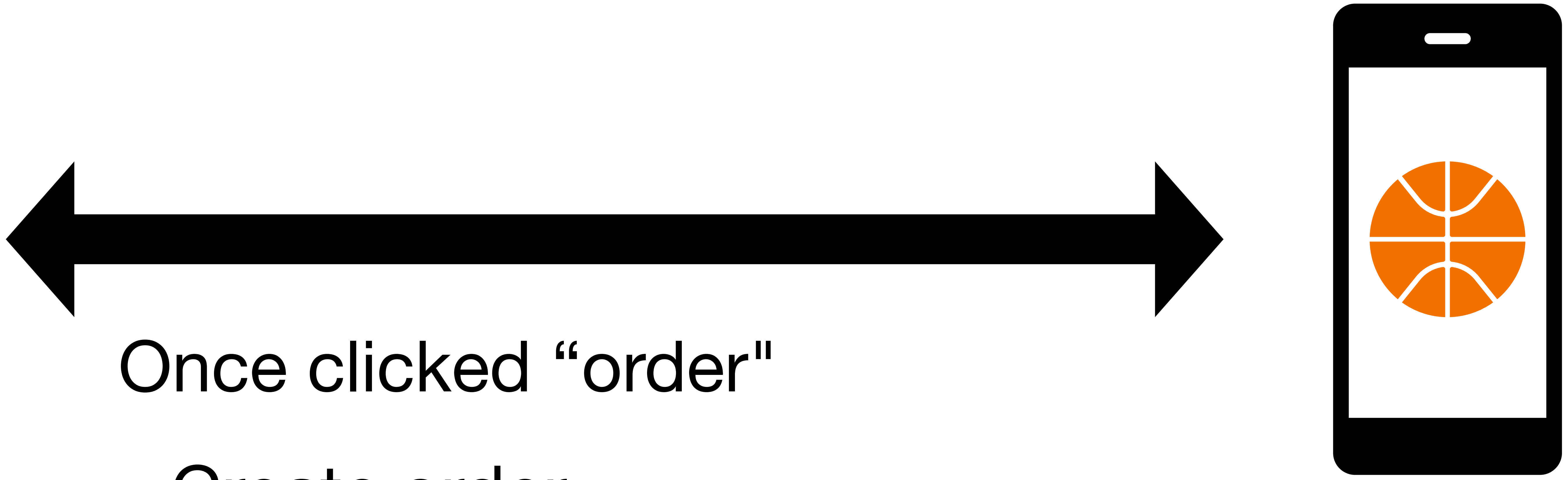
- SaaS
  software as a service
- PaaS
  platform as a service
- IaaS
  infrastructure as a service



Highly Available / Highly Scalable

Mike orders a a basketball





Once clicked “order"

- Create order
- Check inventory
- Process payment
- Approve order ←
- Send to warehouse
- …

System error

fire / flood / electricity / hardware malfunction / software update…

Possible outcomes

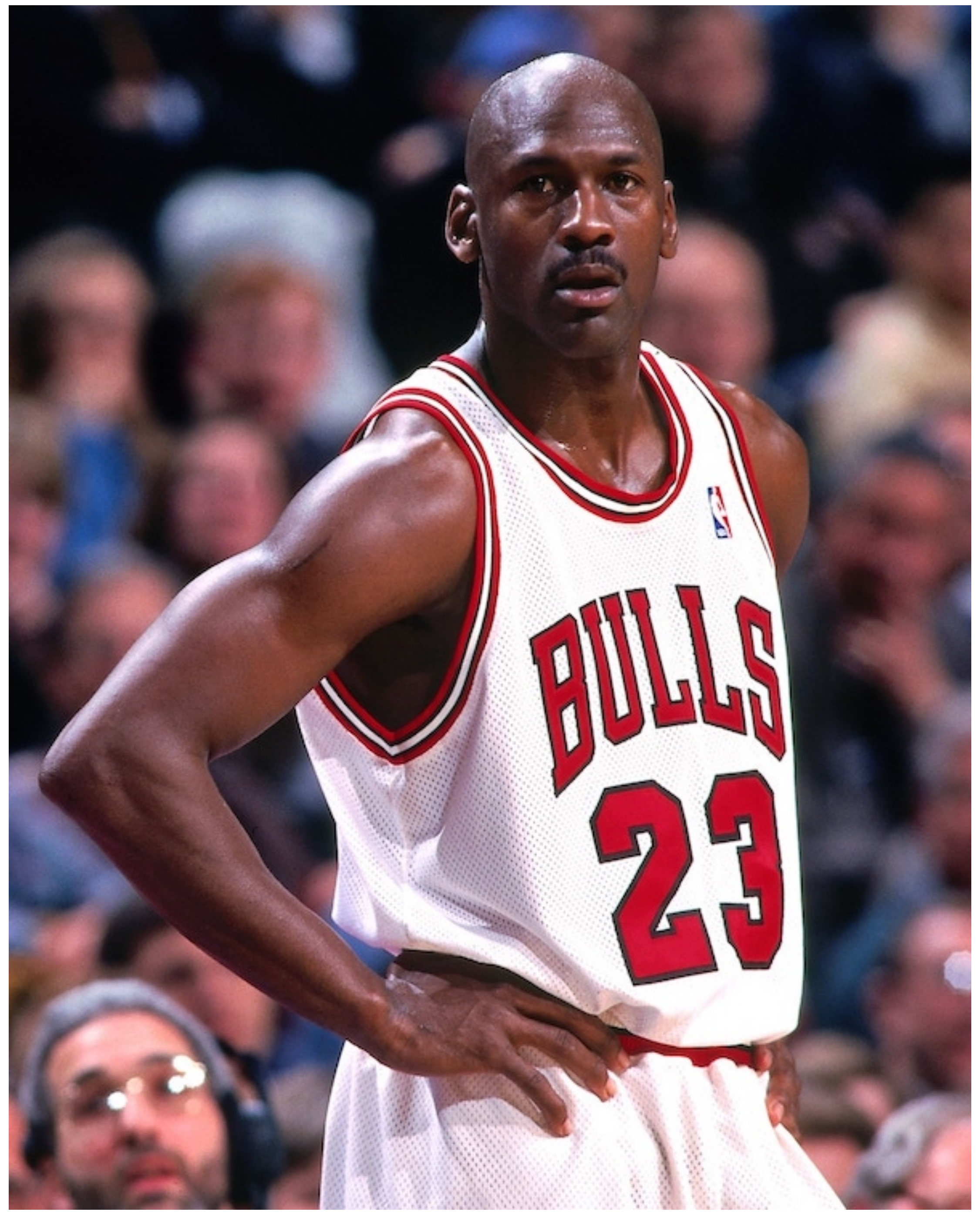
- Service disruption
- Data loss
- Data consistency
- Money lost (direct / reputation)
- A hard problem to solve for Databases
  disaster recovery:
  RTO (Recovery time) / RPO (Recovery point object)

High availability

- “Nines”

Availability	Downtime per day	Downtime per year
90%	2.40 hours	36.53 days
95%	1.20 hours	18.26 days
99%	14.40 minutes	3.65 days
99.9%	1.44 minutes	8.77 hours
99.99%	8.64 seconds	52.60 minutes
99.999%	864.00 milliseconds	5.26 minutes
99.9999%	86.40 milliseconds	31.56 seconds

Mike tweets about a basketball he bought



- Reach millions of users
- Millions of users try to buy the same basketball at the same time

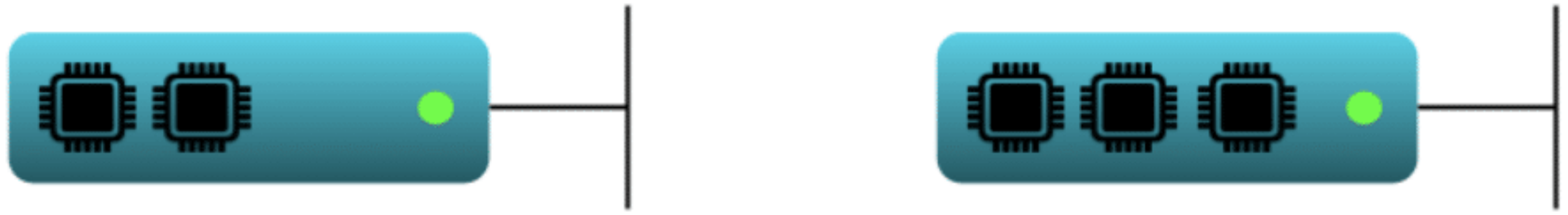
System error

Too many requests

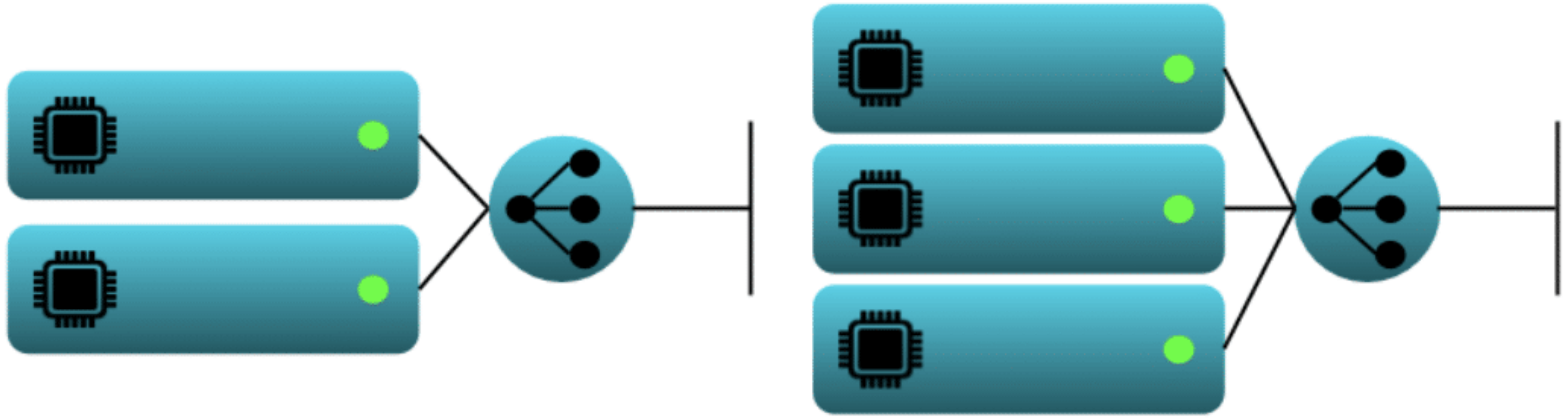
High scalability

- Scale up vs scale out
- Commodity computing
- Stateless

  amazon’s shopping cart is stateless?
- Microservices
- Sharding

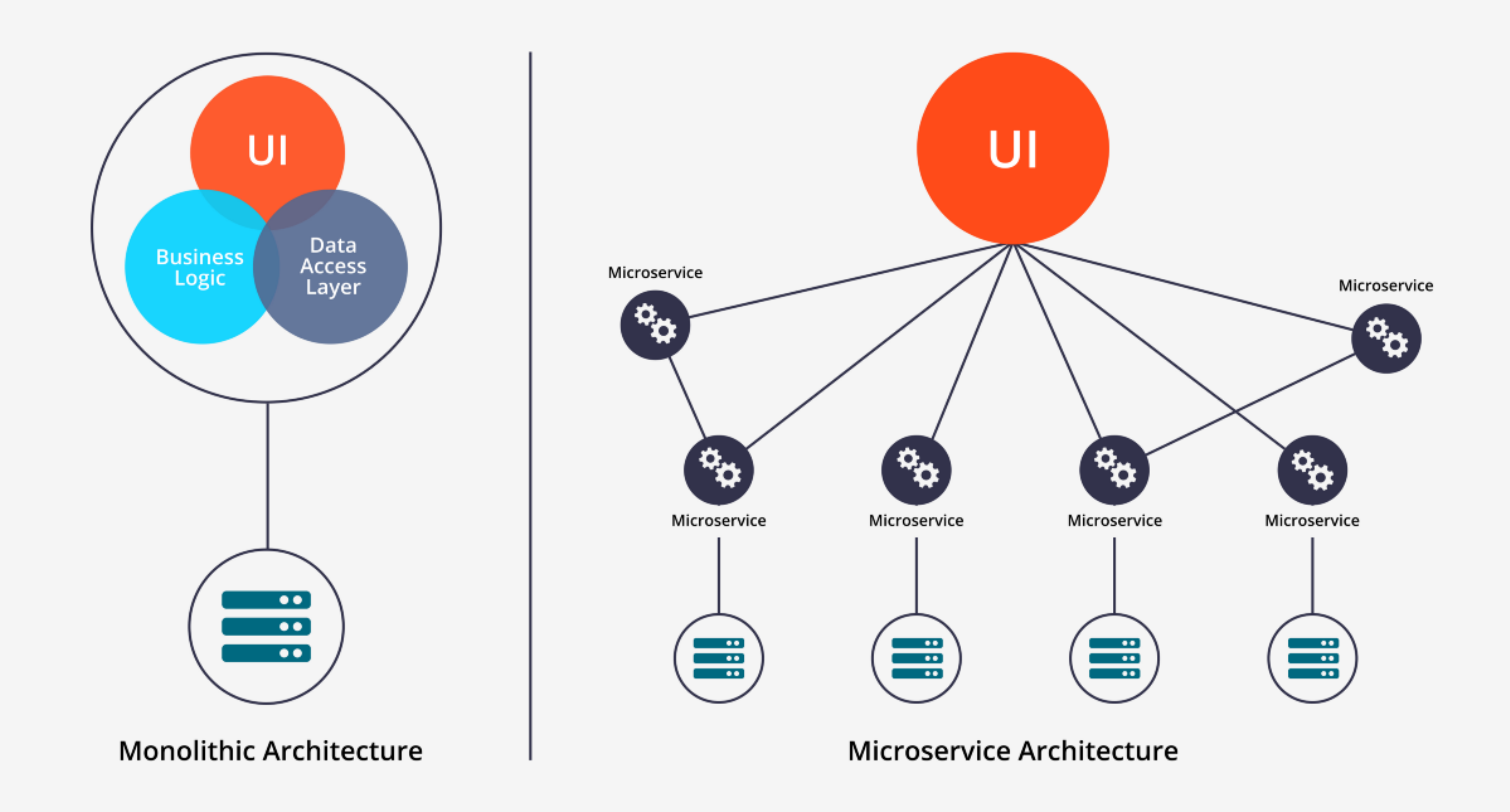


Scaling up from two to three CPUs

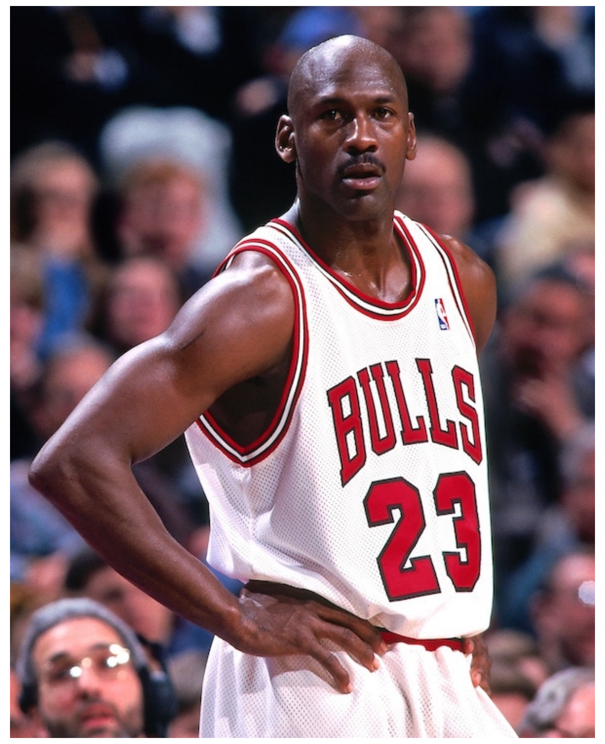


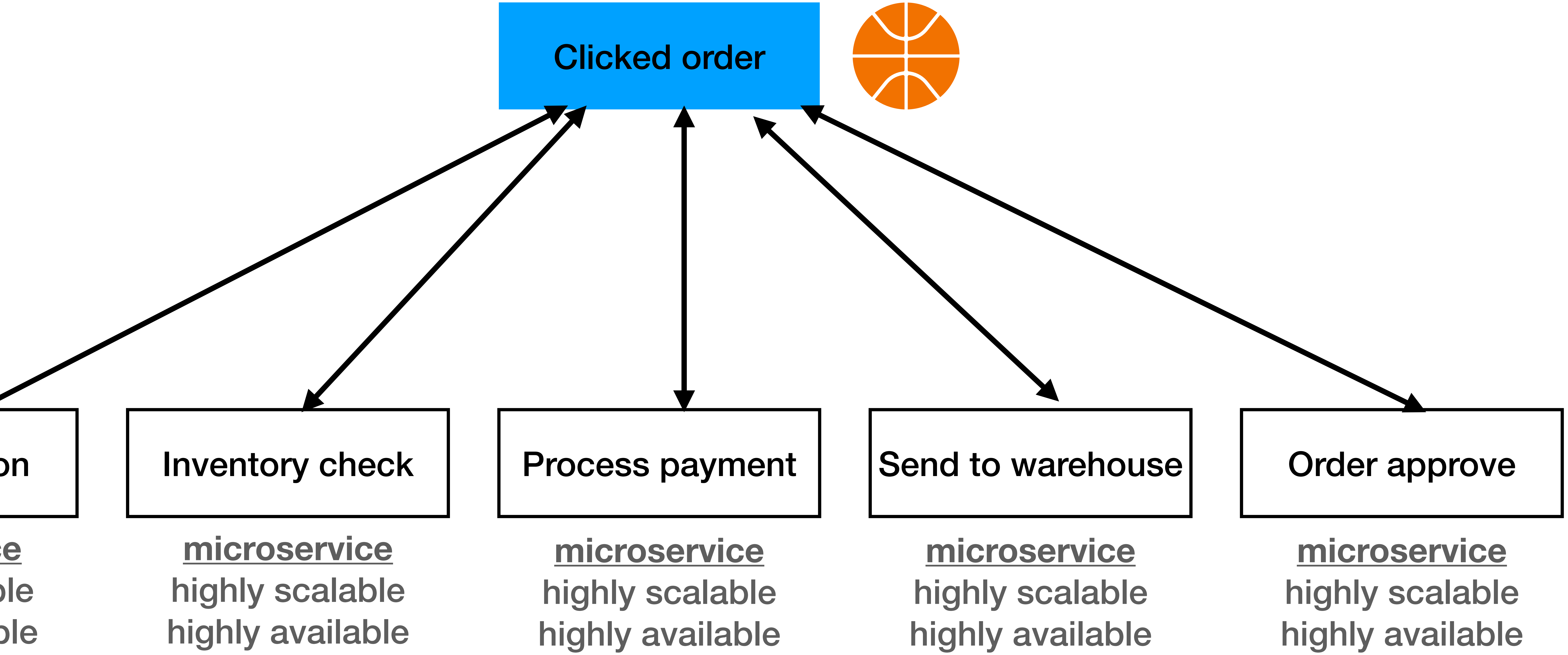
Scaling out from two to three CPUs

Microservices



Ordering a basketball





microservice
highly scalable
highly available

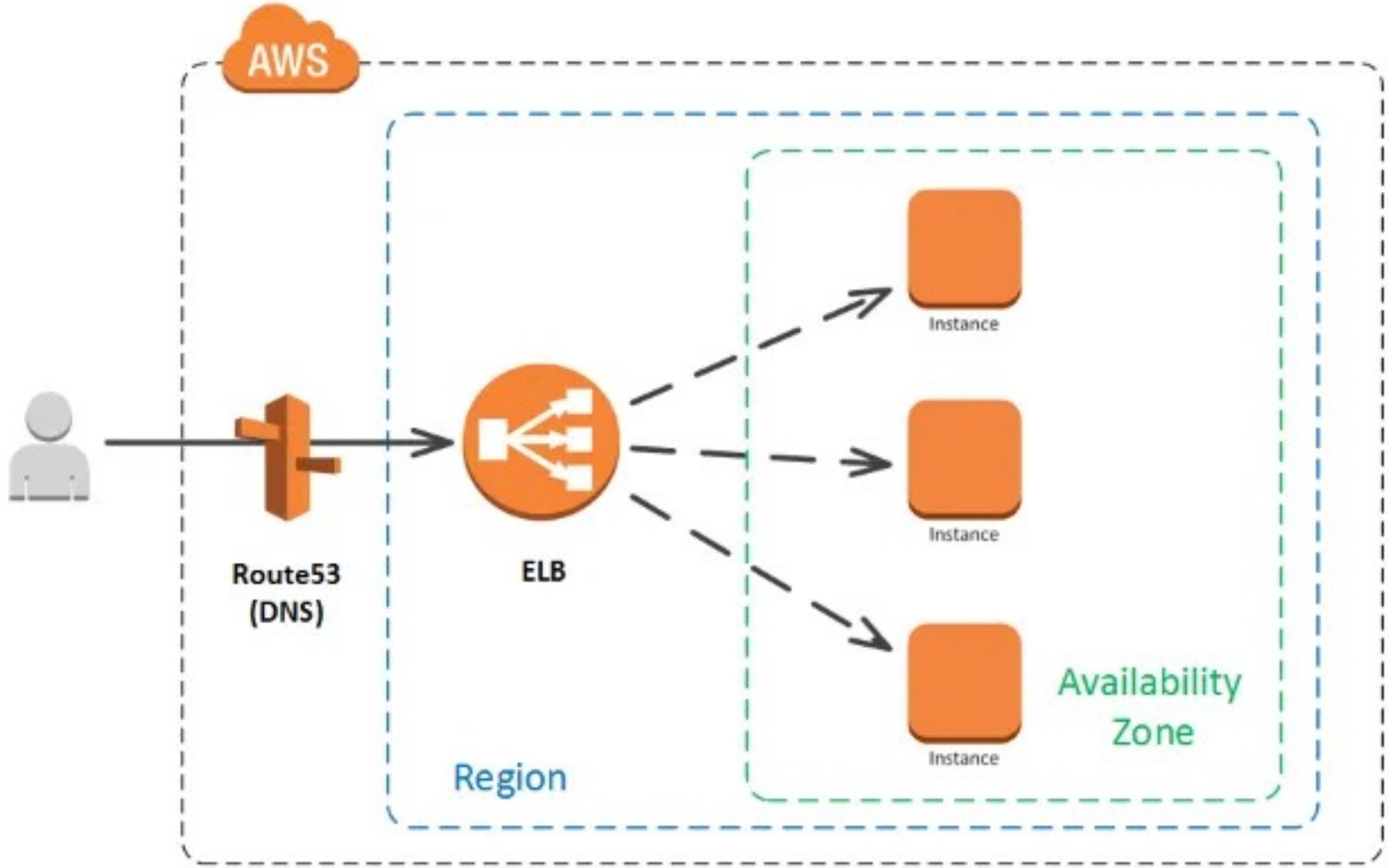
microservice
highly scalable
highly available

microservice
highly scalable
highly available

microservice
highly scalable
highly available

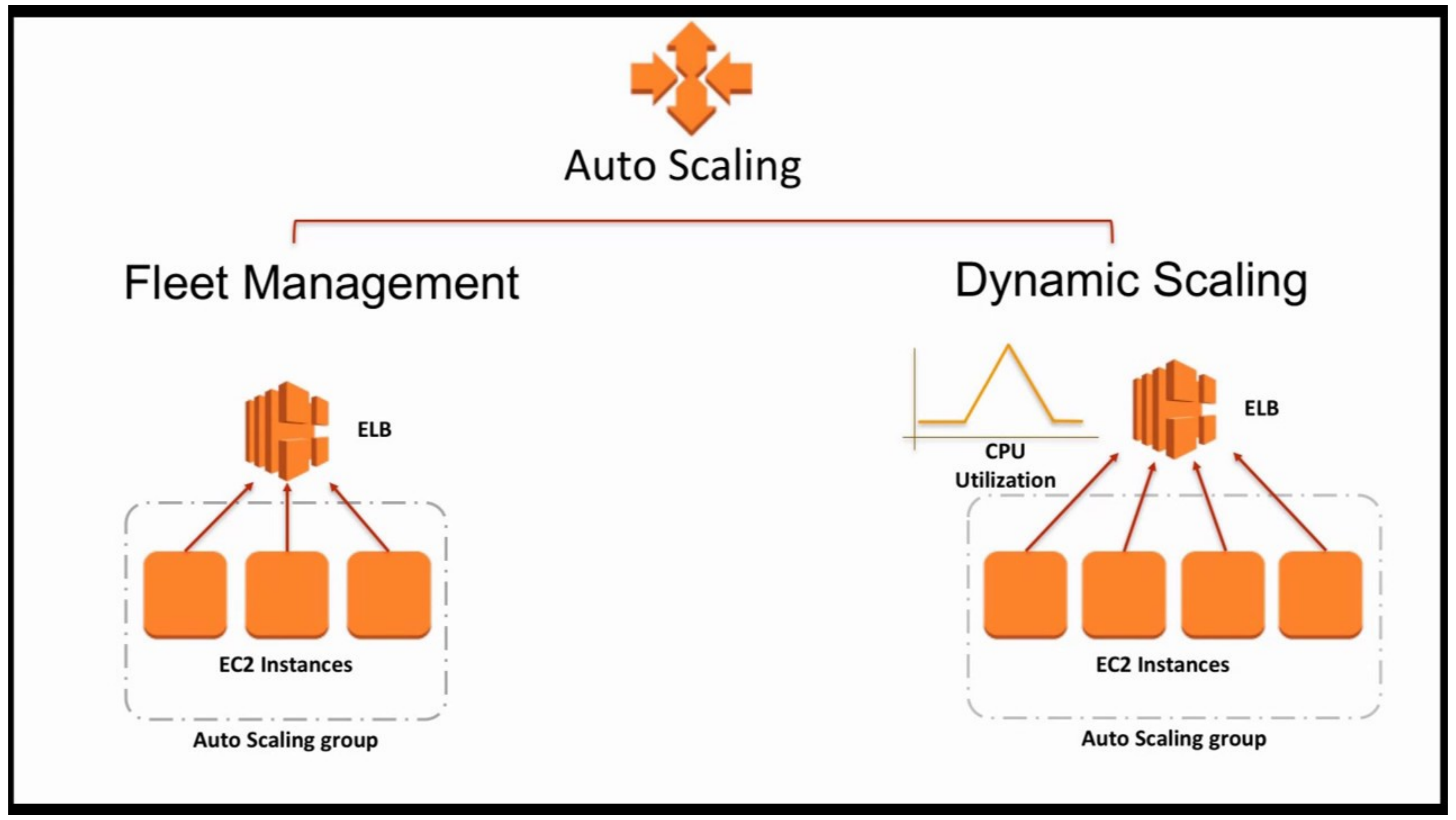
microservice
highly scalable
highly available

Load balancer



Auto scaling

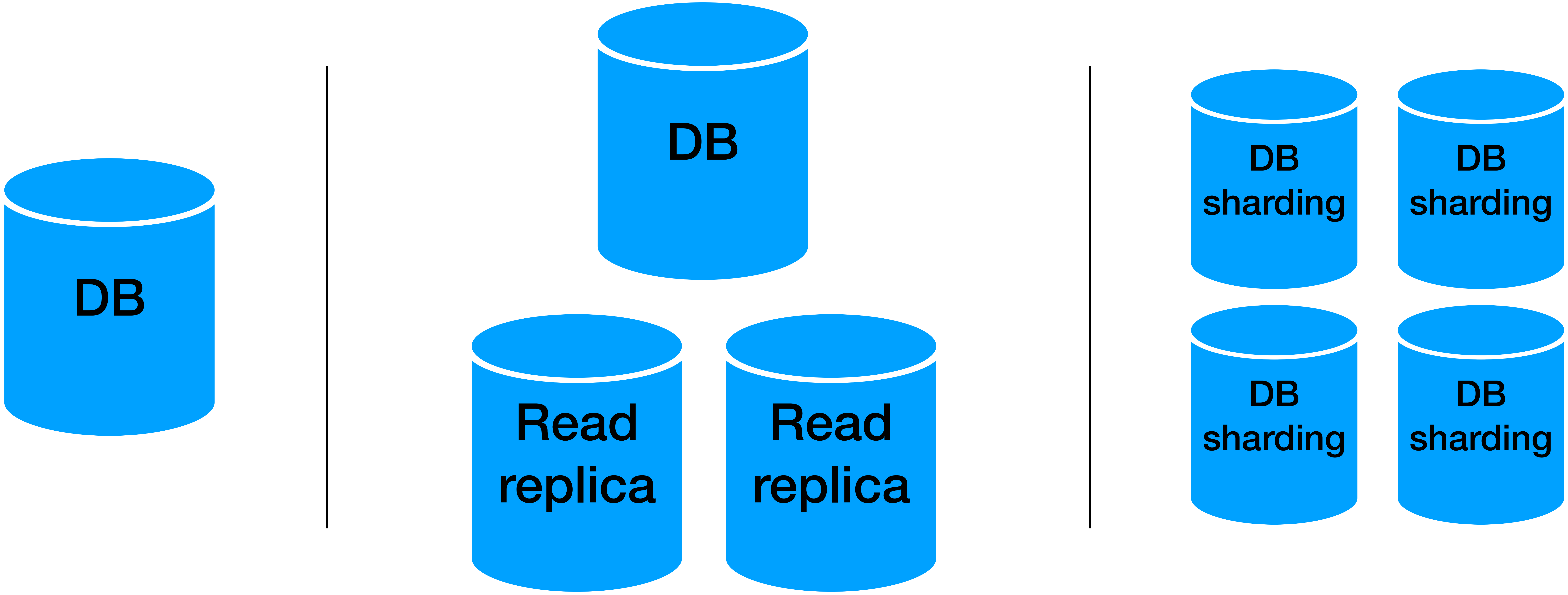
- When threshold occurs (hits / traffic / CPU…), create a new instance with the same logic and add to the load balancer
- When threshold drops, remove the from the load balancer and terminate the instance
- Usually requires stateless logic
  can Cassandra work with auto scale?



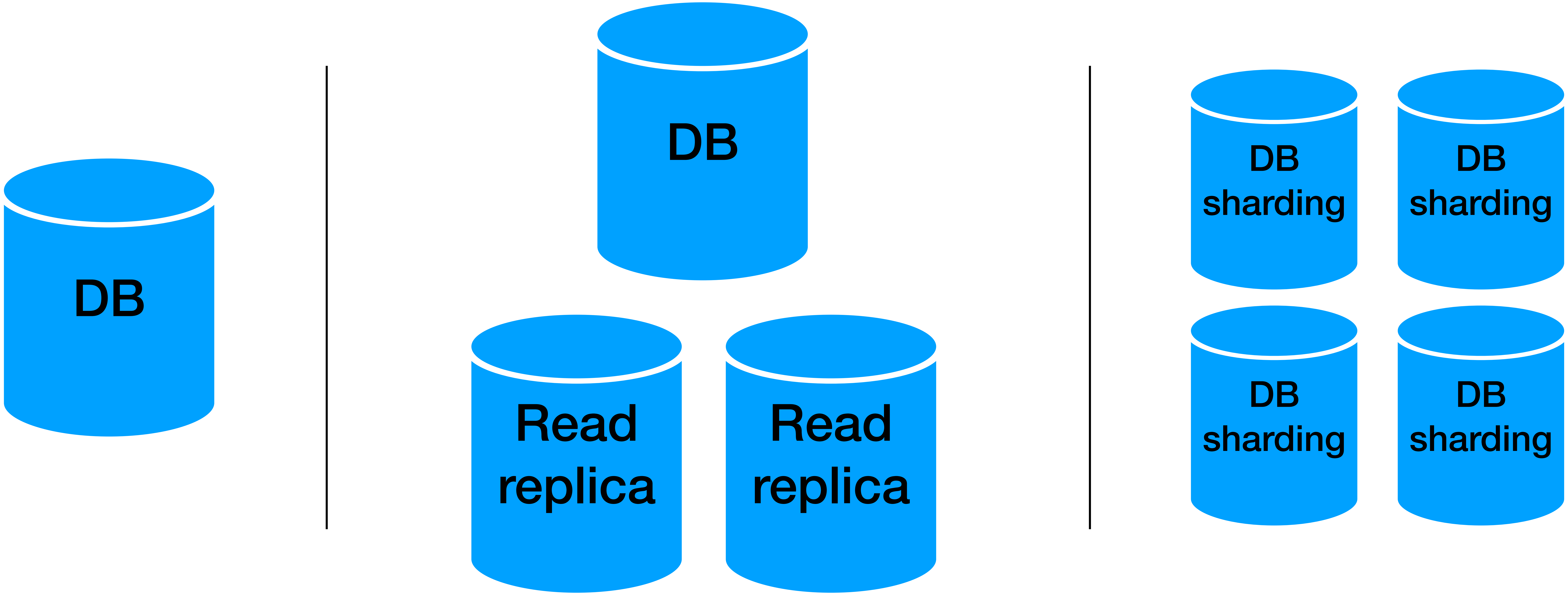
Auto scaling - compute + storage?

- Some applications use both compute and storage (databases)
- Stateless?
- What happens when we scale down?

Scaling databases



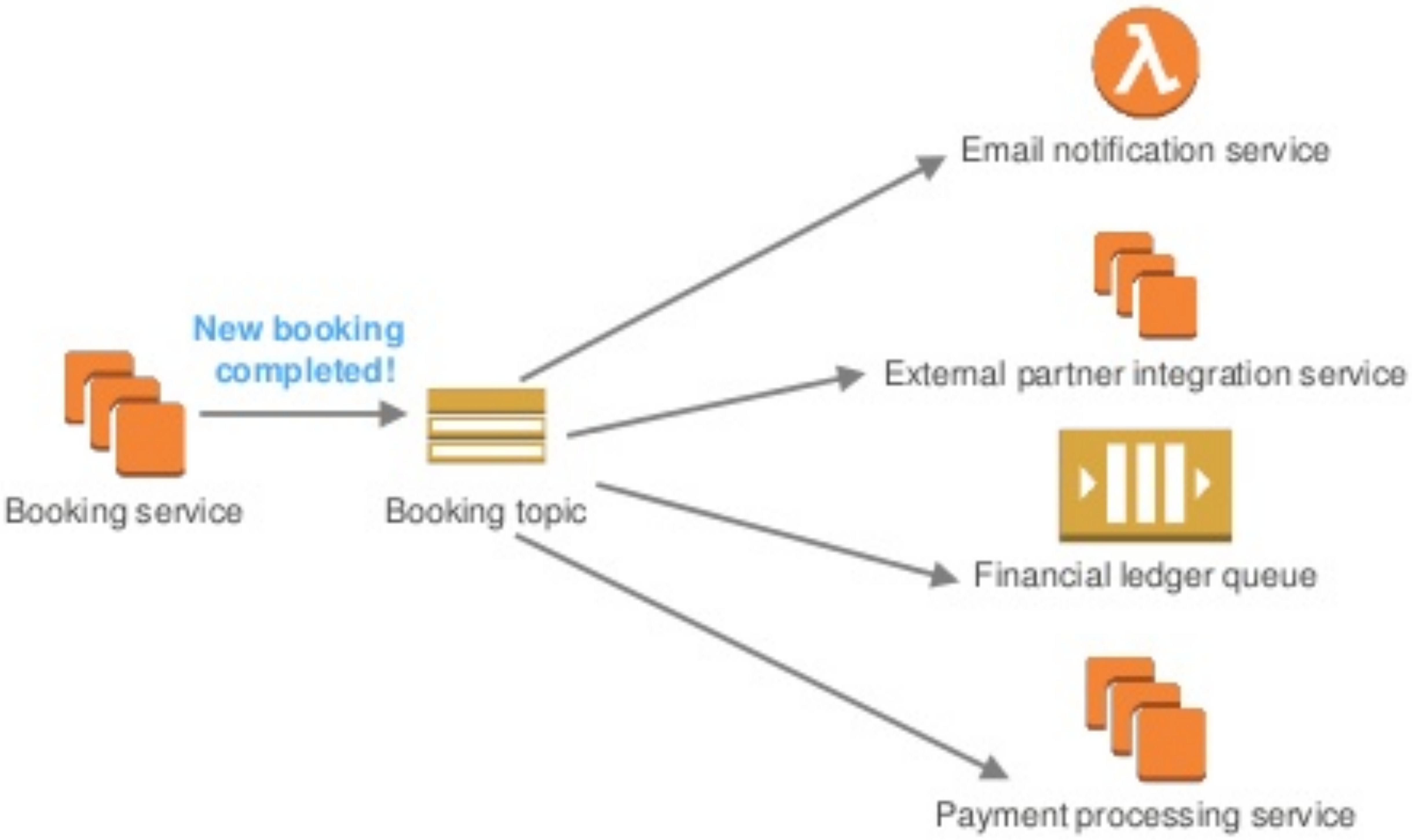
Scaling databases



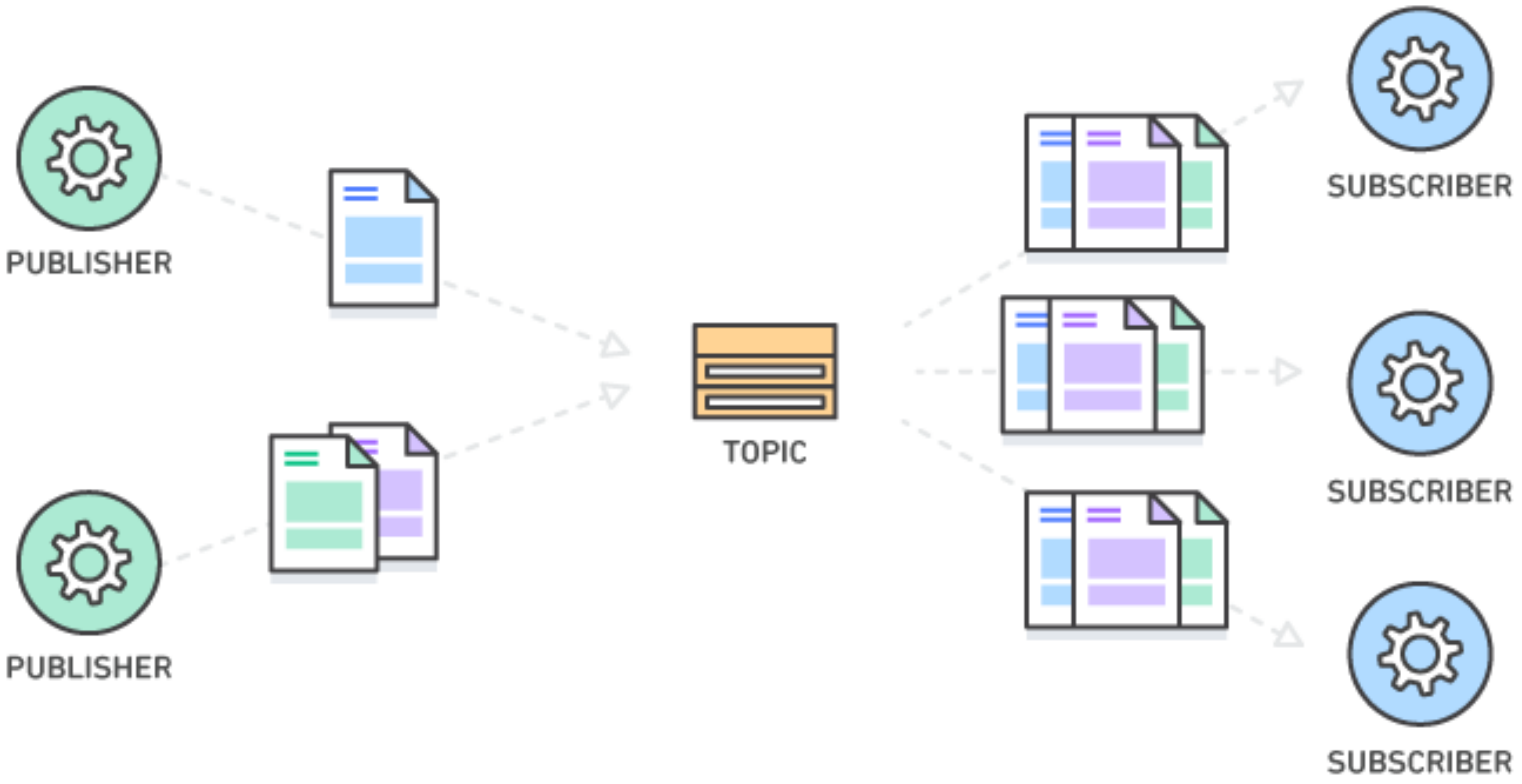
Warning - we will talk about this a lot :)

Decoupling + event based services

- autonomous and unaware of each other services


New booking completed!
Booking service
Booking topic
Email notification service
External partner integration service
Financial ledger queue
Payment processing service

Pub sub


PUBLISHER
PUBLISHER
TOPIC
SUBSCRIBER
SUBSCRIBER
SUBSCRIBER

Managed vs Unmanaged services

Unmanaged service

You are responsible for everything!

- Choosing CPUs, storage, network…
- Installing OS, Java, core software, dependencies…
- Patches, updates
- Security
- Backup
- Monitoring
- Availability

Unmanaged service (2)

Requires different skills

- System
- DevOps
- …

Managed service

- All the stuff we talked about before are managed for you out of the box
- Hardware utilization
- Focus on stuff that really matters for you
- Cost?

Managed service cons

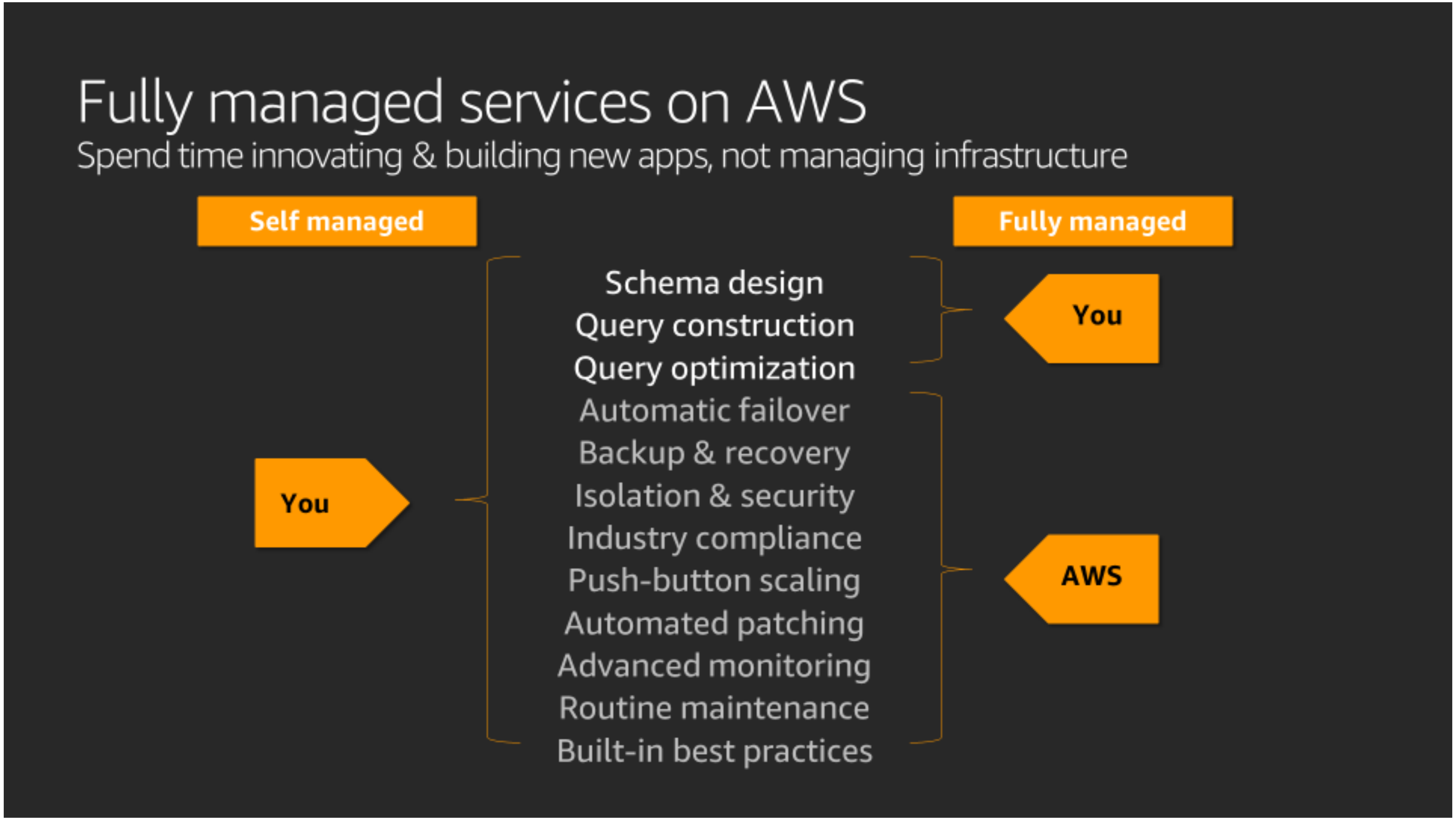
- **Cloud locked in**
- Slightly limited functionality
- Works only in the cloud
- Cost?
  (cheaper to go unmanaged on large scale, but a lot of headaches)

In practice

- Some will be managed and some not
  VMs
  load balancers
  network stuff
  …

- **To go managed or unmanaged with databases is a good question**

Managed vs Unmanaged Databases



Fully managed services on AWS

Spend time innovating & building new apps, not managing infrastructure

Self managed

Fully managed

Schema design
Query construction
Query optimization
Automatic failover
Backup & recovery
Isolation & security
Industry compliance
Push-button scaling
Automated patching
Advanced monitoring
Routine maintenance
Built-in best practices

But how managed service work?

- It is just someone else’s software…

- Do we need to understand how it works behind the scenes?

For databases, YES!

Big Data databases

- Managed big data databases are built on, well, big data databases
- **Data modeling is crucial.**
  (with bad modeling, nothing will work)

To model data correctly,
we need to understand the technology
(it is not just reading the API docs)