Designing Well Architected Framework- Week 1

Reliability, Cost Optimization, and Sustainability

Rohit Bhardwaj

Hands-on Director of Architecture, Salesforce

Founder: ProductiveCloudInnovation.com
Twitter: rbhardwaj1
LinkedIn: www.linkedin.com/in/rohit-bhardwaj-cloud

tinyurl.com/DesigningWellArchitected

https://www.productivecloudinnovation.com/lessons

Design Resilient Architectures

Design Cost-Optimized Architectures

Sustainability Architectures

Design Performant Architectures

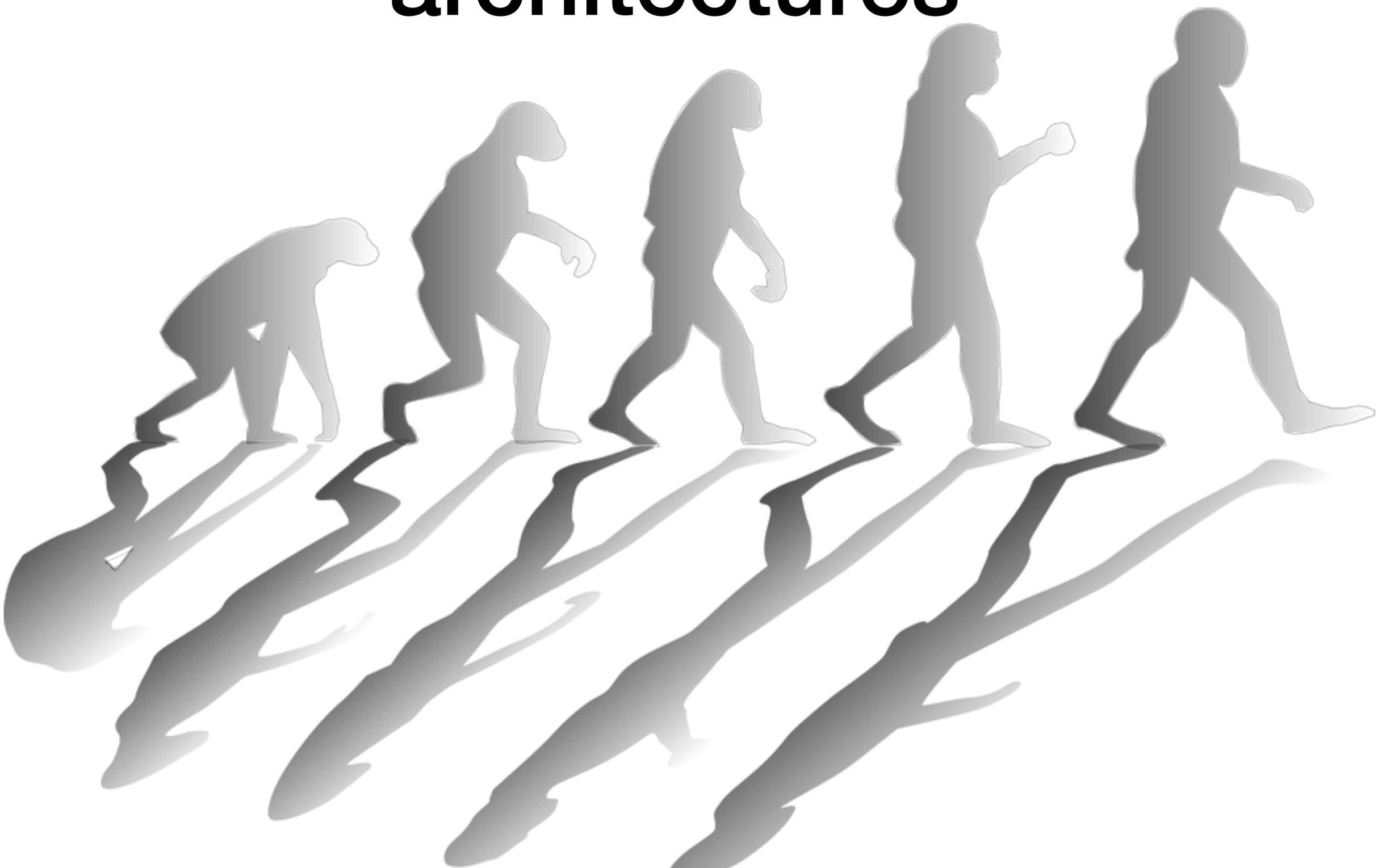
Operationally Excellent Architectures

Specify Secure Applications

Well Architected Framework

Design Principles

Allow for evolutionary architectures



Application Enterprise Architecture Diagram

Gateways

- Portal
- B2B Gateway

Sales and Marketing

- Channel Management
- Leads Management
- Sales Force Management
- Campaign Management
- Quoting

Customer Interaction

- Contact Management
- Customer Management
- Order Entry
- Trouble Tracking

Finance

- Customer Risk Management
- Revenue Assurance
- Billing
- Medication and Rating
- Accounting
- Vendor Billing Reconcillation

Business Intelligence

- Corporate Data Warehouse
- Decision Support Systems
- Reporting

Product Management

- Product Catalog

Service Delivery

- Order Management
- Inventory
- Service Image
- Assign and Design
- Workforce Management
- Vendor Order Management
- Capacity Planning
- Provisioning Interconnection

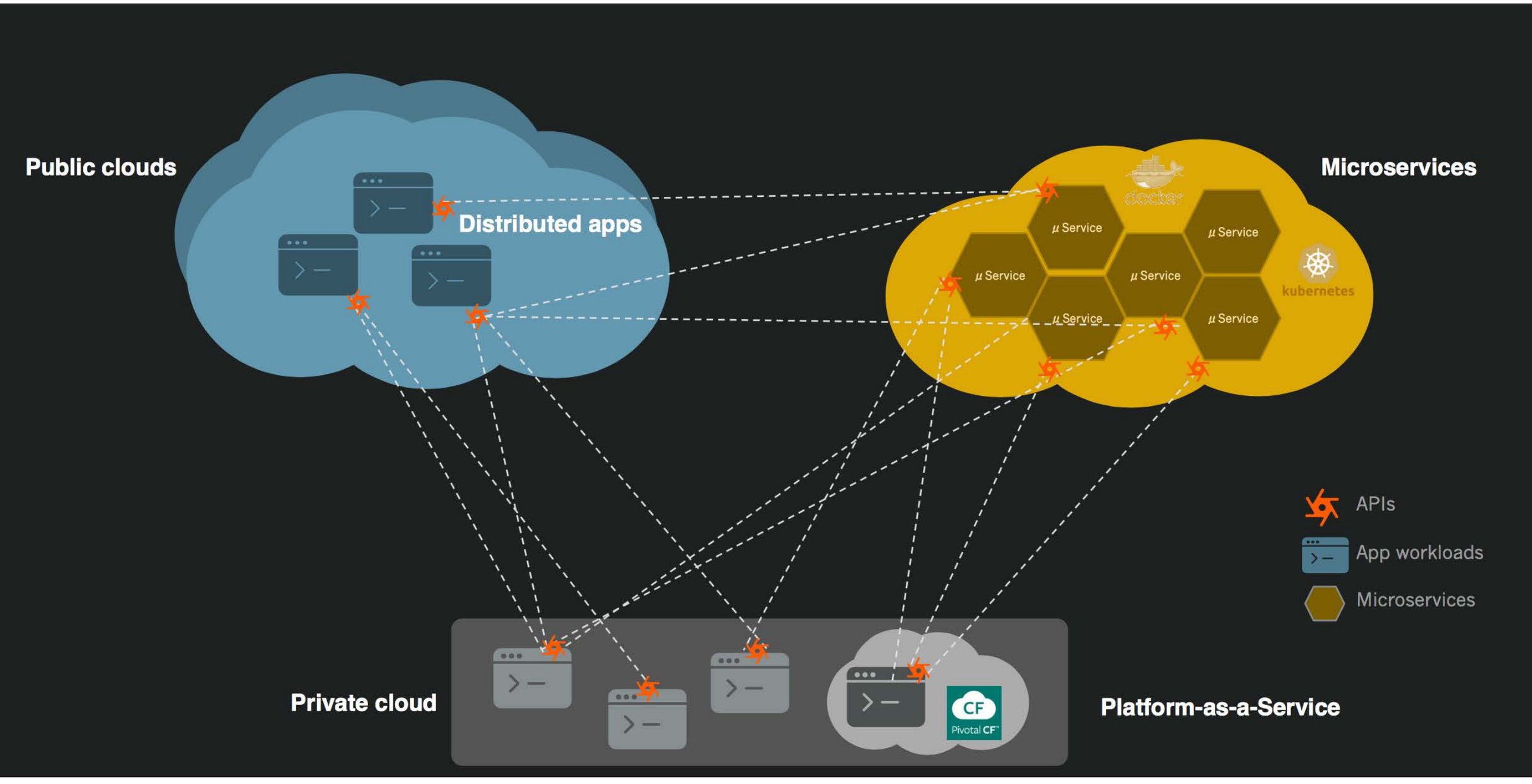
Service Assurance

- Activation Management
- Performance Management
- Trouble Management
- Testing and Diagnostics
- Service Level Management
- Usage Data Collection
- Fault Management
- Network / Element Management

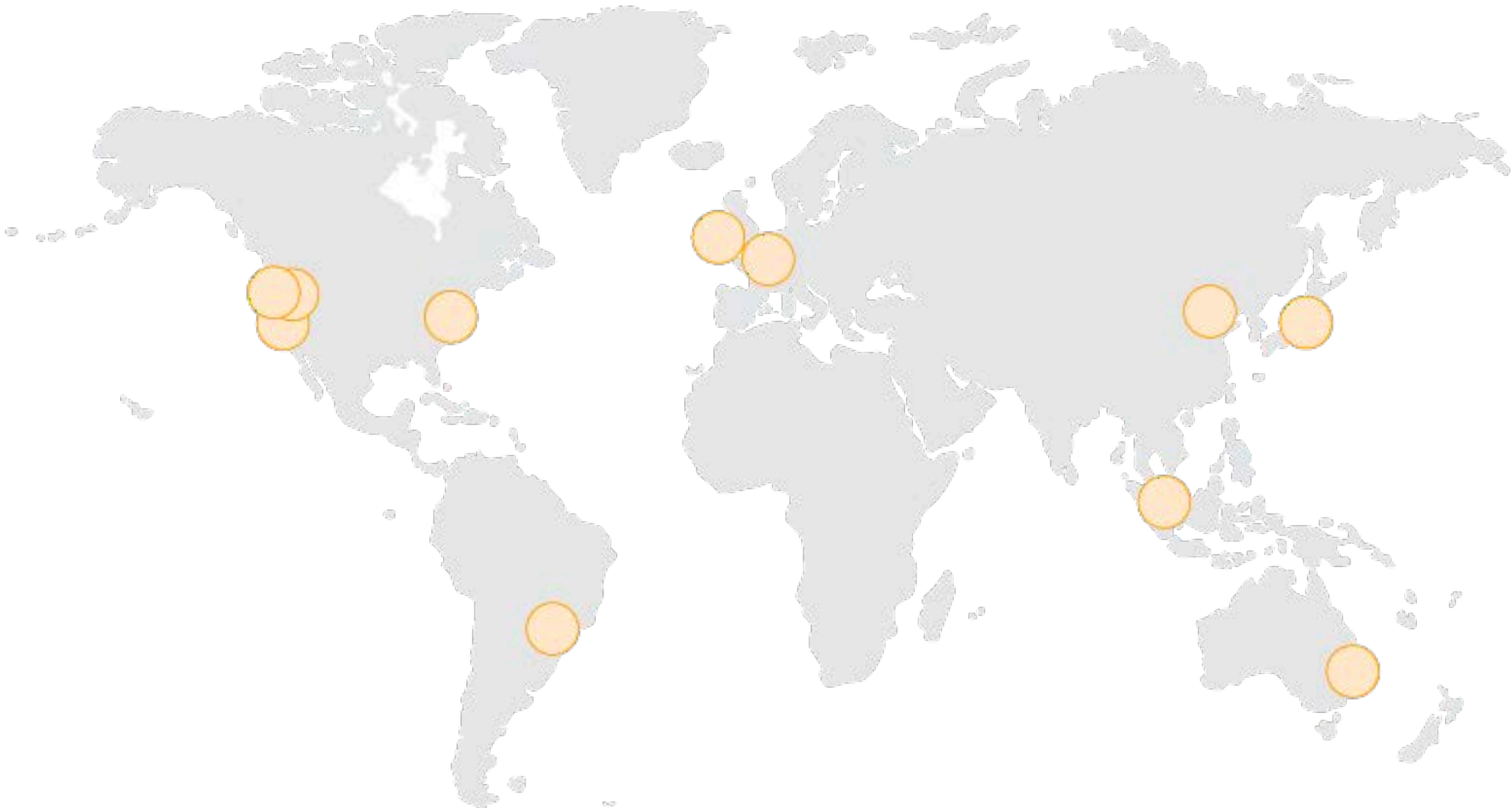
Corporate

- Assest Management
- Human Resources
- Tax and Regulatory
- Legal
- Corp Governance
- Document Managment

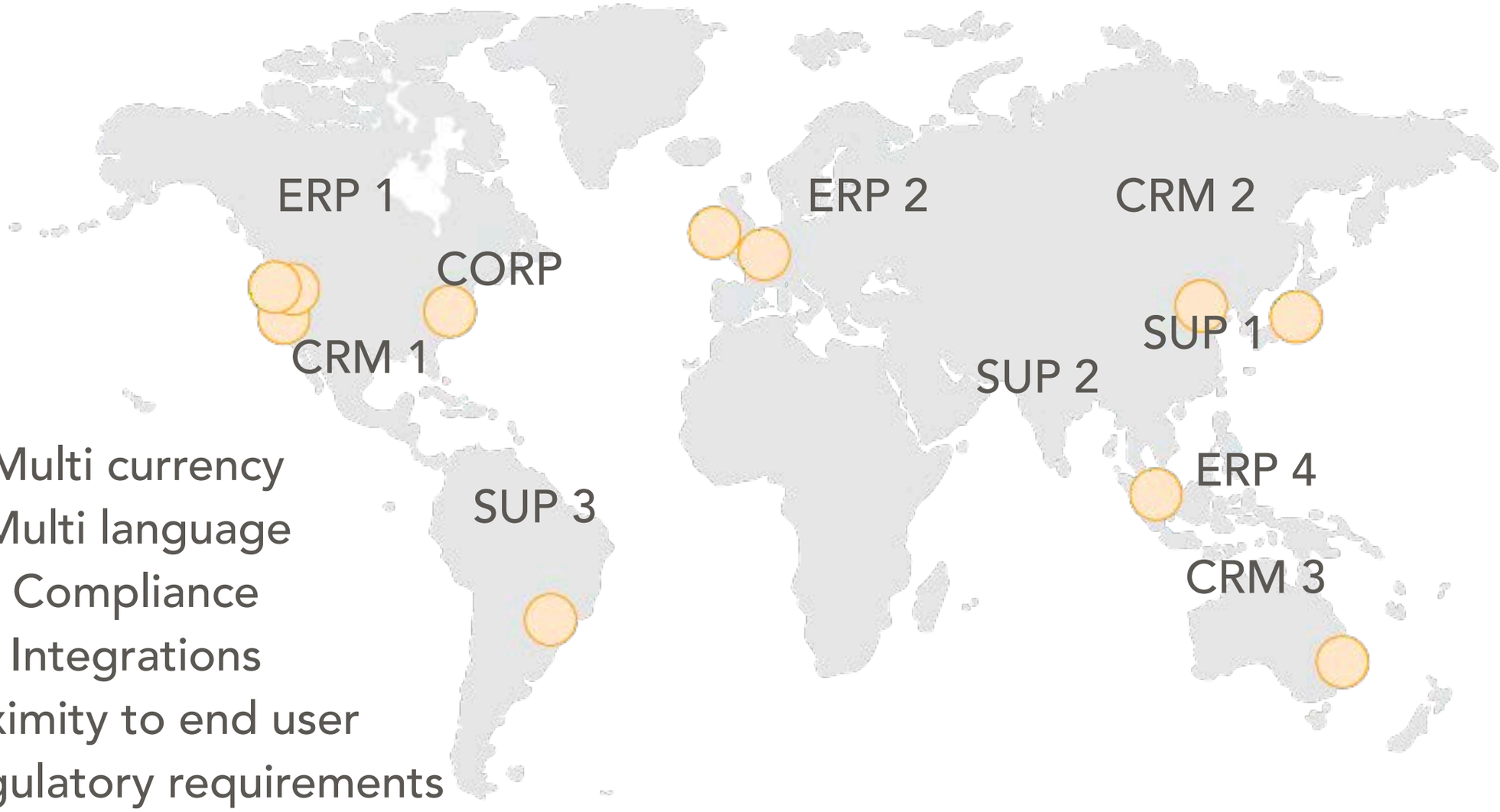
Weakest Link



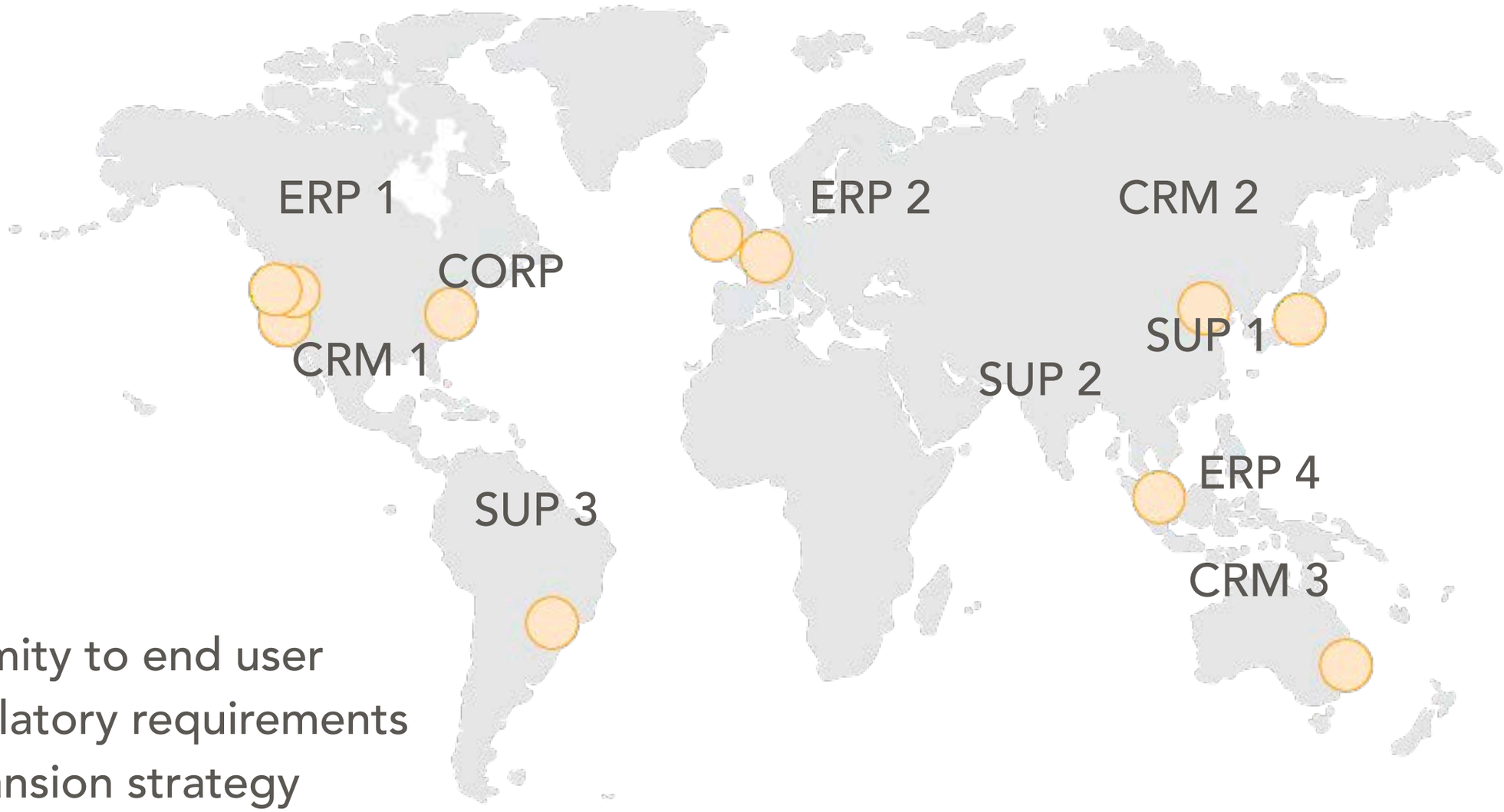
What are the Design Considerations for deploying across globe?



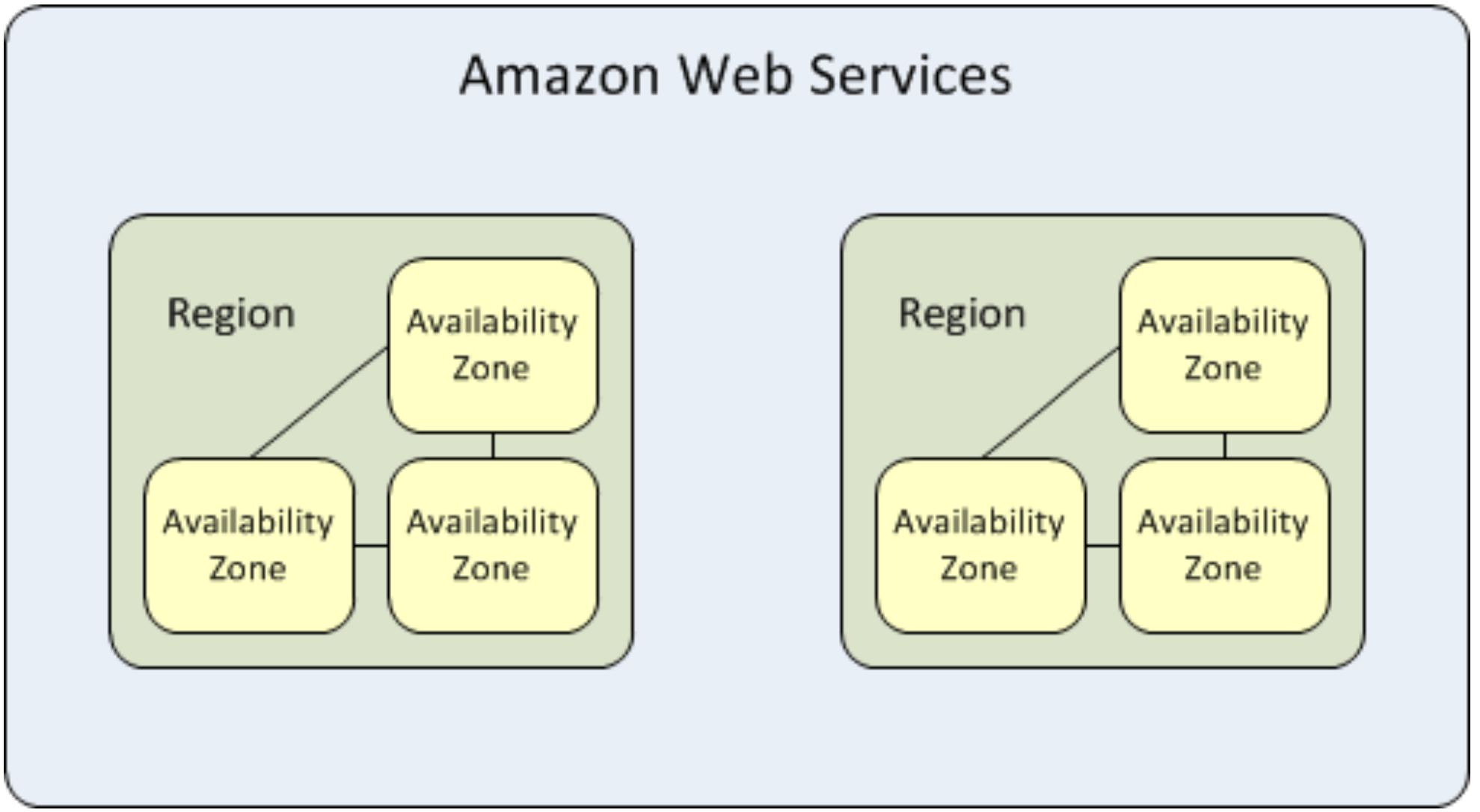
Global Rollout



Multi currency
Multi language
Compliance
Integrations
Proximity to end user
Data regulatory requirements
Expansion strategy
Cost
Service availability



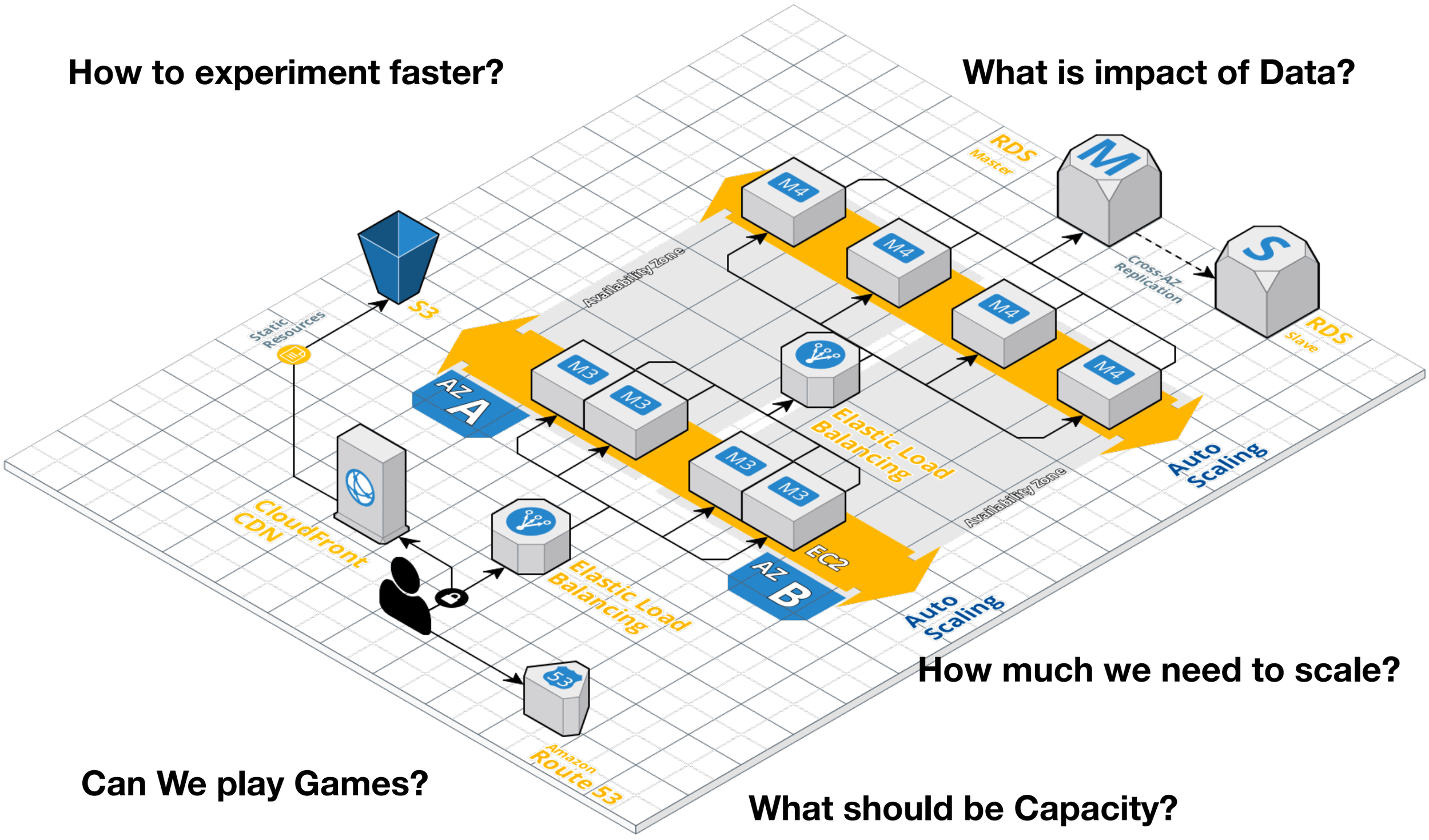
Proximity to end user
Data regulatory requirements
Expansion strategy
Cost
Service availability


Amazon Web Services
Region
Availability Zone
Availability Zone
Availability Zone
Region
Availability Zone
Availability Zone
Availability Zone

How to allow Evolutionary Architecture?

How to experiment faster?

What is impact of Data?

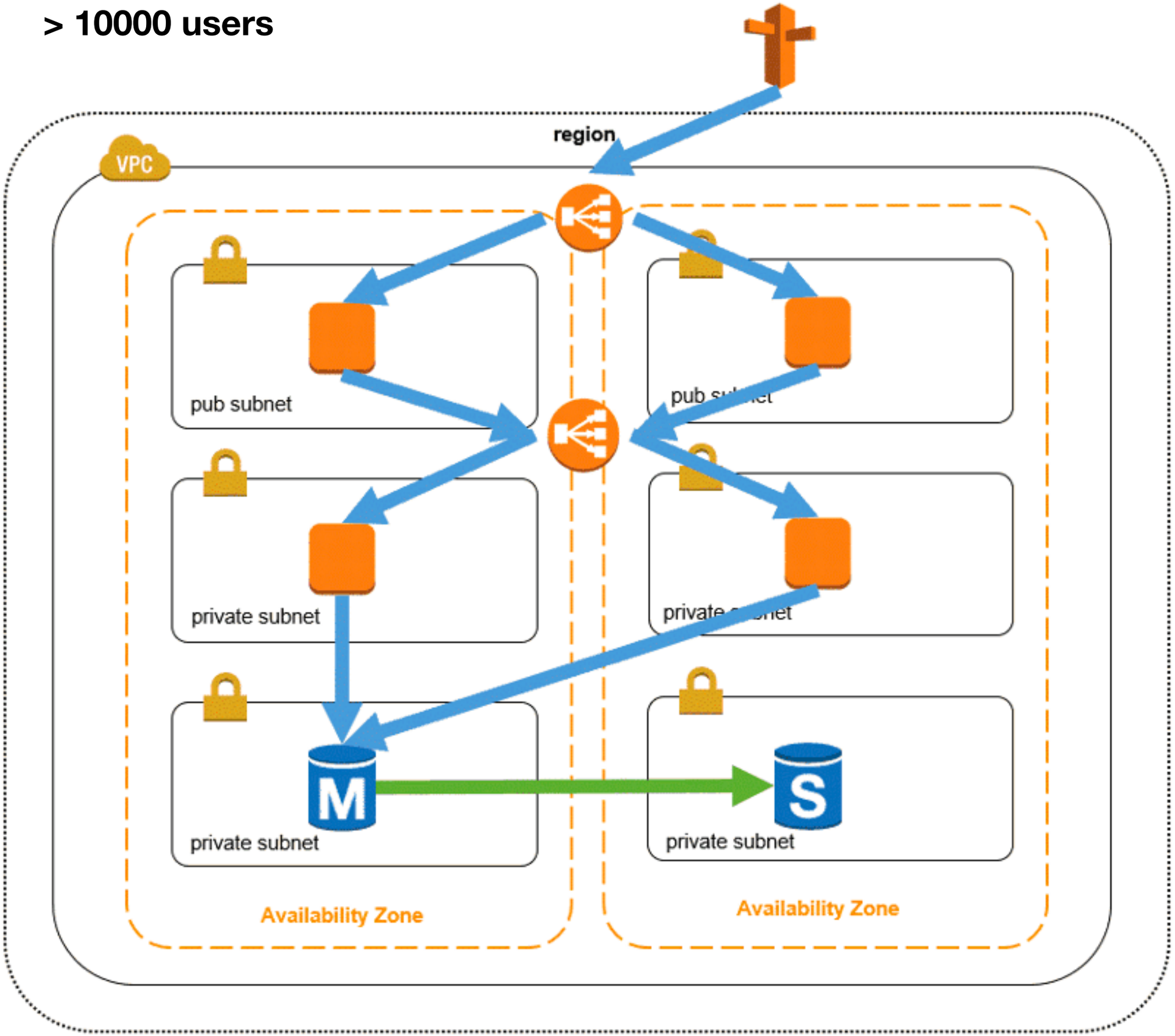


How much we need to scale?

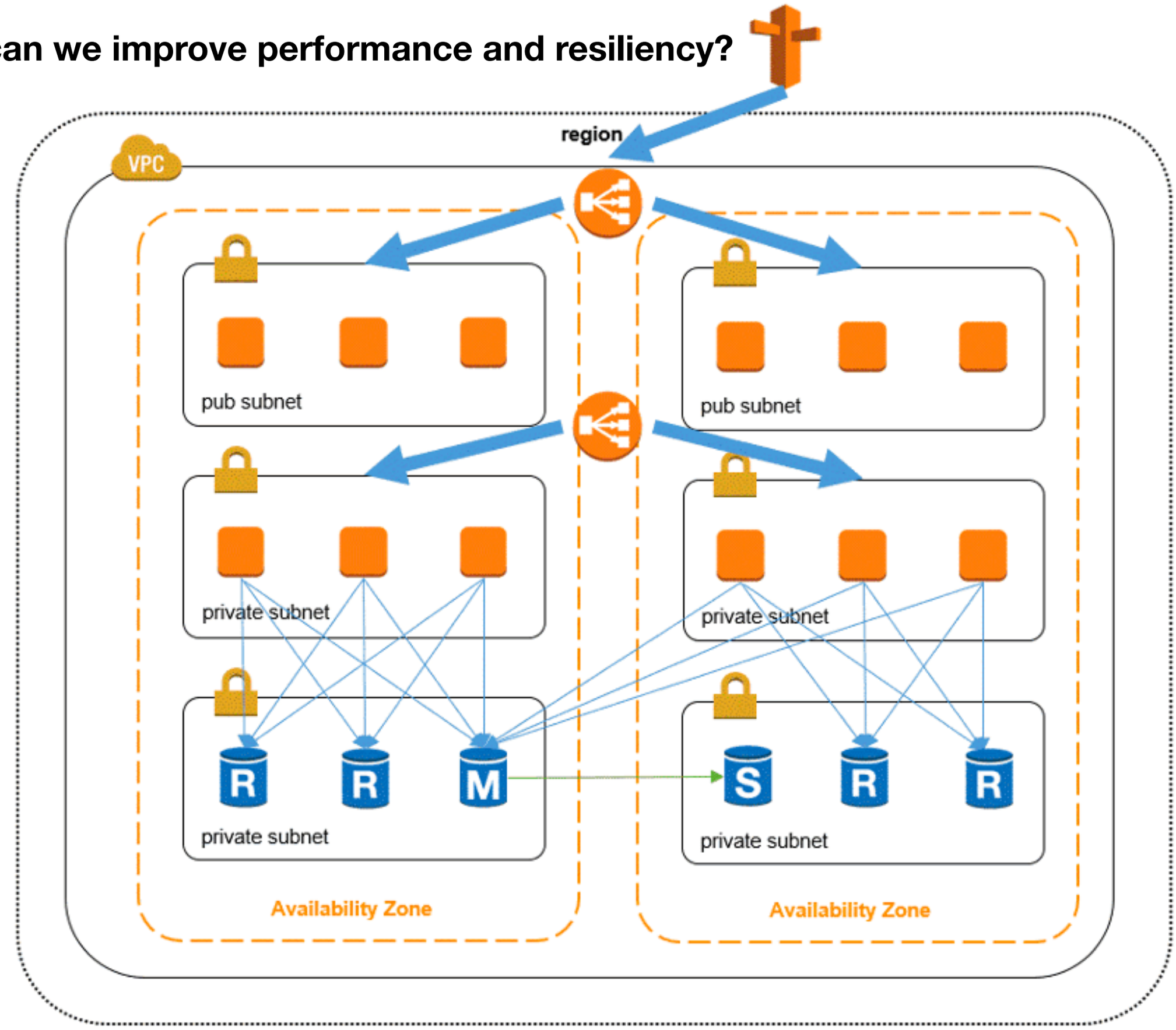
Can We play Games?

What should be Capacity?

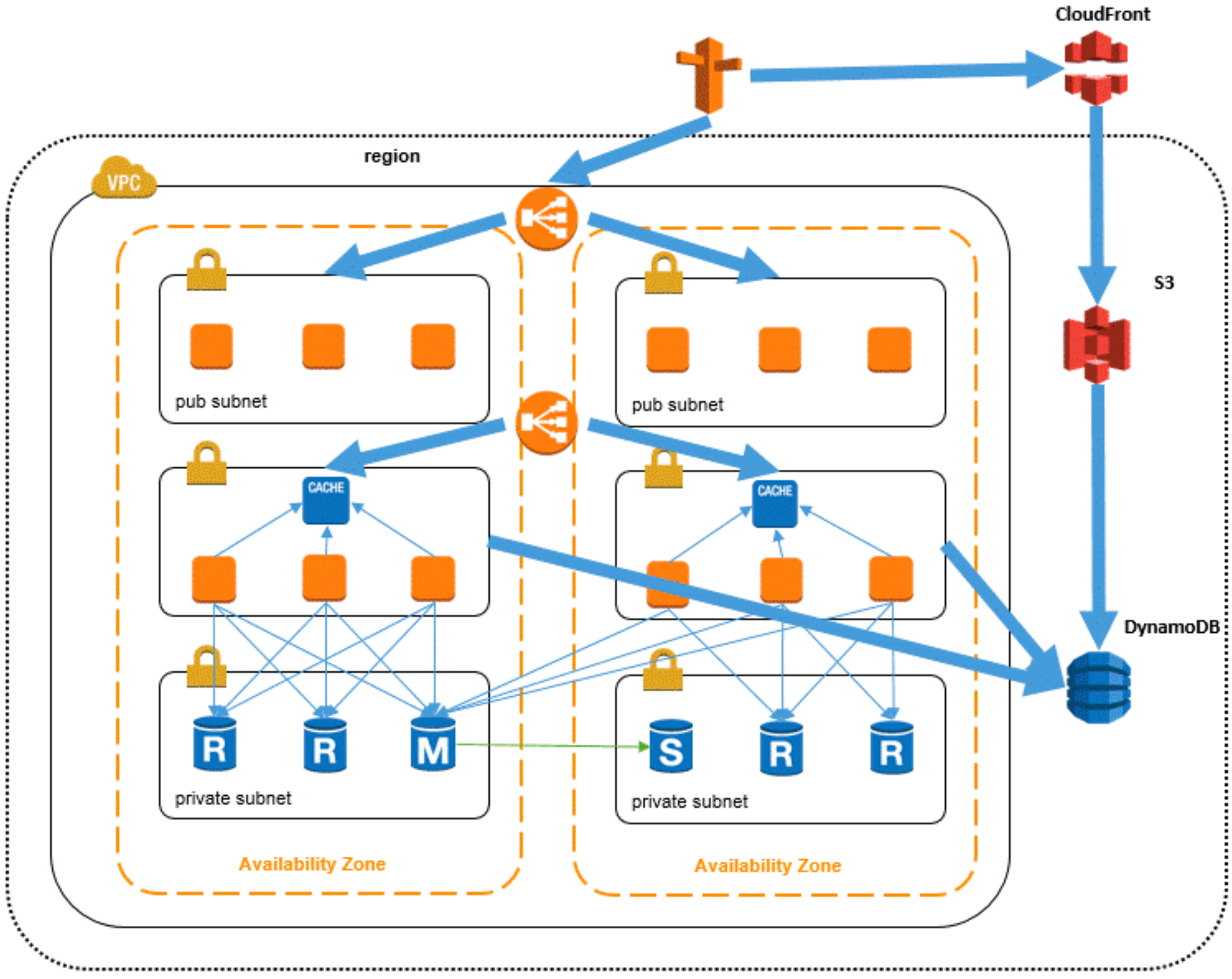
> 10000 users


region
VPC
pub subnet
pub subnet
private subnet
private subnet
private subnet
private subnet
M
S
Availability Zone
Availability Zone

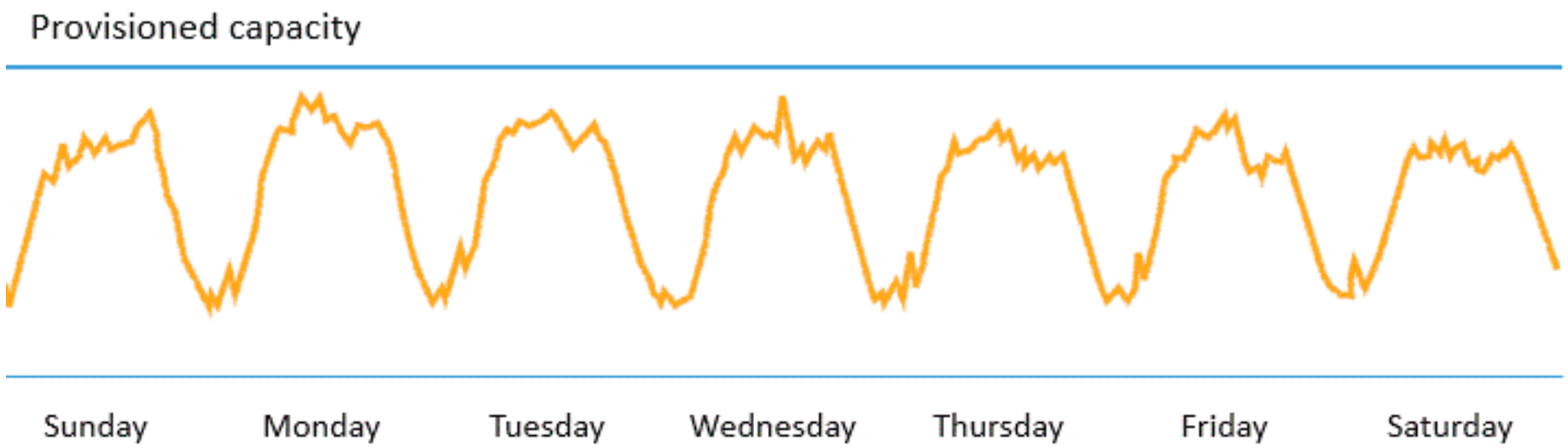
How can we improve performance and resiliency?



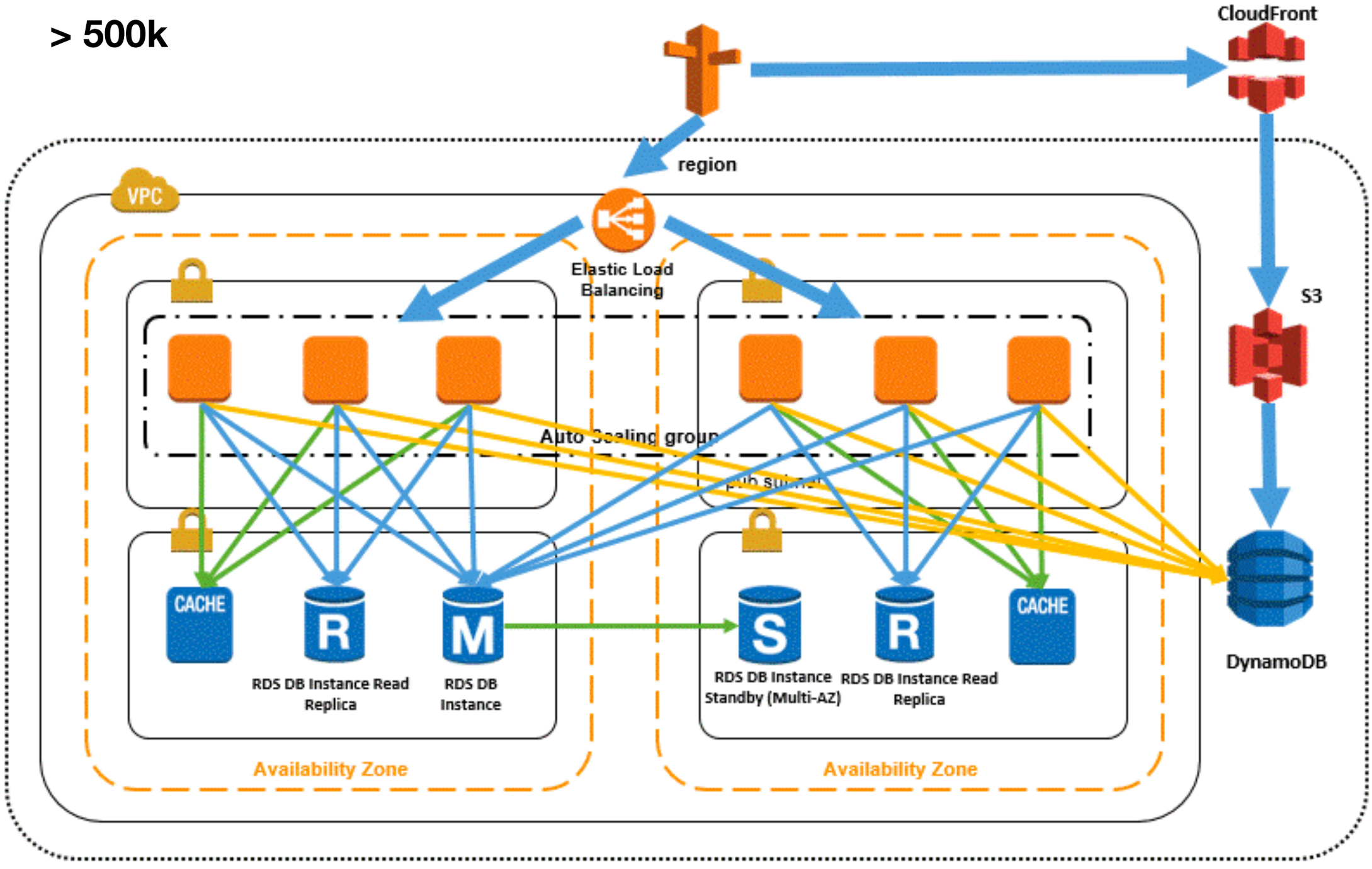
How to take care of variable load?



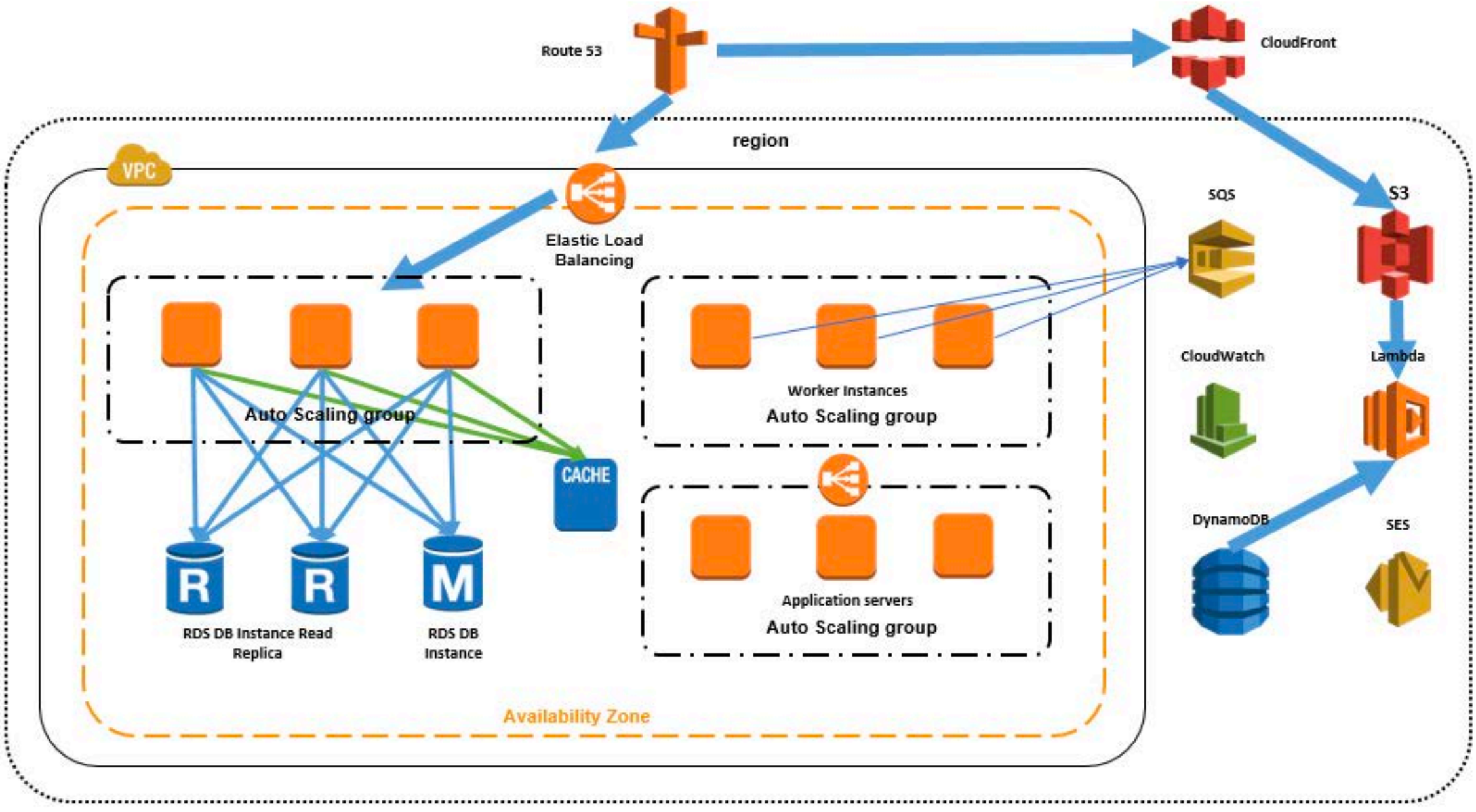
> 500k


Provisioned capacity
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday

> 500k


CloudFront
region
VPC
Elastic Load Balancing
S3
Auto Scaling group
CACHE
R
M
S
R
CACHE
RDS DB Instance Read Replica
RDS DB Instance
RDS DB Instance Standby (Multi-AZ)
RDS DB Instance Read Replica
DynamoDB
Availability Zone
Availability Zone

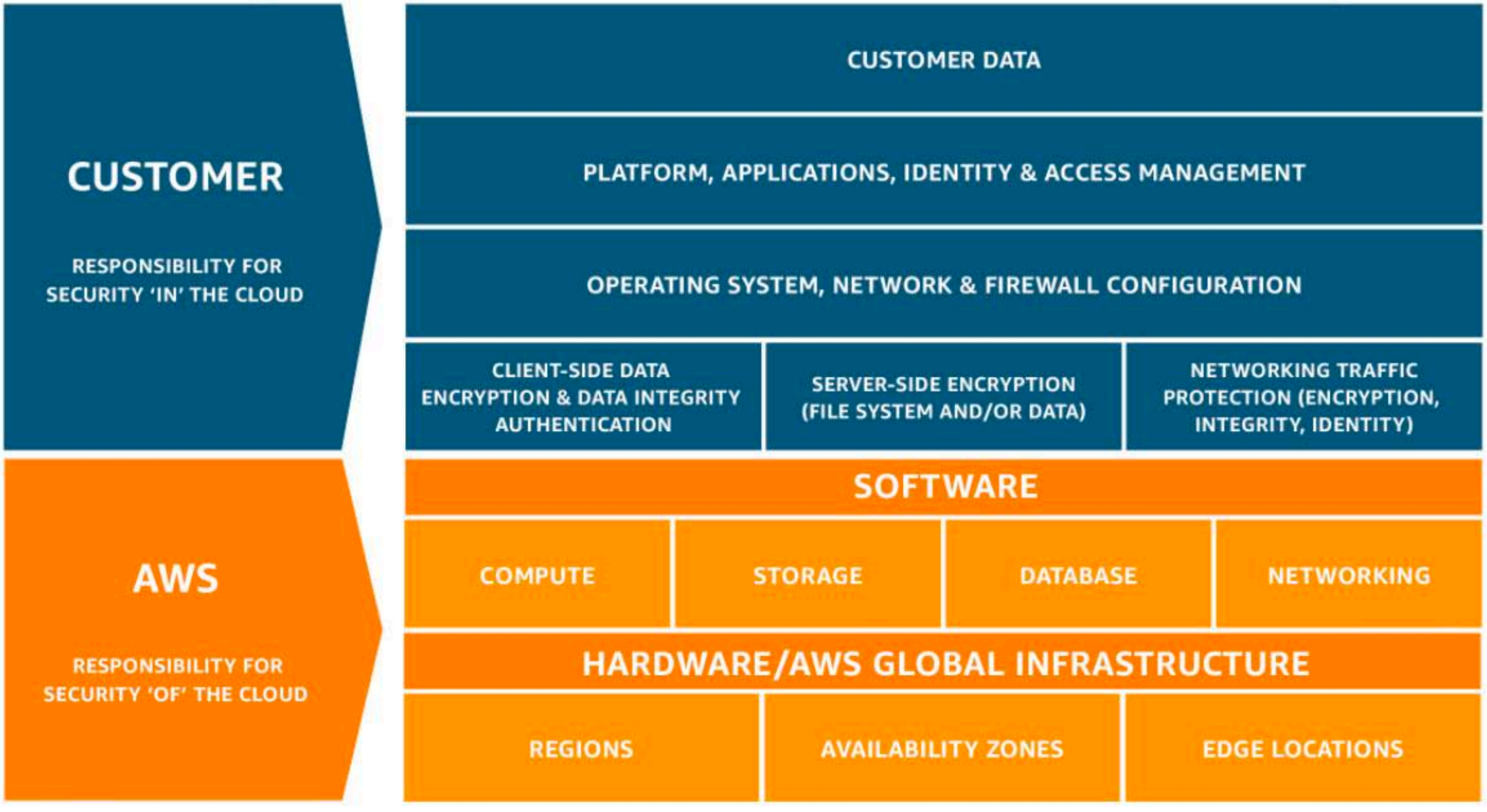
Users 10 to 100 Million What else can we do to improve and scale?



How does Amazon design for peak, brust, average loads during black Friday?

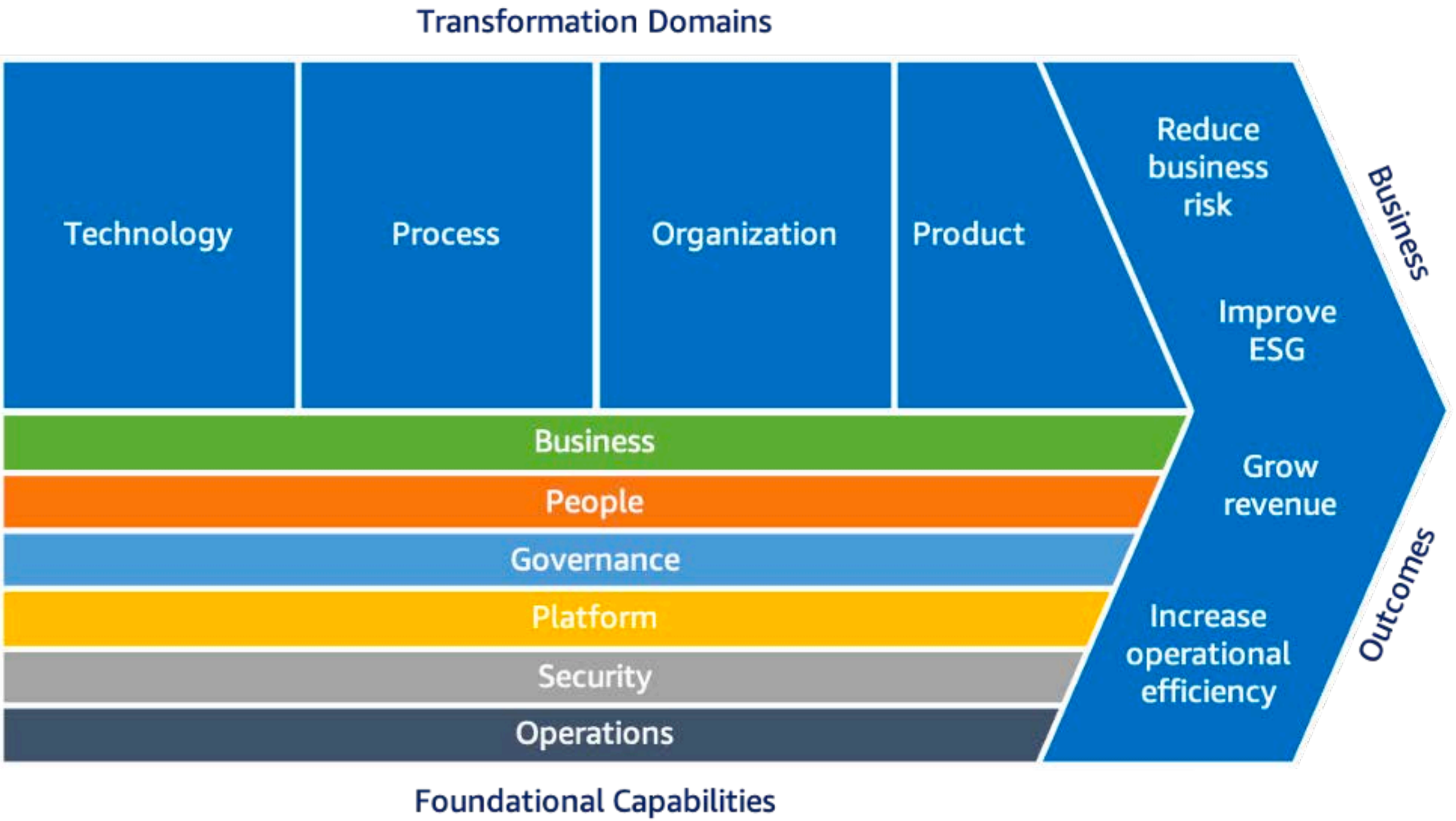
Users 10 to 100 Million

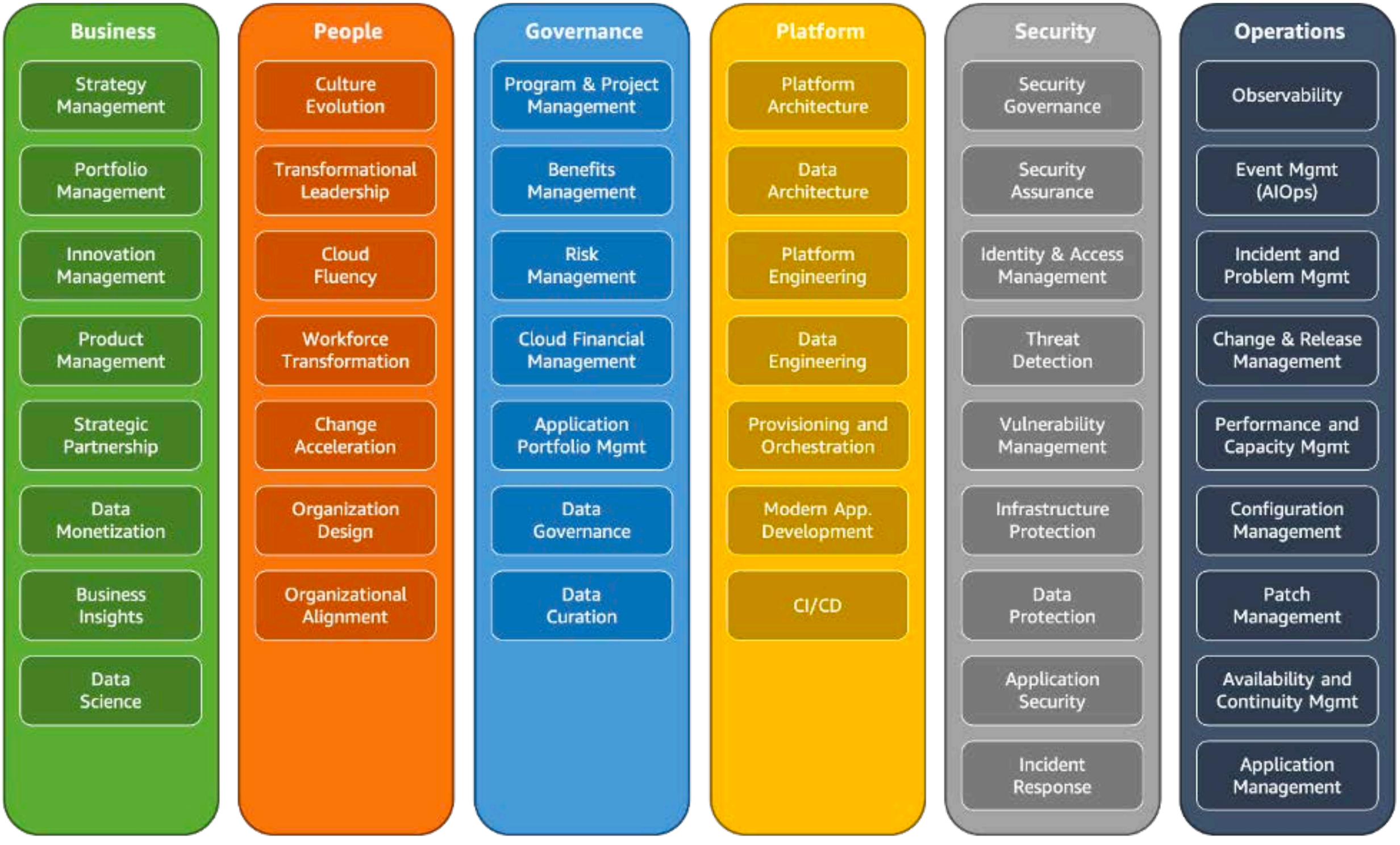
- DB Federation: separating databases by function
- NoSql- Dynamo db
- DB Sharding, Micro services
- Eventual Consistency
- Multi AZ to Multi Region
- Container service



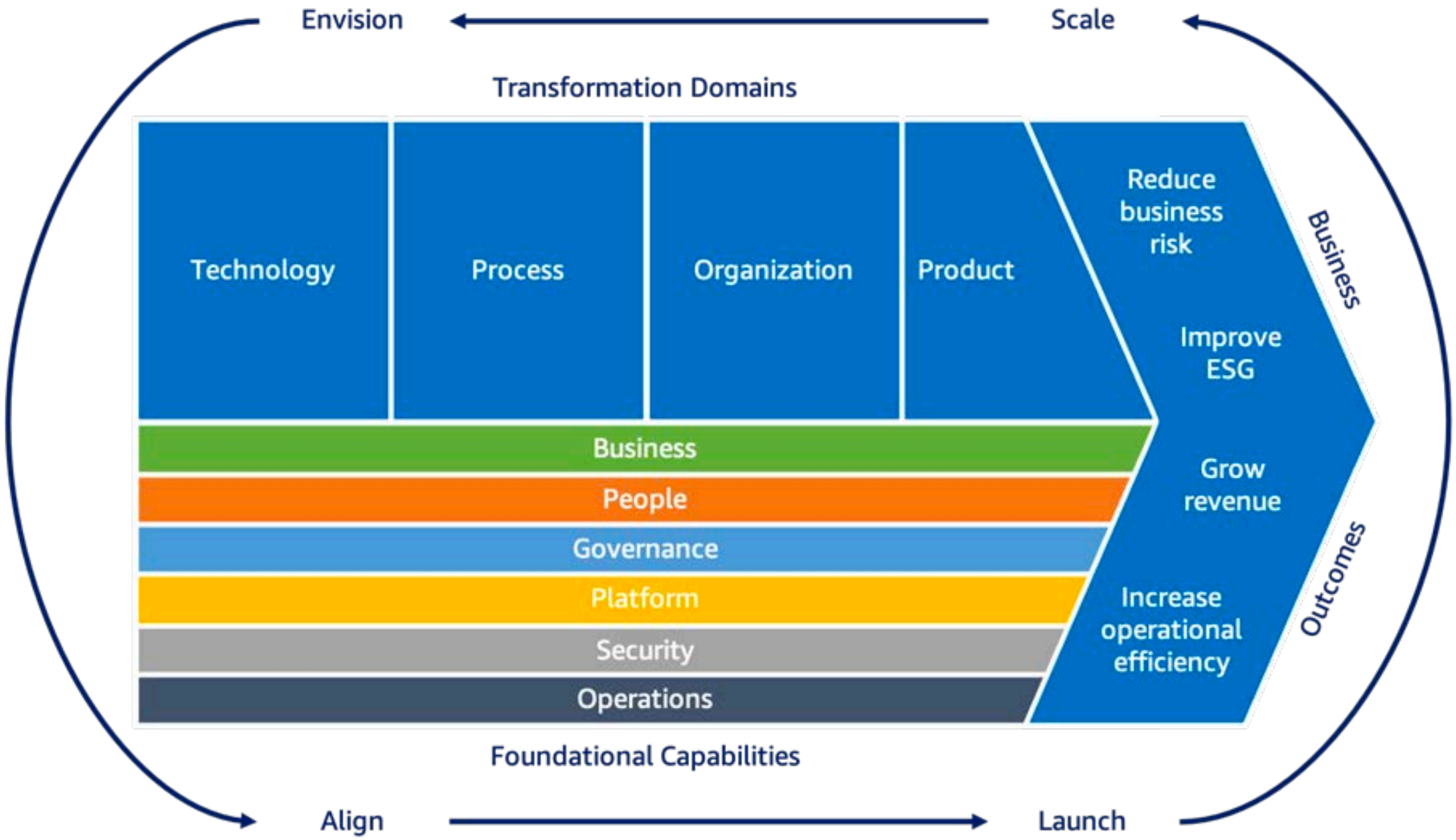
https://aws.amazon.com/compliance/shared-responsibility-model/

AWS Cloud Adoption Framework (CAF) 3.0




Business
Strategy Management
Portfolio Management
Innovation Management
Product Management
Strategic Partnership
Data Monetization
Business Insights
Data Science
People
Culture Evolution
Transformational Leadership
Cloud Fluency
Workforce Transformation
Change Acceleration
Organization Design
Organizational Alignment
Governance
Program & Project Management
Benefits Management
Risk Management
Cloud Financial Management
Application Portfolio Mgmt
Data Governance
Data Curation
Platform
Platform Architecture
Data Architecture
Platform Engineering
Data Engineering
Provisioning and Orchestration
Modern App. Development
CI/CD
Security
Security Governance
Security Assurance
Identity & Access Management
Threat Detection
Vulnerability Management
Infrastructure Protection
Data Protection
Application Security
Incident Response
Operations
Observability
Event Mgmt (AIOps)
Incident and Problem Mgmt
Change & Release Management
Performance and Capacity Mgmt
Configuration Management
Patch Management
Availability and Continuity Mgmt
Application Management

AWS Cloud Adoption Framework (CAF) 3.0



SaaS
Enablement

Marketplace
Custom Packaging
Premium CDN & DNS
Built-In Billing

Cloud Scripting
WHMCS
verizon
CloudBlue

PaaS
Management

App Deployment
Auto-Scaling & Clustering
CI/CD Automation
Container Orchestration

GitHub
GitLab
kubernetes
docker

IaaS
Optimization

Containers
Virtual Machines
Network
Storage

Virtuozzo
KVM
OPEN CONTAINER INITIATIVE
ceph
openstack
AWS
AZURE

https://www.aalpha.net/blog/the-difference-between-paas-iaas-and-saas/

Design Resilient Architectures

Design Cost-Optimized Architectures

Sustainability Architectures

Design Performant Architectures

Operationally Excellent Architectures

Specify Secure Applications

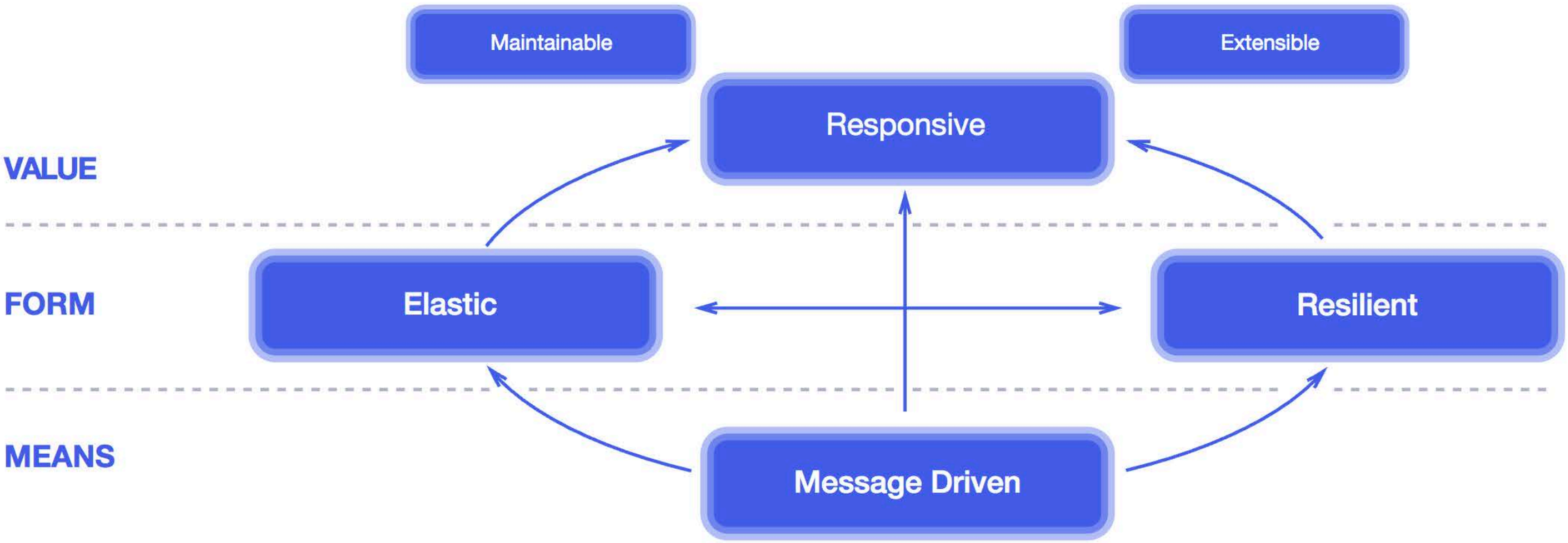
Well Architected Framework

Design Resilient Architectures

Design Resilient Architectures

- 1.1 Choose reliable/resilient storage.
- 1.2 Determine how to design decoupling mechanisms using AWS services.
- 1.3 Determine how to design a multi-tier architecture solution.
- 1.4 Determine how to design high availability and/or fault tolerant architectures.

Resilient Architecture



$$Availability = \frac{Available\ for\ Use\ Time}{Total\ Time}$$

Python - infinity timeout!

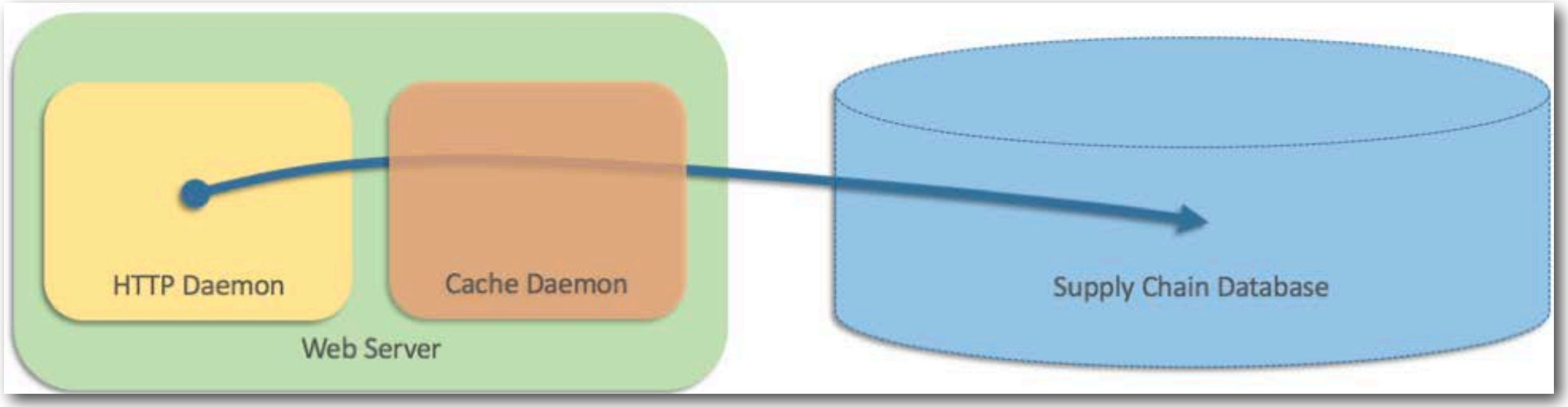
- connect_timeout=31 # number of seconds to wait for connection to be established
- read_timeout=5 # maximum number of seconds between data reads

- r = requests.get('https://github.com', timeout=(connect_timeout, read_timeout))

MYSQL 5.7 or higher

- SELECT /*+ MAX_EXECUTION_TIME(1000) */ status, count(*) FROM articles GROUP BY status ORDER BY status;
- SET SESSION MAX_EXECUTION_TIME=2000;
- SET GLOBAL MAX_EXECUTION_TIME=2000;

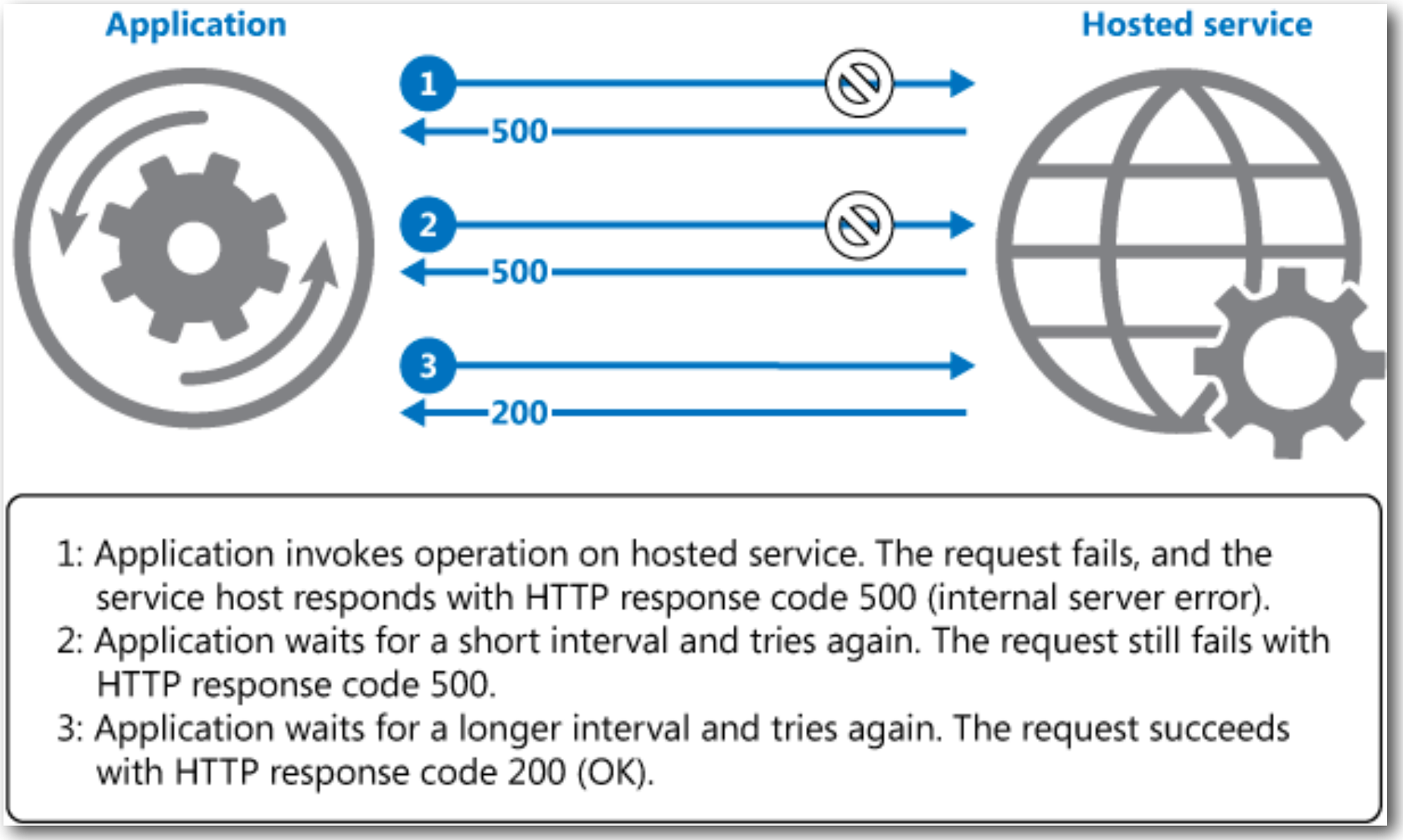
Distributed Systems - No Fallback



https://aws.amazon.com/builders-library/avoiding-fallback-in-distributed-systems/

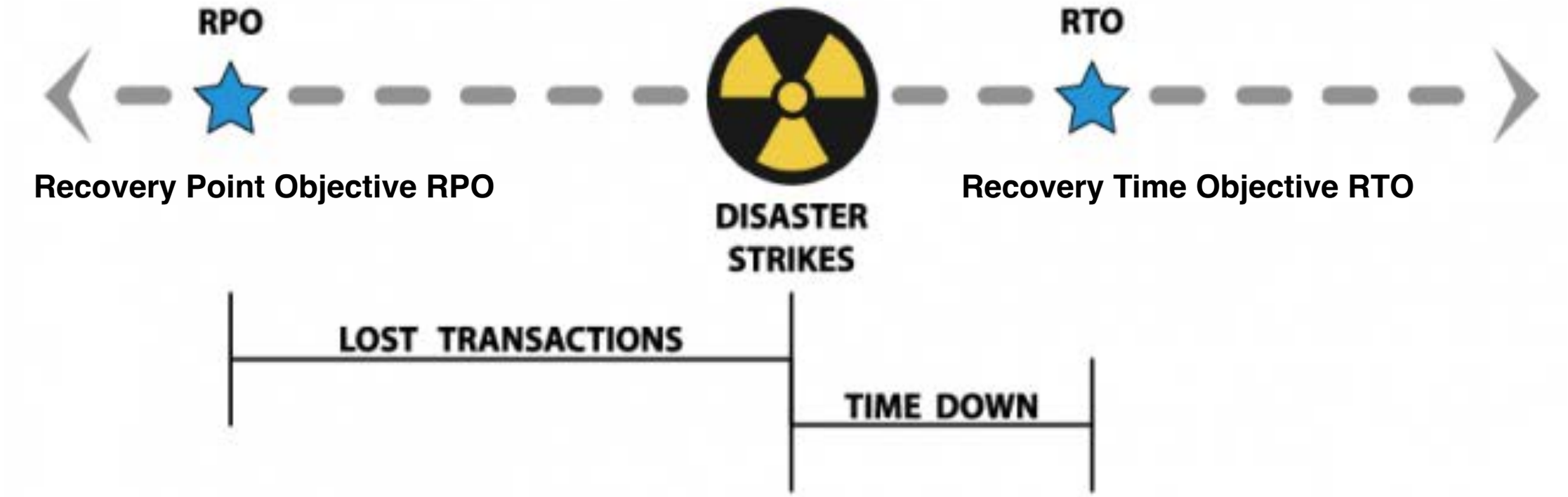
Fail Fast

Retry



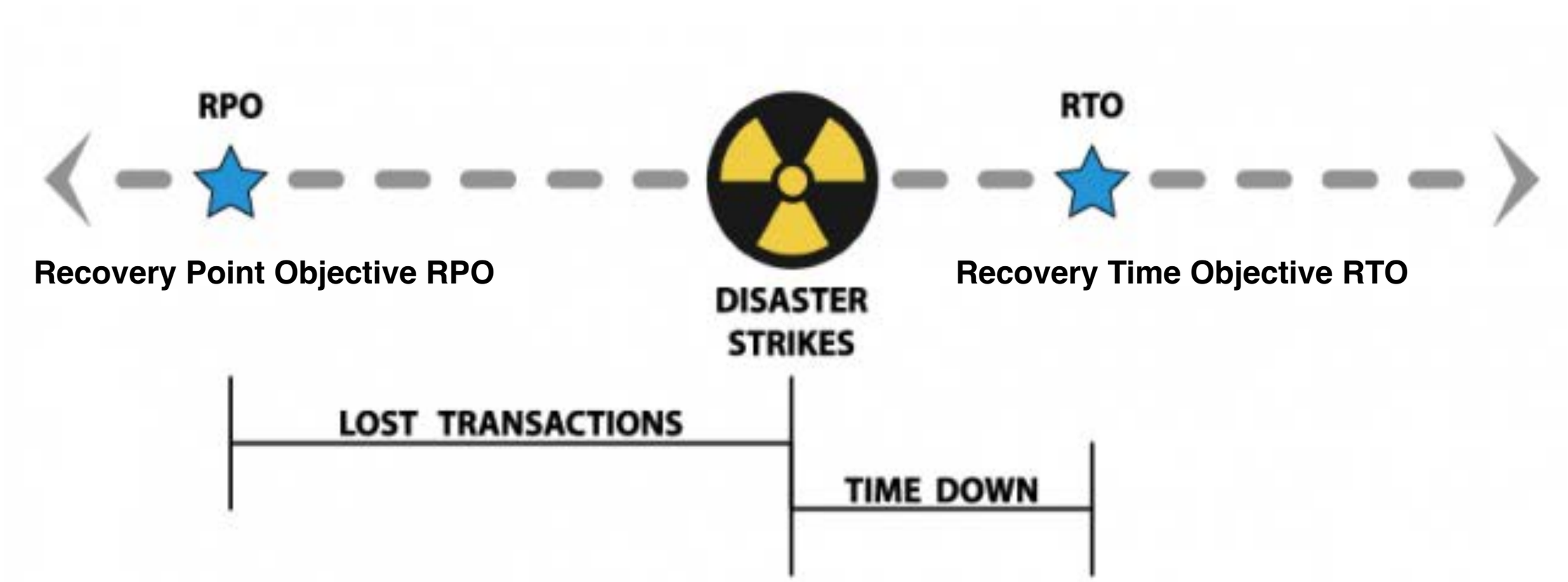
1: Application invokes operation on hosted service. The request fails, and the service host responds with HTTP response code 500 (internal server error).
2: Application waits for a short interval and tries again. The request still fails with HTTP response code 500.
3: Application waits for a longer interval and tries again. The request succeeds with HTTP response code 200 (OK).

https://docs.microsoft.com/en-us/azure/architecture/patterns/retry



Recovery Point Objective RPO
Recovery Time Objective RTO
WRT: Work recovery time
MTDT: Maximum Tolerable Down Time = RTO + WRT

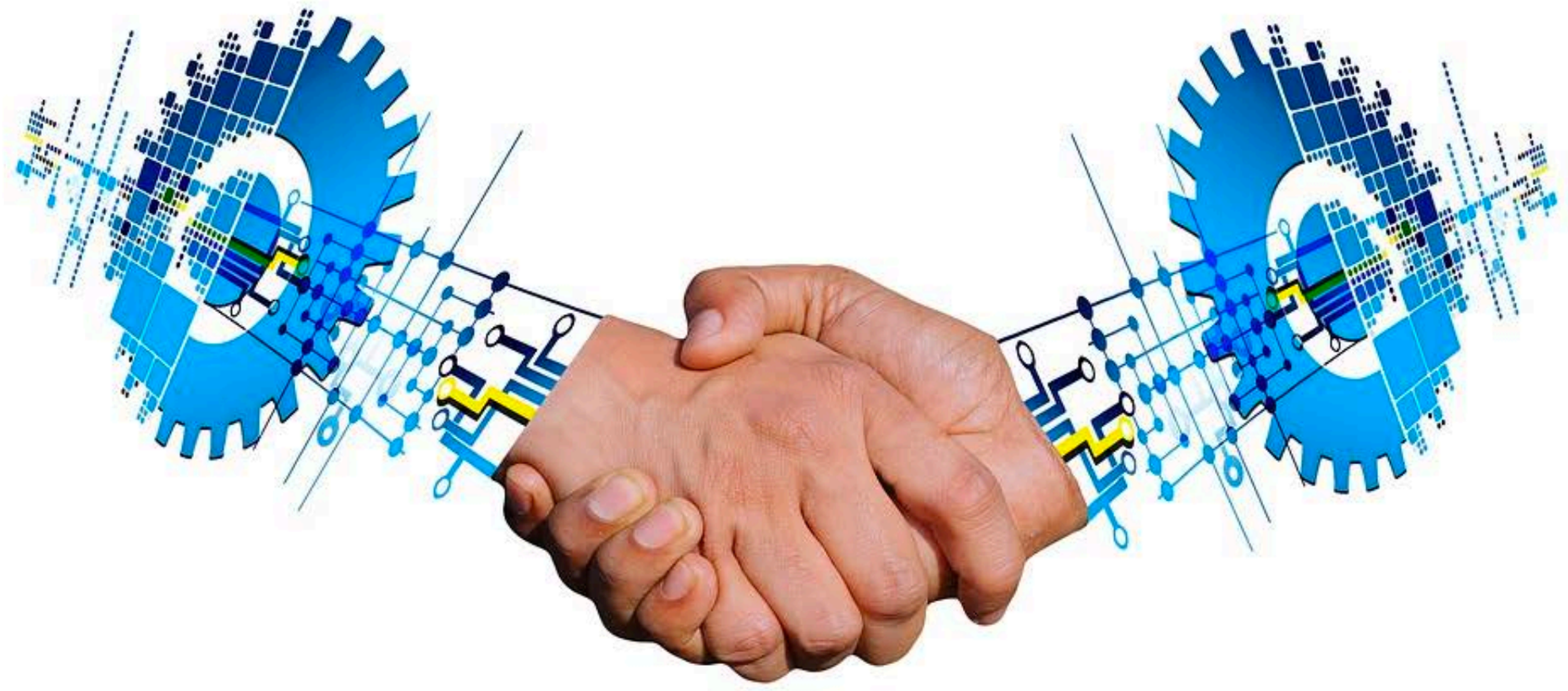
Self Healing



Recovery Point Objective RPO
Recovery Time Objective RTO
WRT: Work recovery time
MTDT: Maximum Tolerable Down Time = RTO + WRT

2 9s (99%) Scenario

3 Days 15 hours per year



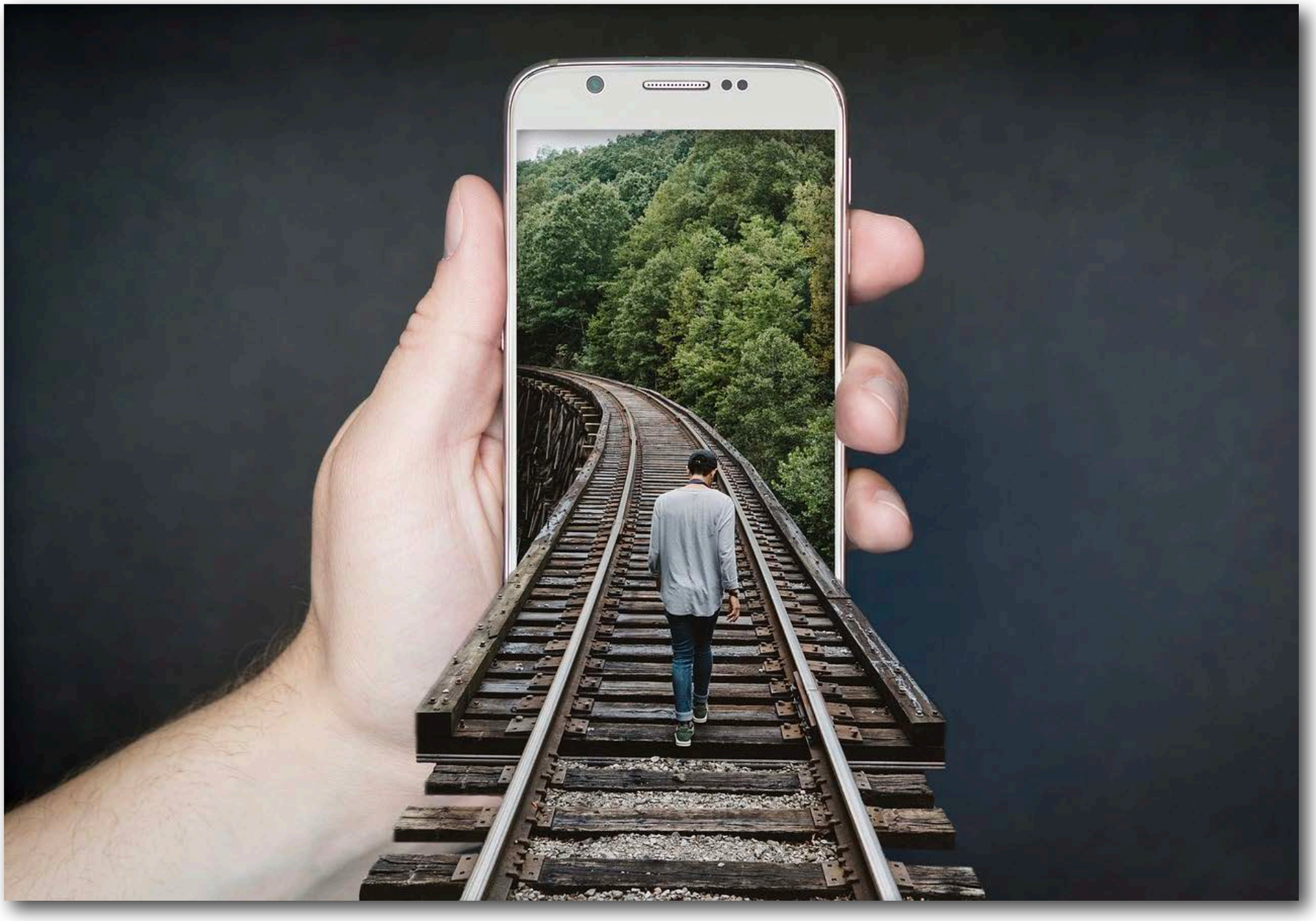
Extract, Transform, Load, Batch Processing

2 9s (99%) Scenario

Topic	Implementation
Monitor resources	Site health check only; no alerting.
Adapt to changes in demand	Vertical scaling via re-deployment.
Implement change	Runbook for deploy and rollback.
Back up data	Runbook for backup and restore.
Architect for resiliency	Complete rebuild; restore from backup.
Test resiliency	Complete rebuild; restore from backup.
Plan for disaster recovery (DR)	Encrypted backups, restore to different Availability Zone if needed.

https://docs.aws.amazon.com/wellarchitected/latest/reliability-pillar/s-99-scenario.html

3 9s (99.9%) Scenario



Project Tracking, Internal tools

8 hrs 45 minutes

3 9s (99.9%) Scenario

Topic	Implementation
Monitor resources	Site health check only; alerts sent when down.
Adapt to changes in demand	ELB for web and automatic scaling application tier; resizing Multi-AZ RDS.
Implement change	Automated deploy in place and runbook for rollback.
Back up data	Automated backups via RDS to meet RPO and runbook for restoring.
Architect for resiliency	Automatic scaling to provide self-healing web and application tier; RDS is Multi-AZ.
Test resiliency	ELB and application are self-healing; RDS is Multi-AZ; no explicit testing.
Plan for disaster recovery (DR)	Encrypted backups via RDS to same AWS Region.

4 9s (99.99%) Scenario

Online Commerce, Point of Sale

52 minutes

4 9s (99.99%) Scenario

Topic	Implementation
Monitor resources	Health checks at all layers and on KPIs;
Adapt to changes in	ELB for web and automatic scaling application tier; automatic scaling storage and read replicas in multiple zones
Implement change	Automated deploy via canary or blue/green and automated rollback when KPIs or alerts indicate undetected problems
Back up data	Automated backups via RDS to meet RPO and automated restoration that is practiced regularly in a game day.
Architect for resiliency	Implemented fault isolation zones for the application; auto scaling to provide self-healing web and application tier;
Test resiliency	Component and isolation zone fault testing is in pipeline and practiced with operational staff regularly in a game day
Plan for disaster	Encrypted backups via RDS to same AWS Region that is practiced in a game day.

5 9s (99.999%) Scenario

ATM Transactions, Telecommunication

5 minutes

5 9s (99.999%) Scenario

Topic	Implementation
Monitor resources	Health checks at all layers, including DNS health at AWS Region level, and on KPIs; alerts
Adapt to changes in	ELB for web and automatic scaling application tier; automatic scaling storage and read replicas. Multi Region
Implement change	Automated deploy via canary or blue/green and automated rollback when KPIs or alerts indicate undetected problems
Back up data	Automated backups in each AWS Region via RDS to meet RPO and automated restoration that is practiced regularly
Architect for resiliency	Implemented fault isolation zones for the application; auto scaling to provide self-healing web and application tier
Test resiliency	Component and isolation zone fault testing is in pipeline and practiced with operational staff regularly in a game day day
Plan for disaster	Active-Active deployed across at least two regions. Infrastructure is fully scaled and statically stable across regions.

Serial Availability

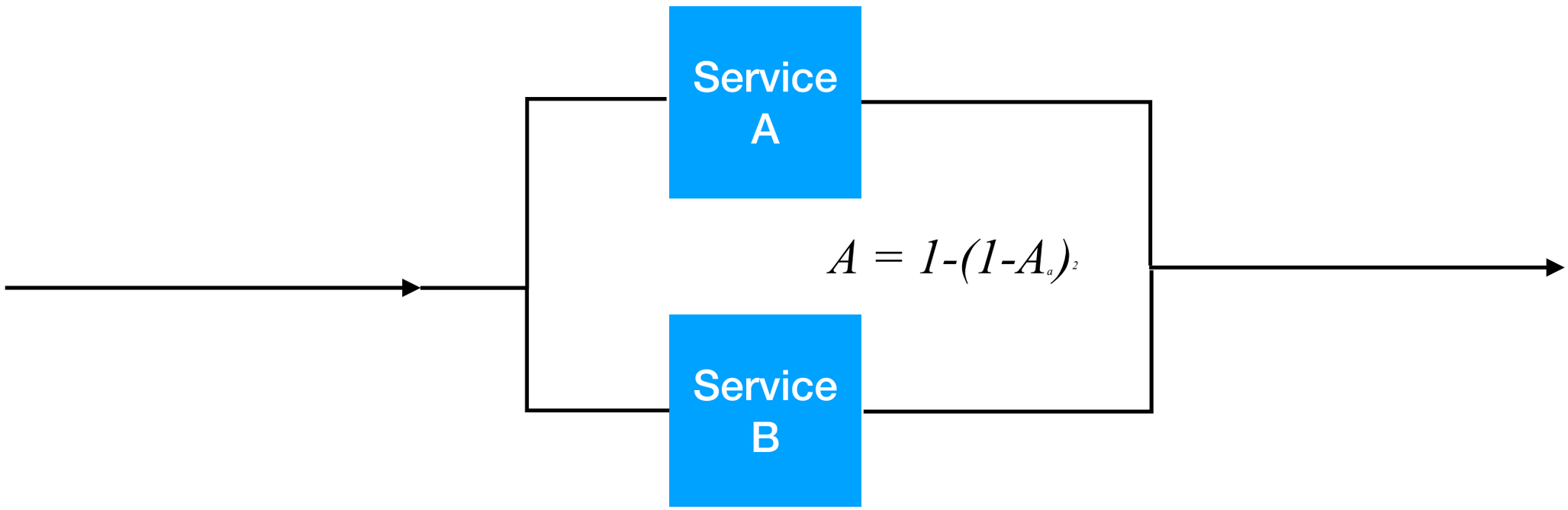
Service	Availability	Downtime
Service A	99%	3.65 days/year
Service B	99.99%	52 minutes/year
A + B	98.99%	3.69 days/year

Calculating availability with hard dependencies

$$Avail_{workload} = Avail_{invok} \times Avail_{dep1} \times Avail_{dep2}$$

$$99.99\% \times 99.99\% \times 99.99\% = 99.97\%$$

Parallel Availability



Service	Availability	Downtime
Service A	99%	3.65 days/year
Service B	99.99%	52 minutes/year
A + B	99.9999%	31 seconds year

Calculating availability with redundant components (99.9)

$$\text{Avail}_{\text{workload}} = \textit{Avail}_{\text{MAX}} - ((100\% - \text{Avail}_{\text{dependency}}) \times (100\% - \text{Avail}_{\text{dependency}}))$$

$$100\% - (0.1\% \times 0.1\%) = 99.9999\%$$

Calculating dependency availability

$$Avail_{EST} = \frac{MTBF}{MTBF + MTTR}$$

Mean Time Between Failure (MTBF)

Mean Time to Recover (MTTR)

https://www.delaat.net/rp/2013-2014/p17/report.pdf

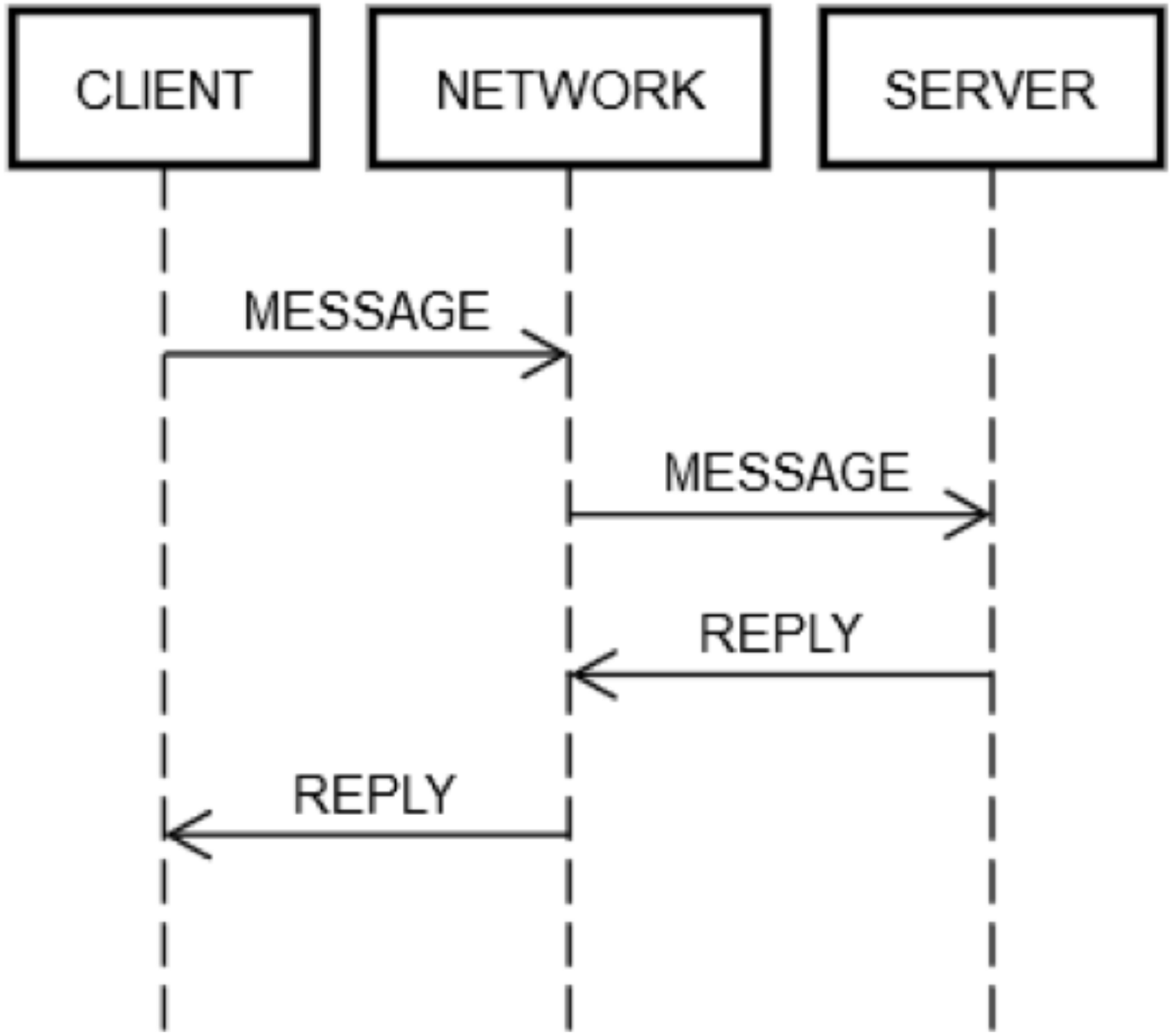
https://docs.aws.amazon.com/wellarchitected/latest/reliability-pillar/appendix-a-designed-for-availability-for-select-aws-services.html

Availability estimate

- MTBF is 150 days and the MTTR is 1 hour

- Availability estimate is 99.97%

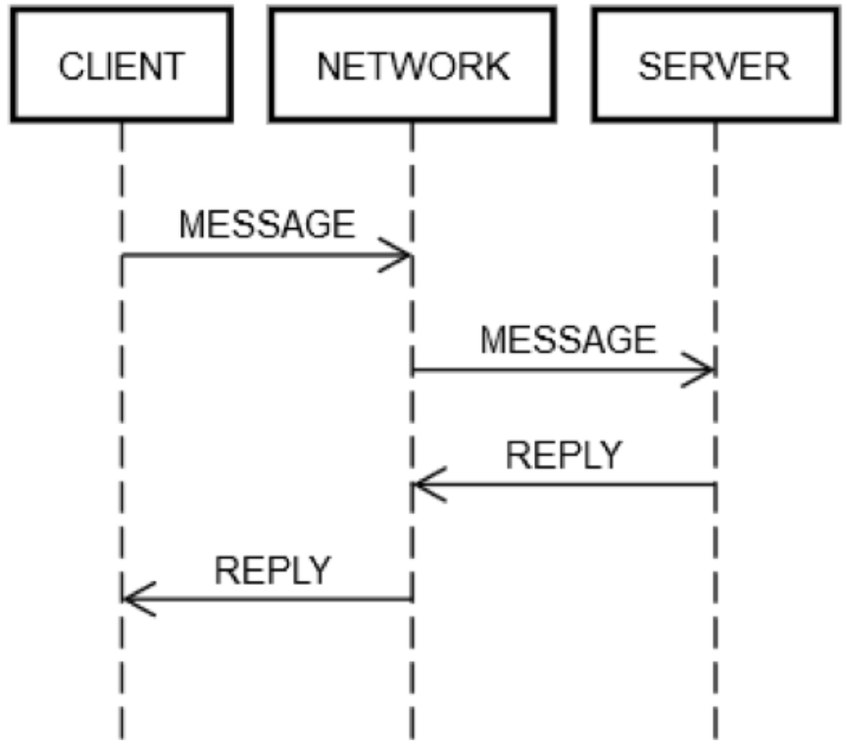
Request/reply messaging across a network



https://aws.amazon.com/builders-library/challenges-with-distributed-systems/?ref=wellarchitected

Request/reply messaging across a network

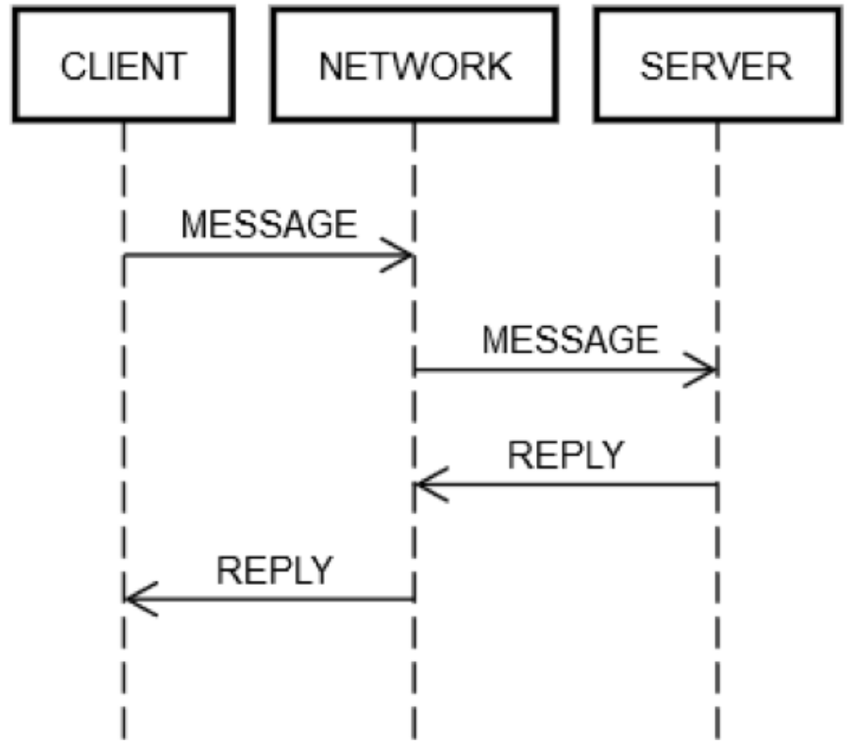
- 1. POST REQUEST: CLIENT puts request MESSAGE onto NETWORK.
- 2. DELIVER REQUEST: NETWORK delivers MESSAGE to SERVER.
- 3. VALIDATE REQUEST: SERVER validates MESSAGE.
- 4. UPDATE SERVER STATE: SERVER updates its state, if necessary, based on MESSAGE.
- 5. POST REPLY: SERVER puts reply REPLY onto NETWORK.
- 6. DELIVER REPLY: NETWORK delivers REPLY to CLIENT.
- 7. VALIDATE REPLY: CLIENT validates REPLY.
- 8. UPDATE CLIENT STATE: CLIENT updates its state, if necessary, based on REPLY.



https://aws.amazon.com/builders-library/challenges-with-distributed-systems/?ref=wellarchitected

Request/reply messaging across a network

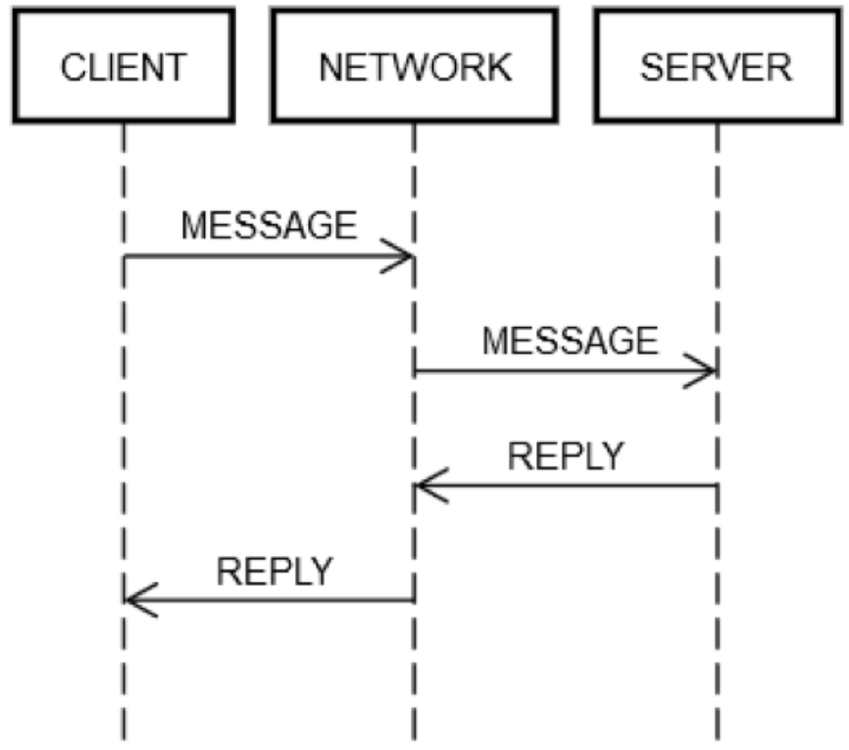
- 1. POST REQUEST fails: Either NETWORK failed to deliver the message (for example, intermediate router crashed at just the wrong moment), or SERVER rejected it explicitly.
- 2. DELIVER REQUEST fails: NETWORK successfully delivers MESSAGE to SERVER, but SERVER crashes right after it receives MESSAGE.
- 3. VALIDATE REQUEST fails: SERVER decides that MESSAGE is invalid. The cause can be almost anything. For example, corrupted packets, incompatible software versions, or bugs on either client or server.
- 4. UPDATE SERVER STATE fails: SERVER tries to update its state, but it doesn’t work.
- 5. POST REPLY fails: Regardless of whether it was trying to reply with success or failure, SERVER could fail to post the reply. For example, its network card might fry just at the wrong moment.
- 6. DELIVER REPLY fails: NETWORK could fail to deliver REPLY to CLIENT as outlined earlier, even though NETWORK was working in an earlier step.
- 7. VALIDATE REPLY fails: CLIENT decides that REPLY is invalid.
- 8. UPDATE CLIENT STATE fails: CLIENT could receive message REPLY but fail to update its own state, fail to understand the message (due to being incompatible), or fail for some other reason.



https://aws.amazon.com/builders-library/challenges-with-distributed-systems/?ref=wellarchitected

Request/reply messaging across a network

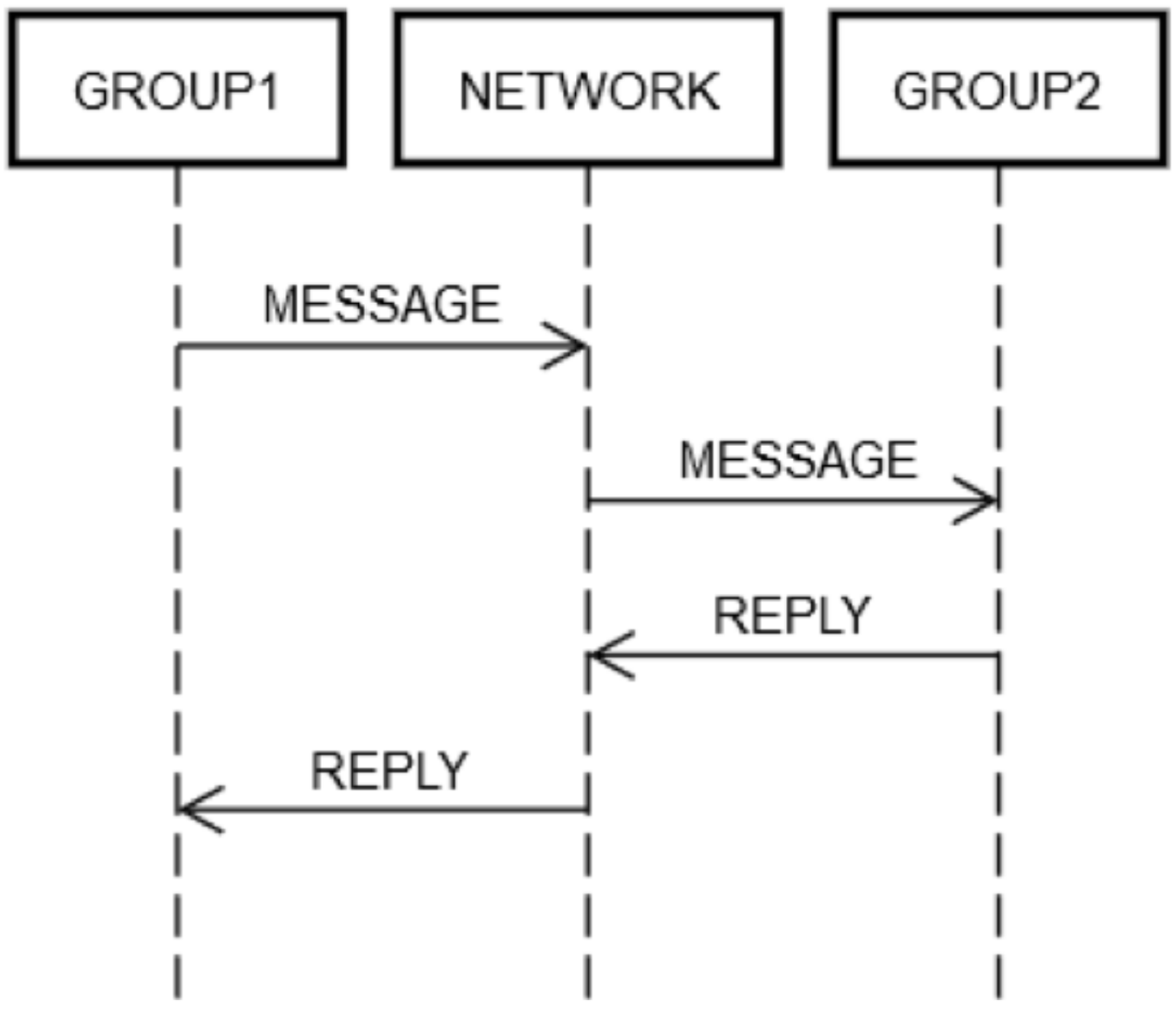
```
(error, reply) = network.send(remote, actionData)
switch error
  case POST_FAILED:
    // handle case where you know server didn't get it
  case RETRYABLE:
    // handle case where server got it but reported transient
failure
  case FATAL:
    // handle case where server got it and definitely doesn't
like it
  case UNKNOWN: // i.e., time out
    // handle case where the *only* thing you know is that the
server received
    // the message; it may have been trying to report SUCCESS,
FATAL, or RETRYABLE
  case SUCCESS:
    if validate(reply)
      // do something with reply object
    else
      // handle case where reply is corrupt/incompatible
```

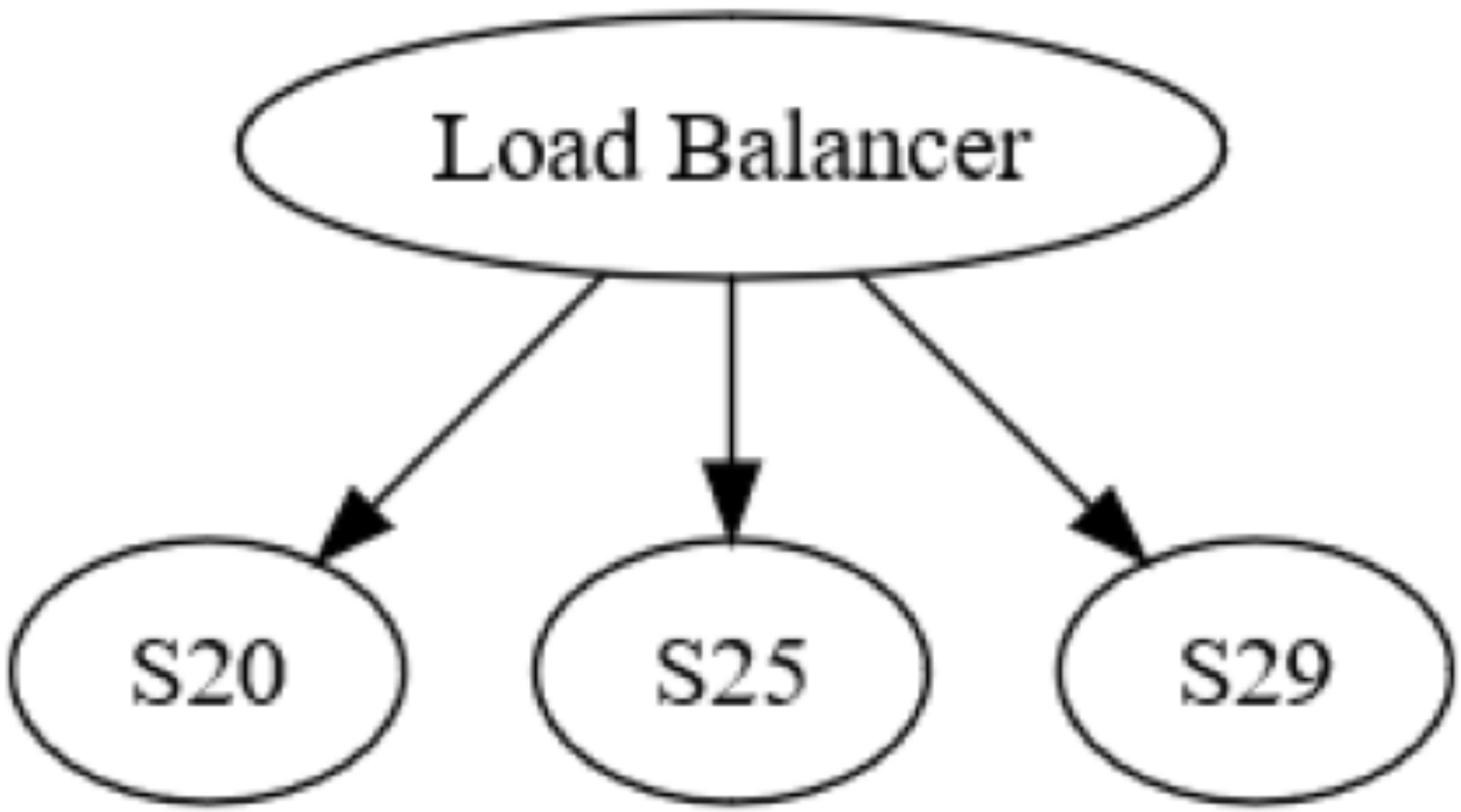


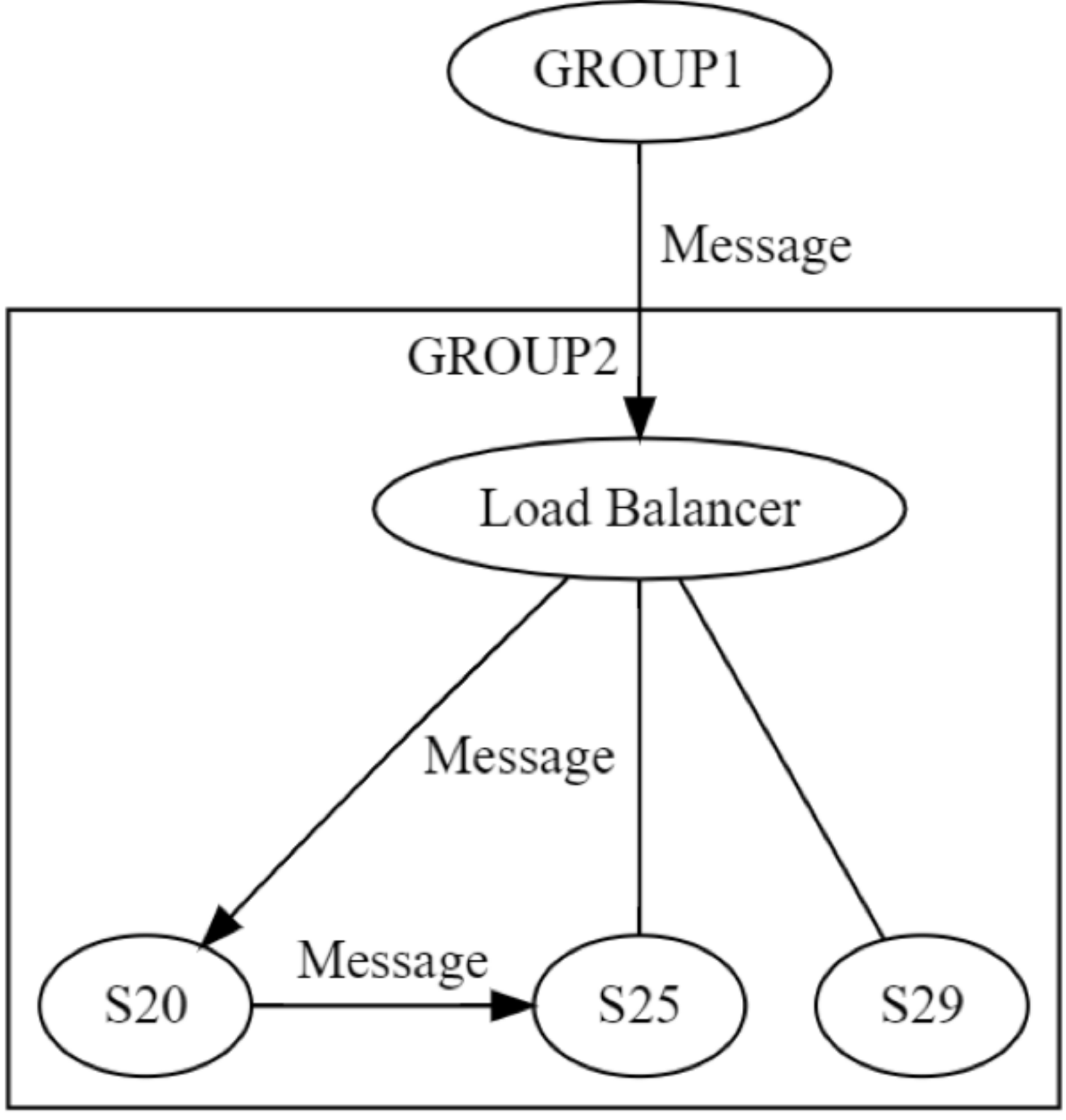
https://aws.amazon.com/builders-library/challenges-with-distributed-systems/?ref=wellarchitected

Herds of hard real-time distributed systems

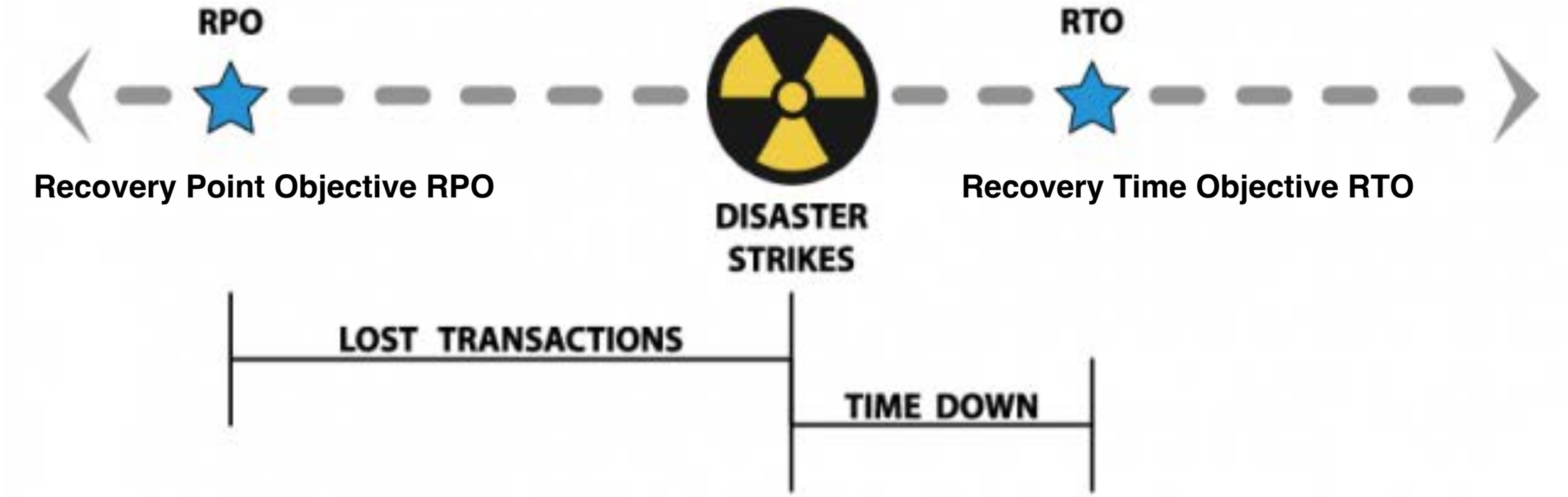
- 1. Individual machines
- 2. Groups of machines
- 3. Groups of groups of machines


GROUP1
NETWORK
GROUP2
MESSAGE
MESSAGE
REPLY
REPLY


GROUP2
Load Balancer
S20
S25
S29

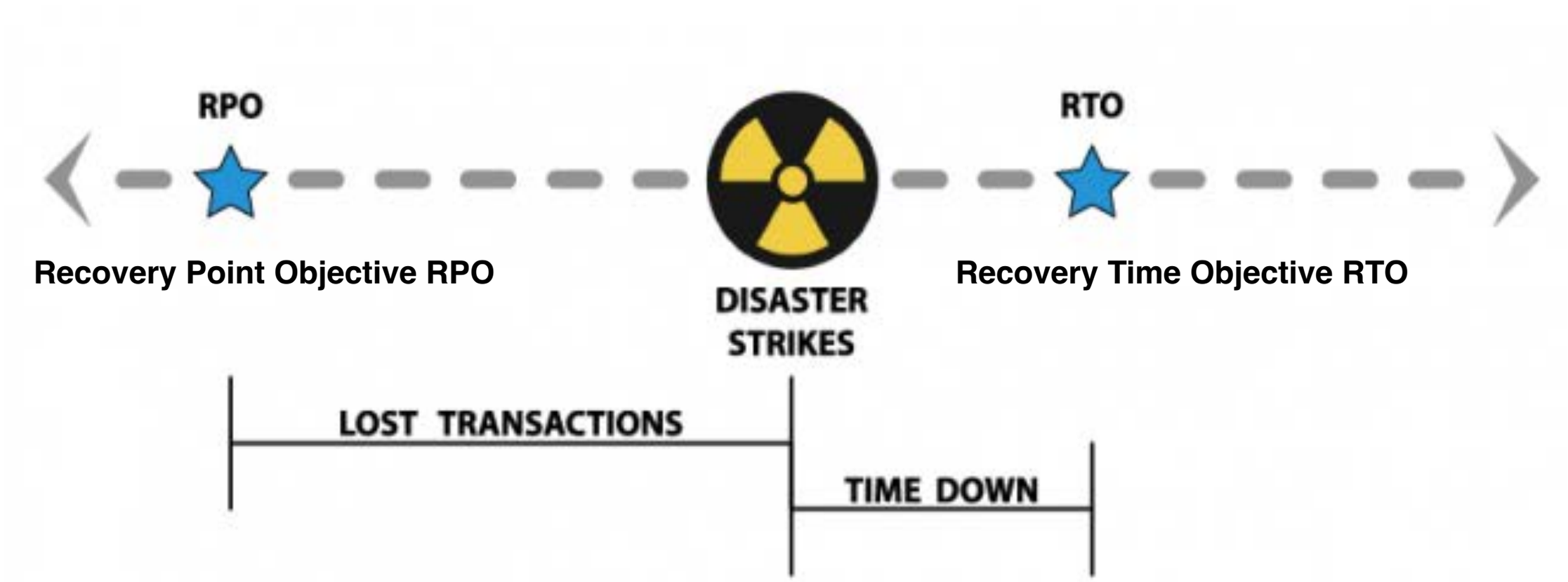


- A test for all eight ways GROUP1 to GROUP2 group-level messaging can fail.
- A test for all eight ways S20 to S25 server-level messaging can fail.



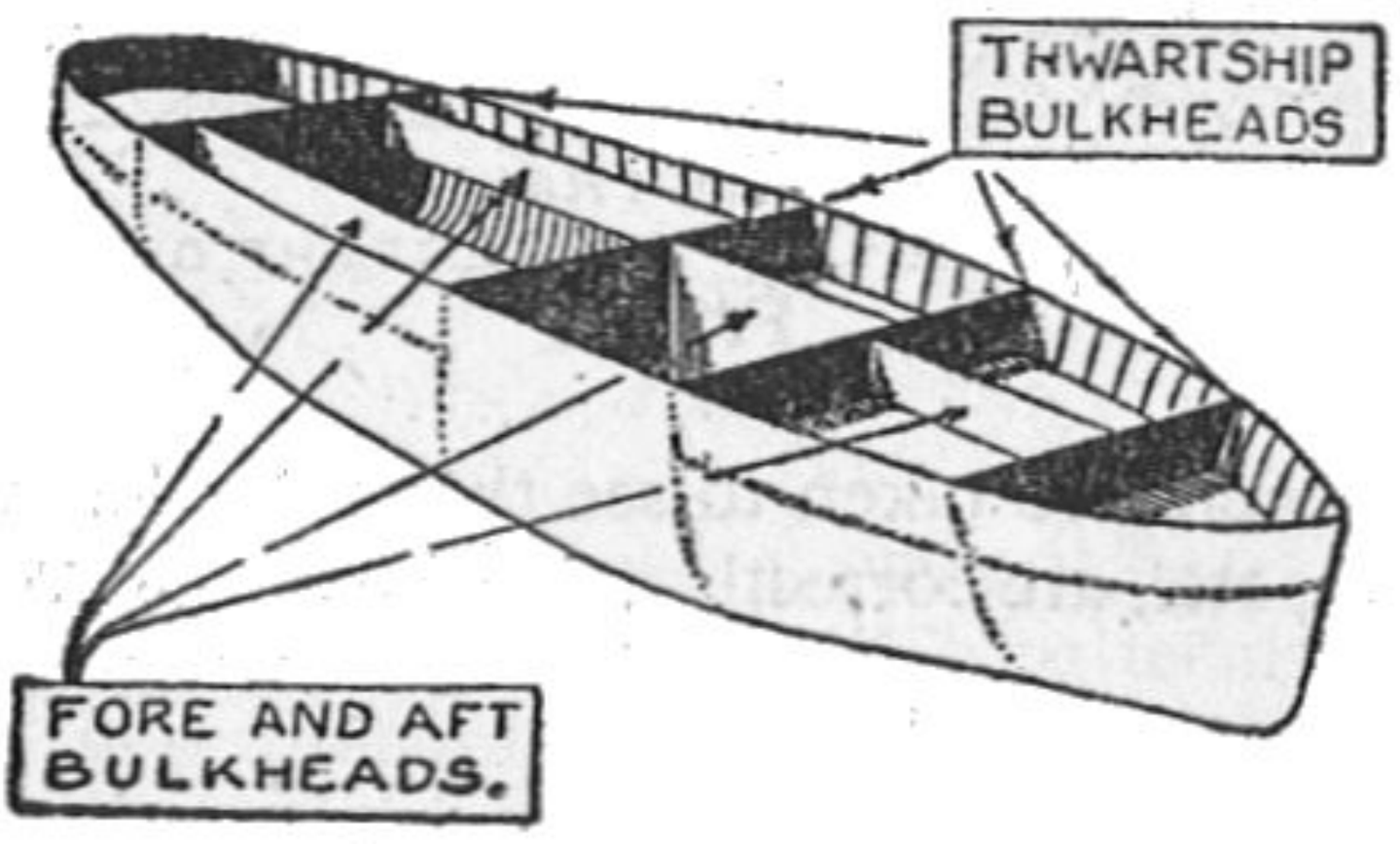
Recovery Point Objective RPO
Recovery Time Objective RTO
WRT: Work recovery time
MTDT: Maximum Tolerable Down Time = RTO + WRT

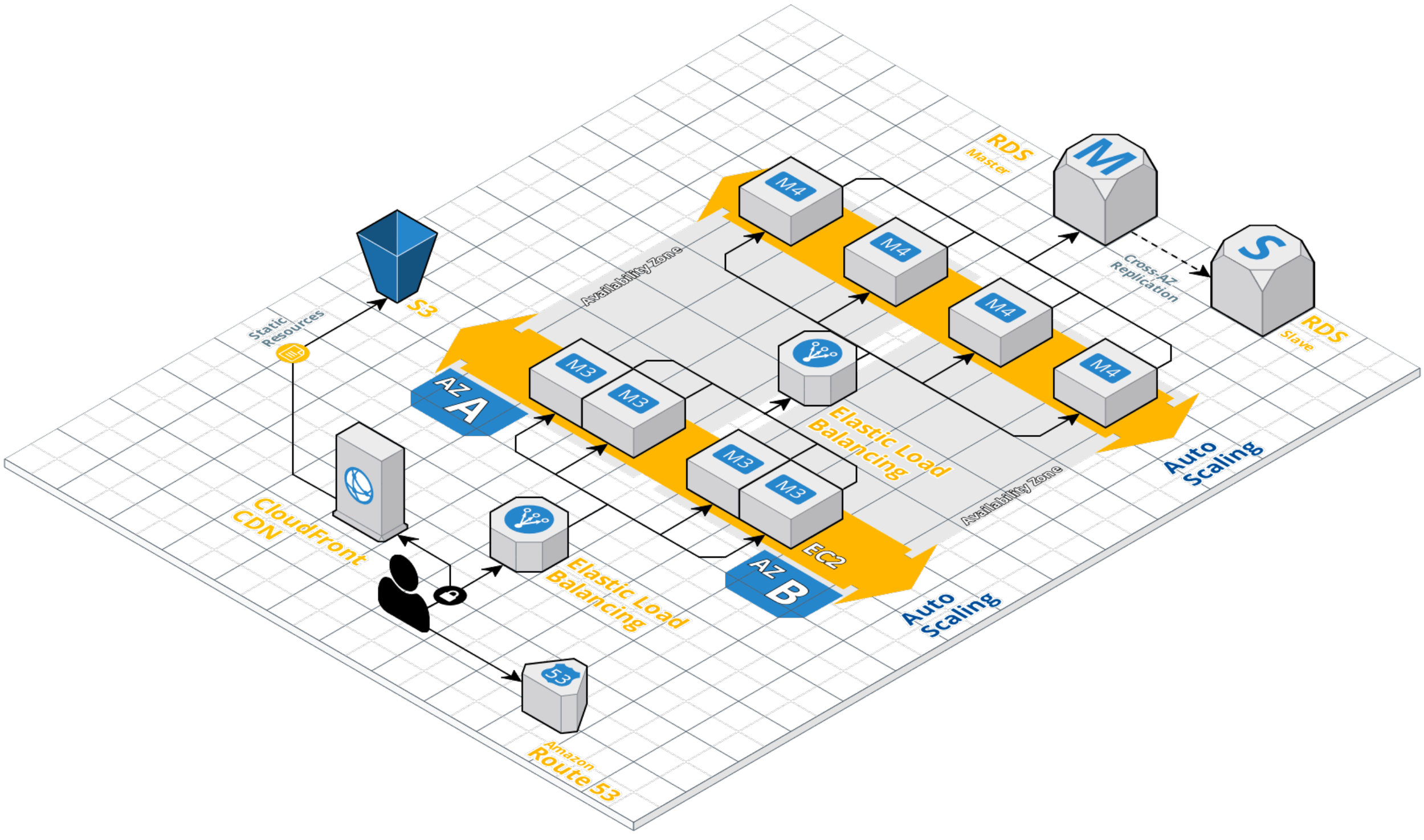
Self Healing

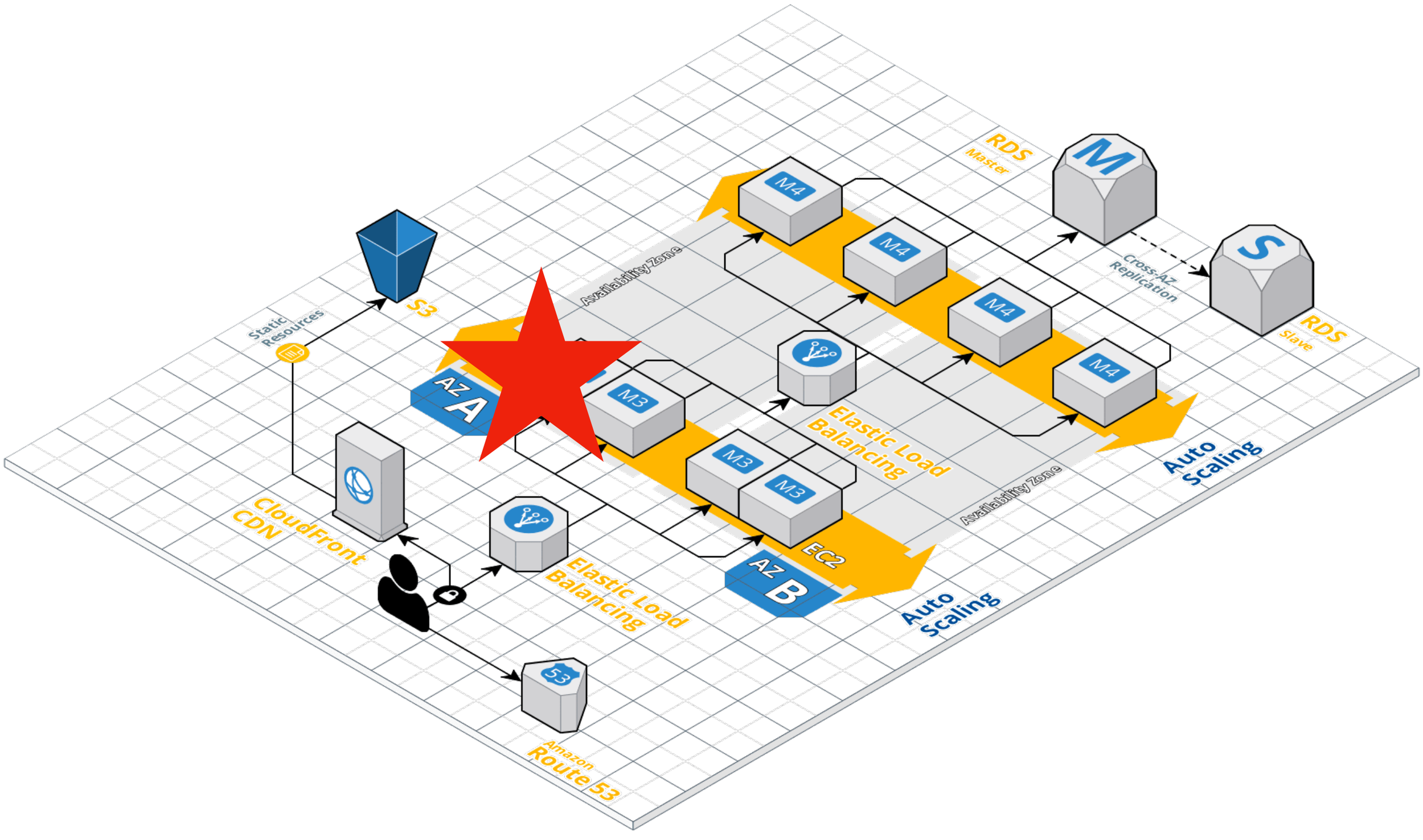


Recovery Point Objective RPO
Recovery Time Objective RTO
WRT: Work recovery time
MTDT: Maximum Tolerable Down Time = RTO + WRT

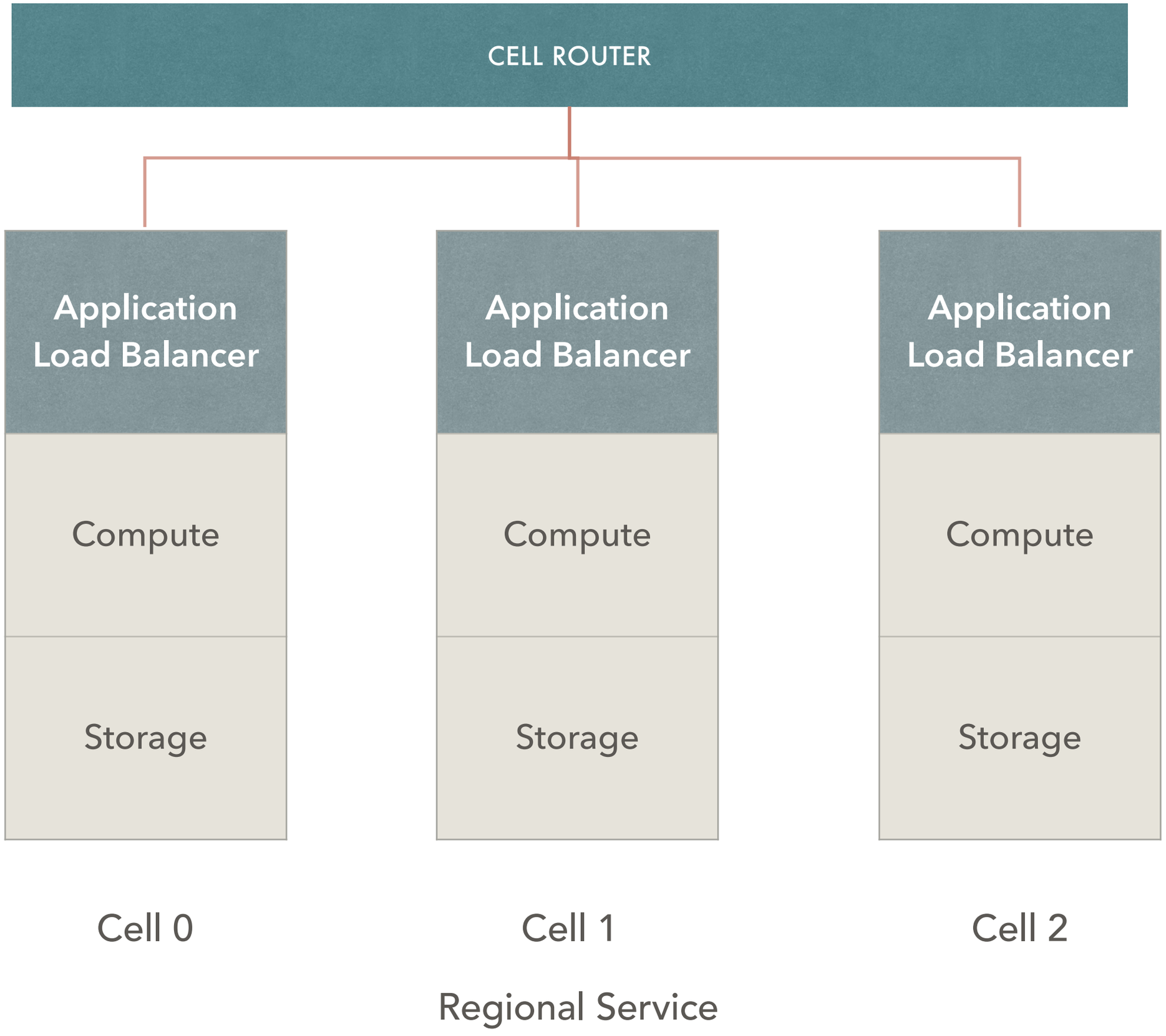
Automatically recover from failure - cell based



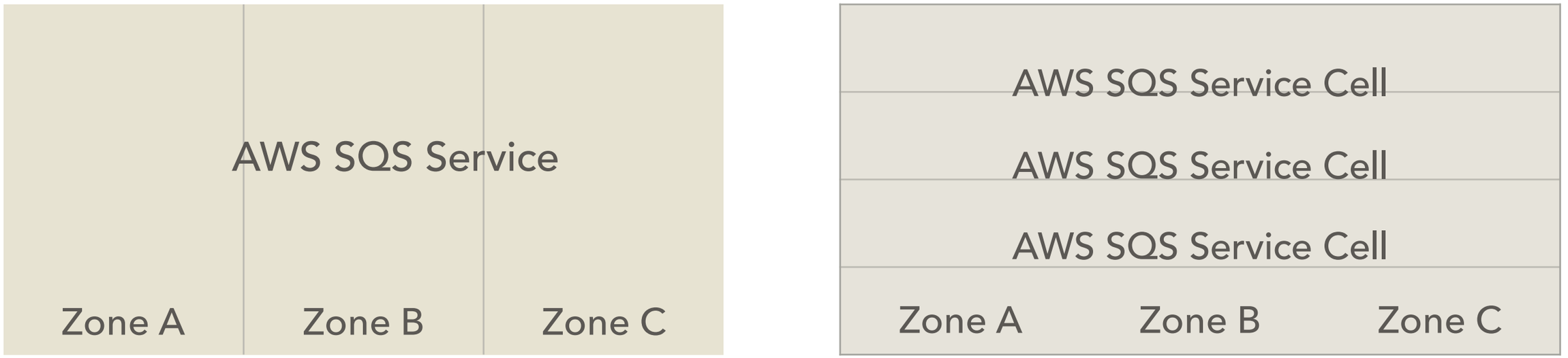

RDS
Master
M
S
RDS
Slave
Cross-AZ
Replication
S3
Static
Resources
Availability Zone
M4
M4
M4
M4
M3
M3
M3
M3
AZ A
AZ B
EC2
Elastic Load
Balancing
Auto
Scaling
Availability Zone
CloudFront
CDN
Elastic Load
Balancing
Amazon
Route 53


RDS
Master
M
M4
M4
M4
M4
S
Cross-AZ
Replication
RDS
Slave
S3
Static
Resources
Availability Zone
AZ
A
M3
M3
M3
Elastic Load
Balancing
Availability Zone
Auto
Scaling
EC2
AZ
B
Auto
Scaling
CloudFront
CDN
Elastic Load
Balancing
53
Amazon
Route 53

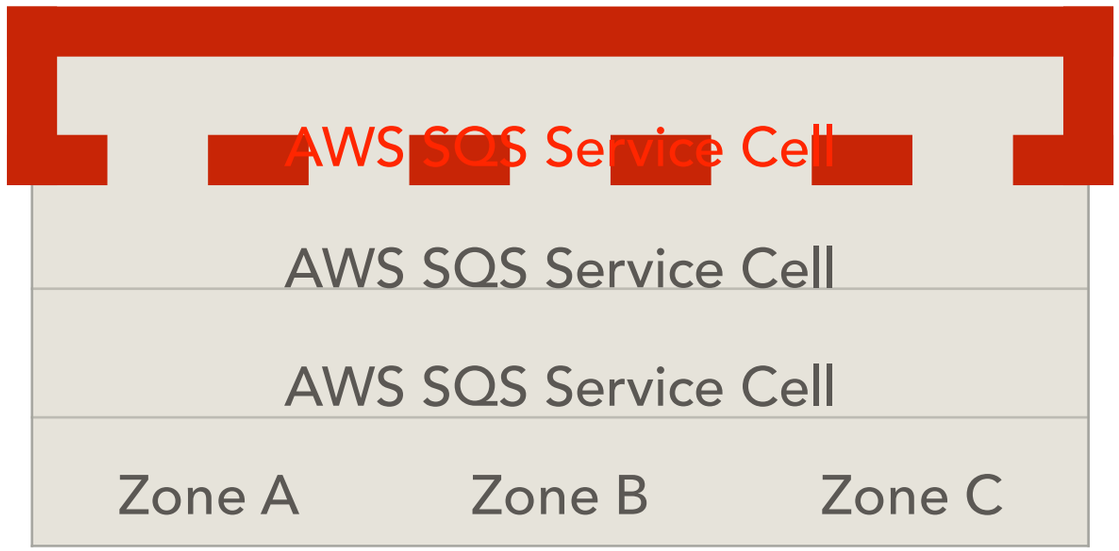
Cell-based architecture



Region service



Region service outage

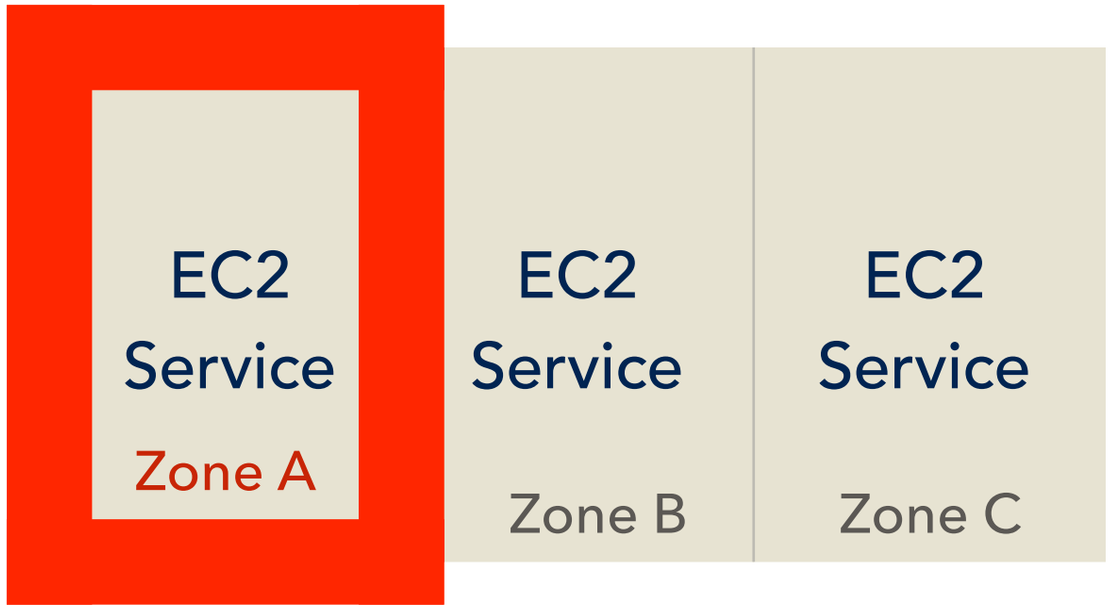


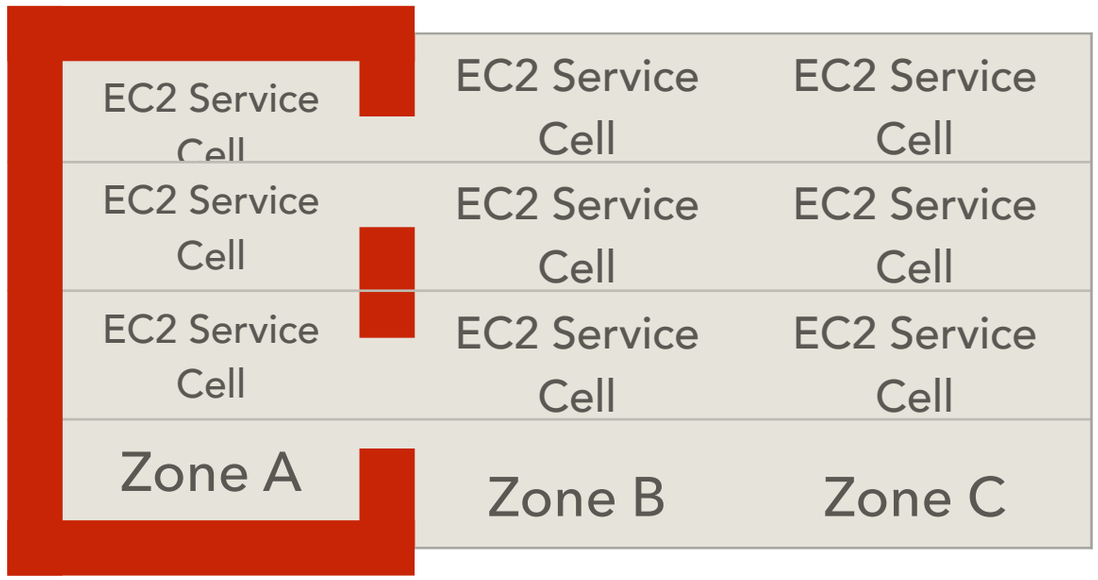
Zonal service

EC2 Service	EC2 Service	EC2 Service
Zone A	Zone B	Zone C

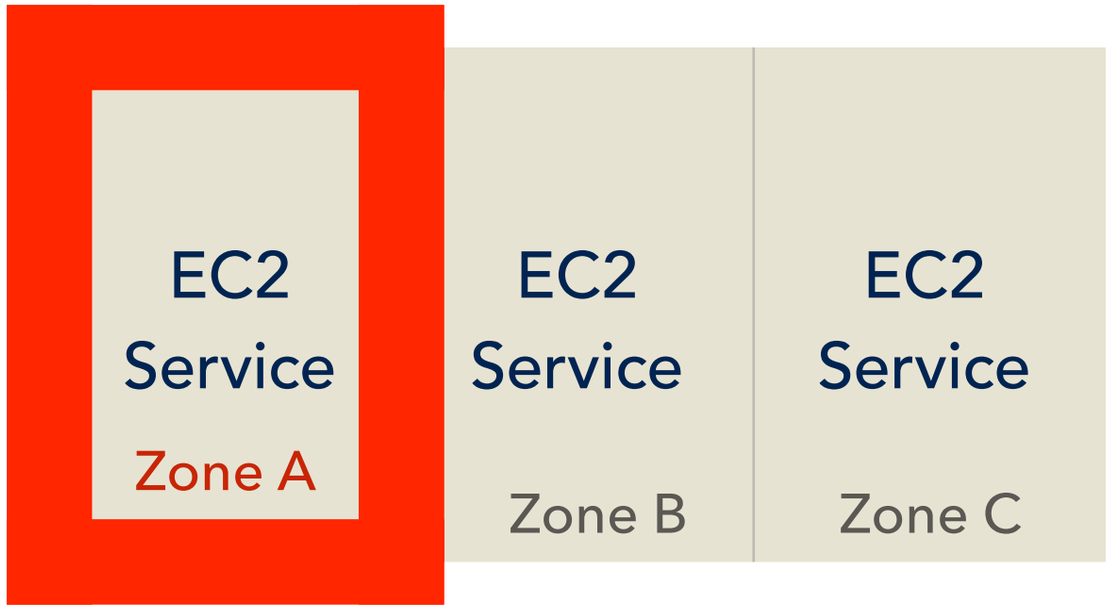
EC2 Service Cell	EC2 Service Cell	EC2 Service Cell
EC2 Service Cell	EC2 Service Cell	EC2 Service Cell
EC2 Service Cell	EC2 Service Cell	EC2 Service Cell
Zone A	Zone B	Zone C

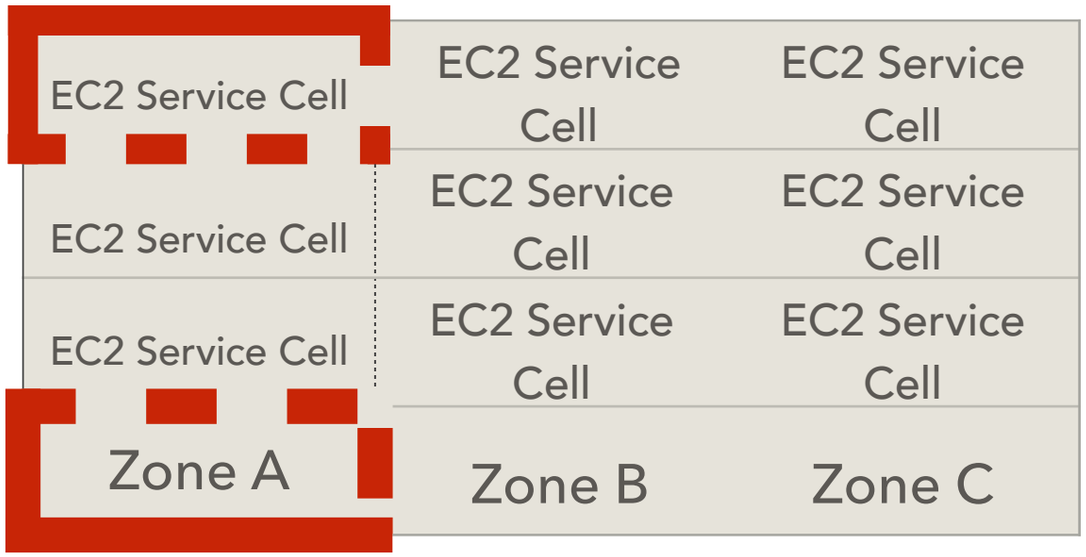
Zonal service outage





Partial Zonal service outage

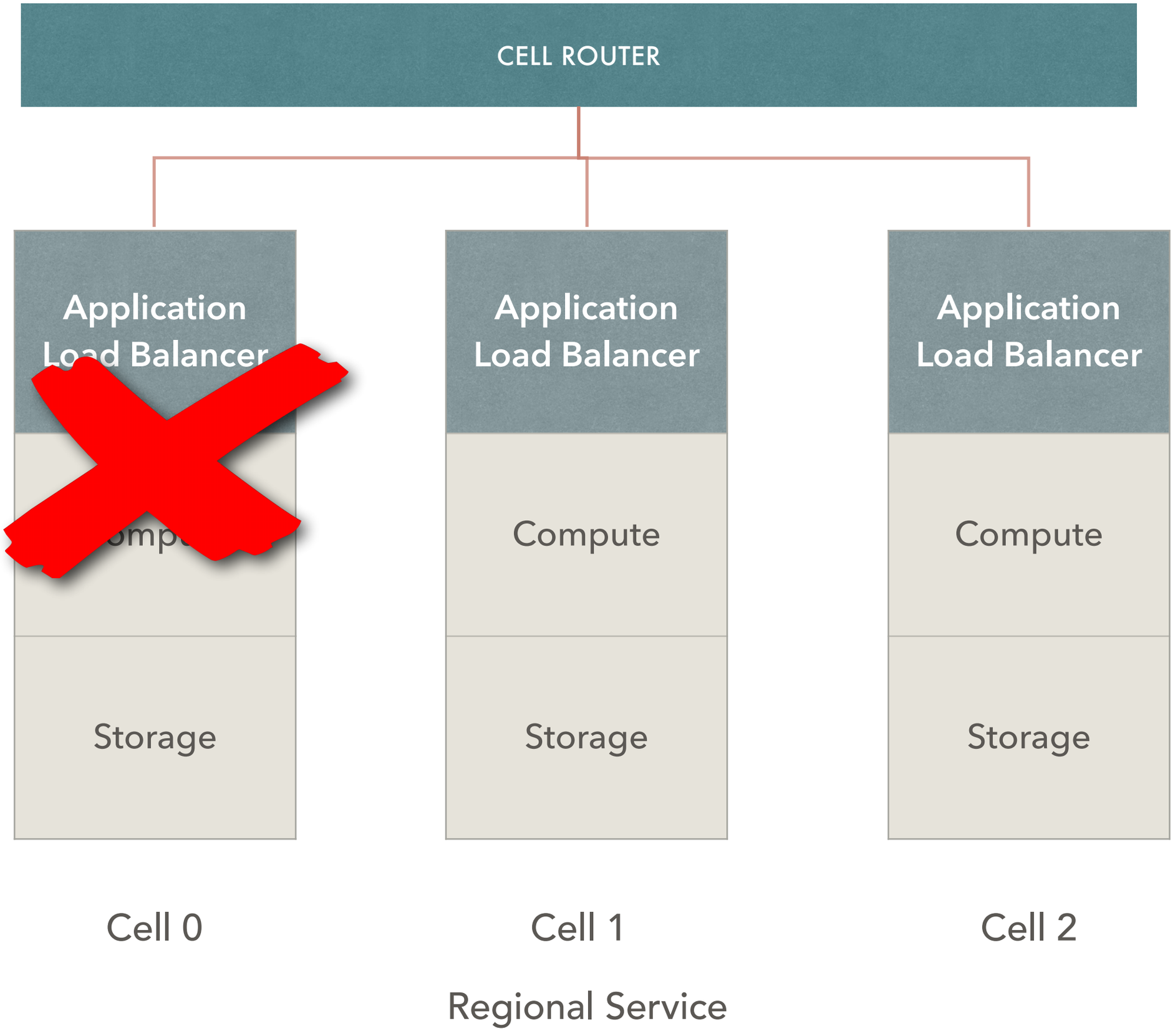


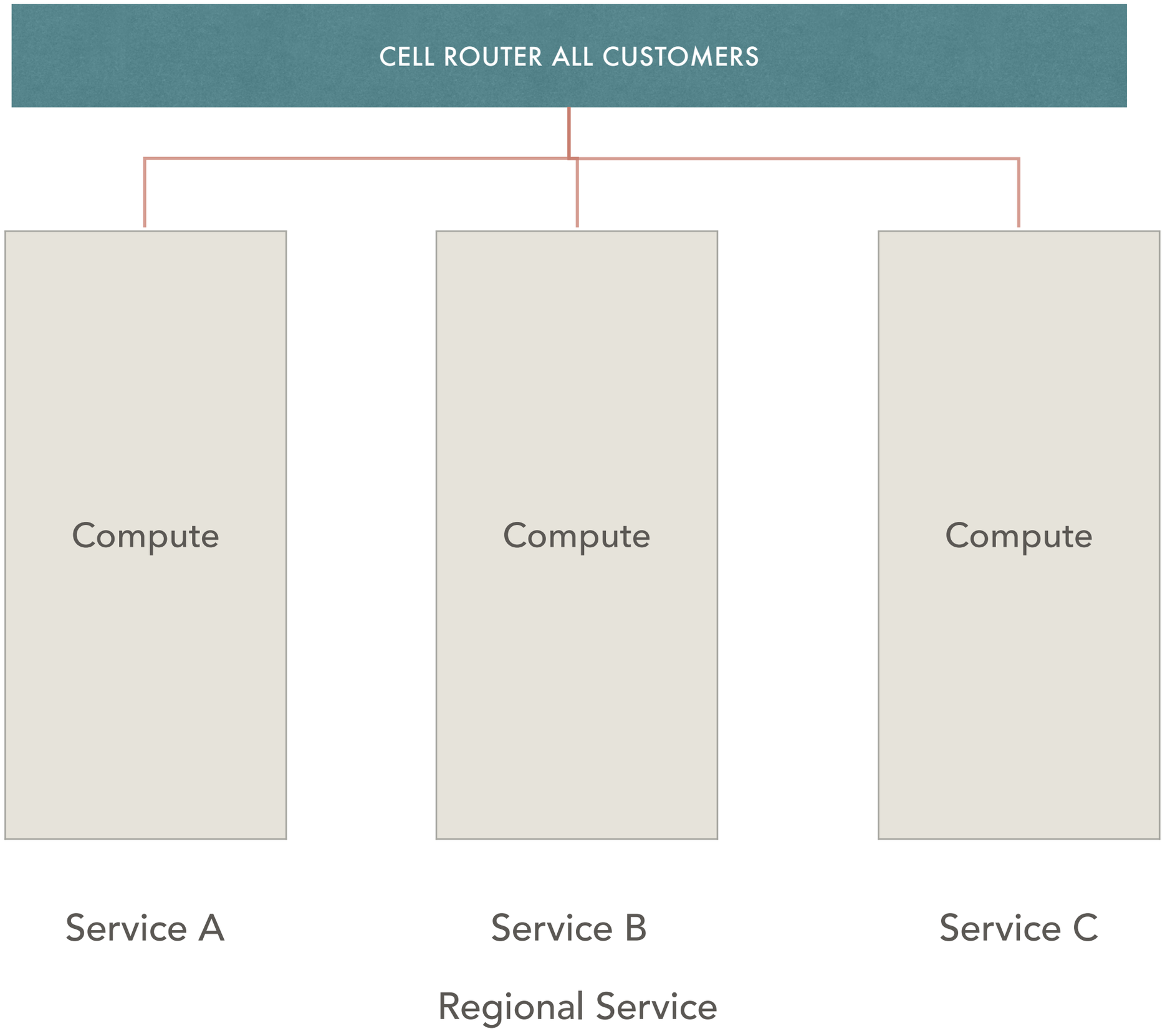


Cell Based Architecture

- Workload isolation
- Failure containment
- Testability
- Manageability

Cell-based architecture




CELL ROUTER ALL CUSTOMERS
Compute
Compute
Compute
Service A
Service B
Regional Service
Service C

CELL ROUTER ALL CUSTOMERS

Compute

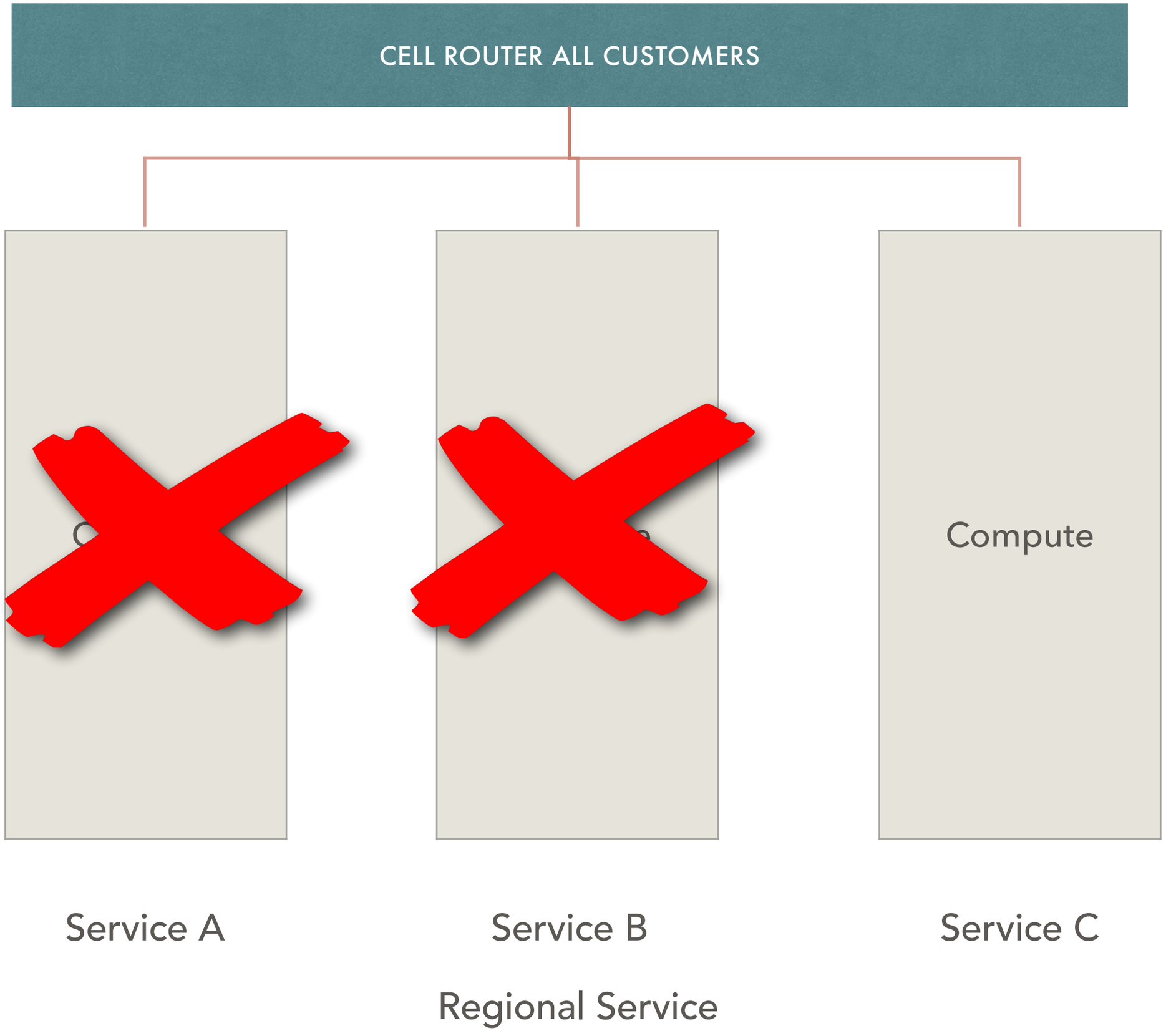
Compute

Service A

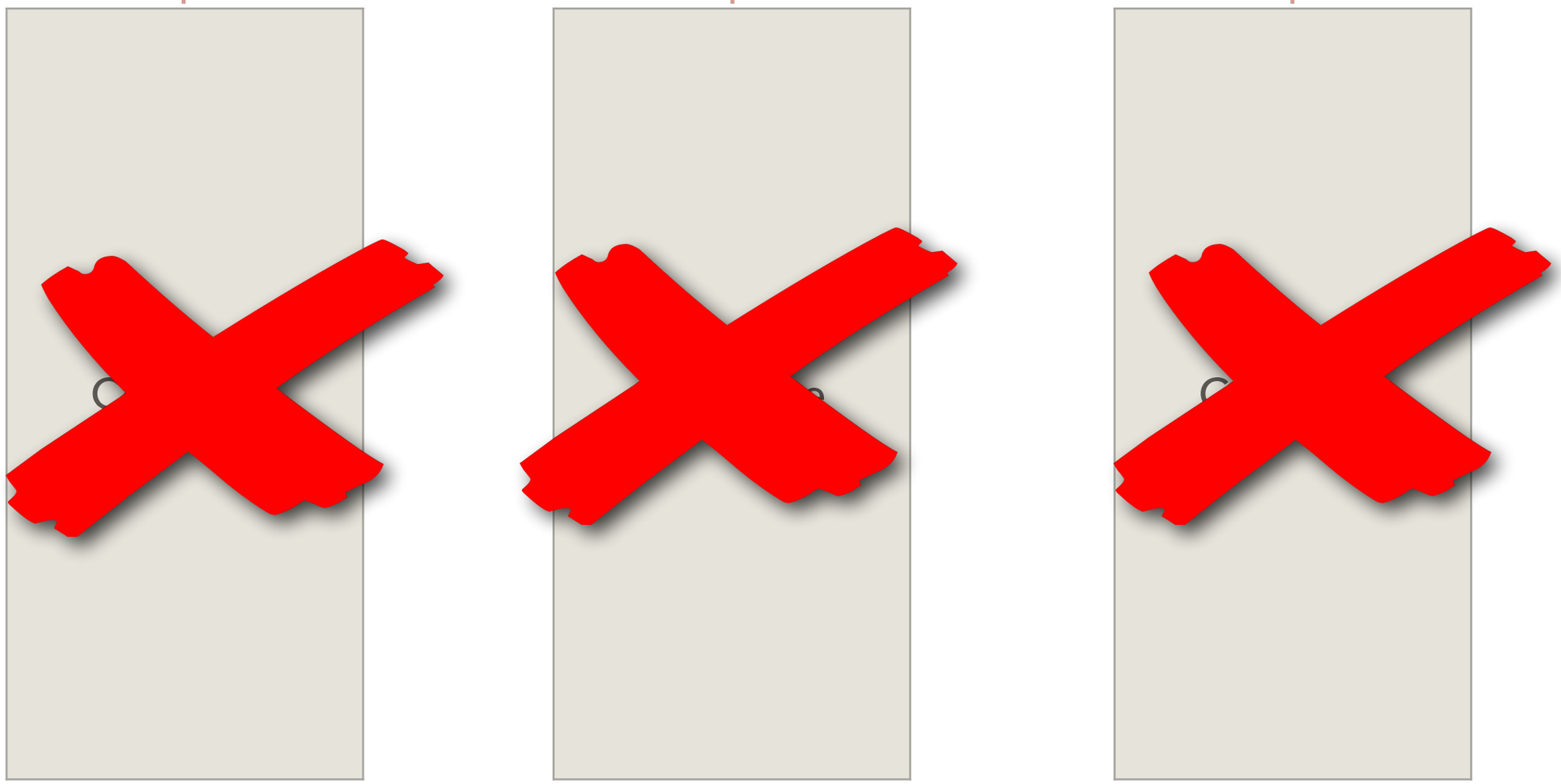
Service B

Regional Service

Service C


CELL ROUTER ALL CUSTOMERS
Compute
Service A
Service B
Regional Service
Service C

CELL ROUTER BY CUSTOMERS

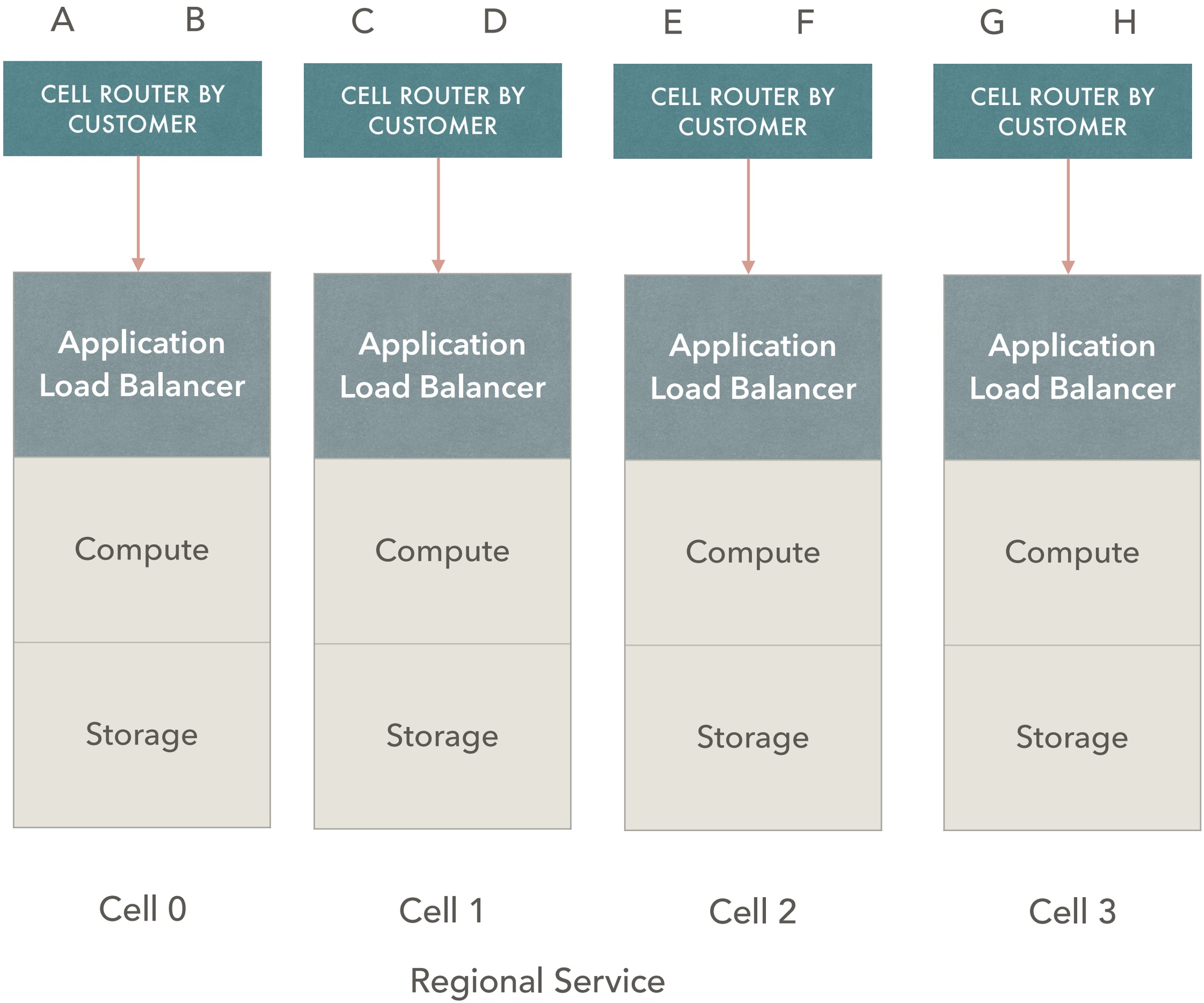


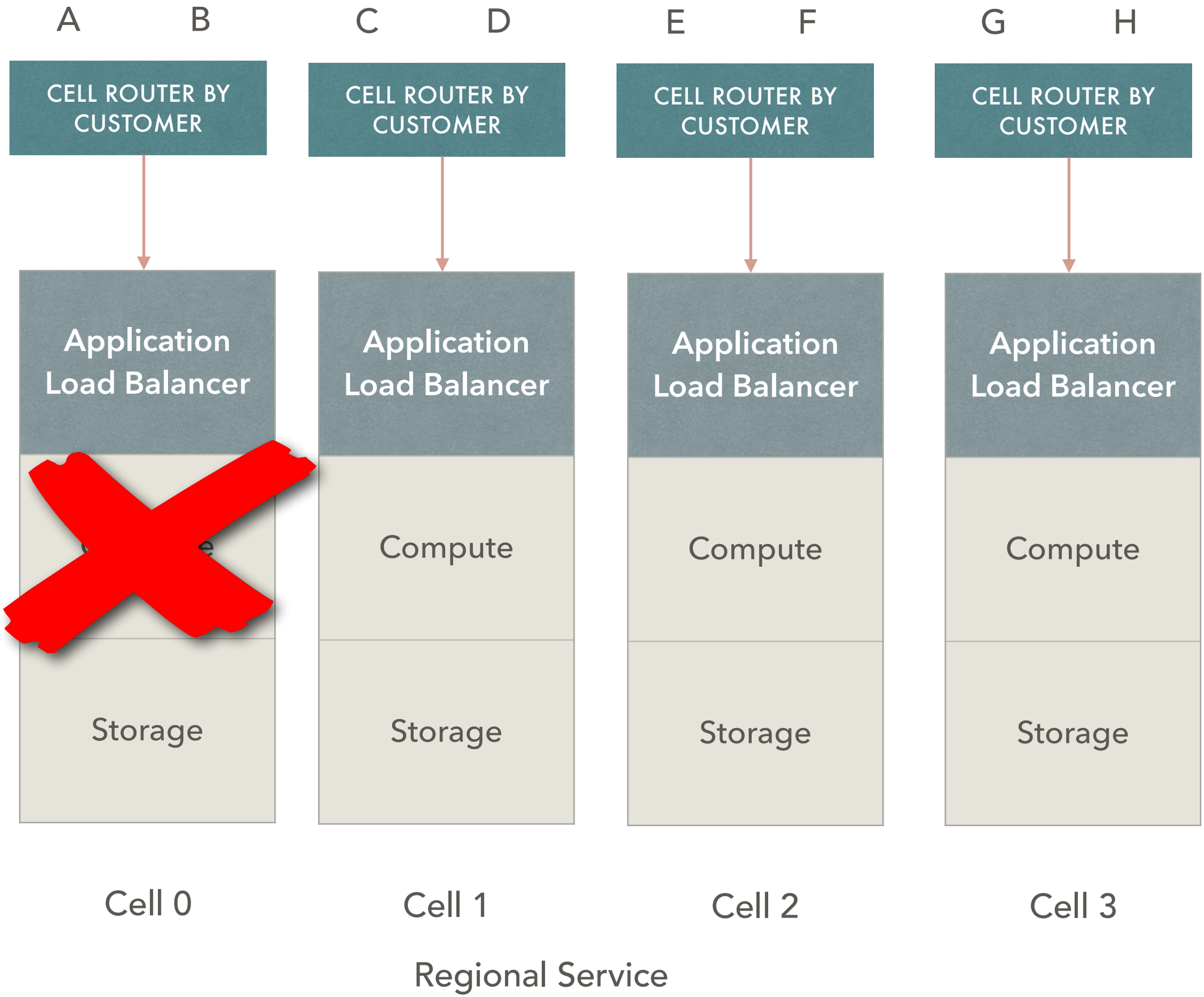
Service A

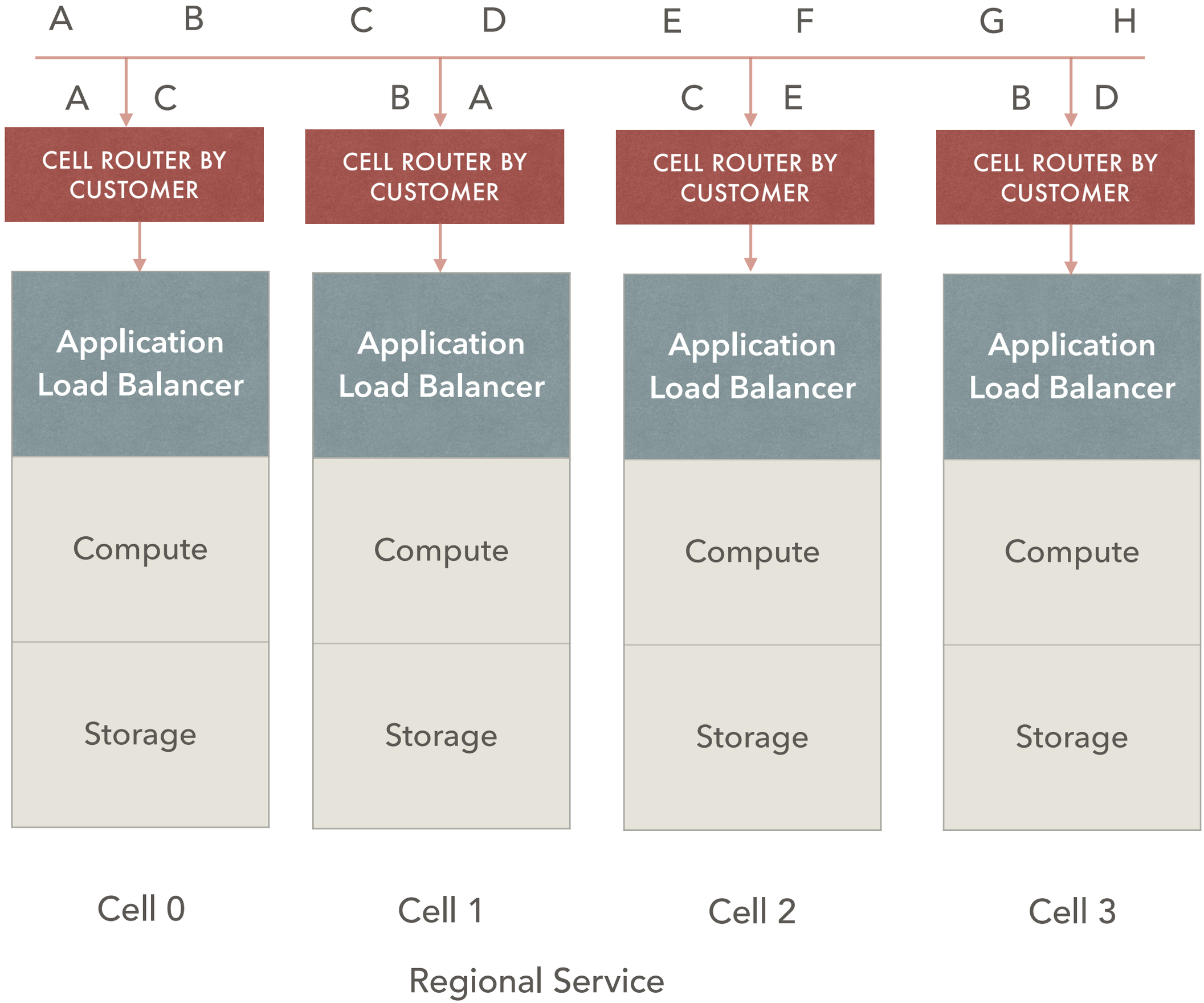
Service B

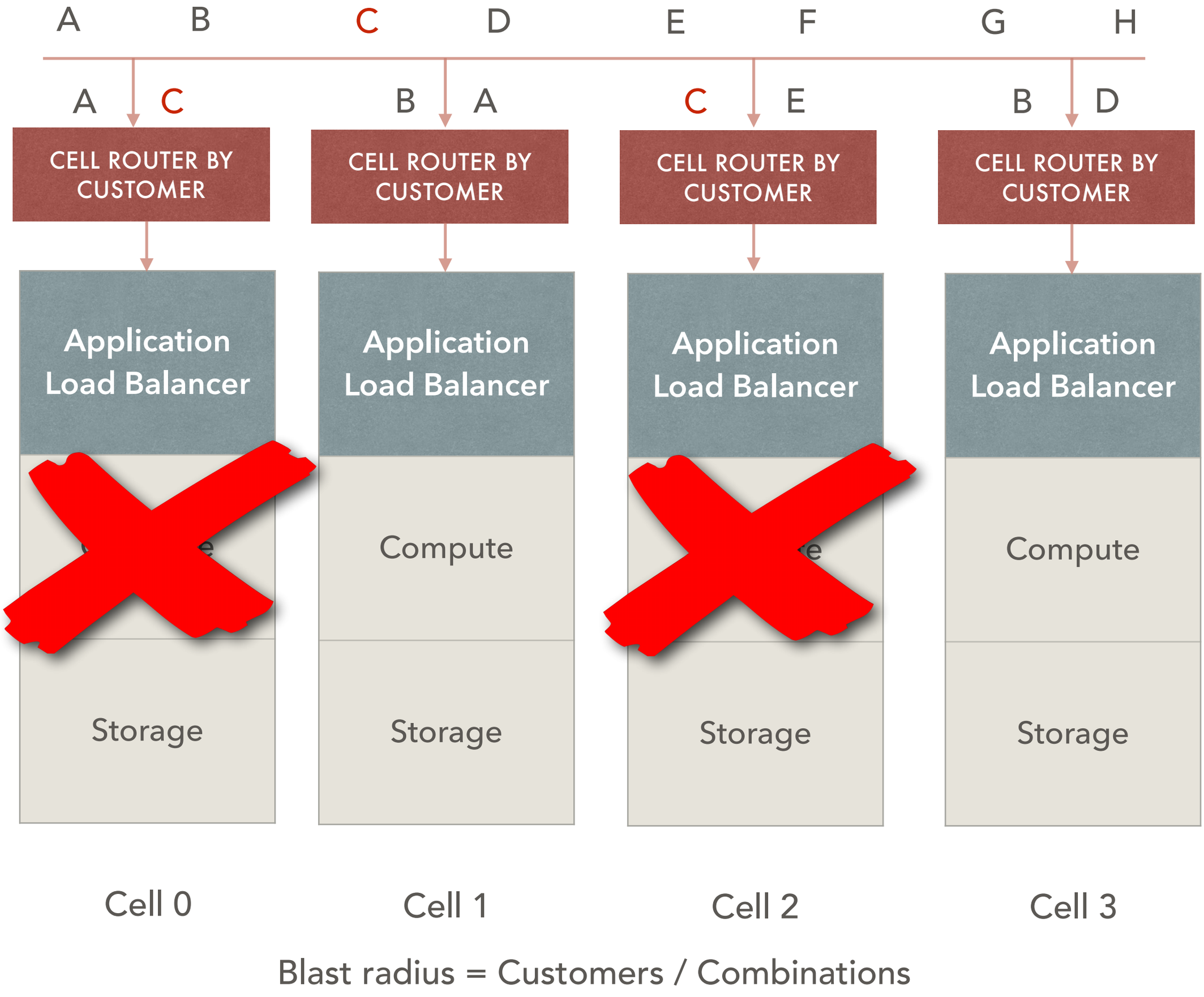
Regional Service

Service C


A
B
C
D
E
F
G
H
CELL ROUTER BY CUSTOMER
CELL ROUTER BY CUSTOMER
CELL ROUTER BY CUSTOMER
CELL ROUTER BY CUSTOMER
Application Load Balancer
Application Load Balancer
Application Load Balancer
Application Load Balancer
Compute
Compute
Compute
Compute
Storage
Storage
Storage
Storage
Cell 0
Cell 1
Cell 2
Cell 3
Regional Service


A
B
C
D
E
F
G
H
CELL ROUTER BY CUSTOMER
CELL ROUTER BY CUSTOMER
CELL ROUTER BY CUSTOMER
CELL ROUTER BY CUSTOMER
Application Load Balancer
Application Load Balancer
Application Load Balancer
Application Load Balancer
Compute
Compute
Compute
Storage
Storage
Storage
Storage
Cell 0
Cell 1
Cell 2
Cell 3
Regional Service


A
B
C
D
E
F
G
H
A
C
B
A
C
E
B
D
CELL ROUTER BY CUSTOMER
CELL ROUTER BY CUSTOMER
CELL ROUTER BY CUSTOMER
CELL ROUTER BY CUSTOMER
Application Load Balancer
Application Load Balancer
Application Load Balancer
Application Load Balancer
Compute
Compute
Compute
Compute
Storage
Storage
Storage
Storage
Cell 0
Cell 1
Cell 2
Cell 3
Regional Service



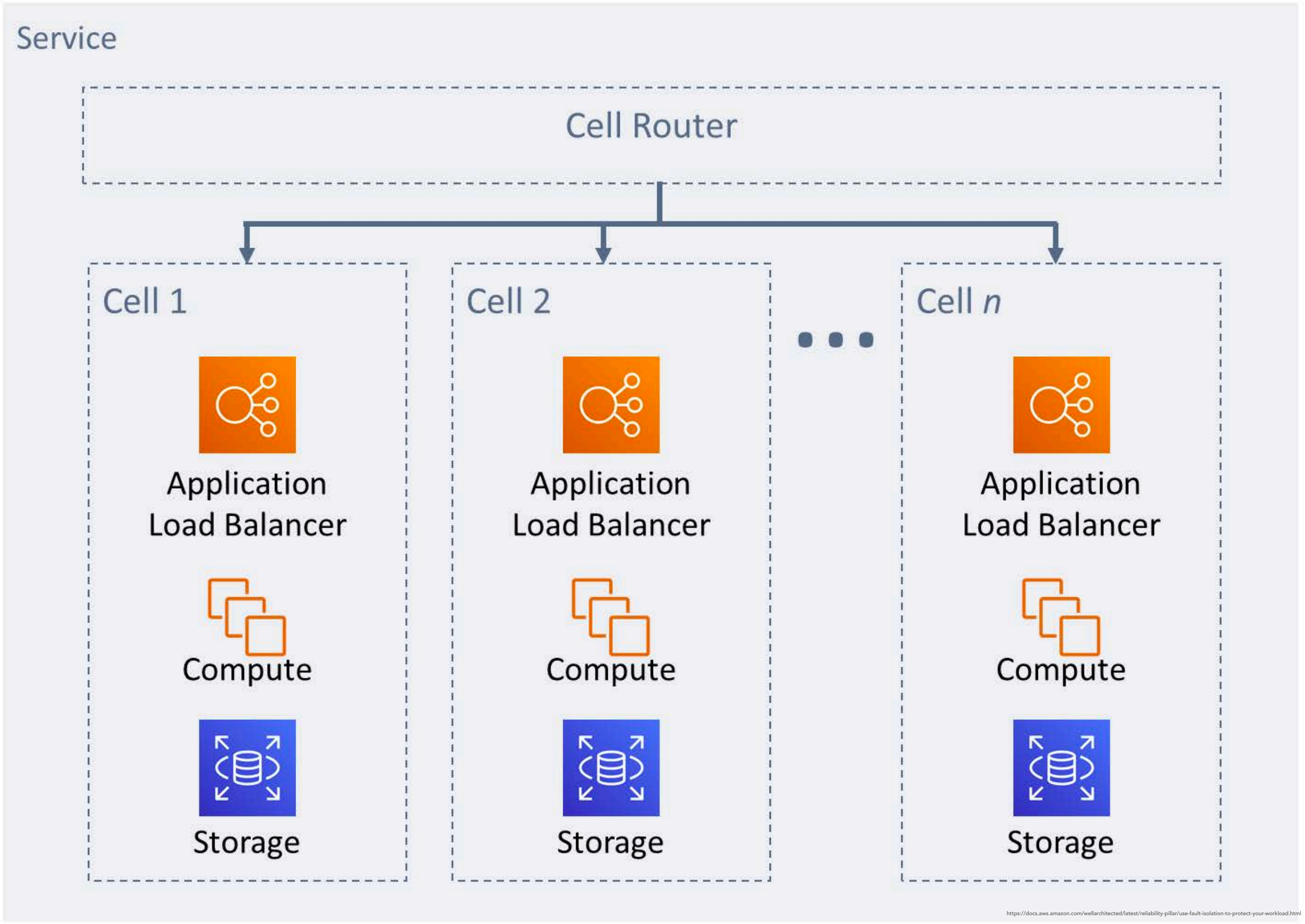
Blast radius = Customers / Combinations

Shuffle Sharding

Nodes = 100 Shard size = 5

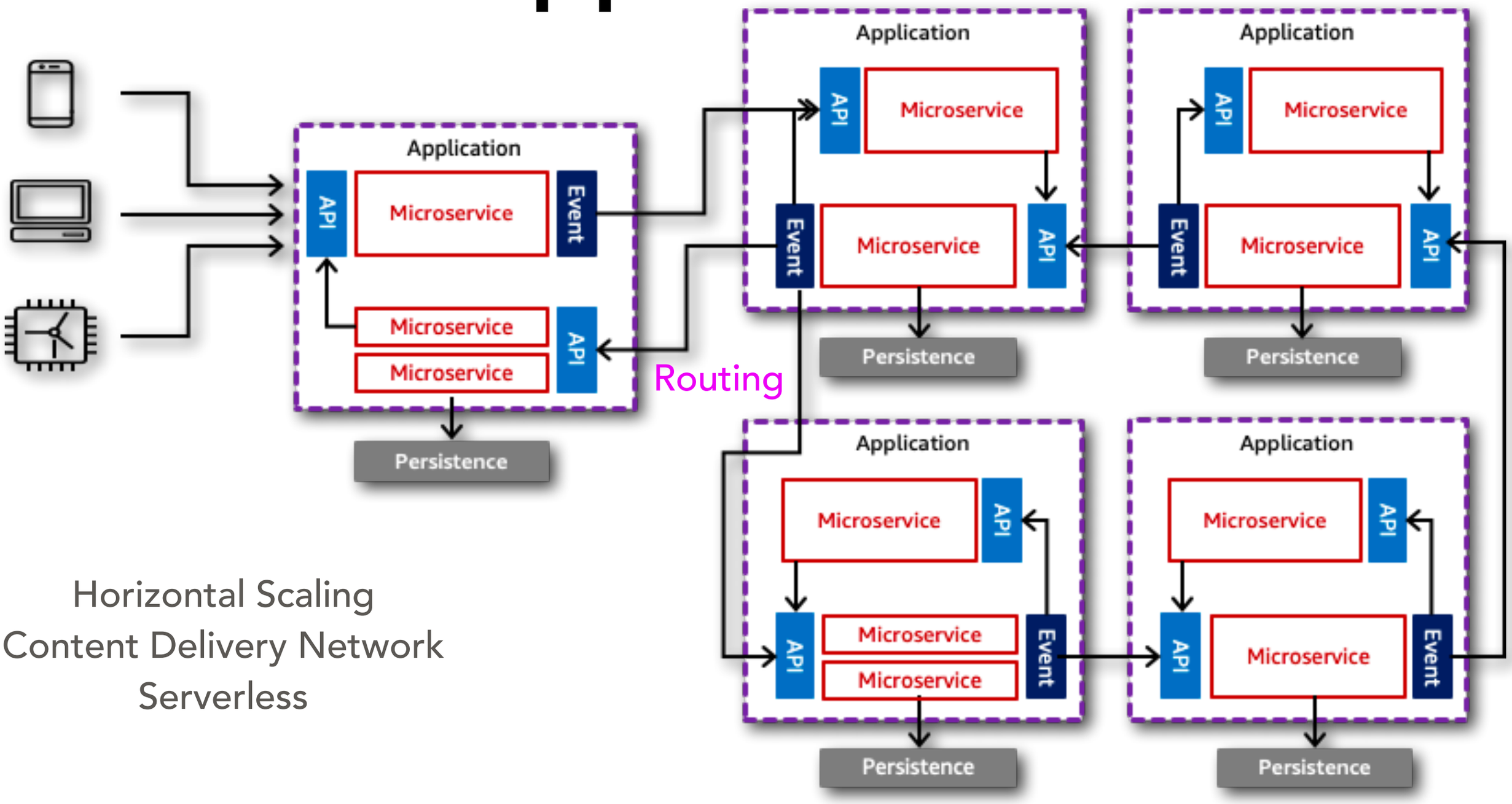
Overlap	% customers impacted
0	77%
1	21%
2	1.8%
3	0.06%
4	0.0006%
5	0.0000013%

https://threadreaderapp.com/thread/1034492056968736768.html



https://docs.aws.amazon.com/wellarchitected/latest/reliability-pillar/use-fault-isolation-to-protect-your-workload.html

Scalable modular applications



Horizontal Scaling
Content Delivery Network
Serverless

https://aws.amazon.com/blogs/architecture/architecting-for-reliable-scalability/

Route53

Domain names

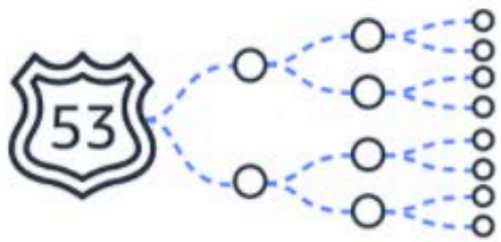
A domain is the name, such as example.com, that your users use to access your application.

Hosted zones

Specify how you want Route 53 to respond to DNS queries for a domain such as example.com.

Health checks

Monitor your applications and web resources, and direct DNS queries to healthy resources.



Traffic flow

Use a visual tool to create policies for multiple endpoints in complex configurations.

Failover

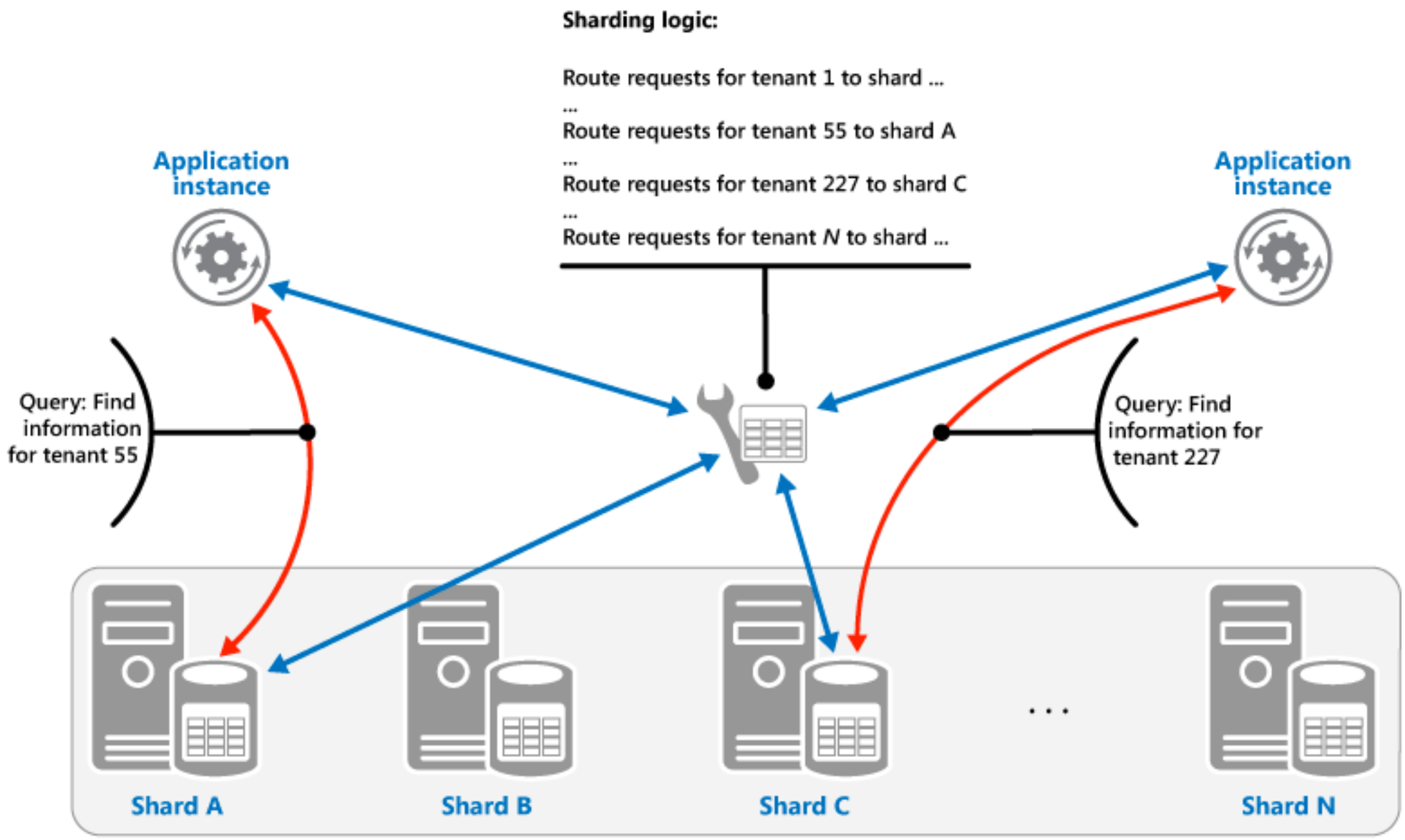
Resolver

Route DNS queries between your VPCs and your network.

Policy	What it Does
Simple	Simple DNS response providing the IP address associated with a name
Failover	If primary is down (based on health checks), routes to secondary destination
Geolocation	Uses geographic location you're in (e.g. Europe) to route you to the closest region
Geoproximity	Routes you to the closest region within a geographic area
Latency	Directs you based on the lowest latency route to resources
Multivalue answer	Returns several IP addresses and functions as a basic load balancer
Weighted	Uses the relative weights assigned to resources to determine which to route to

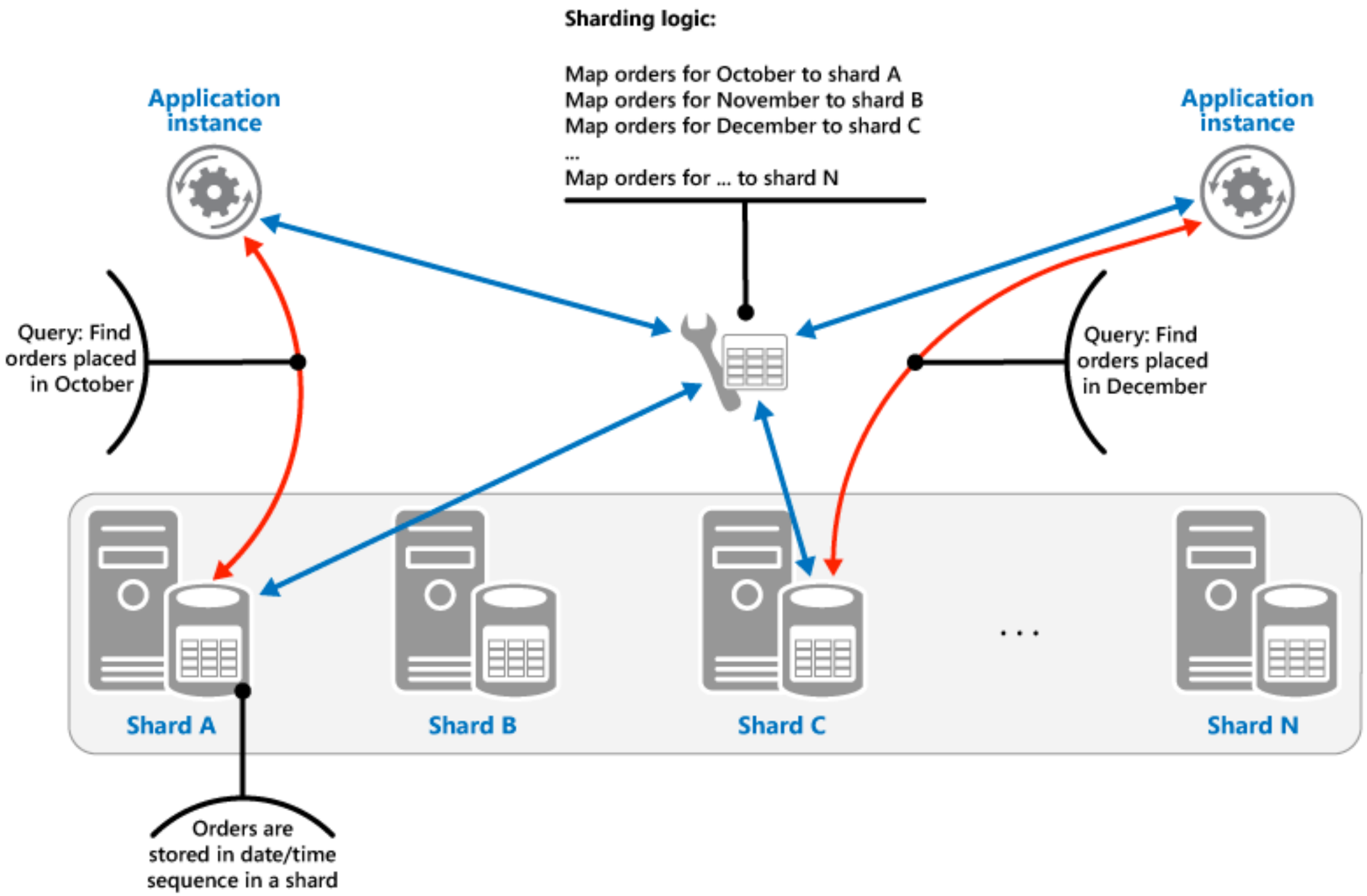
Sharding

Lookup Strategy



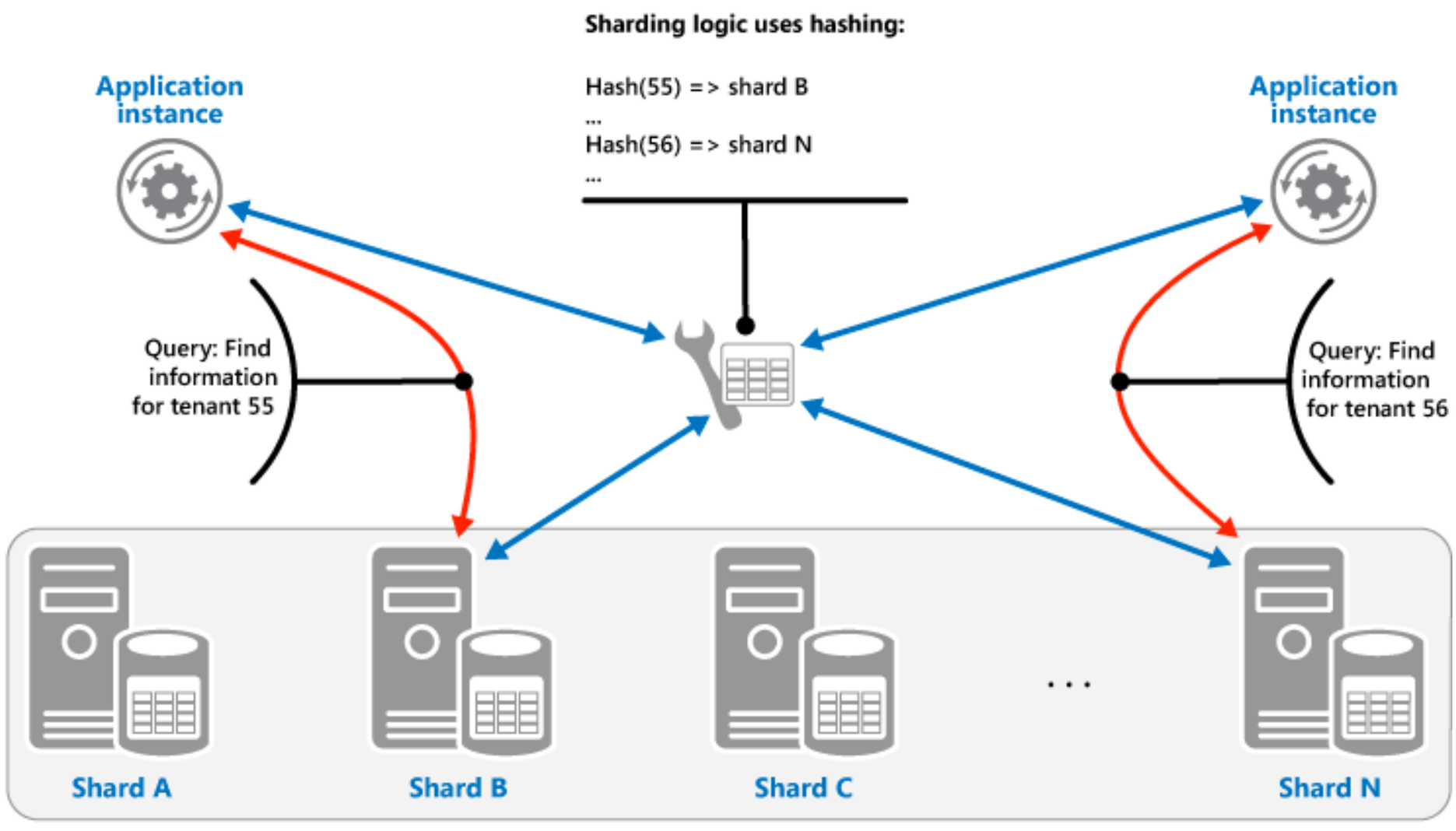
- Tenant ID, keep data together
- More controlled Virtual Shard
- State management
- Rebalance Shards

Range Strategy



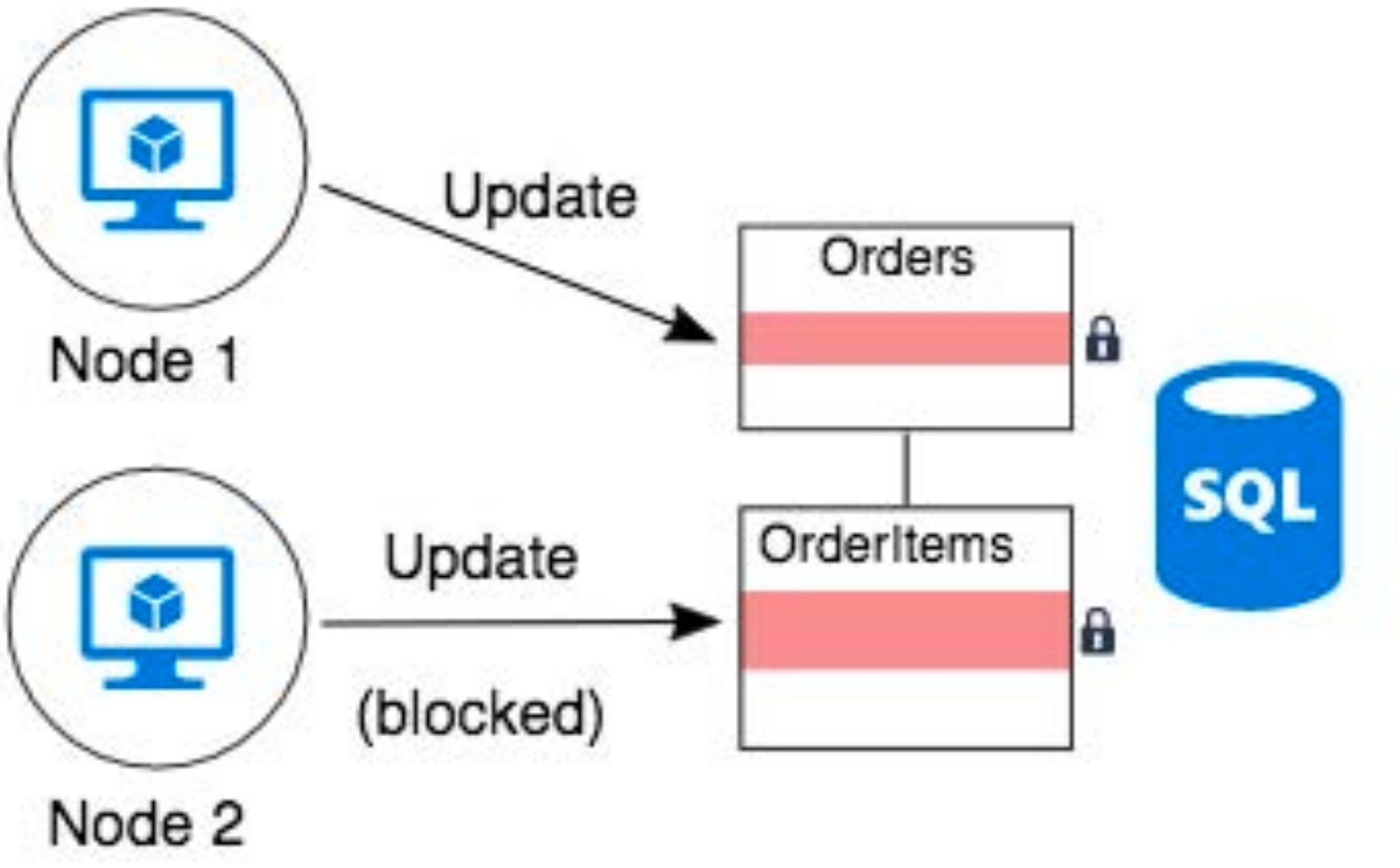
- Given Month
- Given Qtr
- State management
- No balancing between shards

Hash strategy



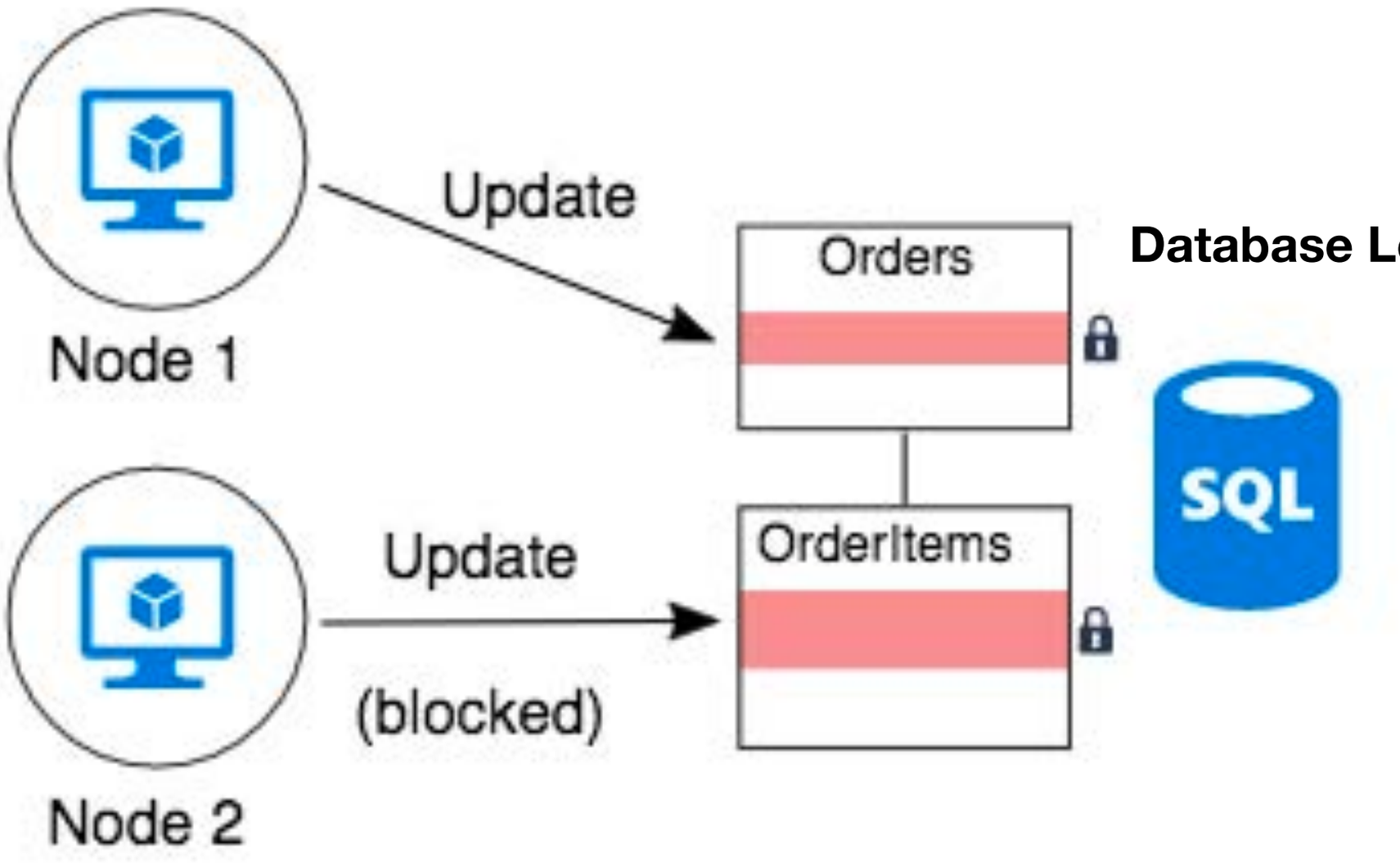
- Data Skew
- Even Distribution
- Rebalancing is difficult

What can be issue with services?



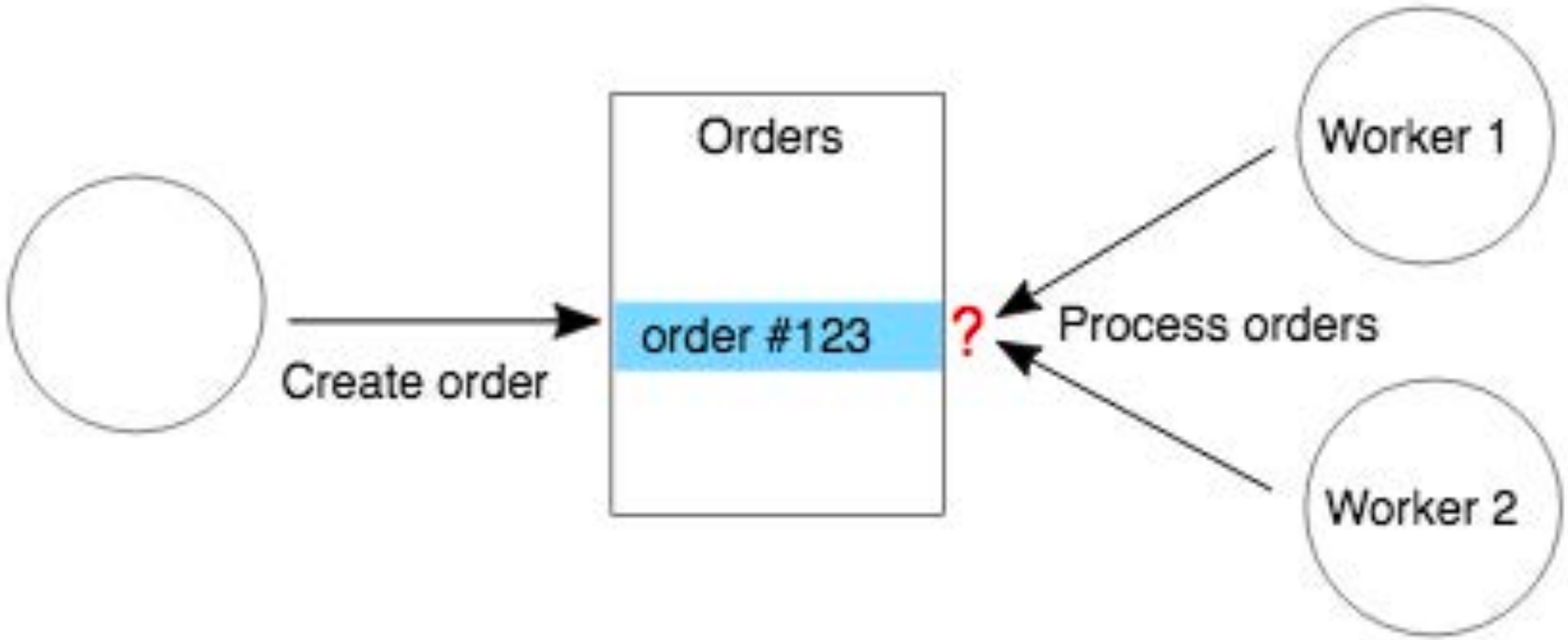
https://docs.microsoft.com/en-us/azure/architecture/guide/design-principles/minimize-coordination

Minimum coordination

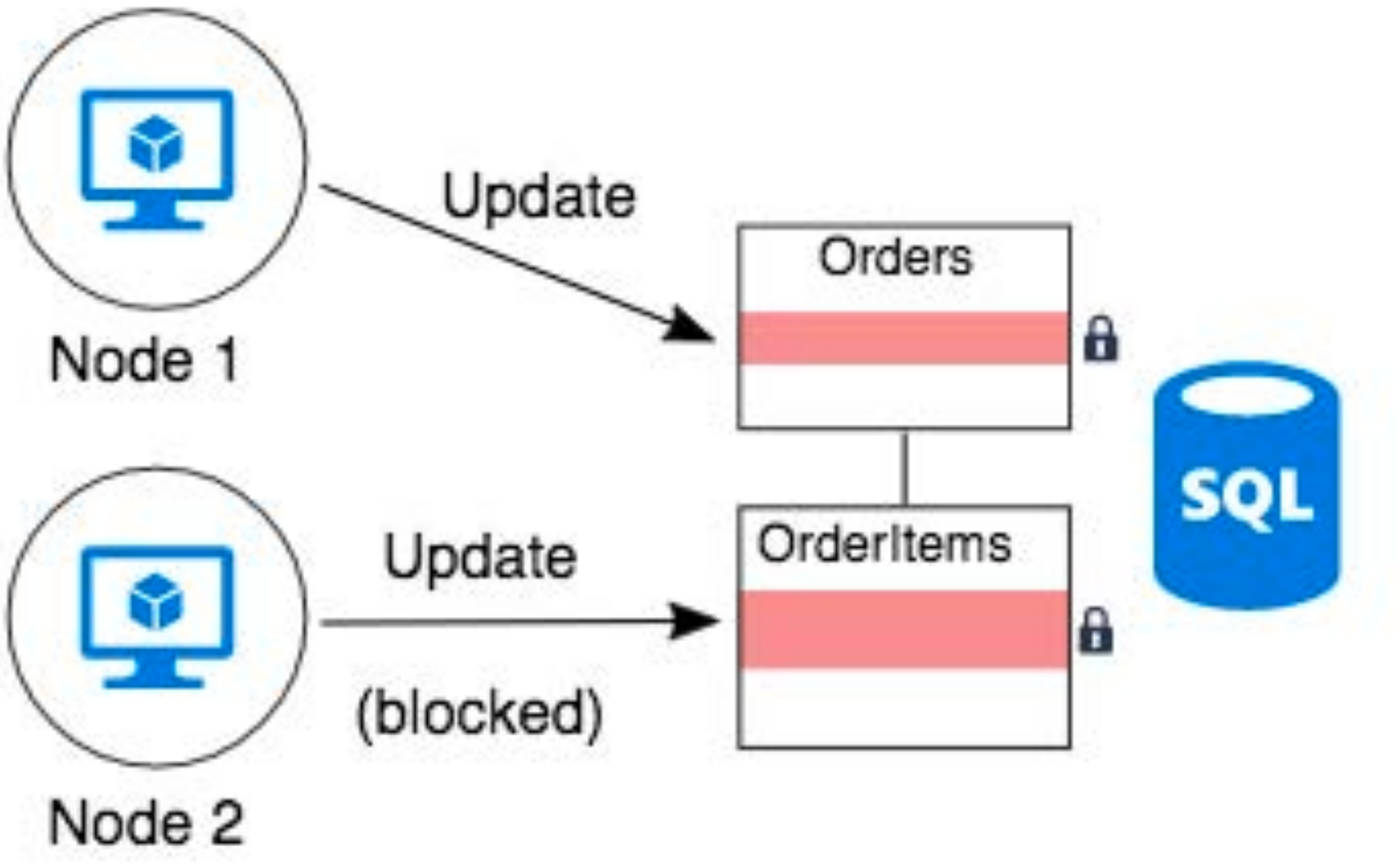


https://docs.microsoft.com/en-us/azure/architecture/guide/design-principles/minimize-coordination

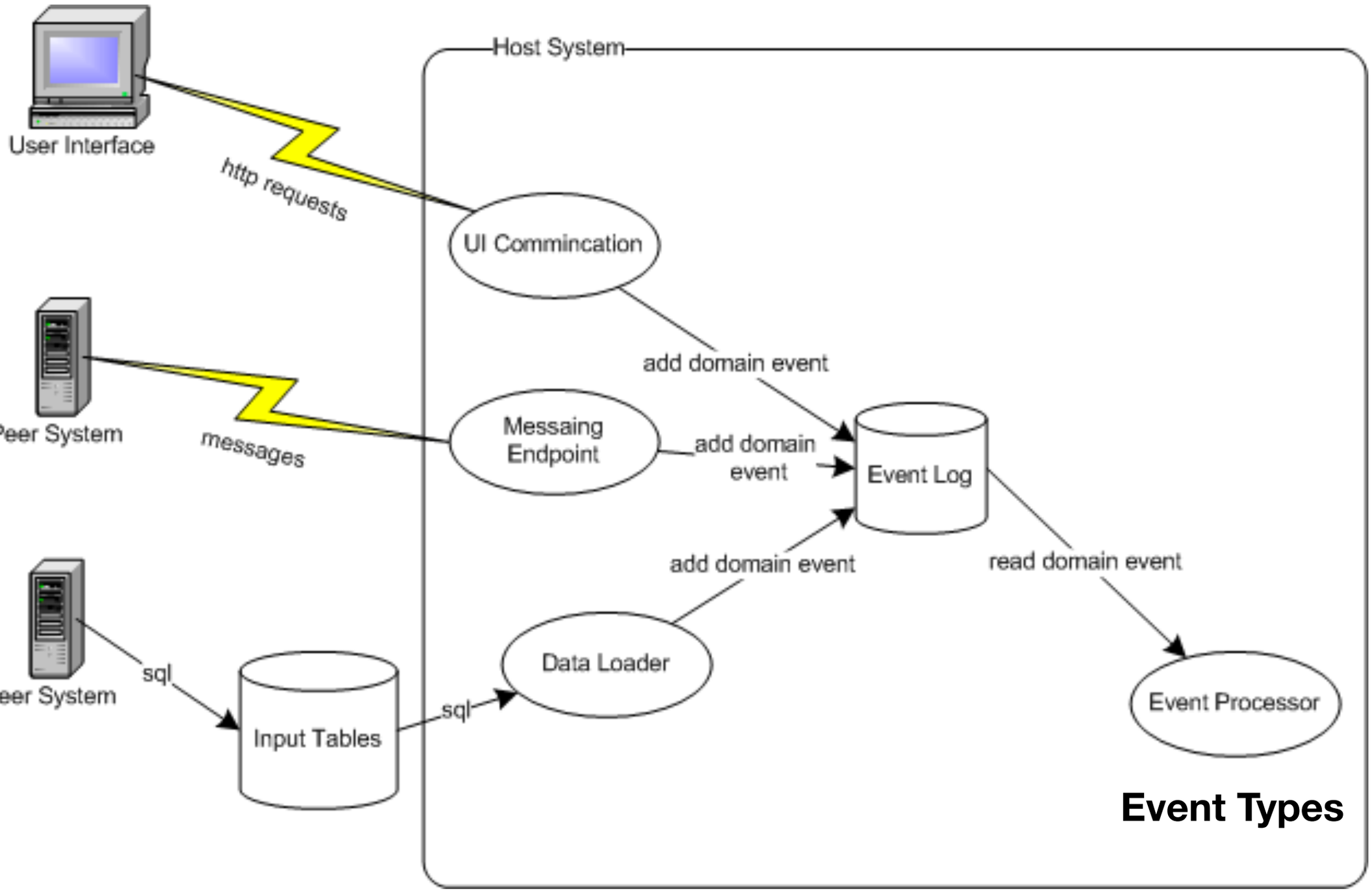
Exactly Once



How to do state management?

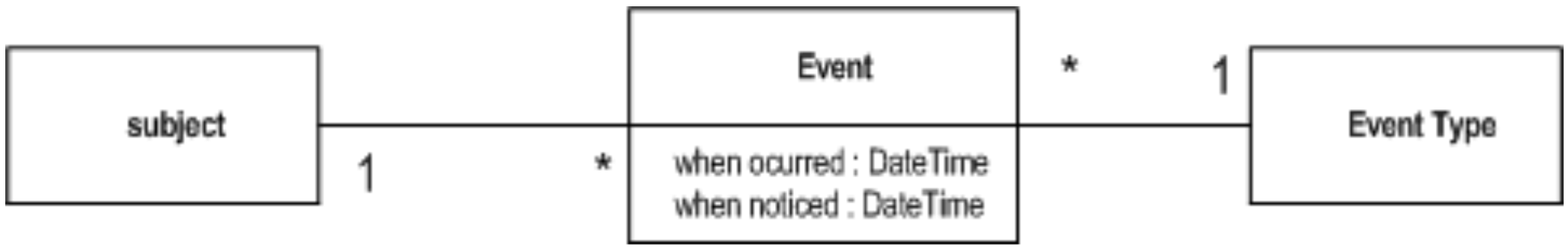


Immutable Audit Log

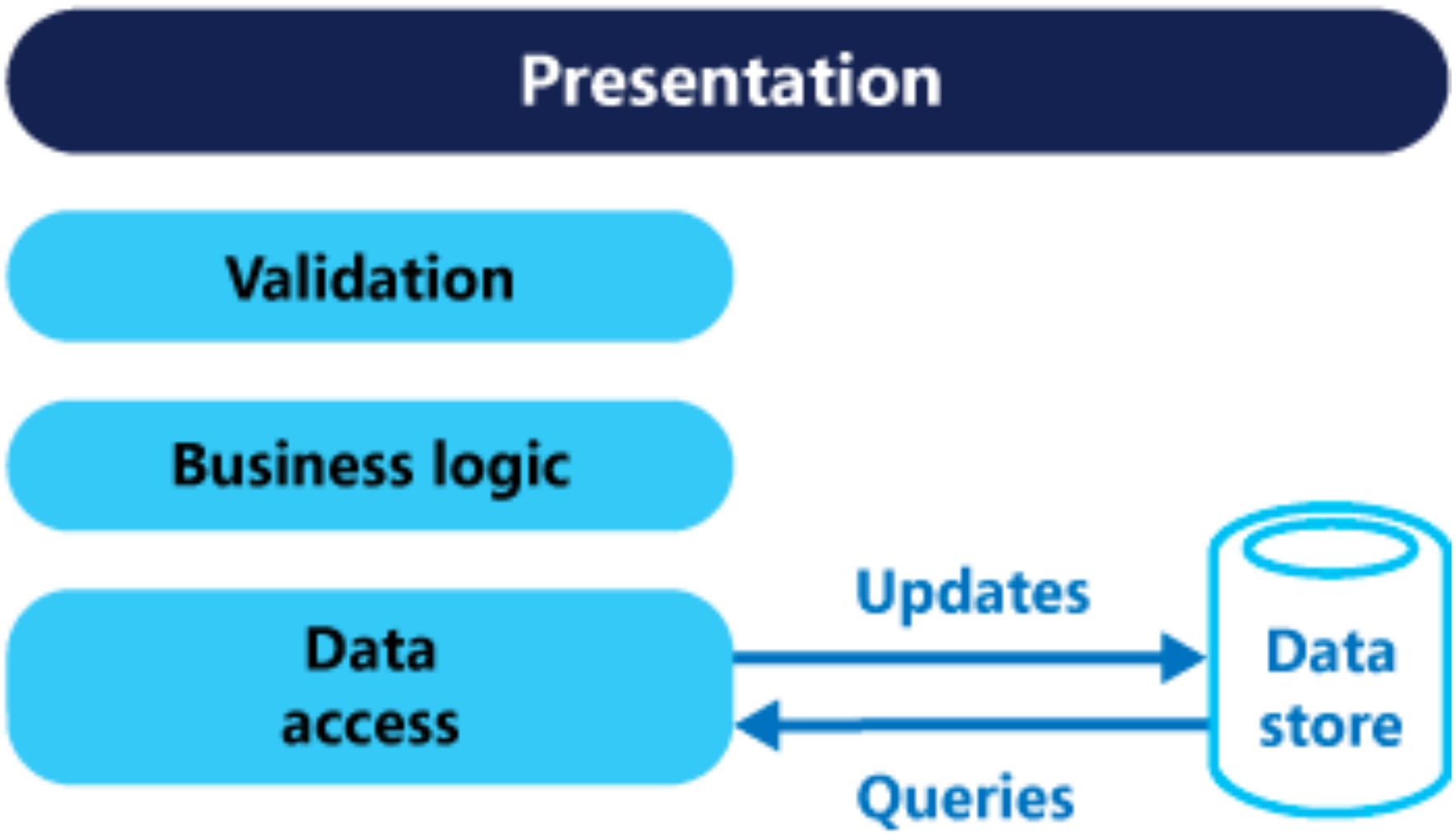


https://martinfowler.com/eaaDev/DomainEvent.html

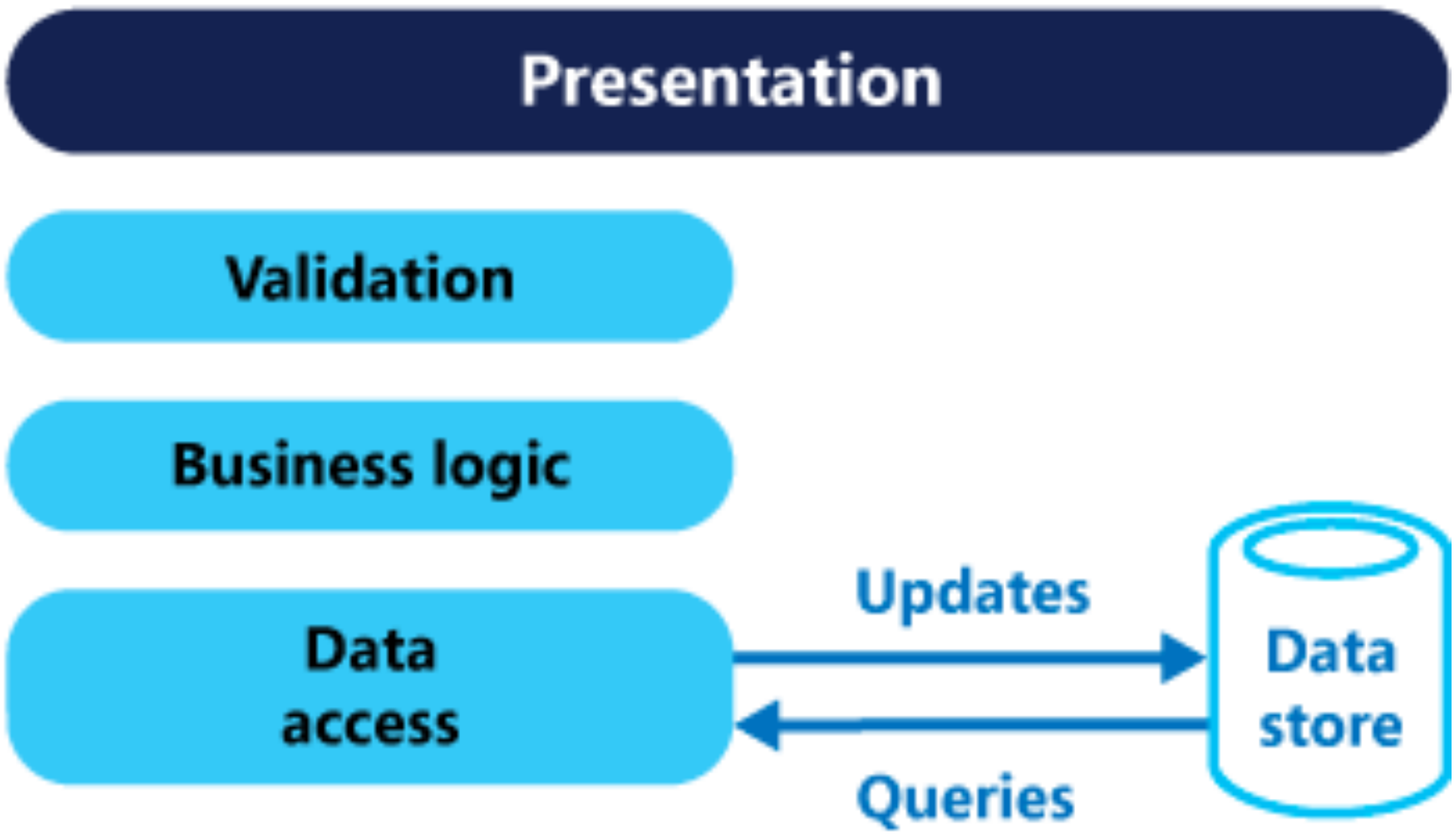
Domain Event



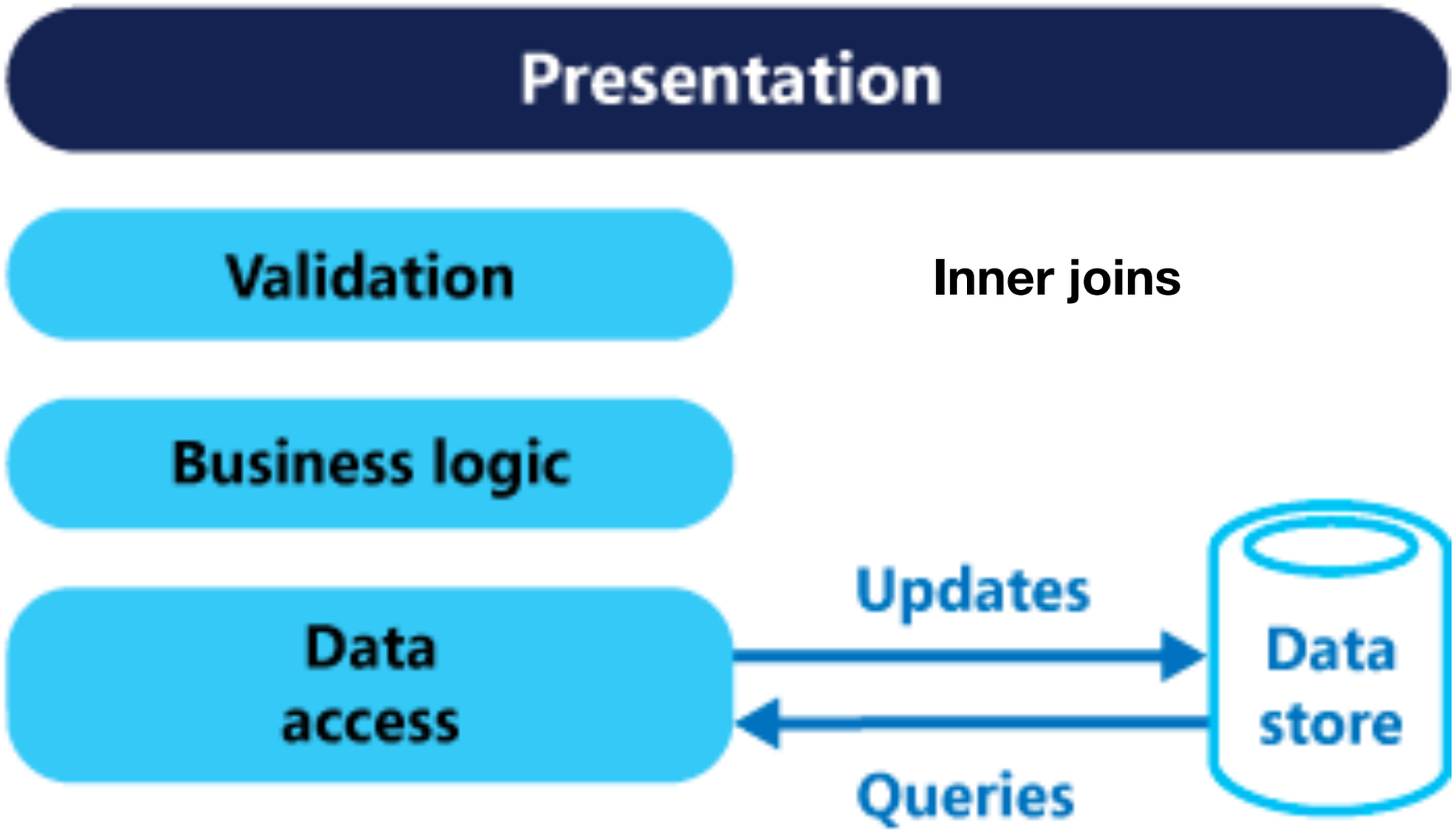
What might be the issue?



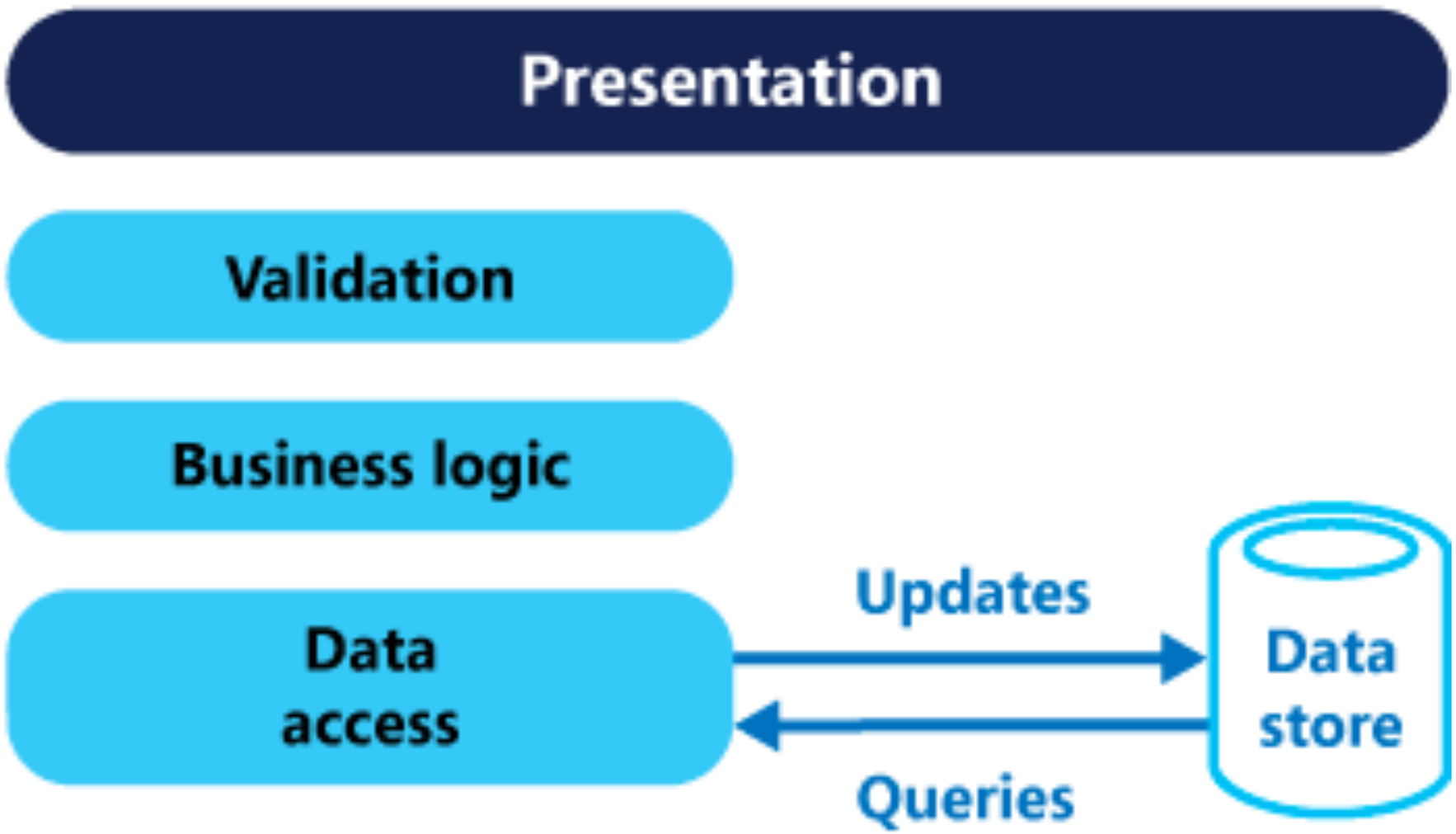
Read Complex data



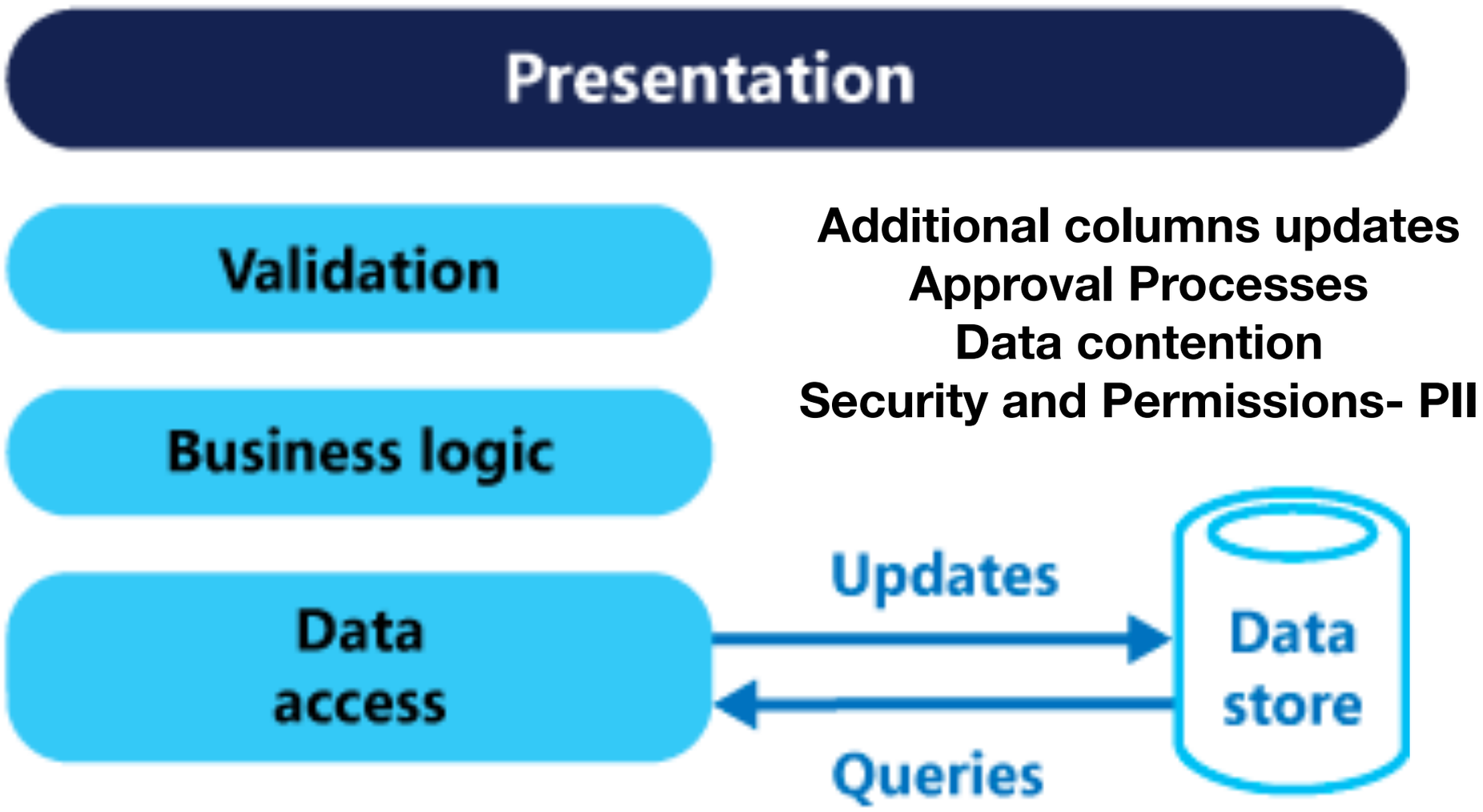
Read Complex data



Write Complex data

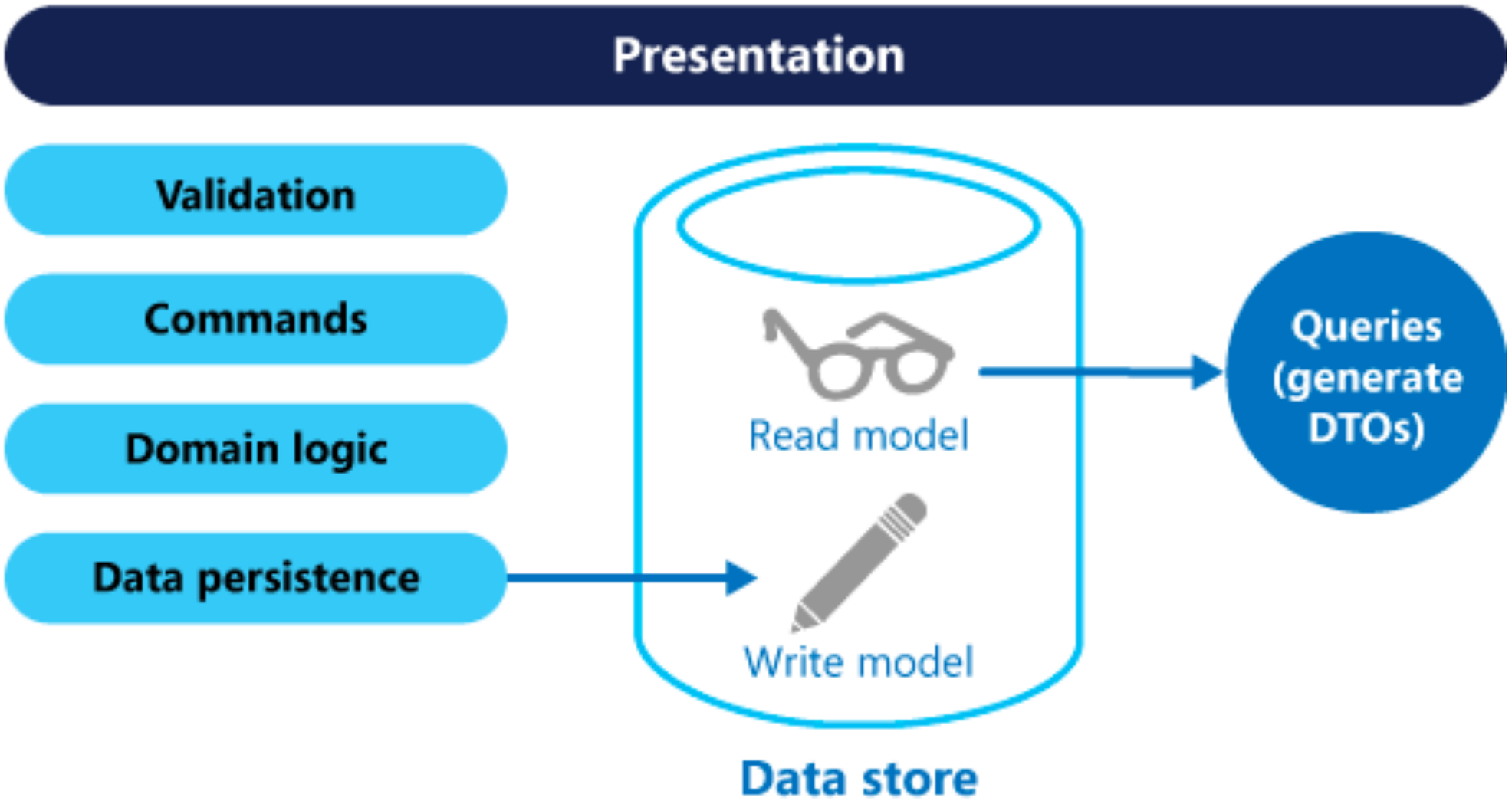


Write Complex data

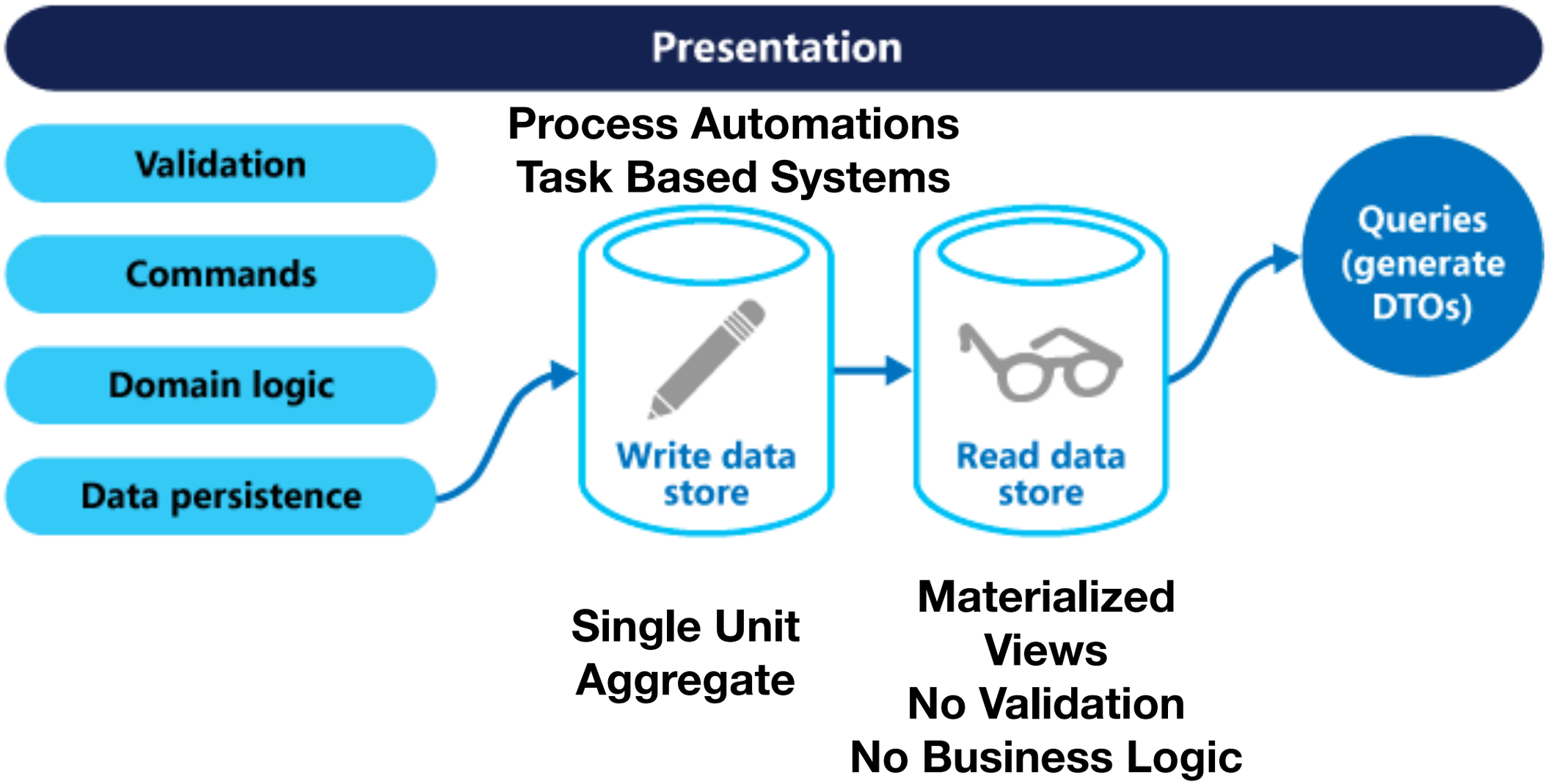


Additional columns updates
Approval Processes
Data contention
Security and Permissions- PII

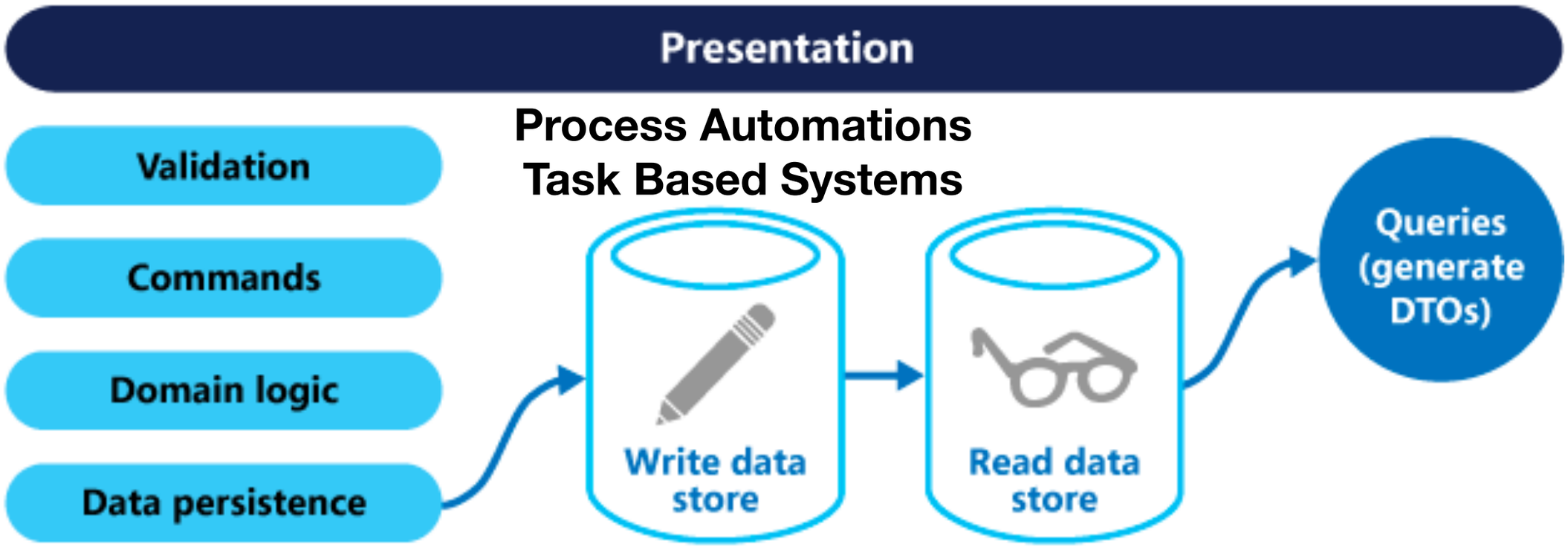
Command and Query Responsibility Segregation (CQRS)



What are the benefits?



What are the benefits?



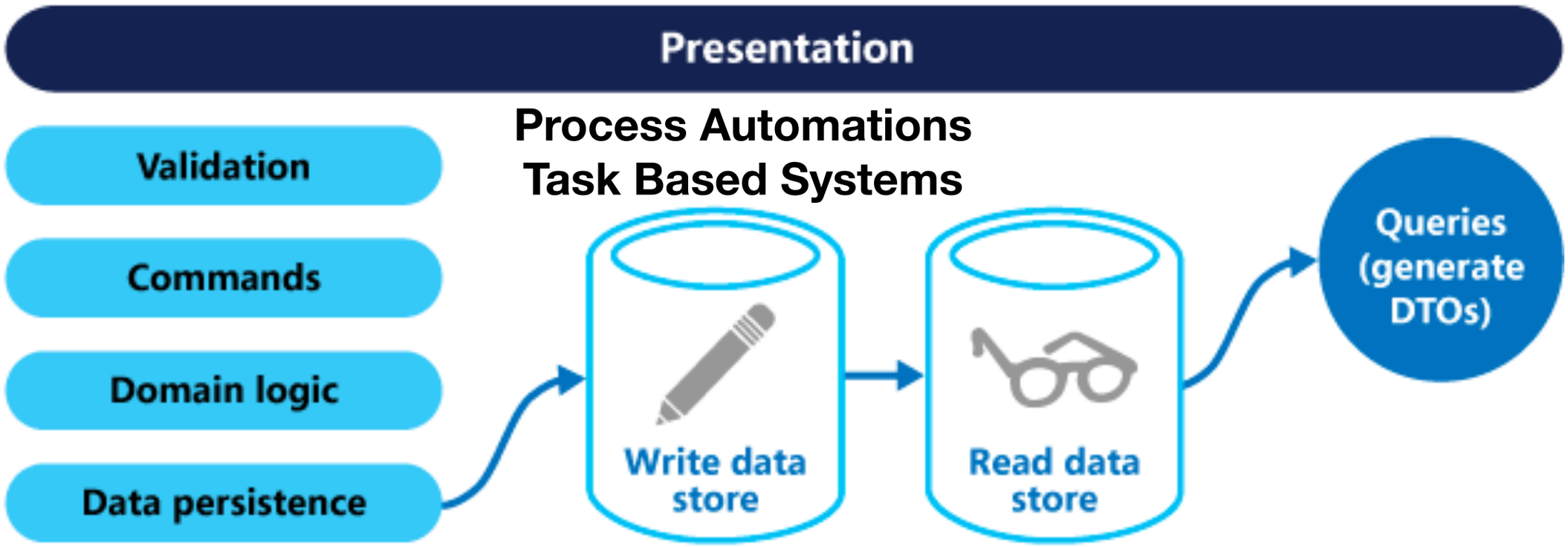
Single Unit Aggregate

Materialized Views
No Validation
No Business Logic

Higher Reads than Writes

Where?

What are the benefits?



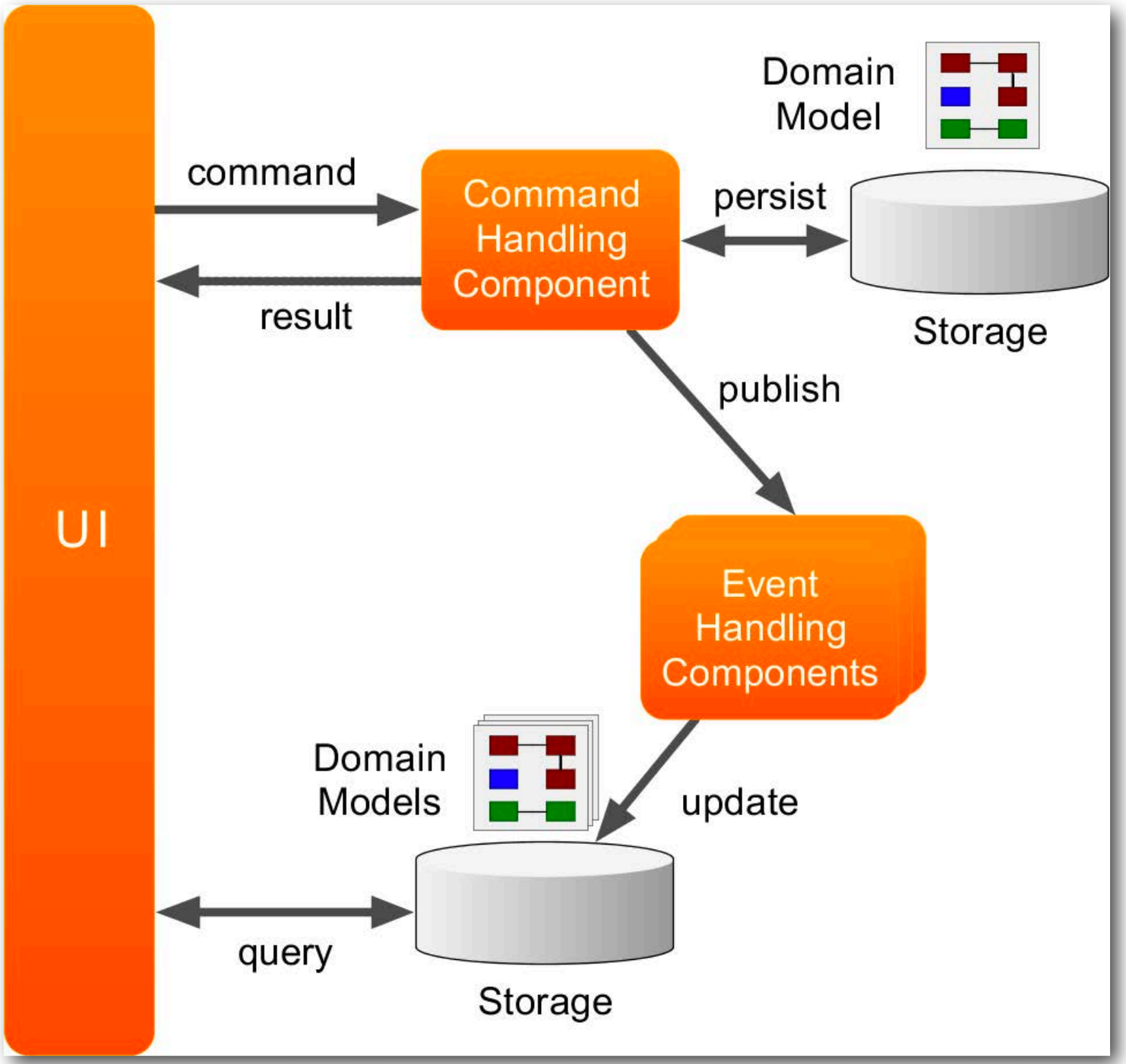
Single Unit Aggregate

Materialized Views
No Validation
No Business Logic

Higher Reads than Writes

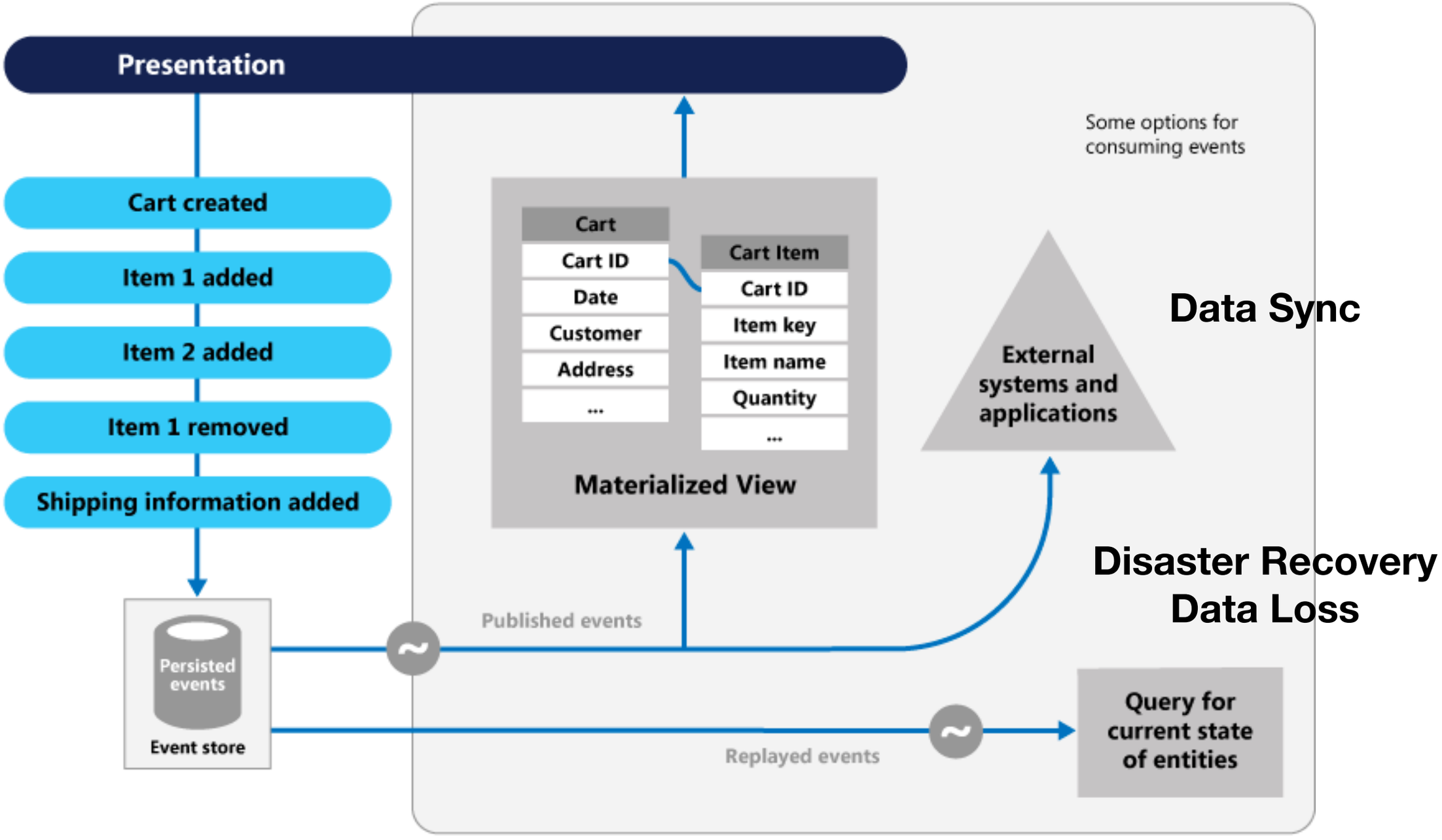
Twitter

Command Query Responsibility Segregation (CQRS)



https://blogs.vmware.com/management/2013/02/scaling-in-complex-domains-using-cqrs-axon-and-spring-insight.html

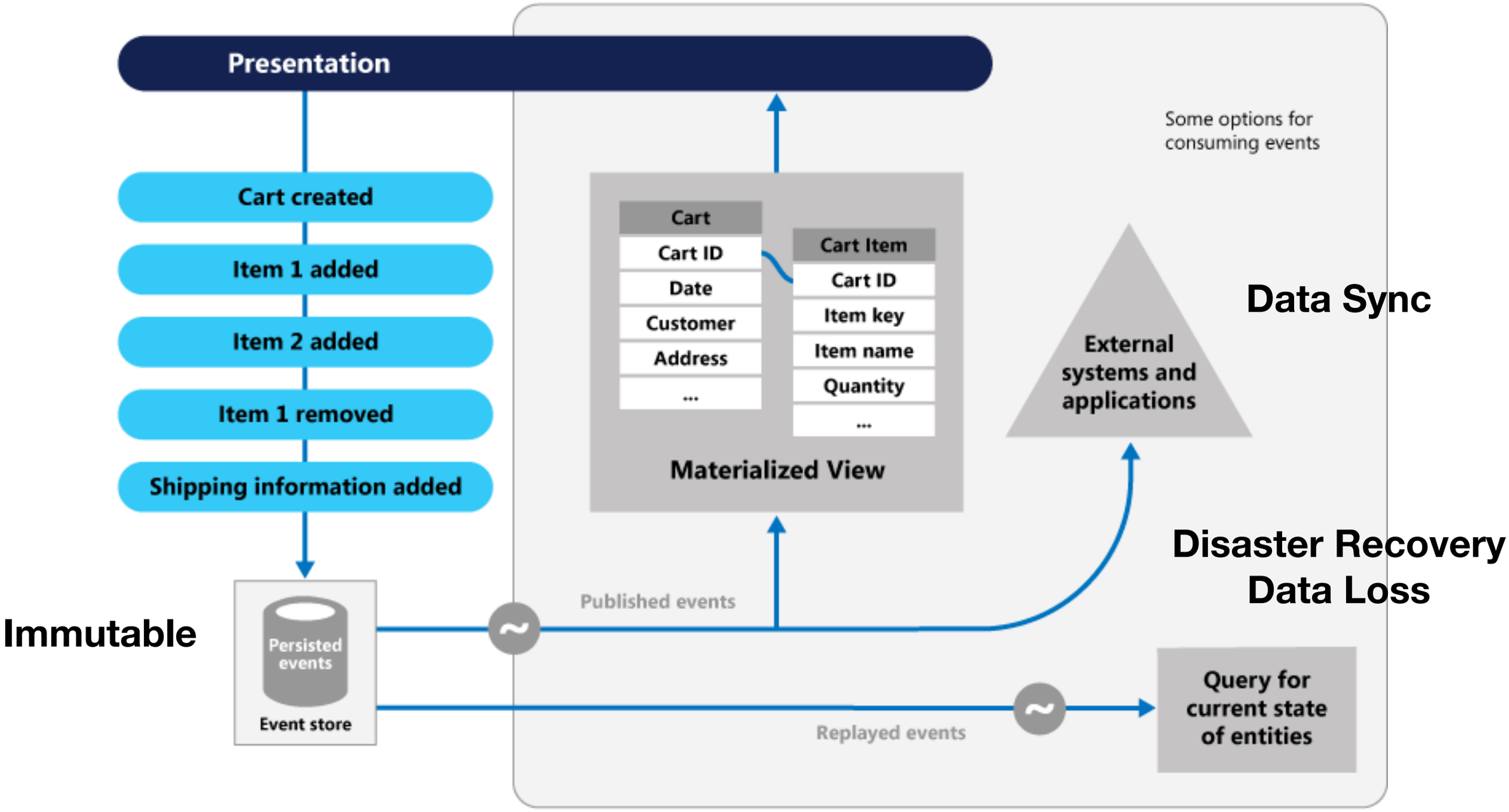
Event Sourcing pattern



Data Sync

Disaster Recovery
Data Loss

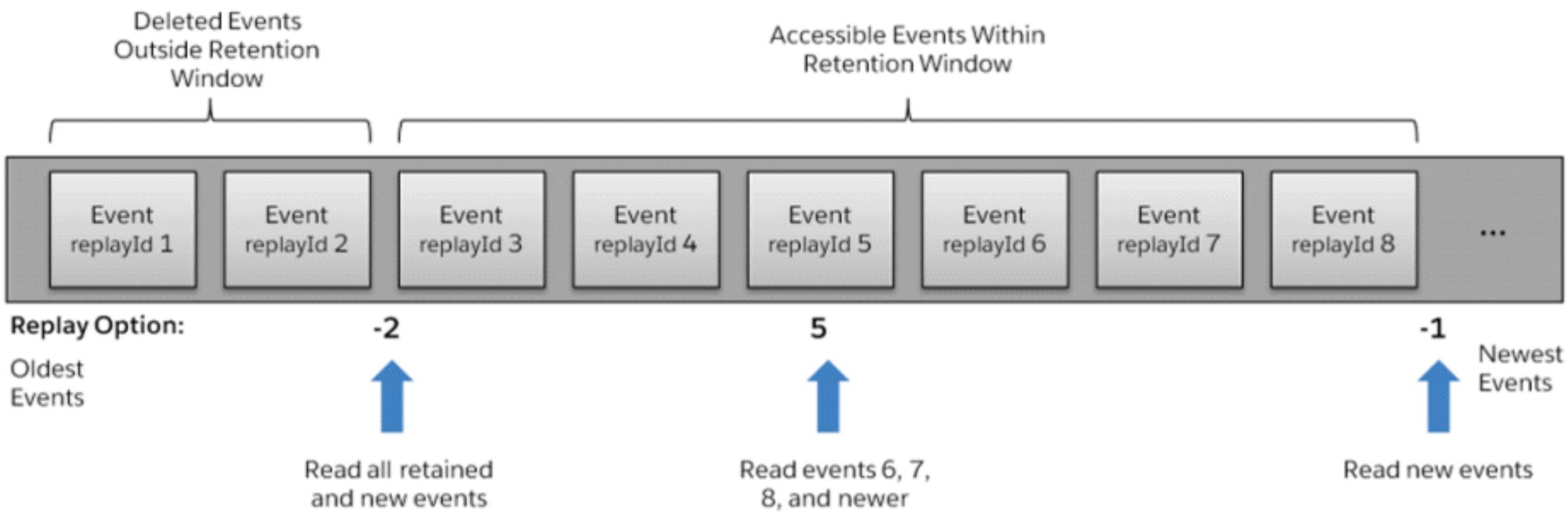
Event Sourcing pattern



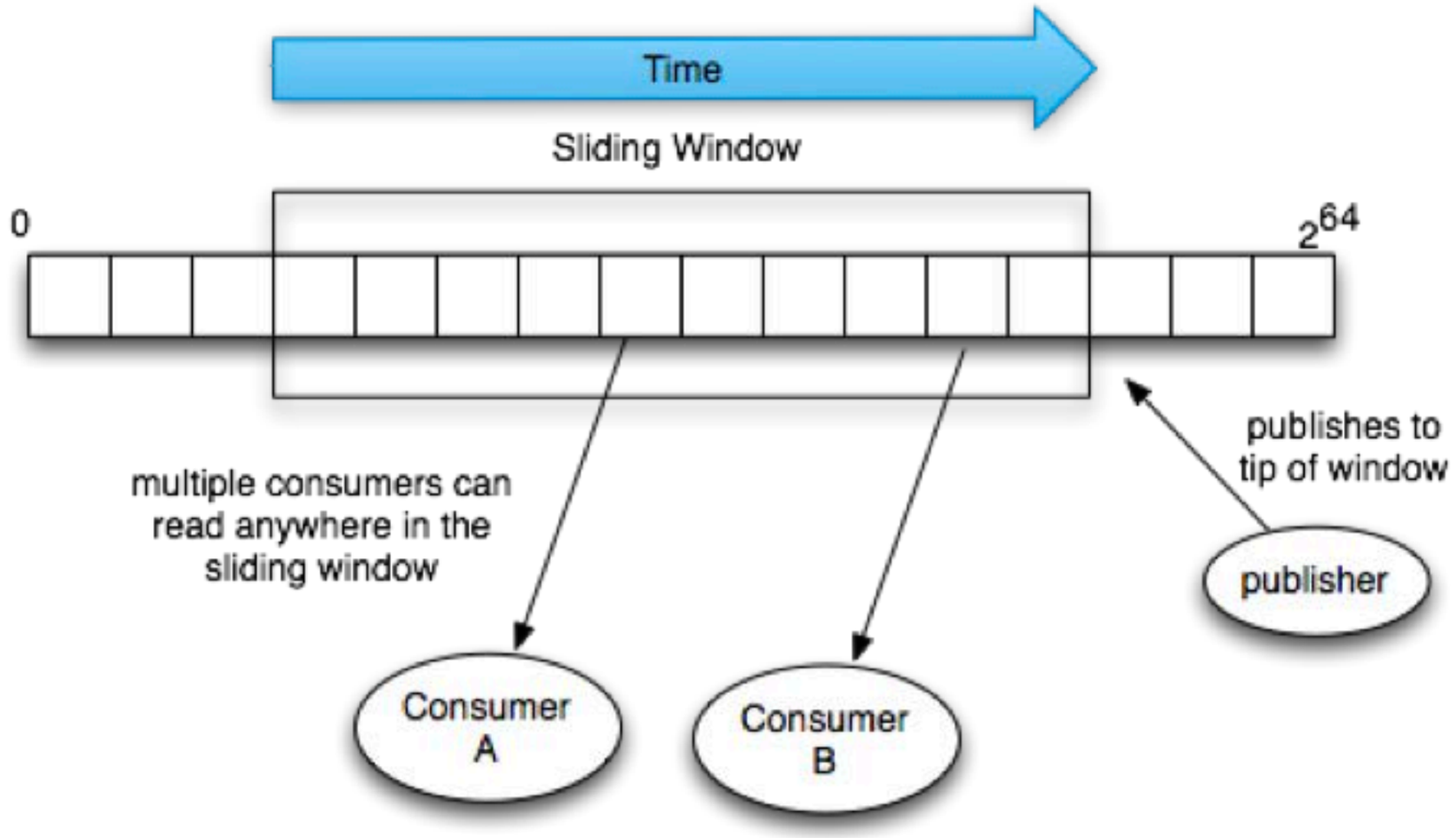
What are the challenges?

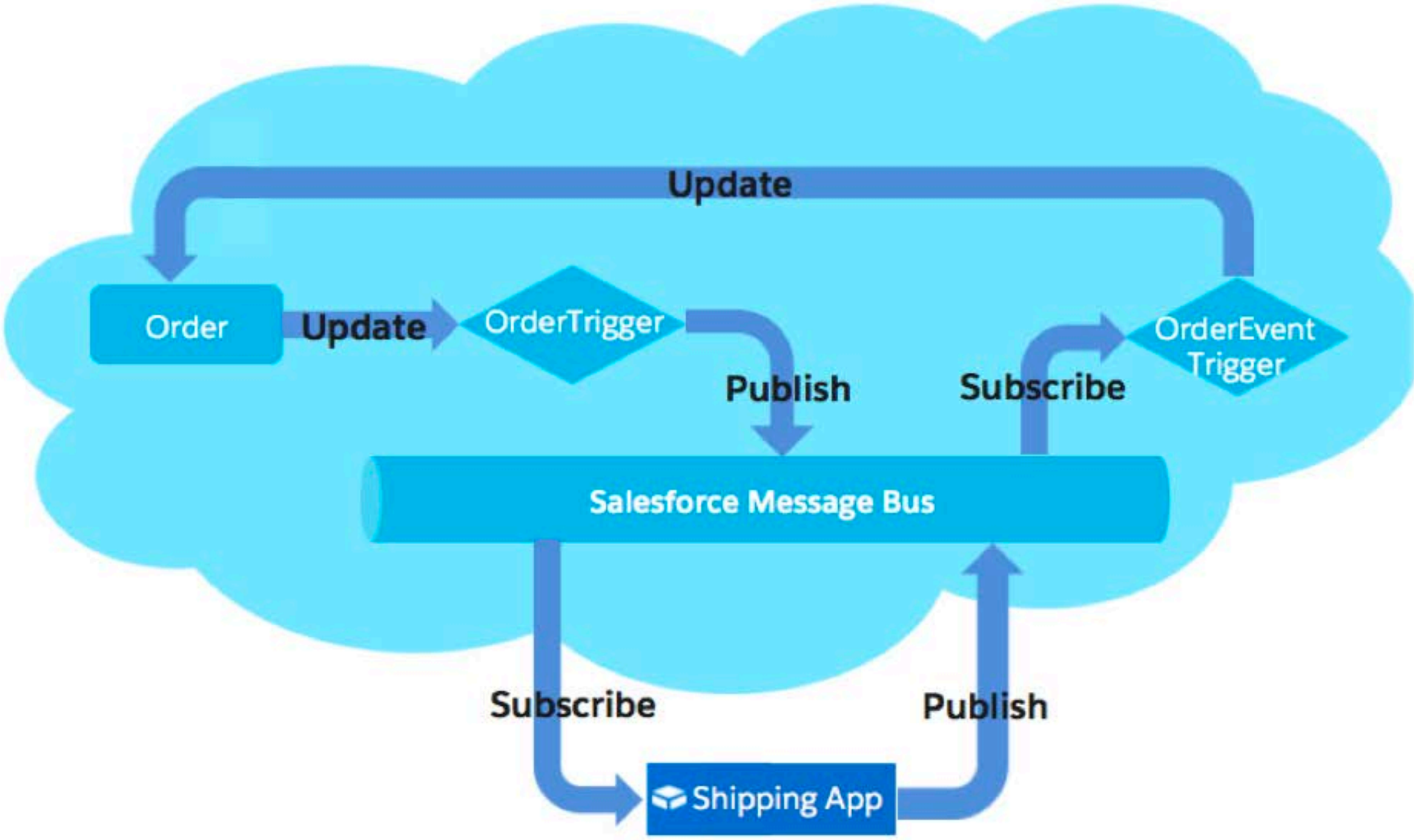
- What about changes?
- How to manage message failures
- How to handle duplicate messages?
- What is the priority of a message?

Replay events



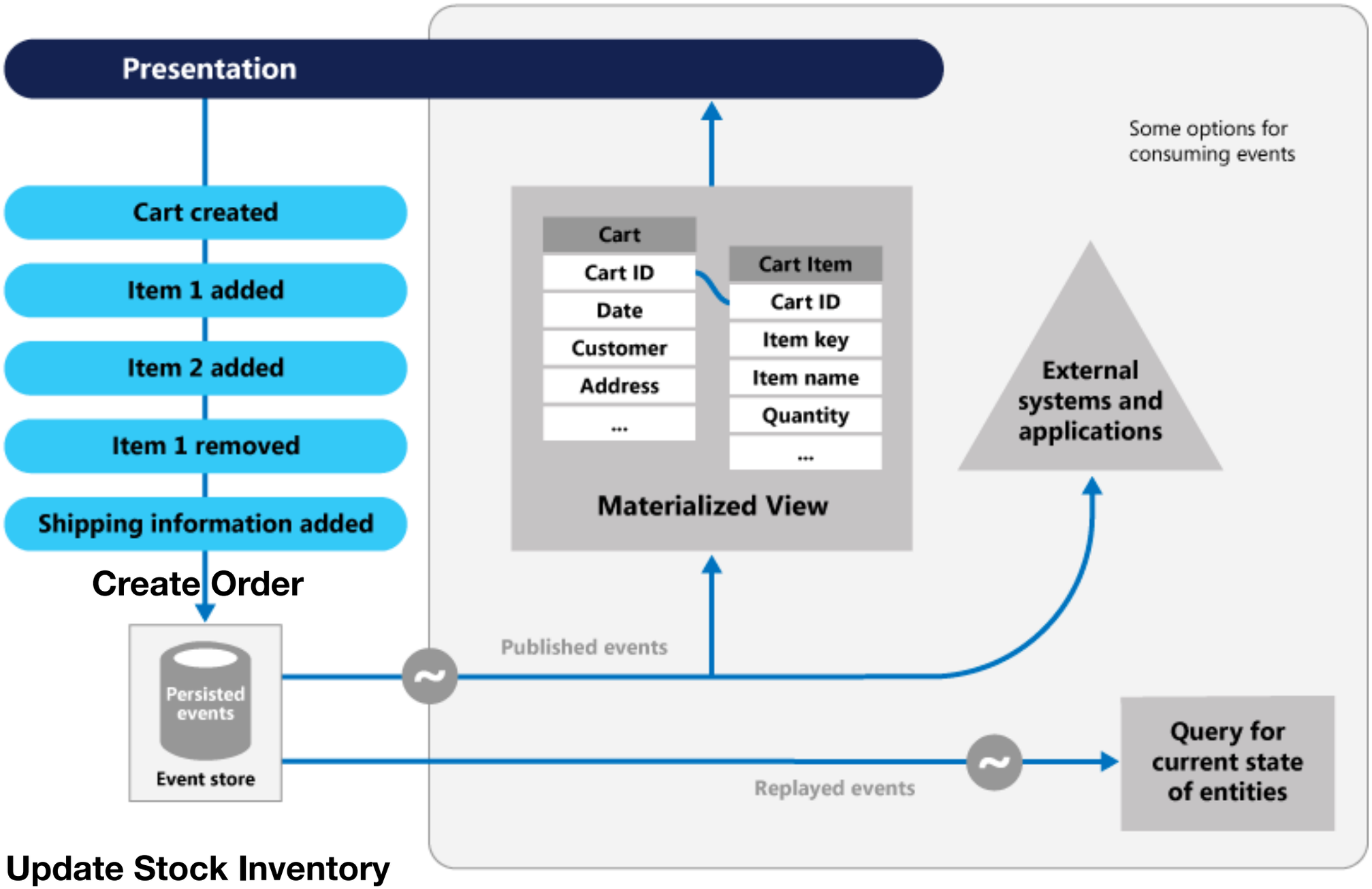
https://developer.salesforce.com/docs/atlas.en-us.api_streaming.meta/api_streaming/using_streaming_api_durability.htm


Time
Sliding Window
0
2^{64}
multiple consumers can read anywhere in the sliding window
publishes to tip of window
publisher
Consumer A
Consumer B

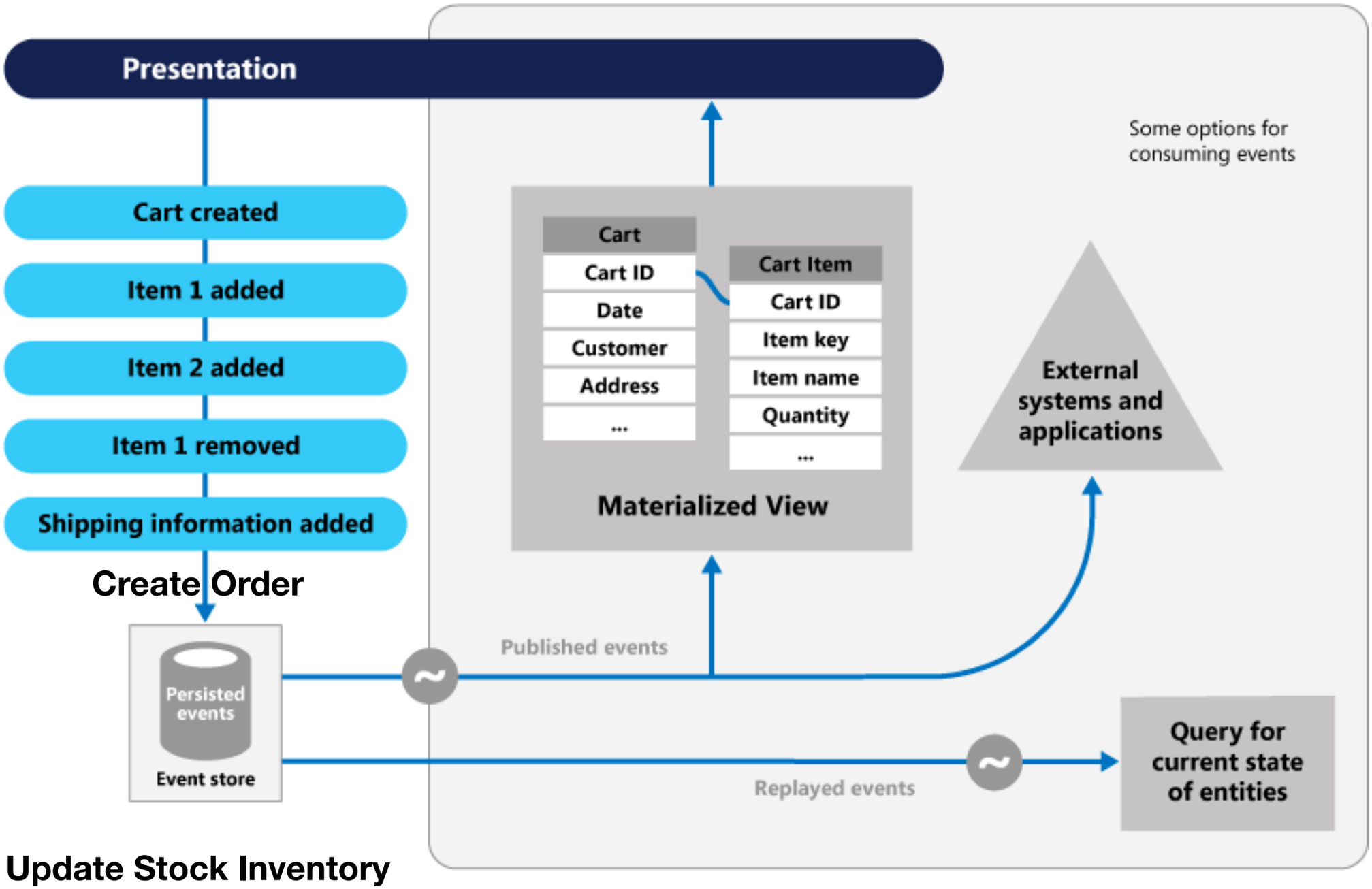


https://developer.salesforce.com/blogs/developer-relations/2017/05/first-impressions-platform-events-salesforce-enterprise-messaging-platform.html

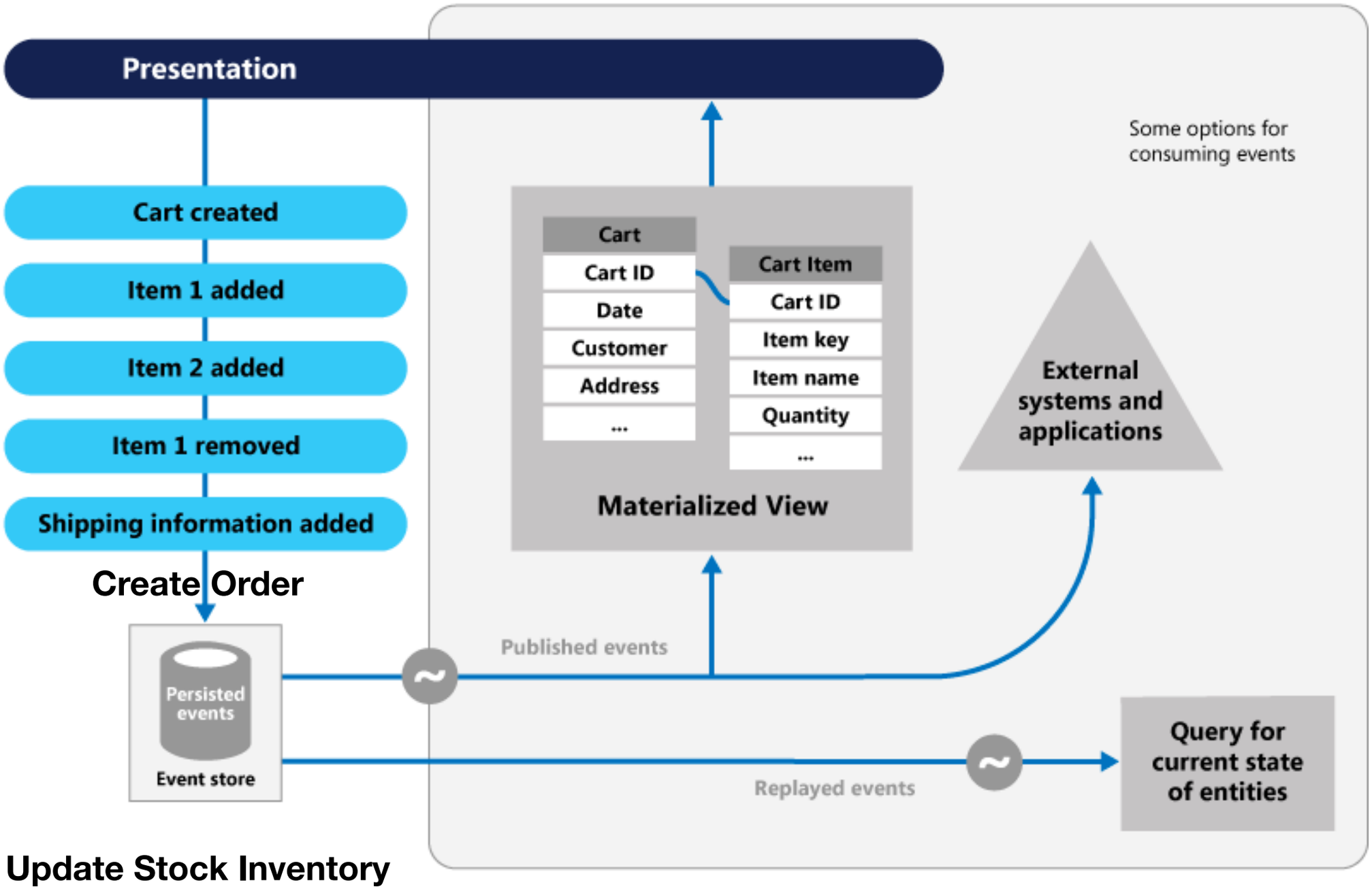
What are the challenges?



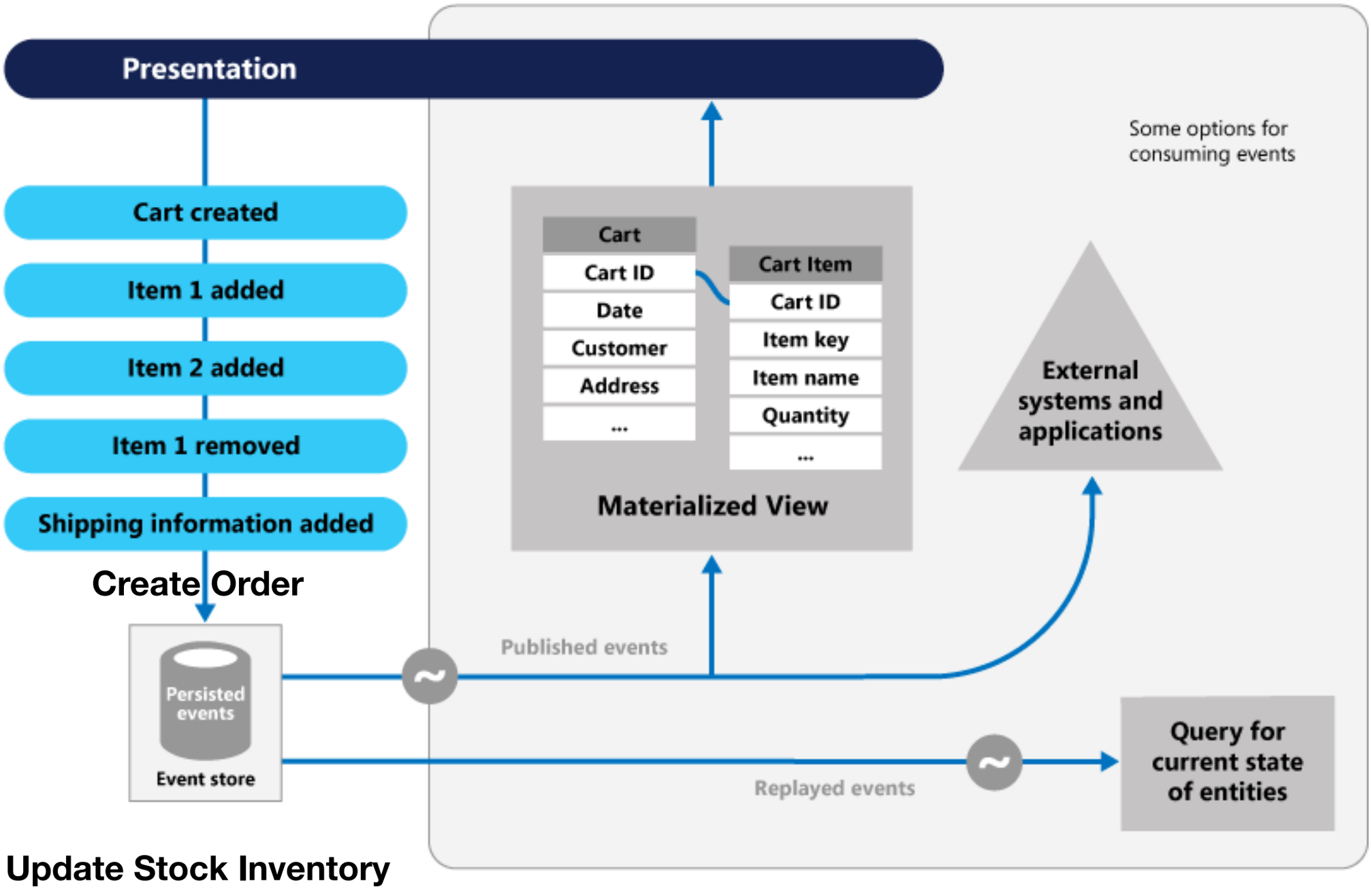
What happens when Stock Inventory is inserted first?



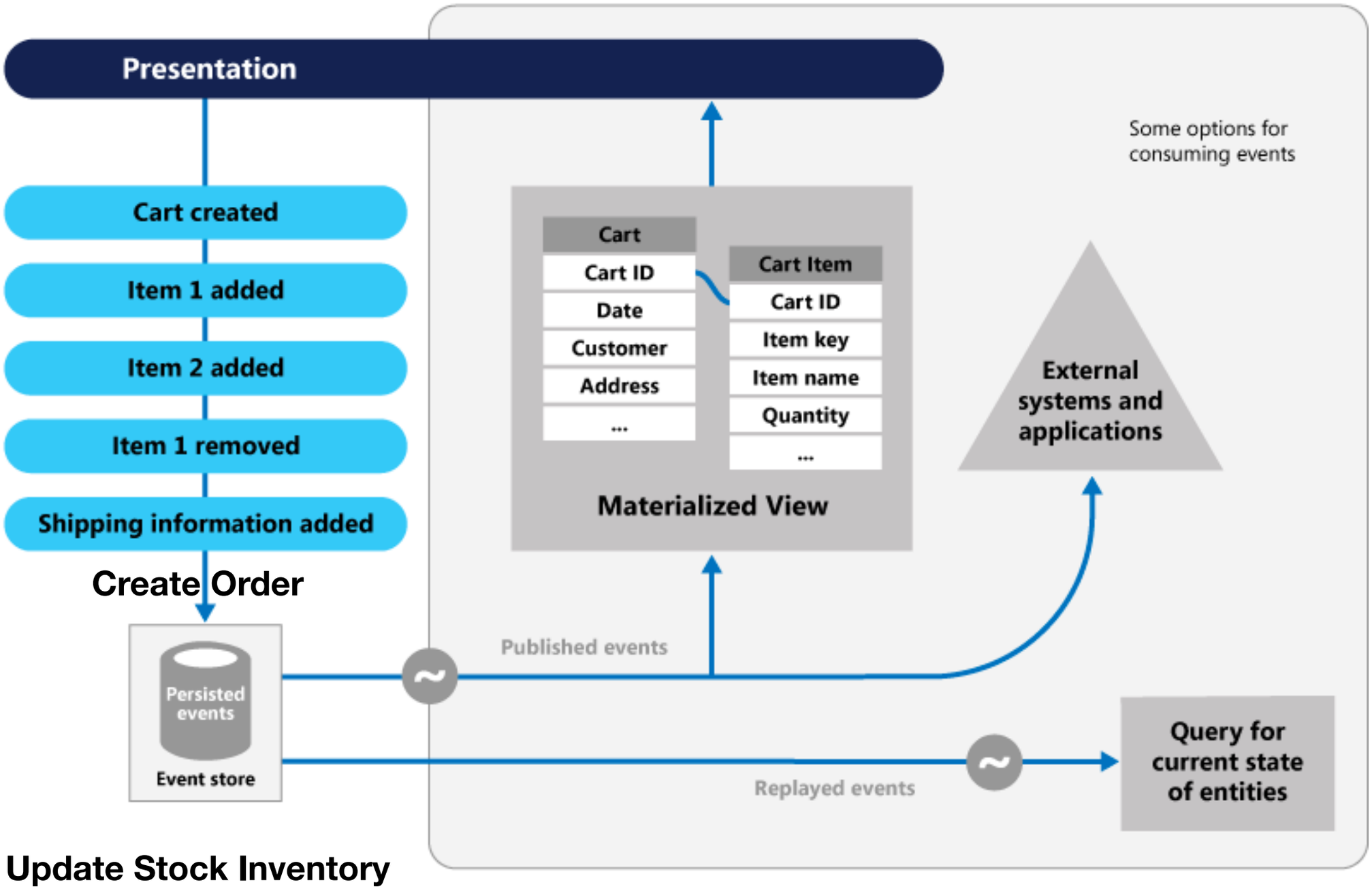
What happens when Stock Inventory is inserted first?- Back Order



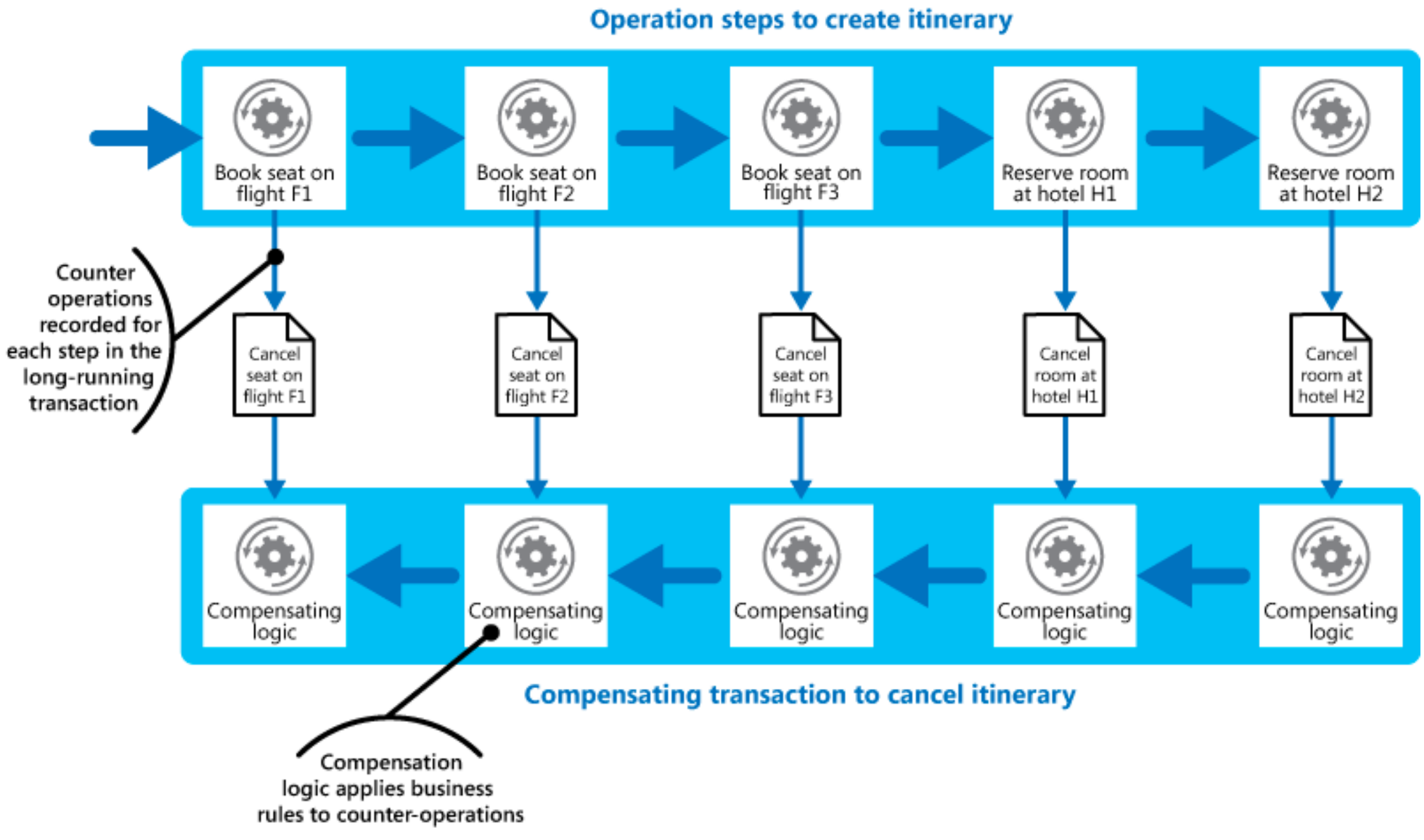
What happens when event arrives twice?- At Least Once



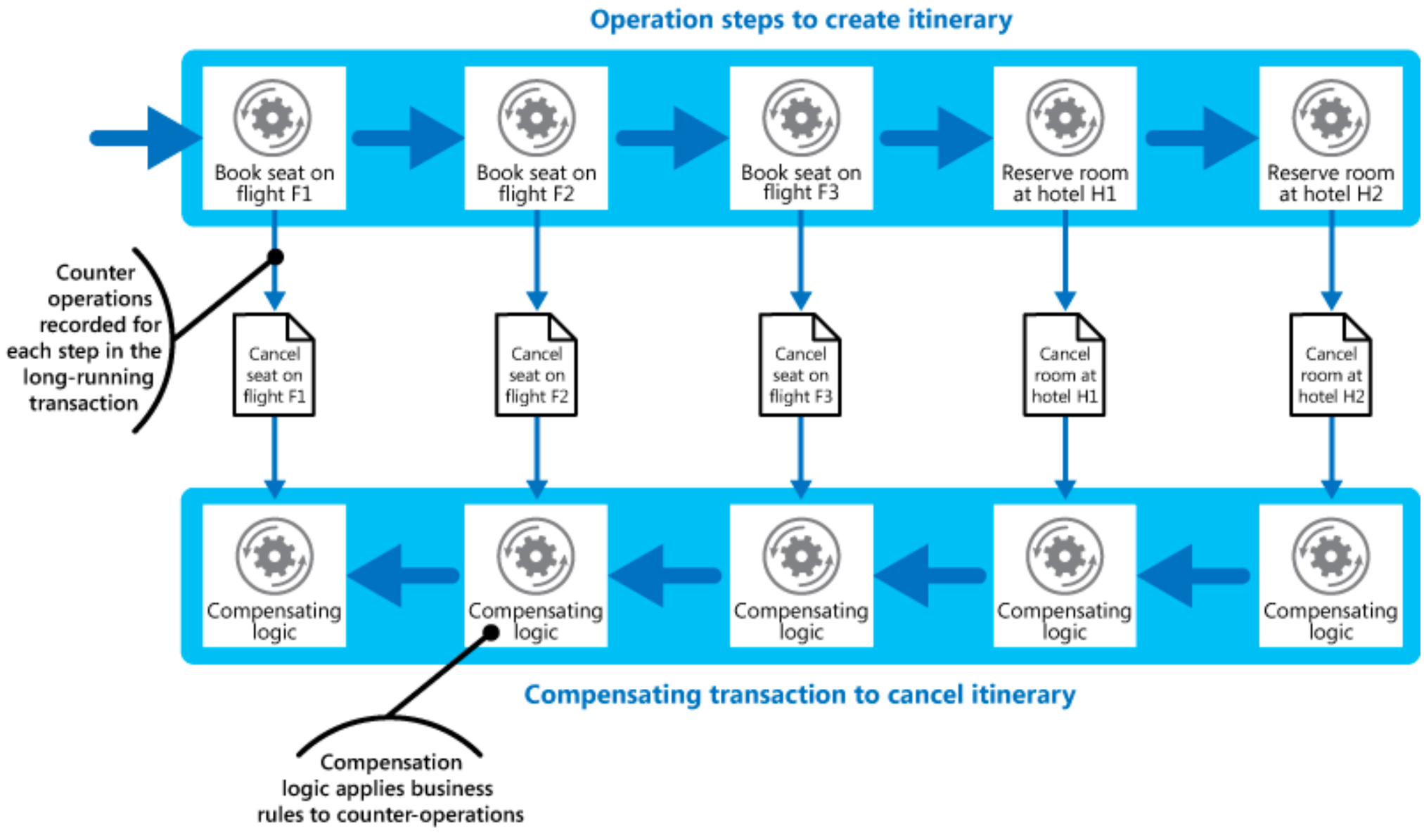
Event need to be Idempotent arrives twice- At Least Once

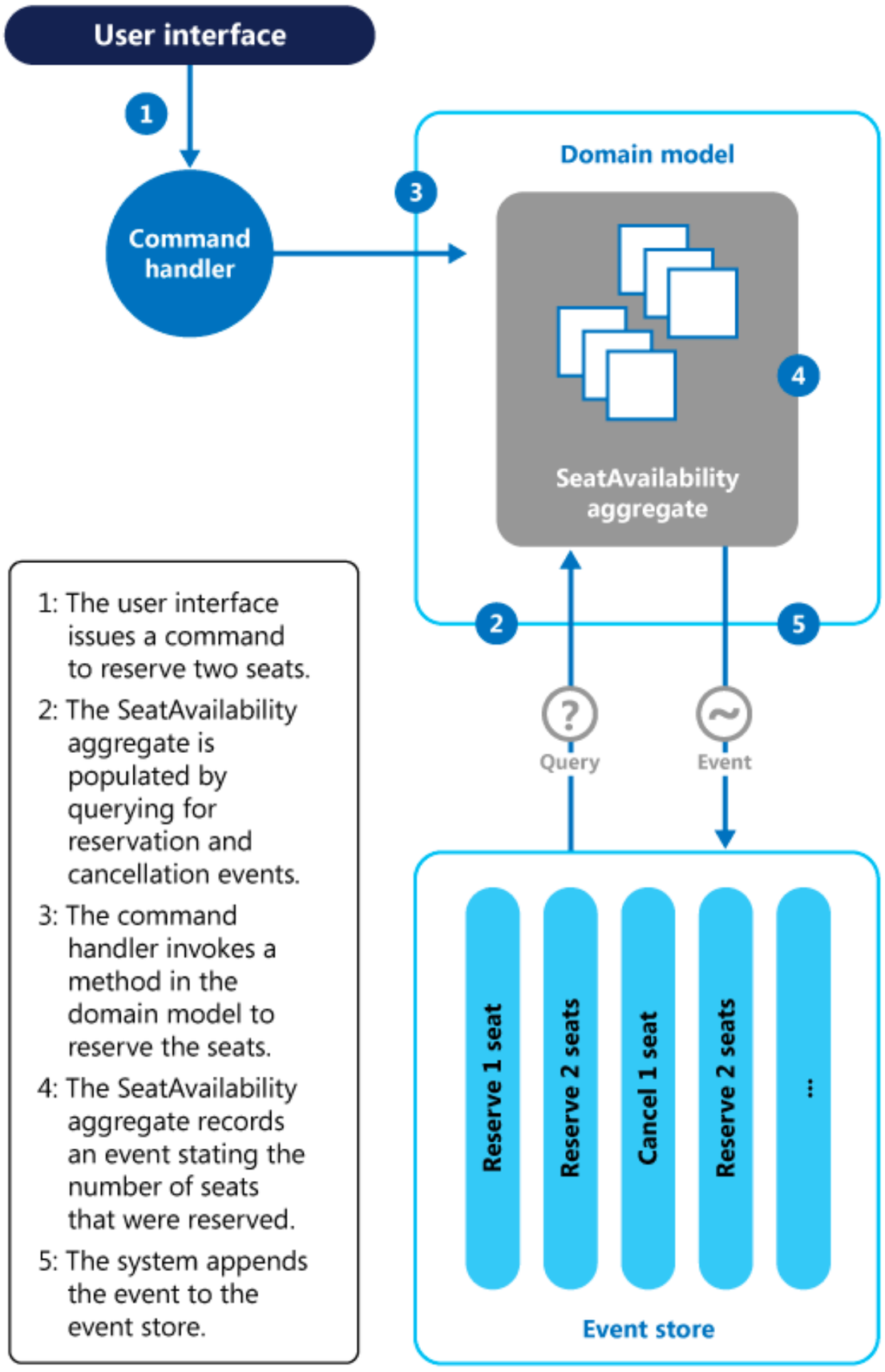


What if transaction is between two databases?



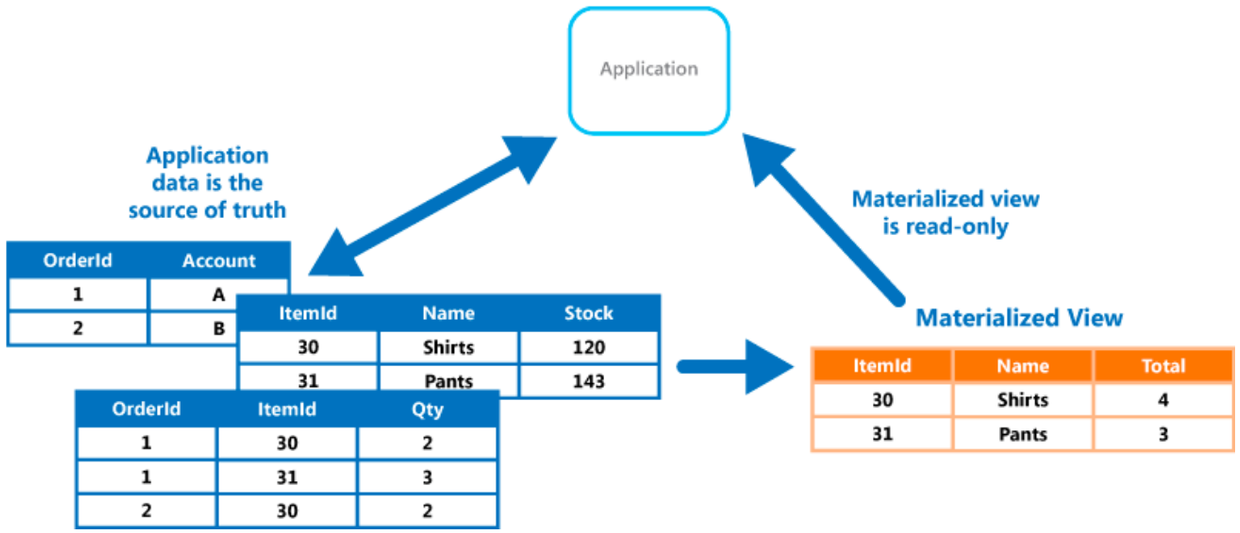
Compensation Pattern




User interface
1
Command handler
3
Domain model
SeatAvailability aggregate
4
2
5
Query
Event
Reserve 1 seat
Reserve 2 seats
Cancel 1 seat
Reserve 2 seats
...
Event store
1: The user interface issues a command to reserve two seats.
2: The SeatAvailability aggregate is populated by querying for reservation and cancellation events.
3: The command handler invokes a method in the domain model to reserve the seats.
4: The SeatAvailability aggregate records an event stating the number of seats that were reserved.
5: The system appends the event to the event store.

Materialized view

- Domain Driven Design
- Different Data stores
- Eventually consistent
- Query performance



Order table

Partition key	Row key	Order date	Shipping address	Total invoice	Order status
001 (Customer ID)	1 (Order ID)	11082013	One Microsoft way Redmond, WA 98052	$400	In process
005	2	11082013	One Microsoft way Redmond, WA 98052	$200	Shipped

OrderItem table

Partition key	Row key	Product	Unit Price	Amount	Total
1 (Order ID)	001_1 (OrderItem ID)	XX	$100	2	$200
1	001_2	YY	$40	5	$200
2	002_1	ZZ	$200	1	$200

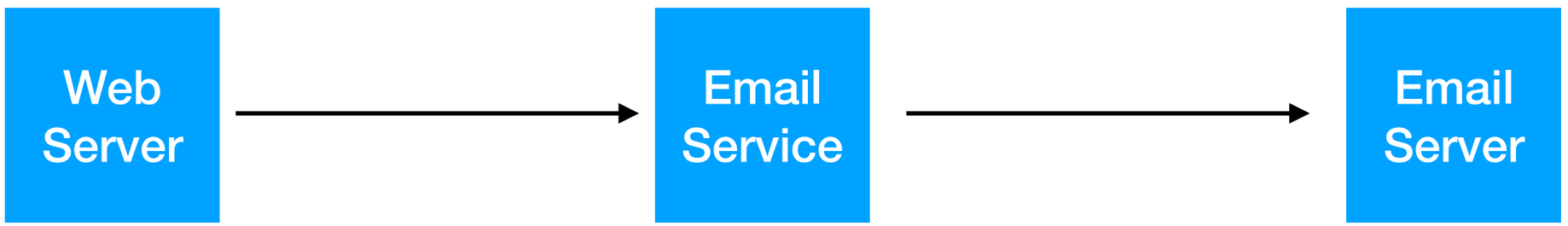
Customer table

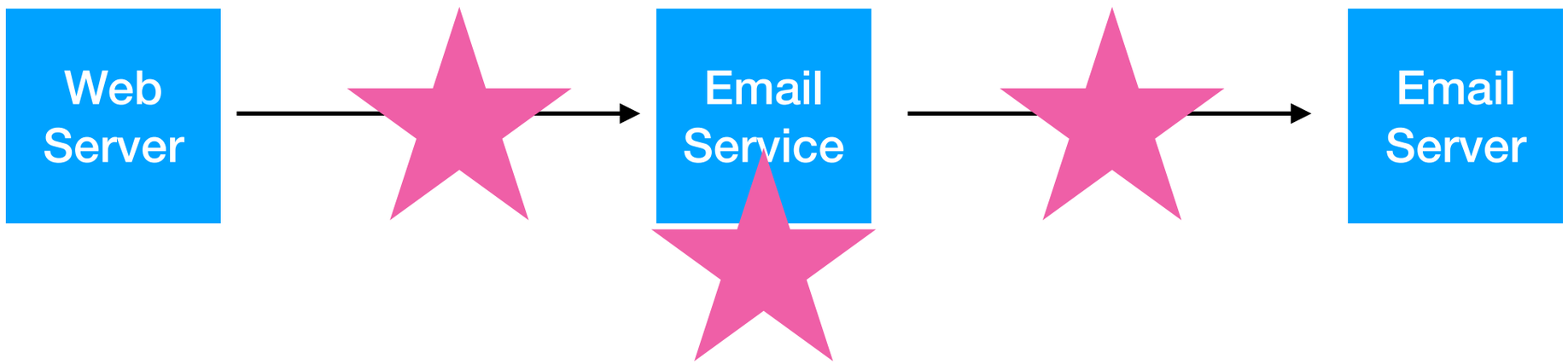
Partition key	Row key	Billing Information	Shipping address	Gender	Age
US East (region)	001 (Customer ID)	*****0001	One Microsoft way Redmond, WA 98052	Female	30
US East	002	*****2006	One Microsoft way Redmond, WA 98052	Male	40

Materialized View

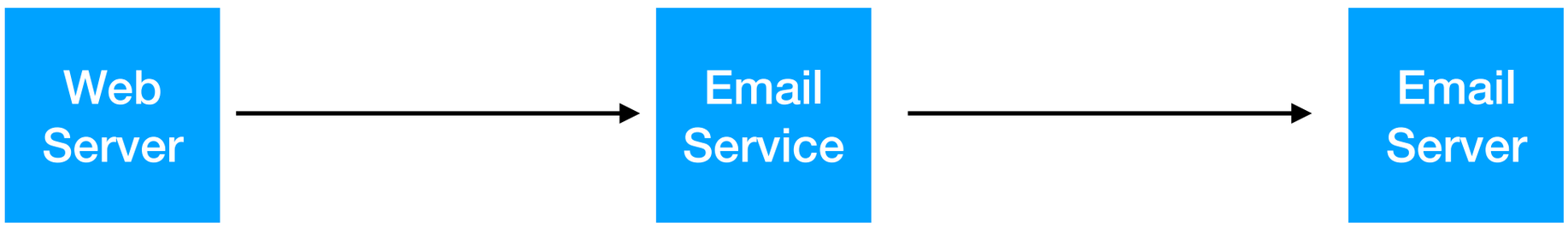
Partition key	Row key	Product Name	Total sold	Number of customers
Electronics (Product category)	001 (Product ID)	XX	$30,000	500
Electronics	002	YY	$100,000	400

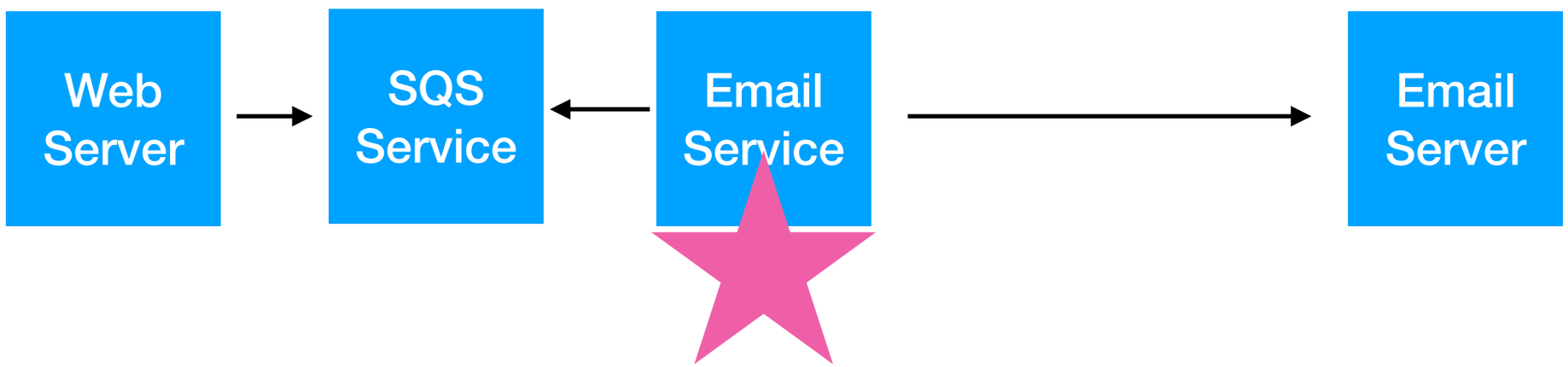
Determine how to design decoupling mechanisms using AWS services.



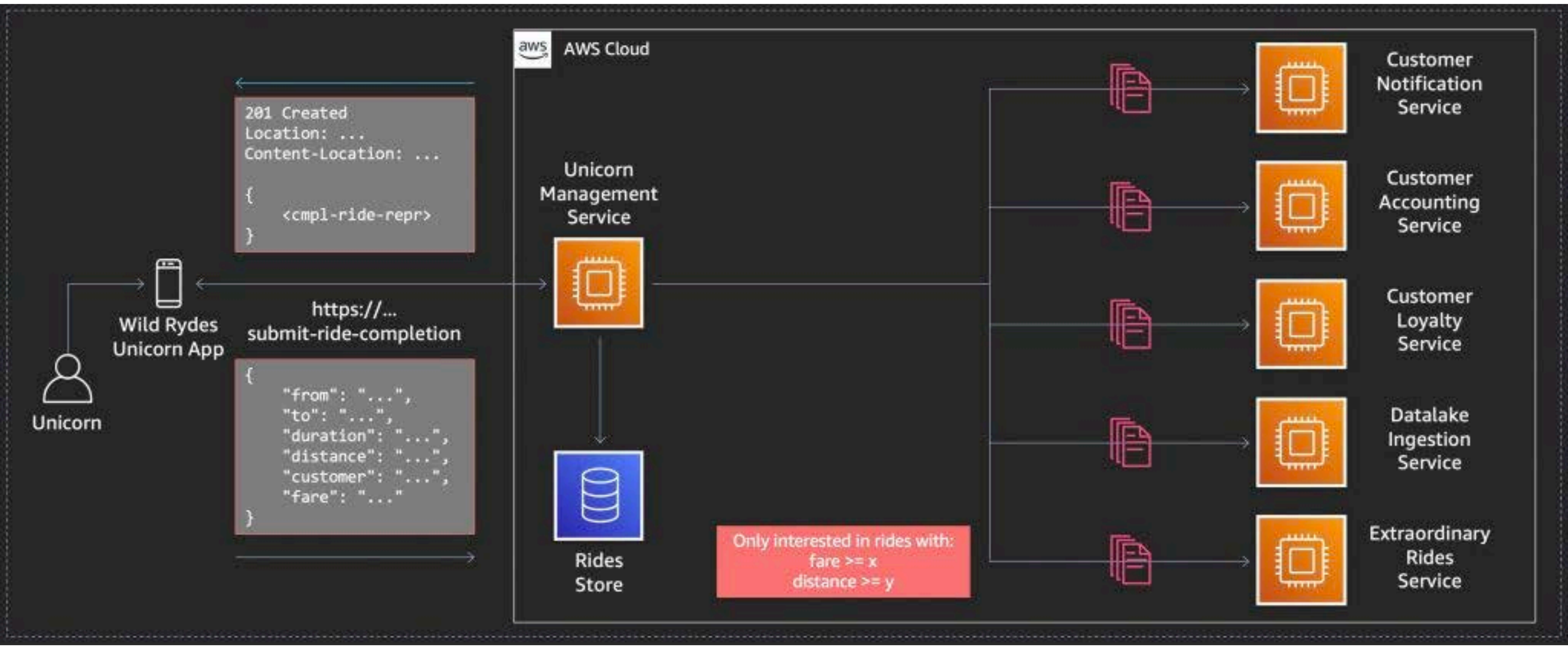


Determine how to design decoupling mechanisms using AWS services.

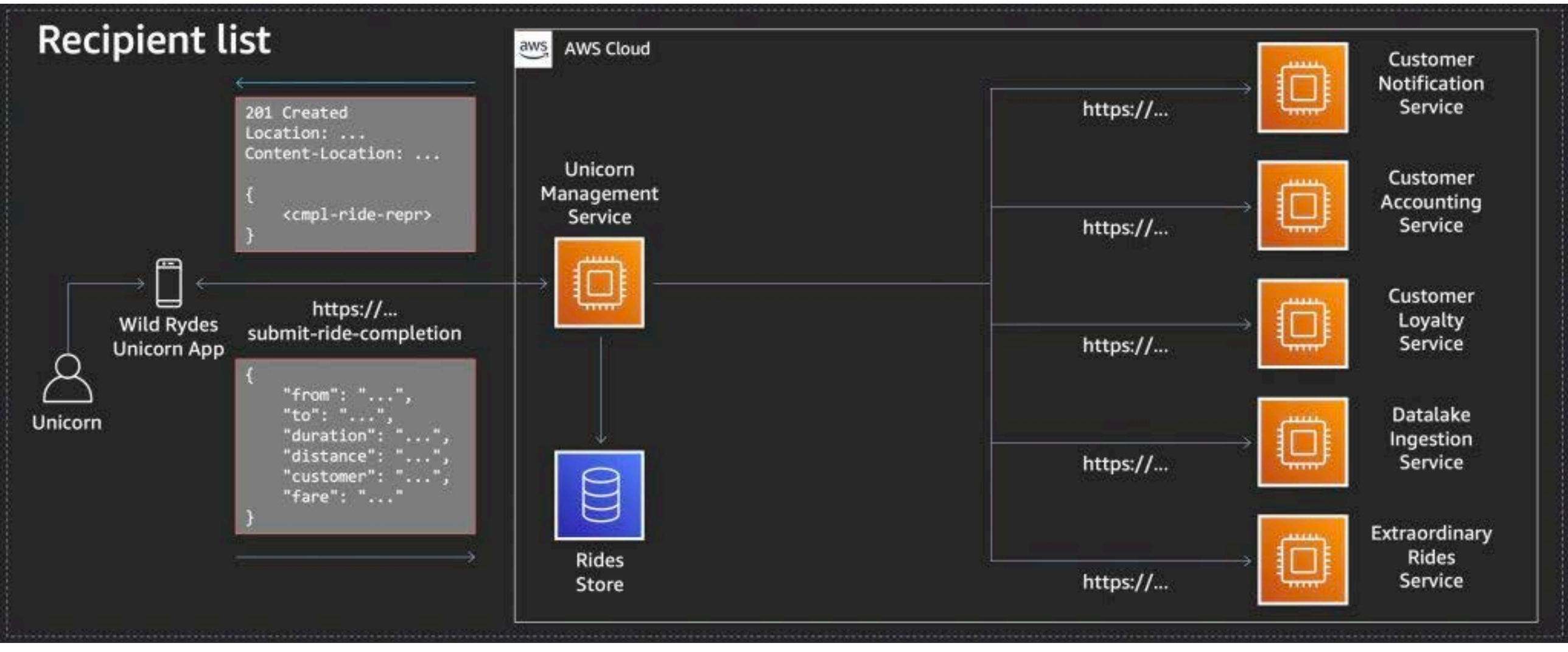




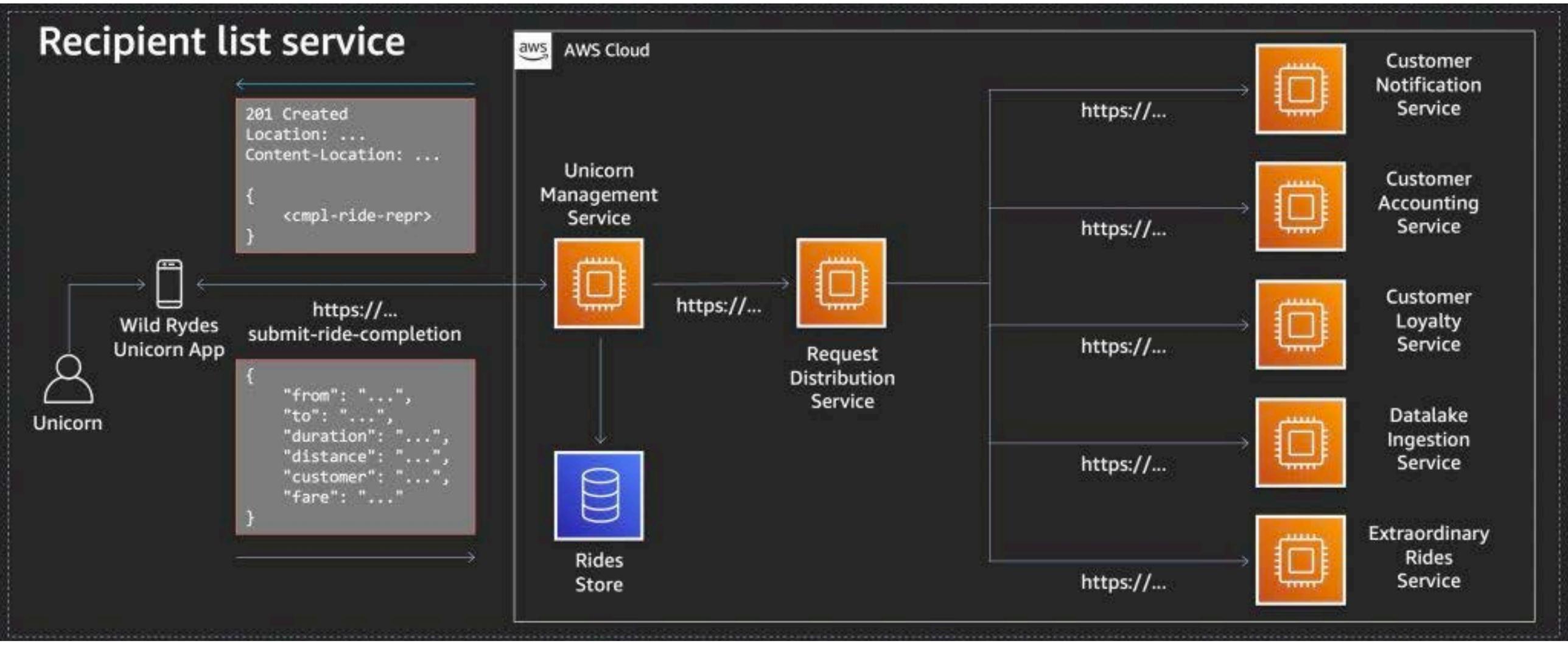
Unicorn Uber



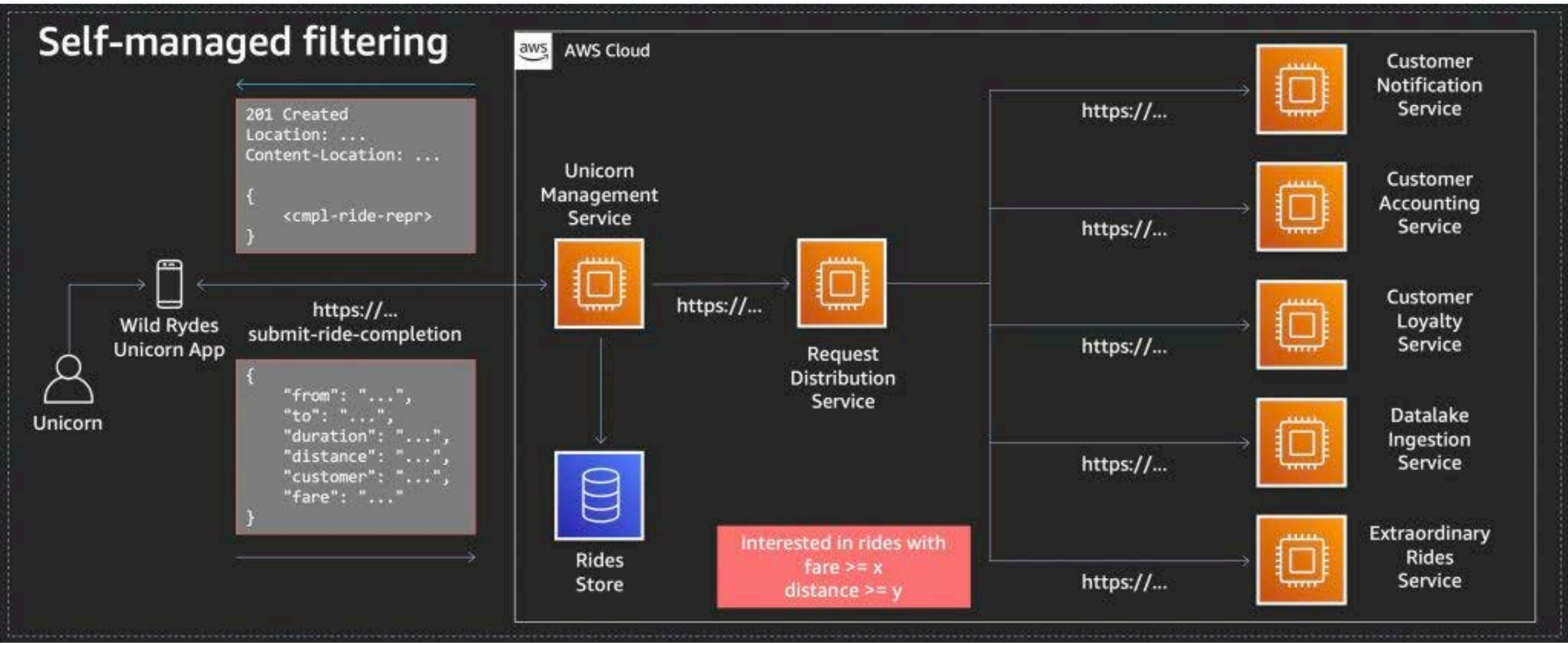
Uber Management service - Call micro services



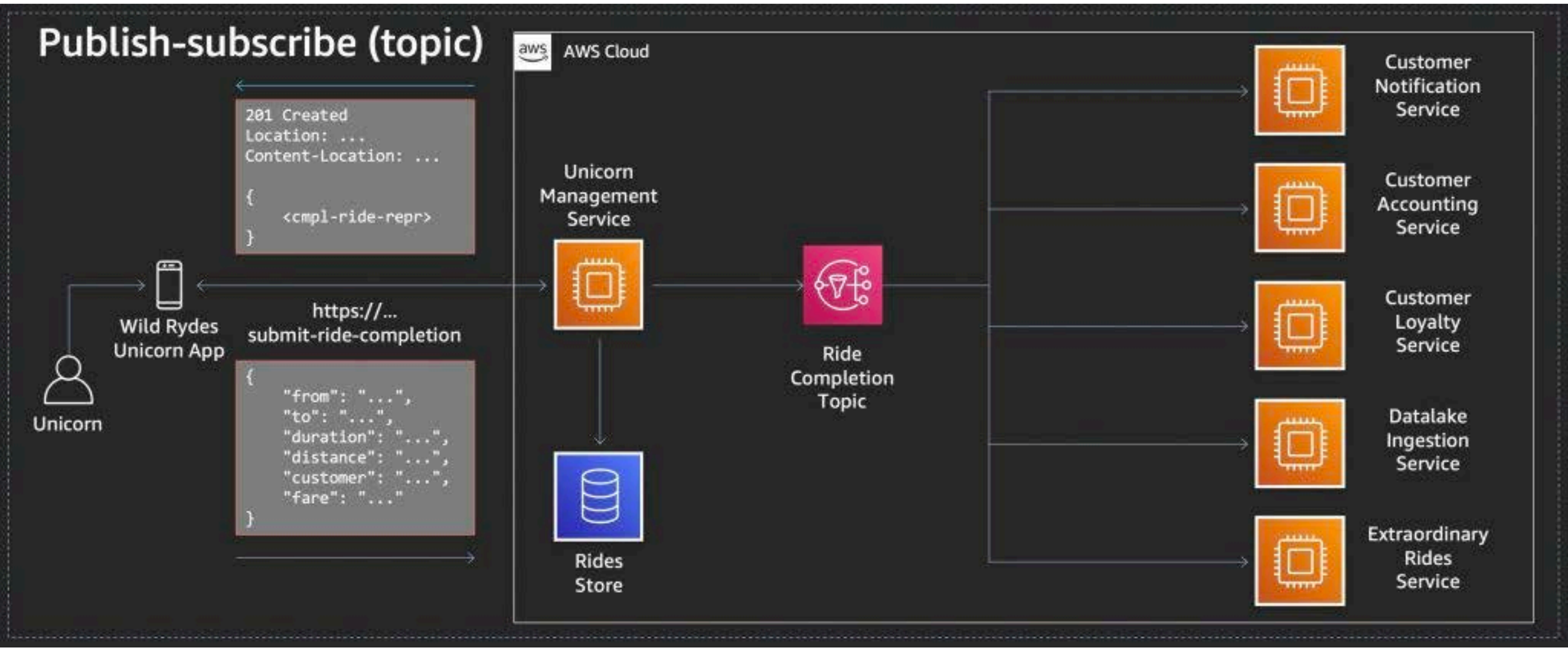
Fan out to services



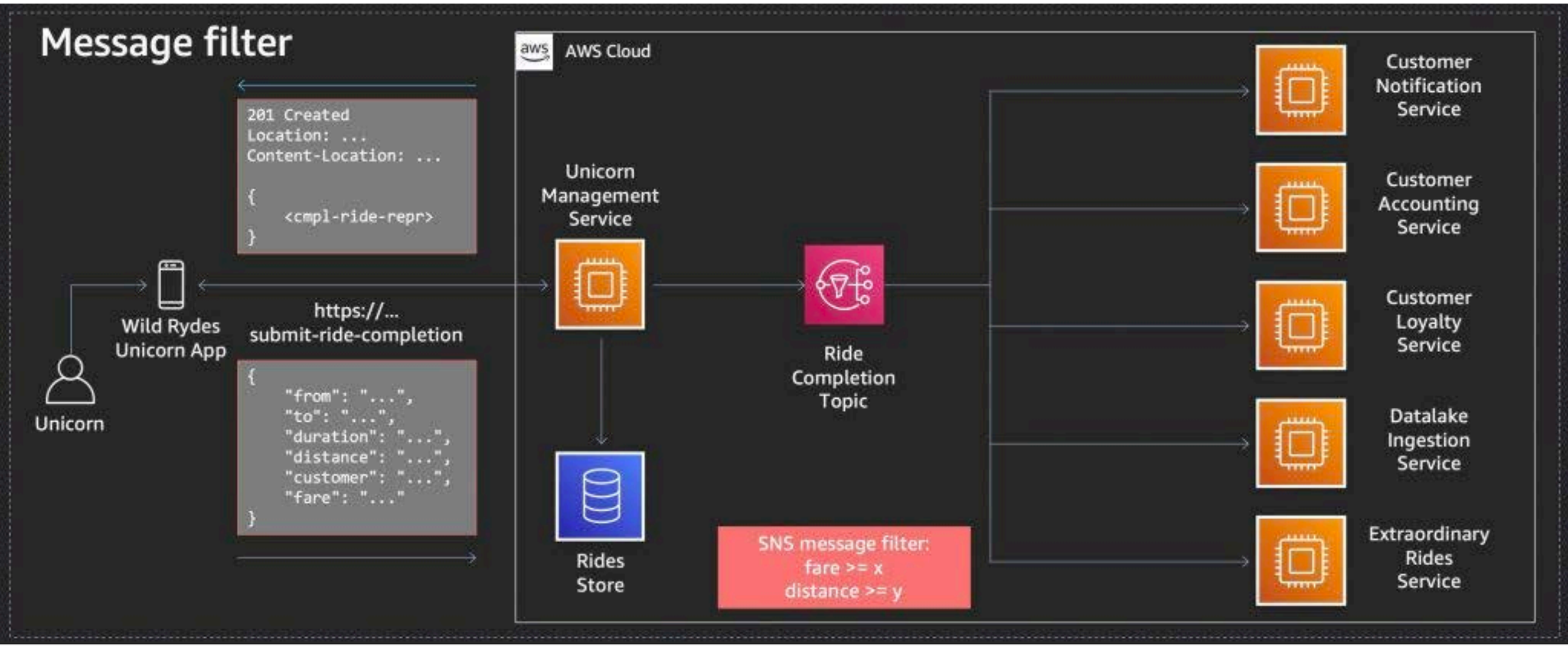
Filtering Uber rides



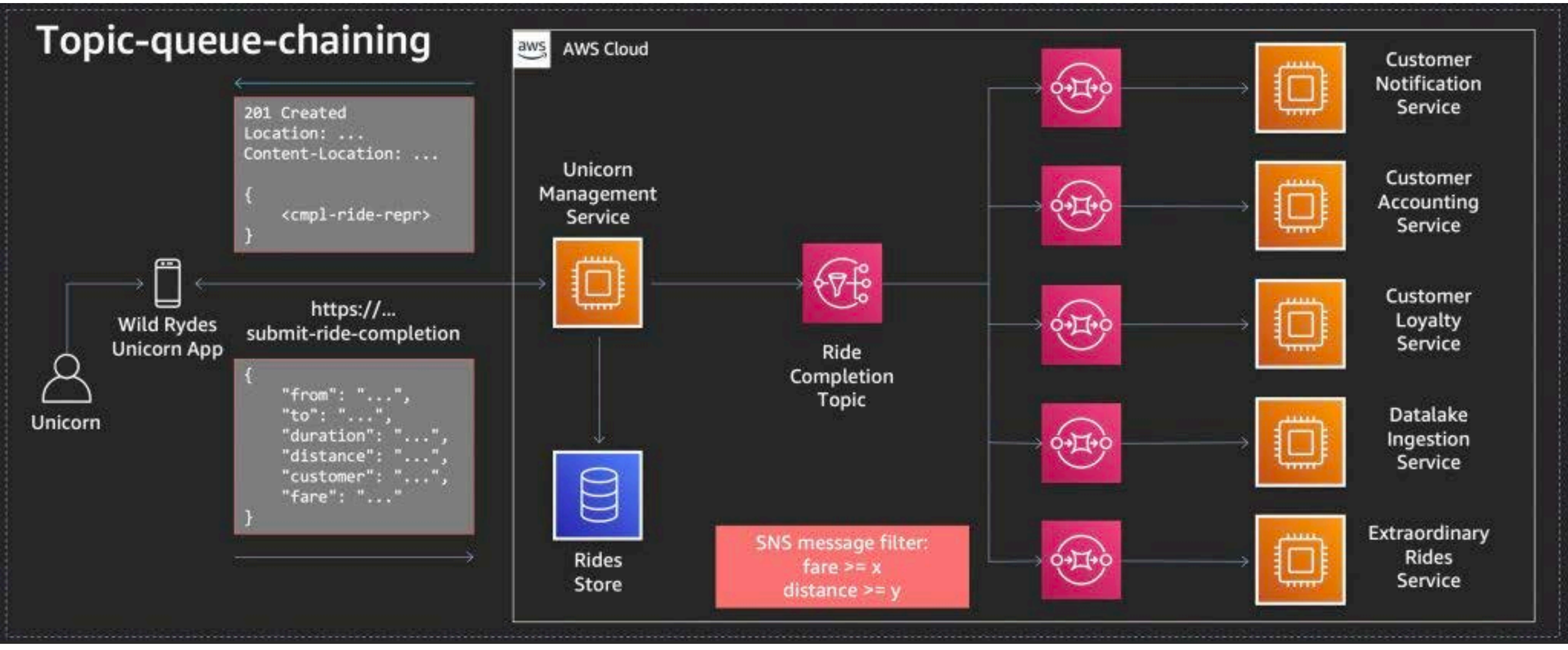
Publish-subscribe pattern



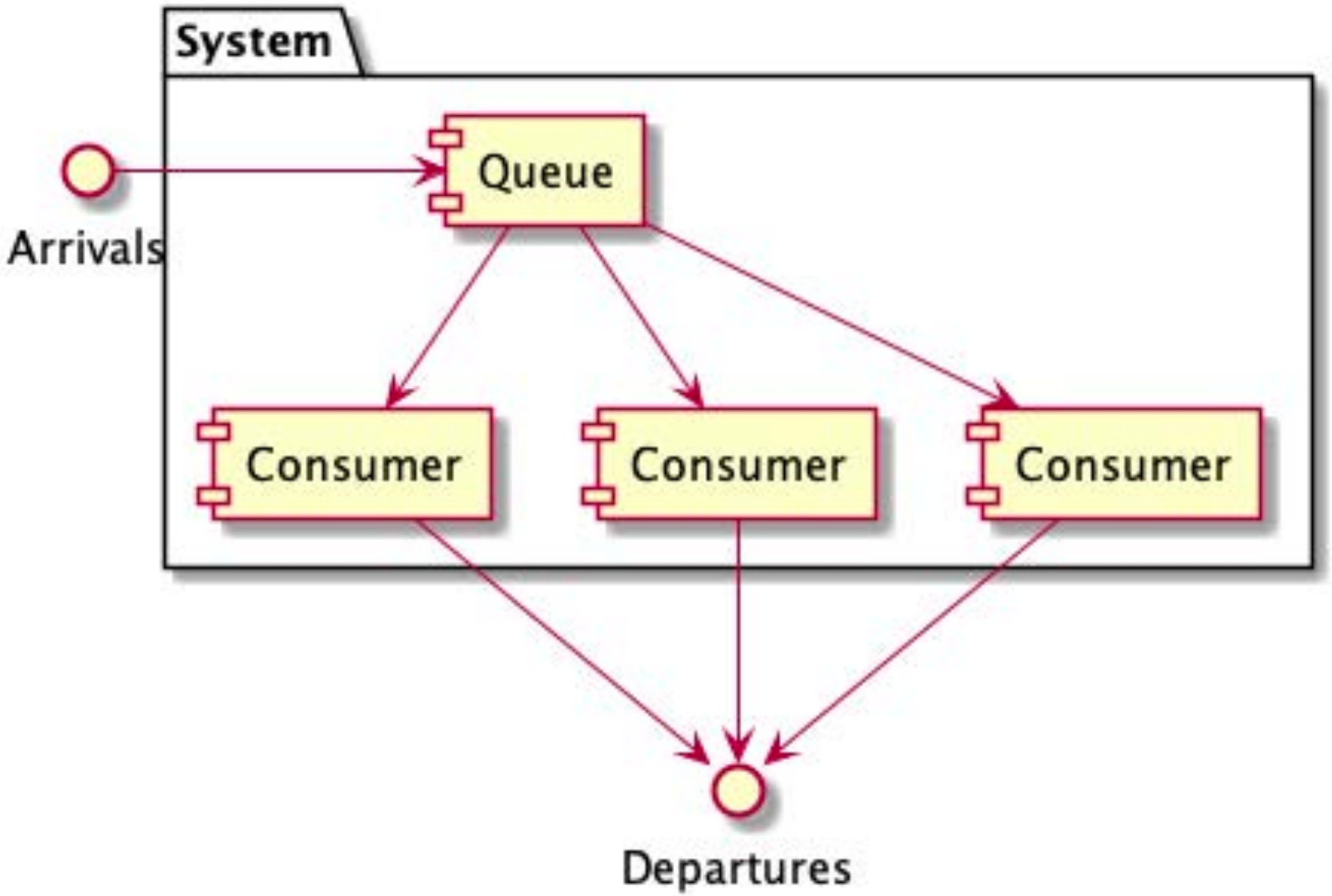
Message filter pattern

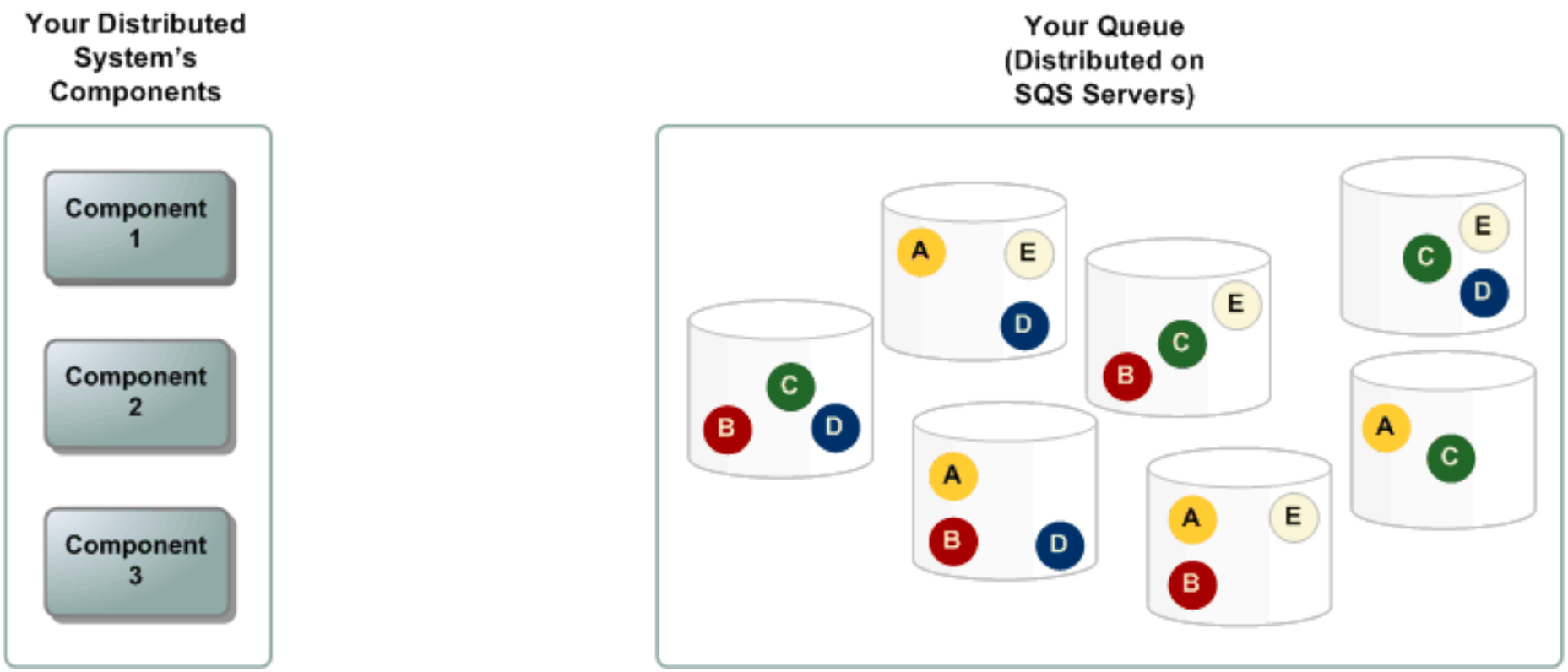


Topic-queue-chaining pattern - flatten peak loads



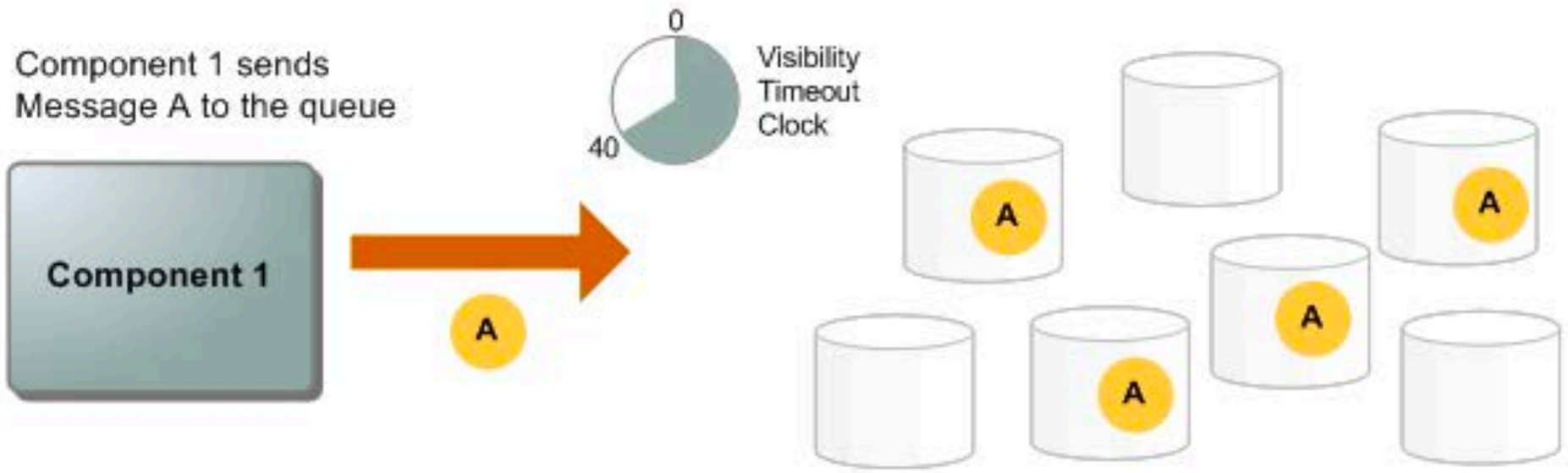
SQS


System
Queue
Arrivals
Consumer
Consumer
Consumer
Departures


Your Distributed System's Components
Component 1
Component 2
Component 3
Your Queue (Distributed on SQS Servers)
A
E
D
C
B
E
C
B
D
E
C
D
A
B
D
A
E
B
A
C

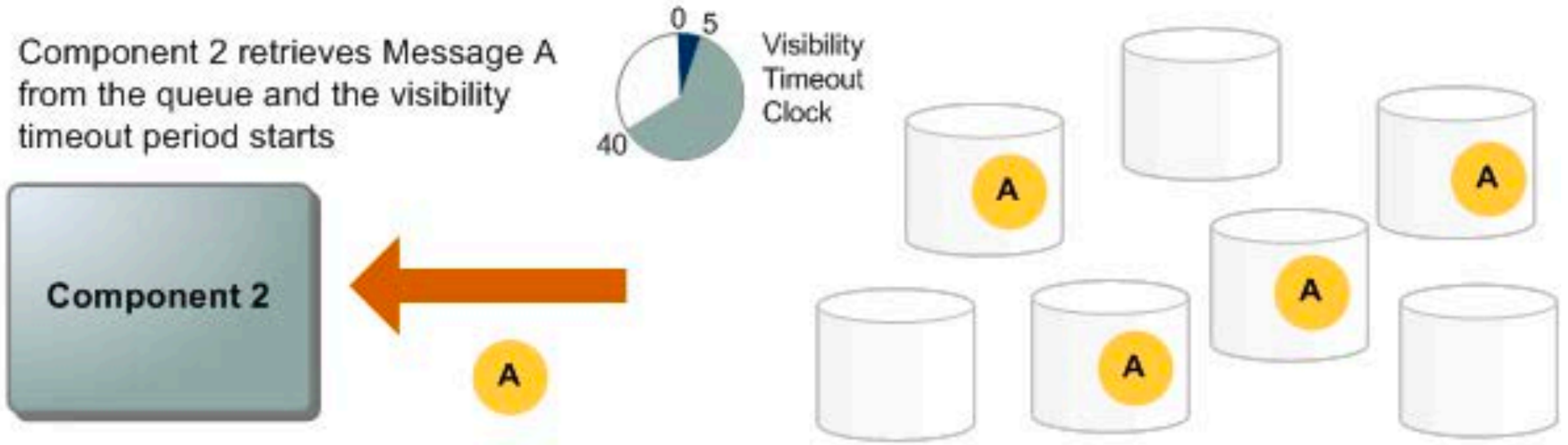
1

Component 1 sends
Message A to the queue



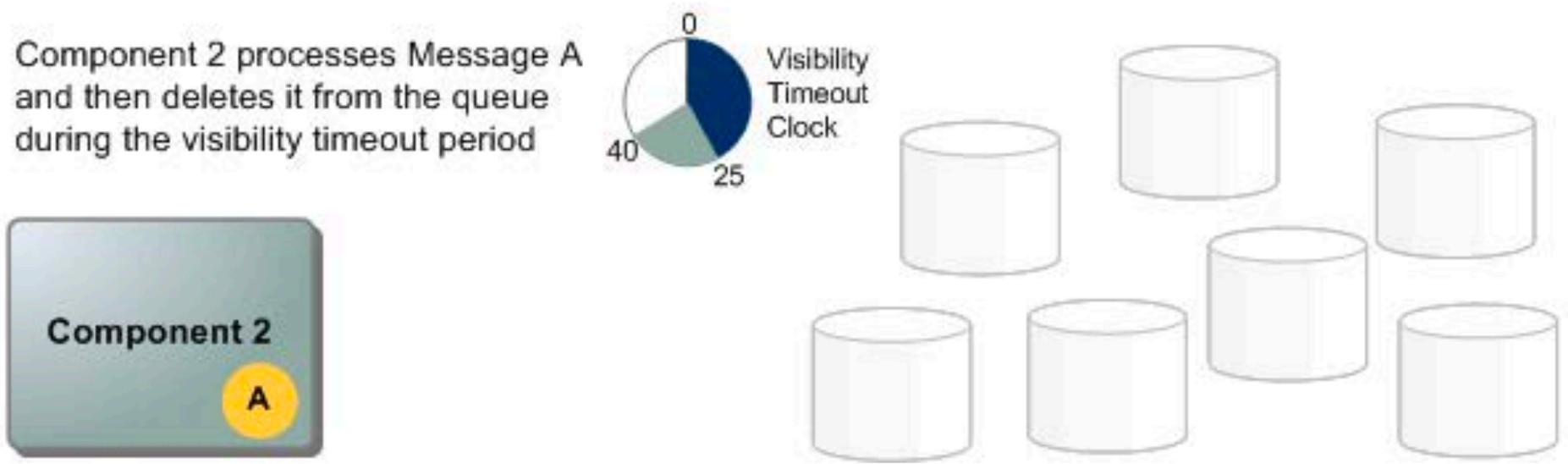
2

Component 2 retrieves Message A
from the queue and the visibility
timeout period starts



3

Component 2 processes Message A
and then deletes it from the queue
during the visibility timeout period



Standard Queue

- Best-effort ordering
- Duplicate messages

Usecases	
Live Users uploads	Media processing
Task allocations	Credit card validation requests
Batch processing	Scheduled job

FIFO Queue

- Right ordering
- Exactly-once processing - SHA-256 hash or dedup unique id

Where can we use FIFO?

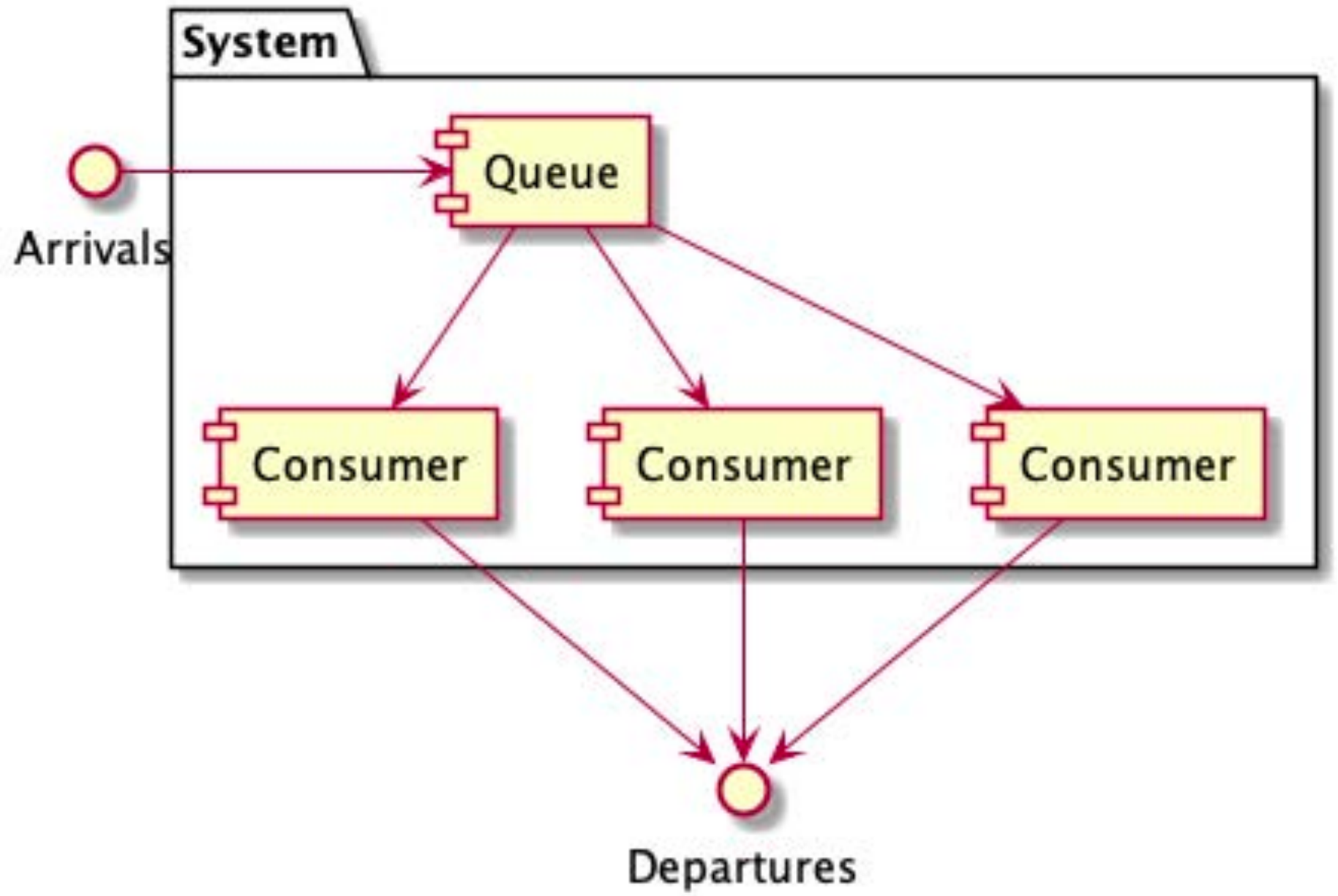
FIFO Queue

- Right ordering
- Exactly-once processing - SHA-256 hash or dedup unique id

- To make sure that user-entered commands are run in the right order.
- To display the correct product price by sending price modifications in the right order.
- To prevent a student from enrolling in a course before registering for an account.

How to calculate response time?

Little’s Law to Measure Response Time



https://sookocheff.com/post/modeling/littles-law/

Little's Law

$$L = \lambda W \rightarrow W = \frac{L}{\lambda}$$

where,

$$\begin{aligned} W &= \text{response time} \\ L &= \text{work in progress} \\ \lambda &= \text{throughput} \end{aligned}$$

With three workers

$$\text{response time} = \frac{\text{work in progress}}{\text{throughput}}$$

$$\mathrm{W} = \frac{L}{\lambda}$$

$$\text{response time} = \frac{10}{30} \text{ seconds}$$

$$\text{response time} = \frac{1}{3} \text{ seconds}$$

Predicting for Black Friday

$$\text{response time} = \frac{\text{work in progress}}{\text{throughput}}$$

$$\mathrm{W} = \frac{L}{\lambda}$$

$$\text{response time} = \frac{50}{30} \text{ seconds}$$

$$\text{response time} = 1\frac{2}{3} \text{ seconds}$$

Planning for Black Friday

$$\text{response time} = \frac{\text{work in progress}}{\text{throughput}}$$

$$\text{W} = \frac{L}{\lambda}$$

$$\text{response time} = \frac{50}{50} \text{ seconds}$$

$$\text{response time} = 1 \text{ second}$$

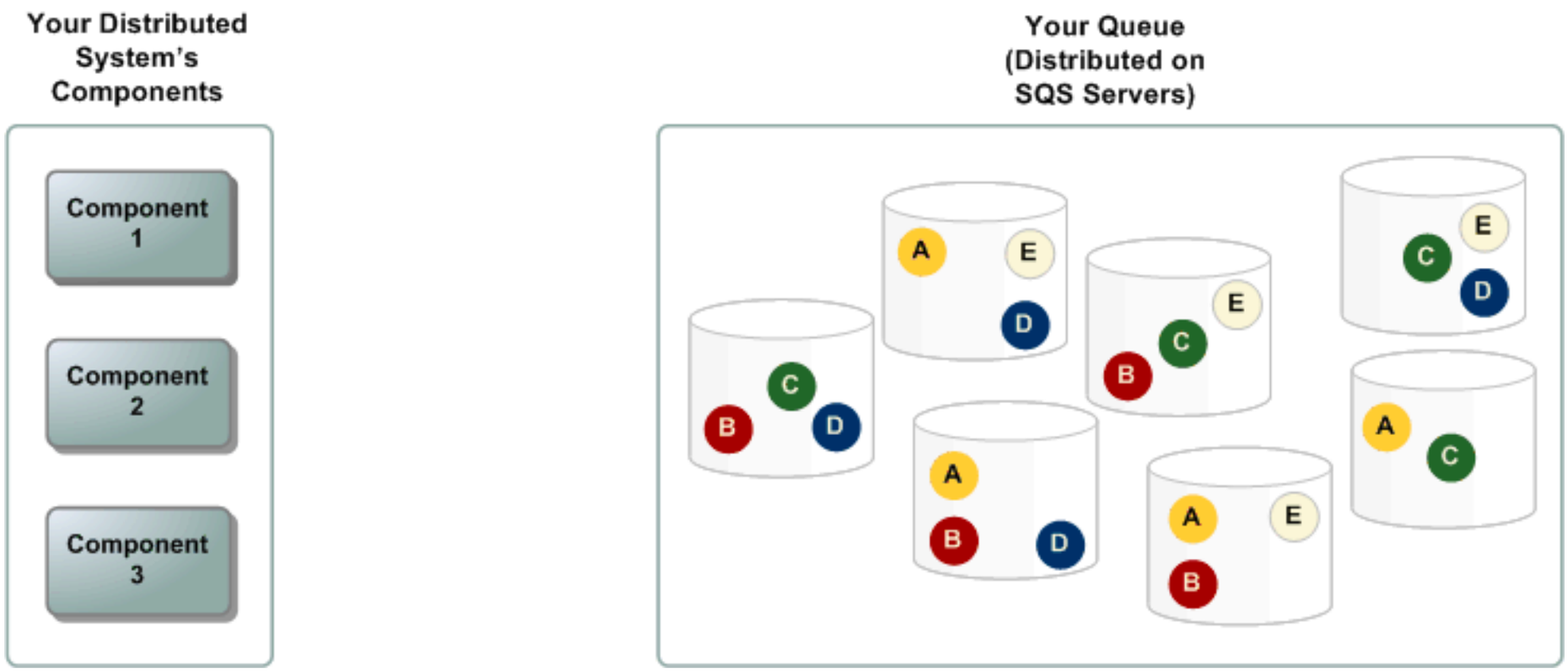
What else we can do to optimize?

Data Partitioning while it is moving

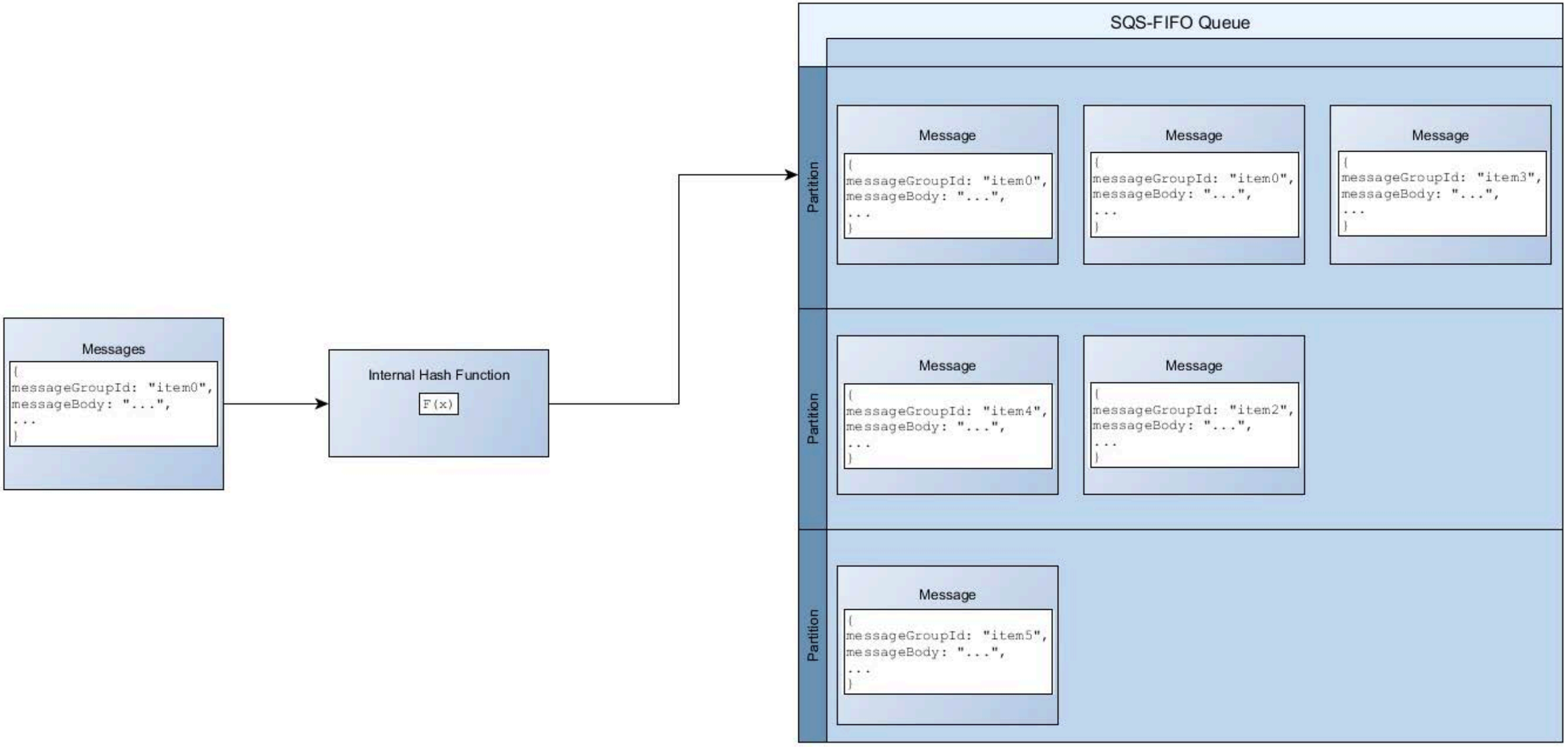
- How to partition data?

- Can we convert Standard Queue to FIFO?

Partitioning a Queue



Group IDs


Messages
{
messageGroupId: "item0",
messageBody: "...",
...
}
Internal Hash Function
F(x)
SQS-FIFO Queue
Partition
Message
{
messageGroupId: "item0",
messageBody: "...",
...
}
Message
{
messageGroupId: "item0",
messageBody: "...",
...
}
Message
{
messageGroupId: "item3",
messageBody: "...",
...
}
Partition
Message
{
messageGroupId: "item4",
messageBody: "...",
...
}
Message
{
messageGroupId: "item2",
messageBody: "...",
...
}
Partition
Message
{
messageGroupId: "item5",
messageBody: "...",
...
}

Message Batch

```
{
messageGroupId: "item0",
messageBody: "...",
...
}
```

```
{
messageGroupId: "item0",
messageBody: "...",
...
}
```

```
{
messageGroupId: "item1",
messageBody: "...",
...
}
```

```
{
messageGroupId: "item2",
messageBody: "...",
...
}
```

```
{
messageGroupId: "item3",
messageBody: "...",
...
}
```

Internal Hash Function

F(x)

Messages

```
{
messageGroupId: "item0",
messageBody: "...",
...
}
```

```
{
messageGroupId: "item0",
messageBody: "...",
...
}
```

```
{
messageGroupId: "item3",
messageBody: "...",
...
}
```

1 Transaction

Messages

```
{
messageGroupId: "item2",
messageBody: "...",
...
}
```

1 Transaction

Messages

```
{
messageGroupId: "item1",
messageBody: "...",
...
}
```

1 Transaction

SQS-FIFO Queue

Partition

Message

```
{
messageGroupId: "item5",
messageBody: "...",
...
}
```

Message

```
{
messageGroupId: "item0",
messageBody: "...",
...
}
```

Message

```
{
messageGroupId: "item3",
messageBody: "...",
...
}
```

Partition

Message

```
{
messageGroupId: "item4",
messageBody: "...",
...
}
```

Message

```
{
messageGroupId: "item2",
messageBody: "...",
...
}
```

Partition

Message

```
{
messageGroupId: "item1",
messageBody: "...",
...
}
```

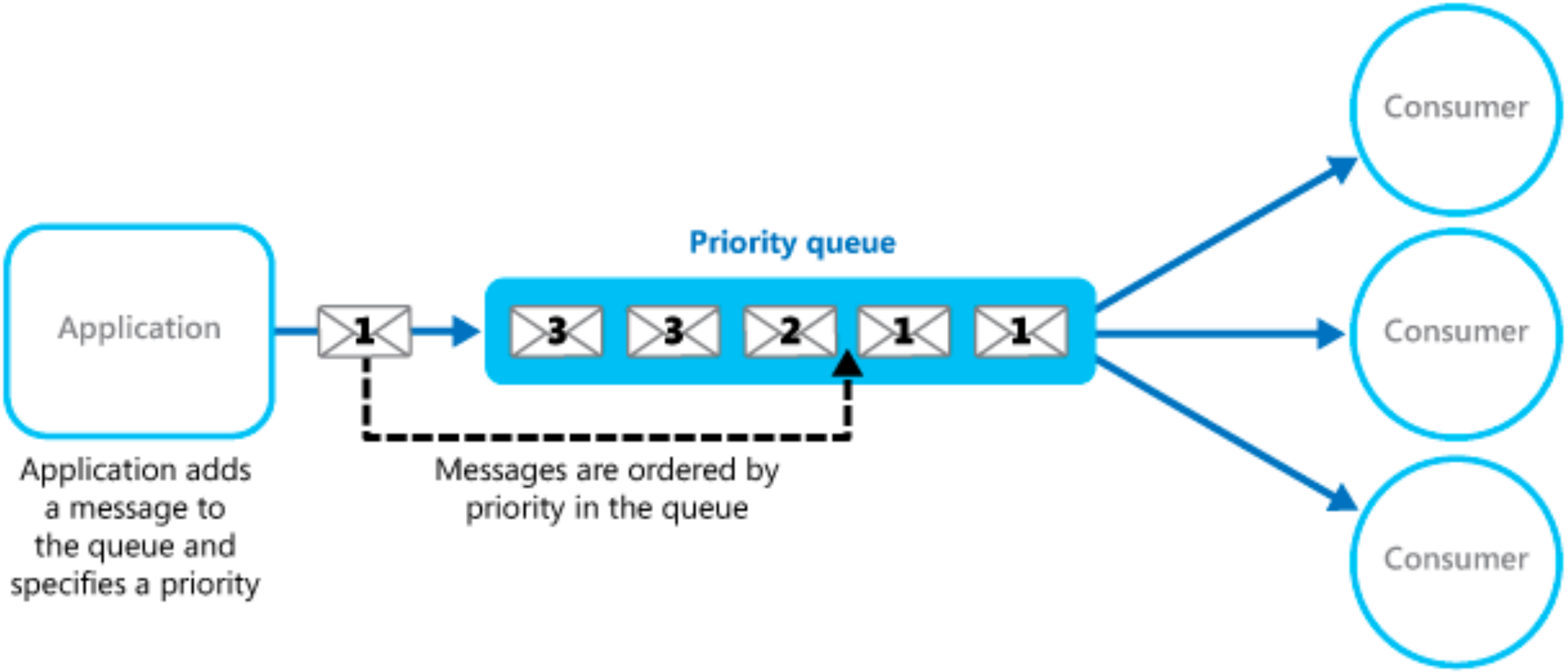
- (SQS) You are designing a new app which processes payments and delivers promotional emails to customers. Your need to ensure that the payment process takes priority over the creation of emails. What is the best way to achieve this?

- (SQS) You are designing a new app which processes payments and delivers promotional emails to customers. Your need to ensure that the payment process takes priority over the creation of emails. What is the best way to achieve this?

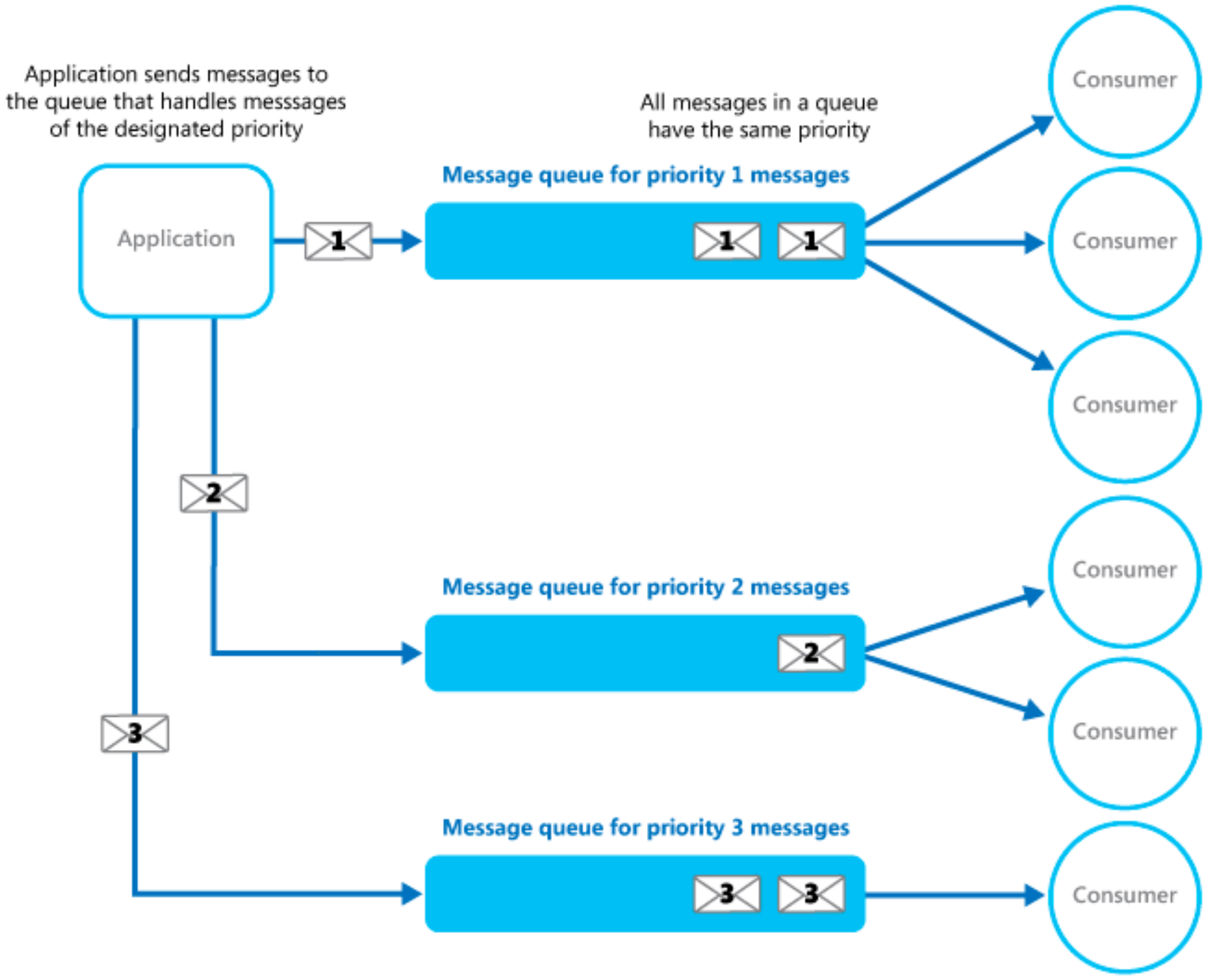
- Use 2 SQS queues. Poll that payment queue first.

What if you have a VIP customer? How to design Queue ?

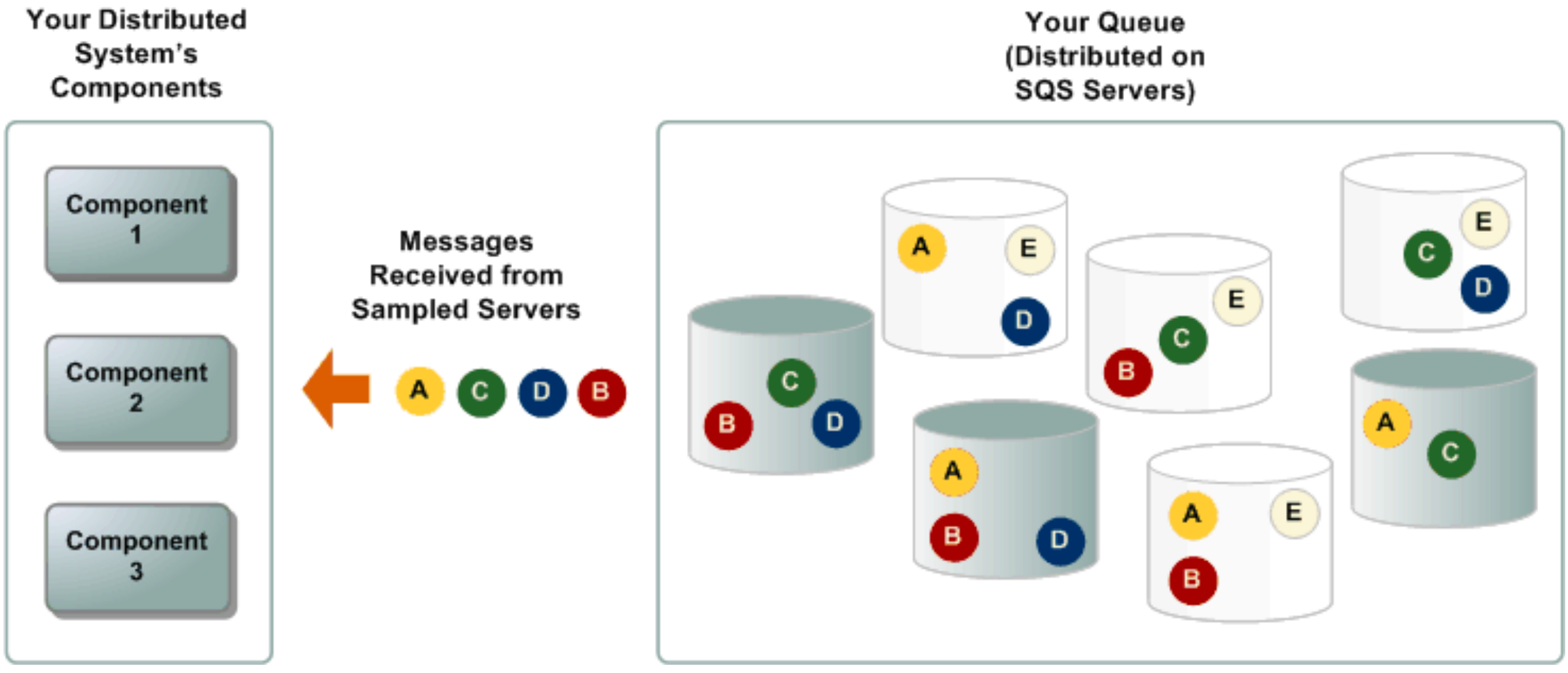
Priority Queue pattern



What if Priority Queue is not possible?

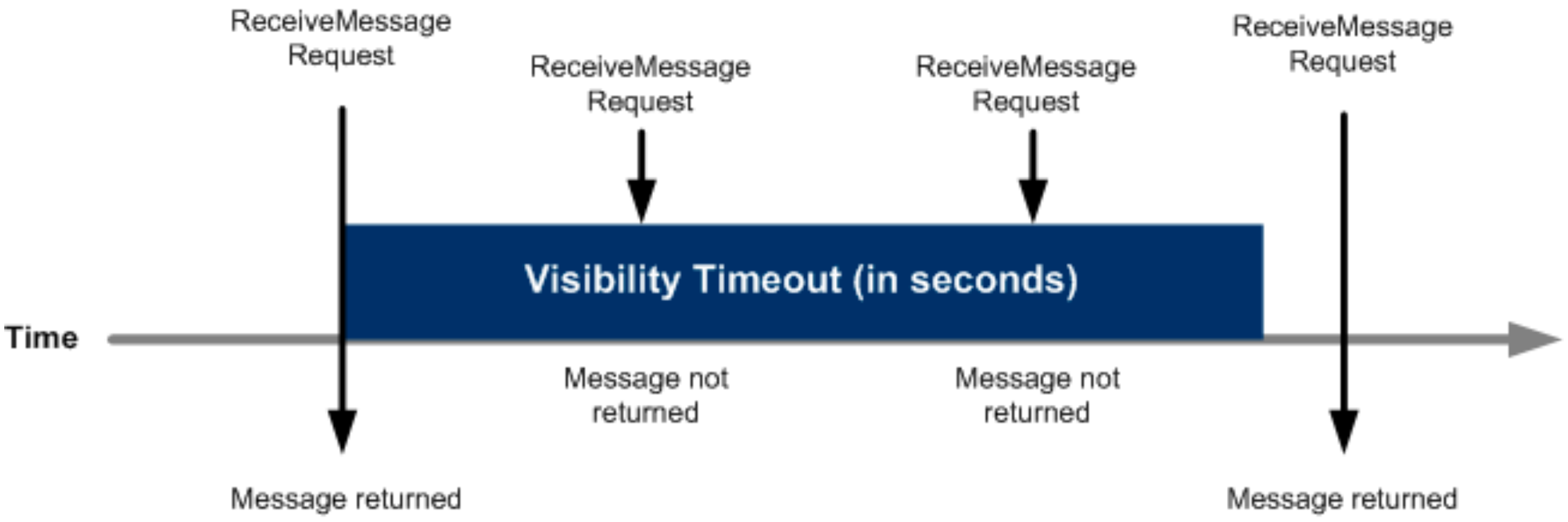

Application sends messages to
the queue that handles messsages
of the designated priority
All messages in a queue
have the same priority
Application
1
Message queue for priority 1 messages
1
1
Consumer
Consumer
Consumer
2
Message queue for priority 2 messages
2
Consumer
Consumer
3
Message queue for priority 3 messages
3
3
Consumer

Short/Long Polling



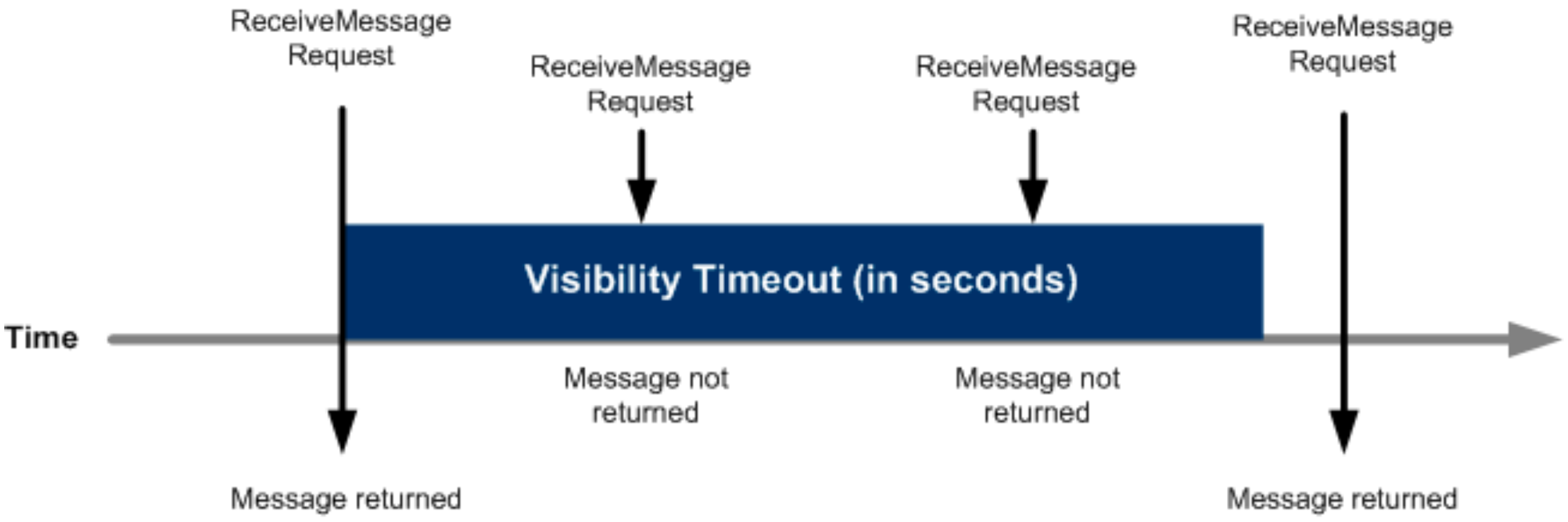
Long Polling-> `ReceiveMessage` API action is greater than 0
Avoid Empty messages, return messages asap when available

Amazon SQS visibility timeout



- A visibility timeout is a period of time during which AWS SQS prevents other consuming components from receiving and processing the message?

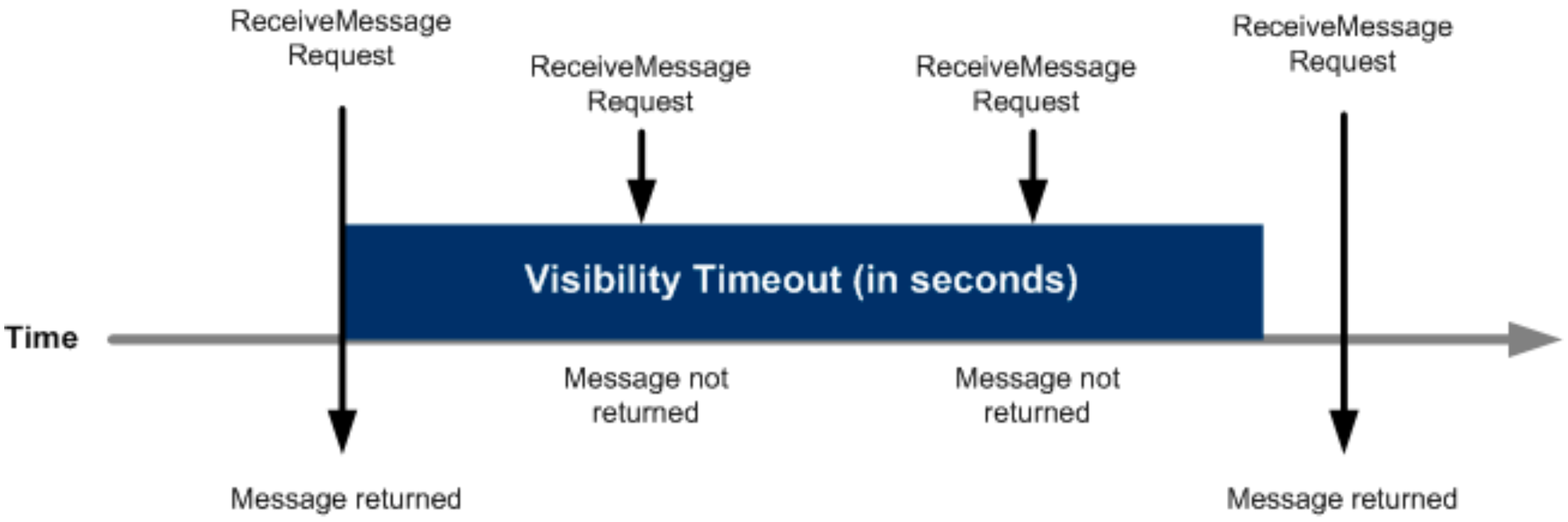
Amazon SQS visibility timeout



What are lifecycle steps for a message?

- A visibility timeout is a period of time during which AWS SQS prevents other consuming components from receiving and processing the message?

Amazon SQS visibility timeout

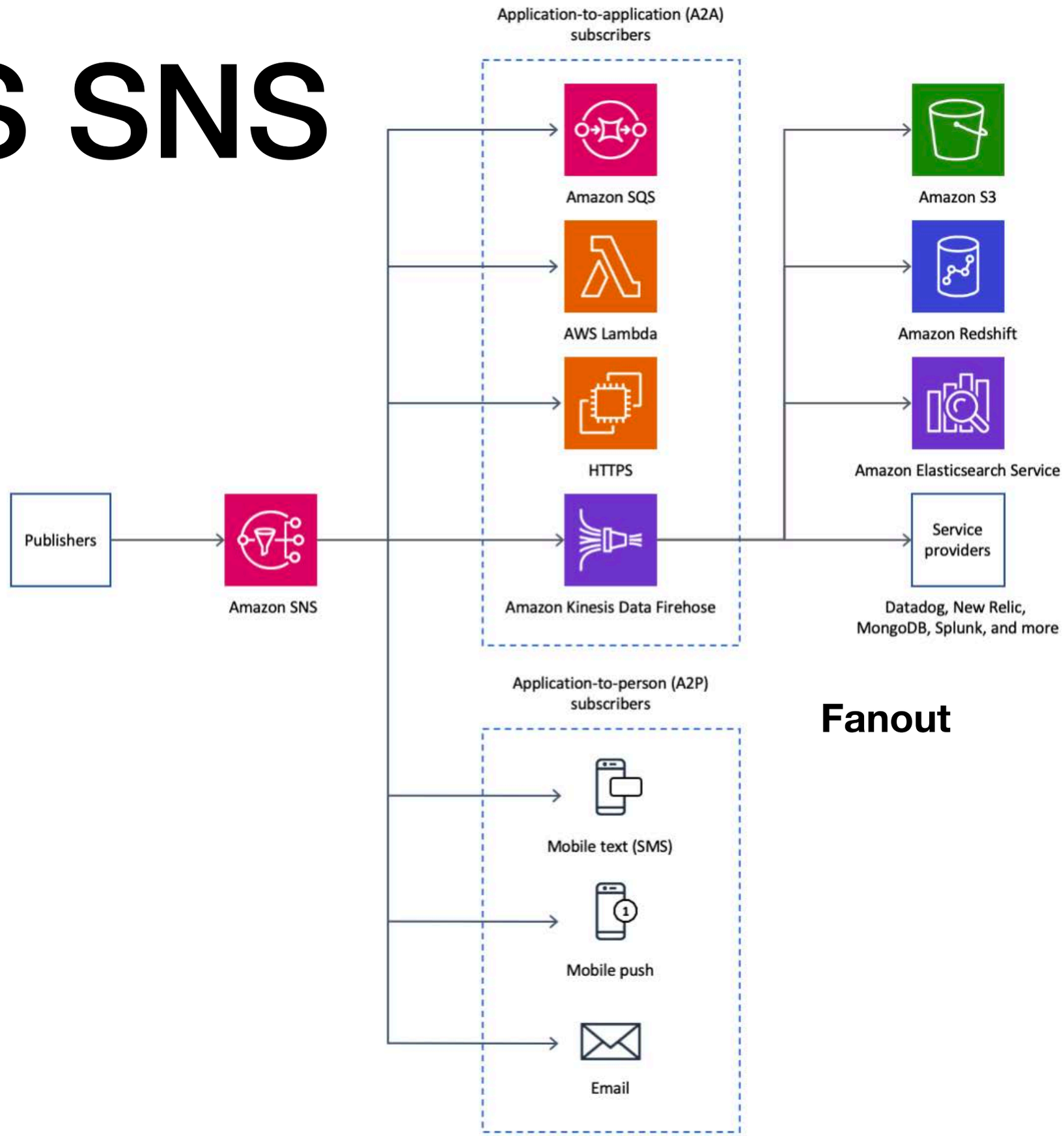


1. Sent to a queue by a producer.
2. Received from the queue by a consumer.
3. Deleted from the queue.

- A visibility timeout is a period of time during which AWS SQS prevents other consuming components from receiving and processing the message?

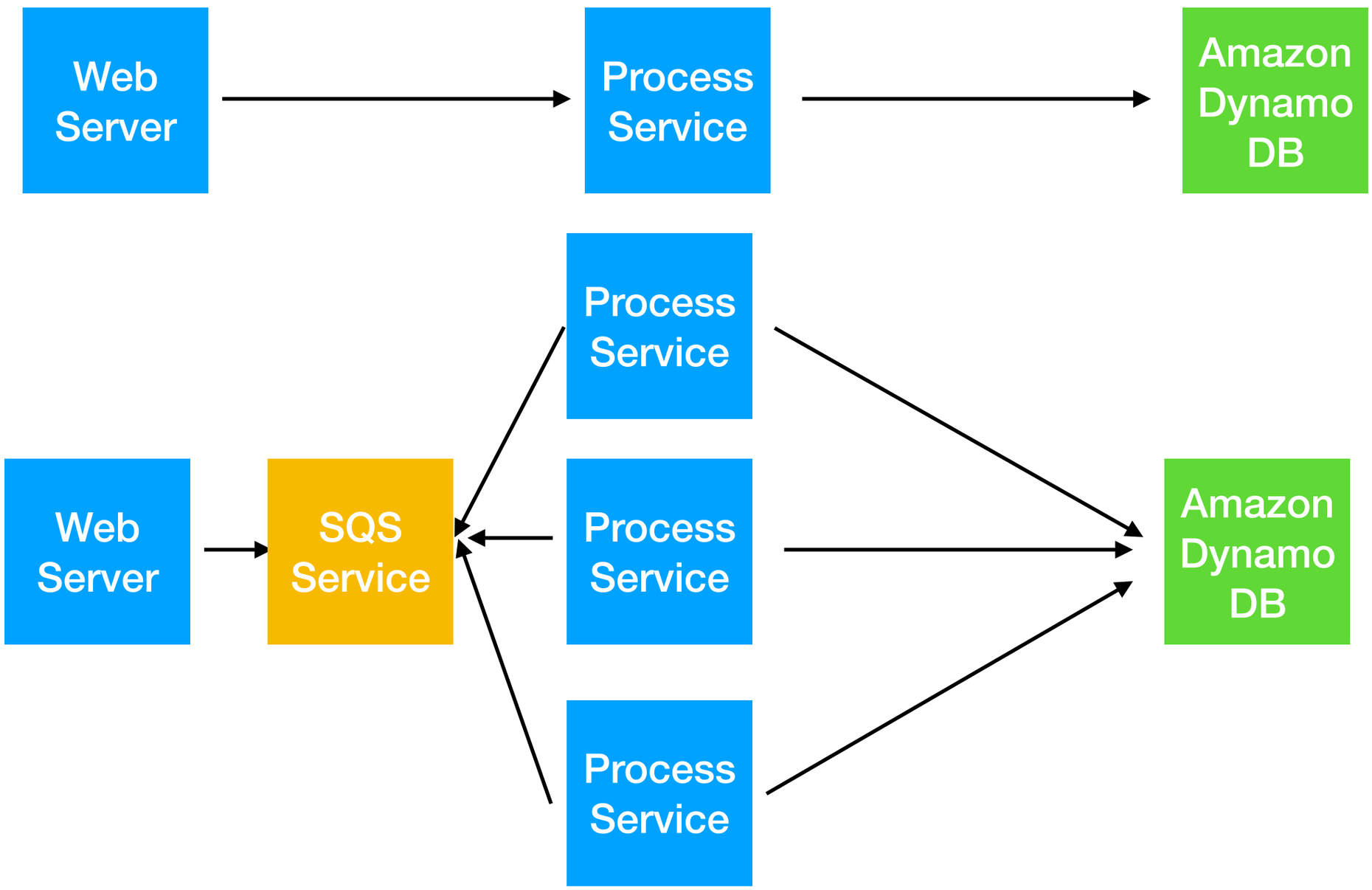
How to fanout to multiple customers?

AWS SNS



Fanout

Decouple for scalability

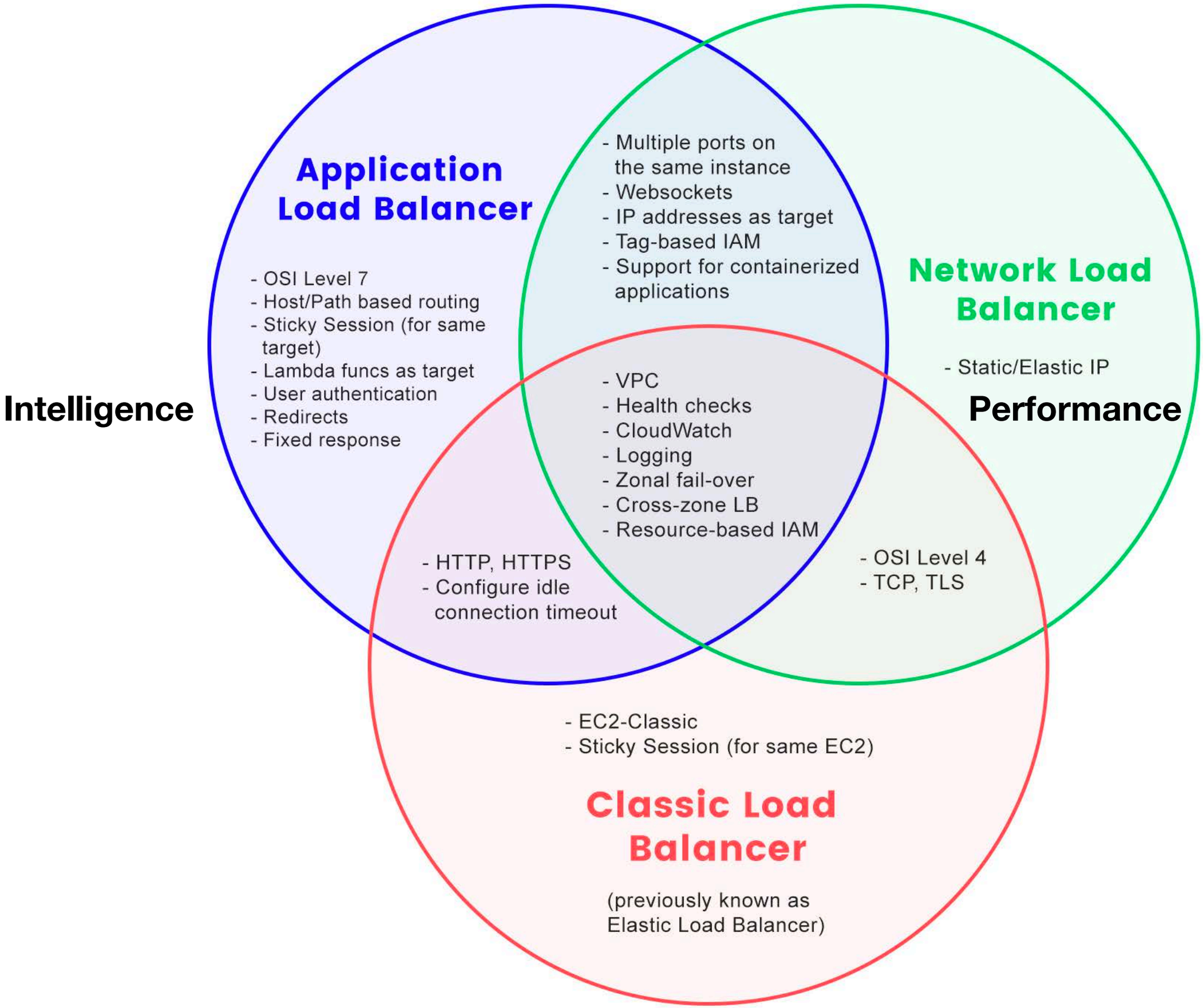


Elastic Load Balancer

- ELB offers features:
    - Detection of unhealthy EC2 instances.
    - Spreading EC2 instances across healthy channels only.
    - Centralized management of SSL certificates.
    - Optional public key authentication.
    - Support for both IPv4 and IPv6.

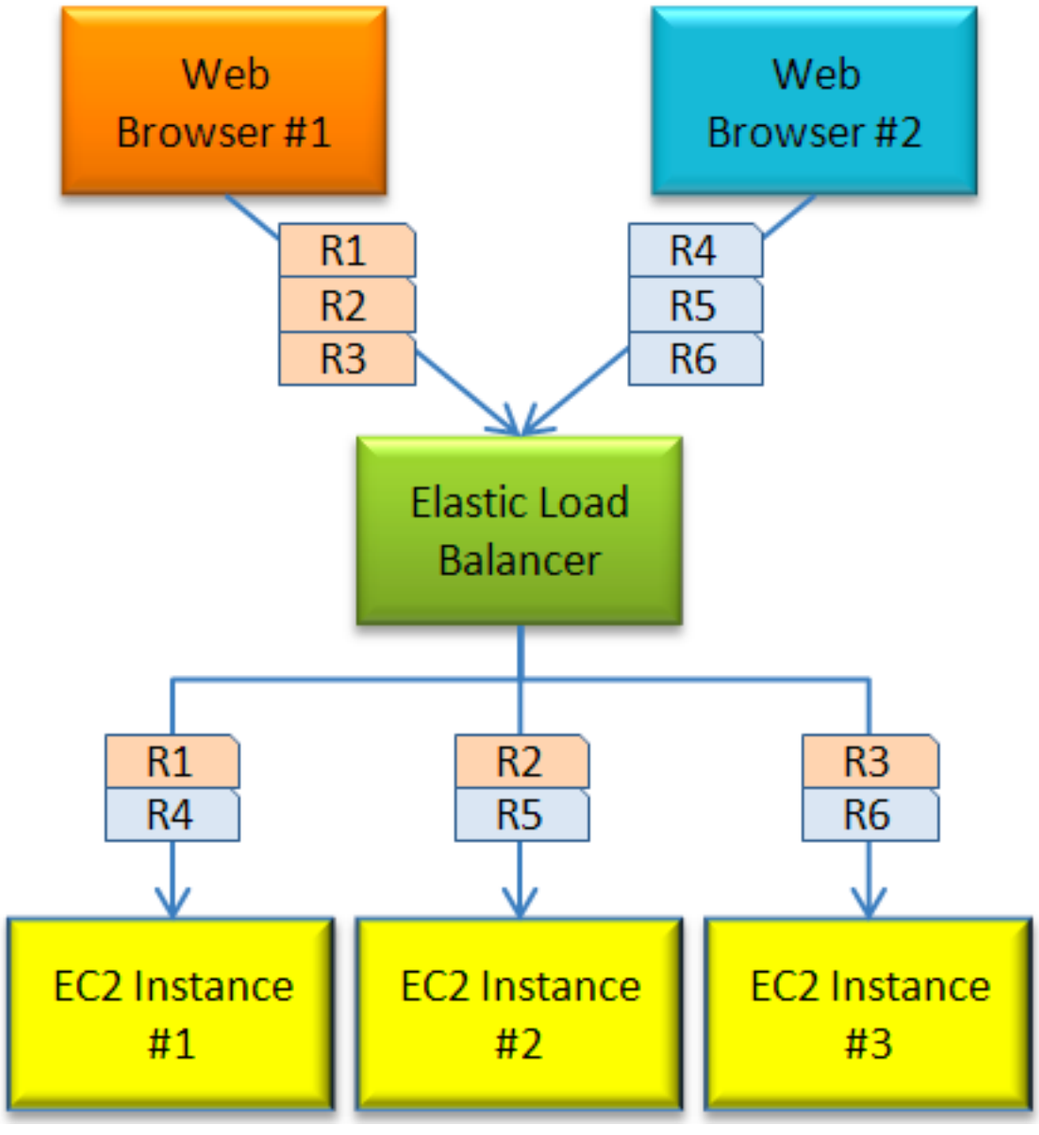
ELB

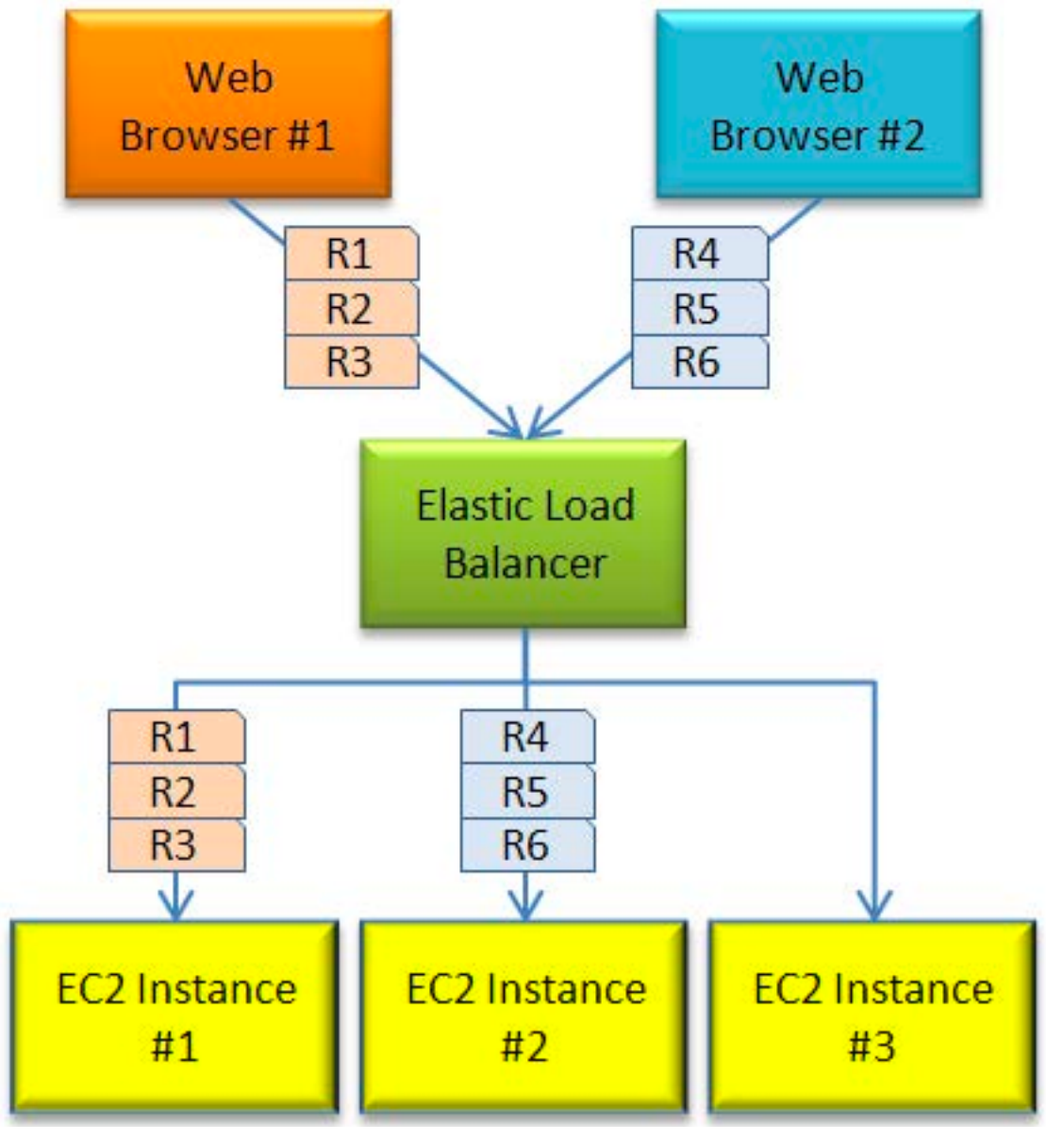
- ELB supports 3 types of load balancers:
    - Application Load Balancers (Intelligence)
    - Network Load Balancers (performance)
    - Classic Load Balancers



https://www.nclouds.com/blog/what-type-of-aws-elastic-load-balancing-aws-elb-is-right-for-you/

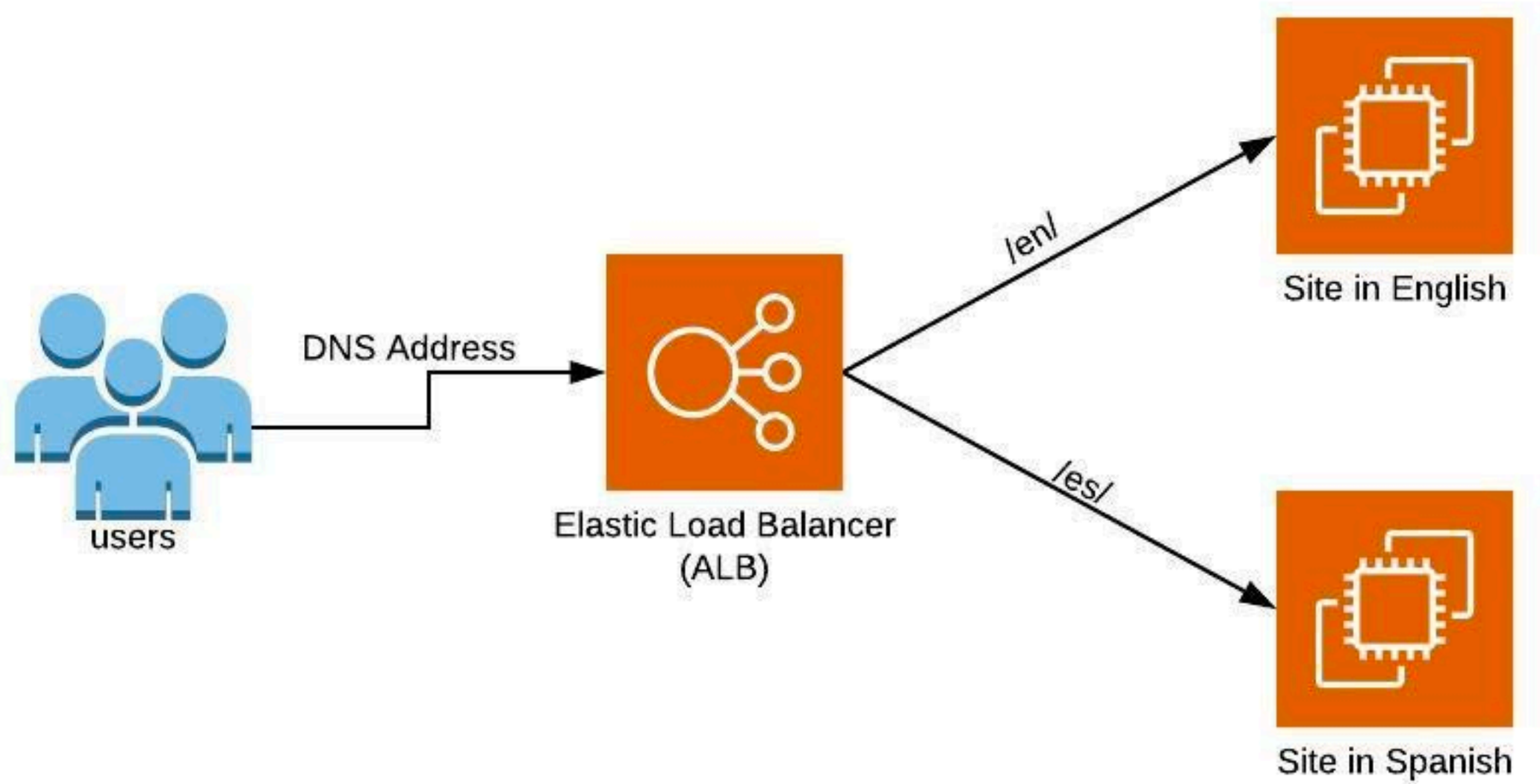
Elastic Load Balancing Feature: Sticky Sessions





Local Storage

Path based routing- Application load balancer



https://zeebaig.medium.com/path-based-routing-in-aws-application-load-balancer-b5a91a79d7f9

Question

- You configured ELB to perform health checks on EC2 instances. If an instance fails to pass health checks, which statement will be true? (choose 1 correct answer)

1. The instance gets quarantined by the ELB for root cause analysis.

2. The instance is replaced automatically by the ELB.

3. The instance gets terminated automatically by the ELB.

4. The ELB stops sending traffic to the instance that failed its health check.

Question

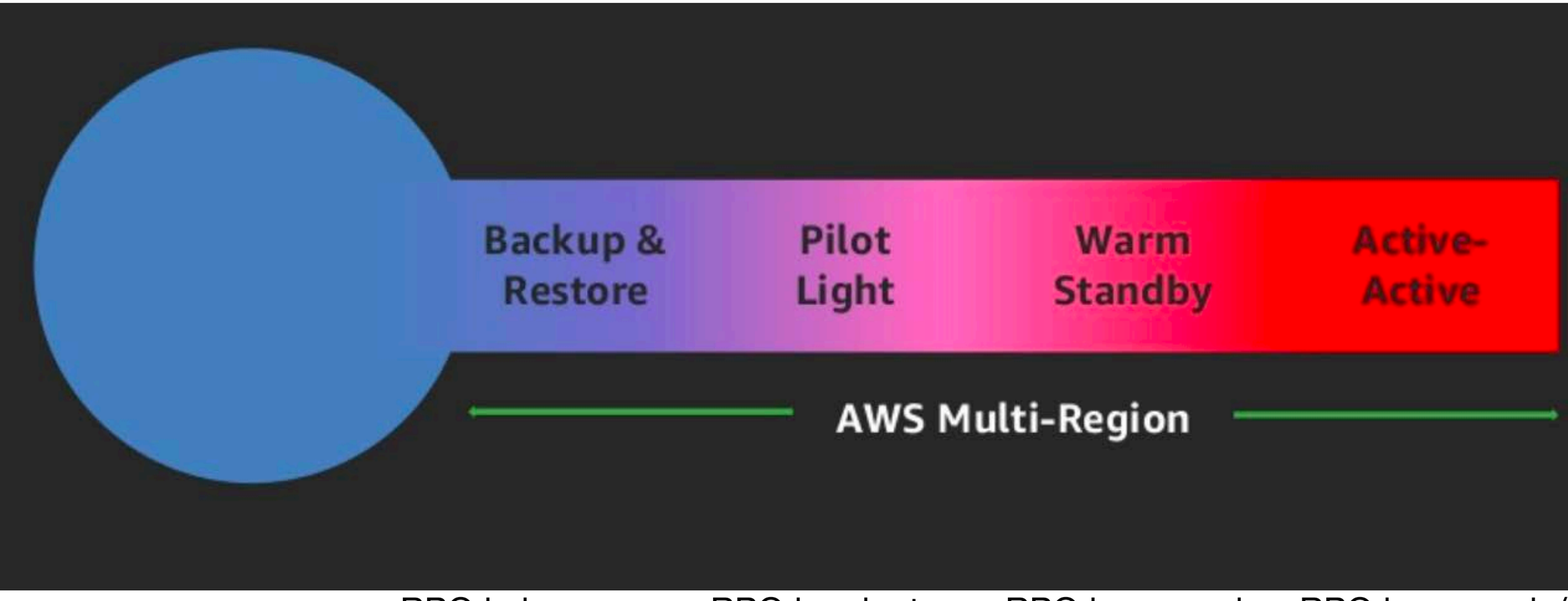
- You configured ELB to perform health checks on EC2 instances. If an instance fails to pass health checks, which statement will be true? (choose 1 correct answer)

1. The instance gets quarantined by the ELB for root cause analysis.

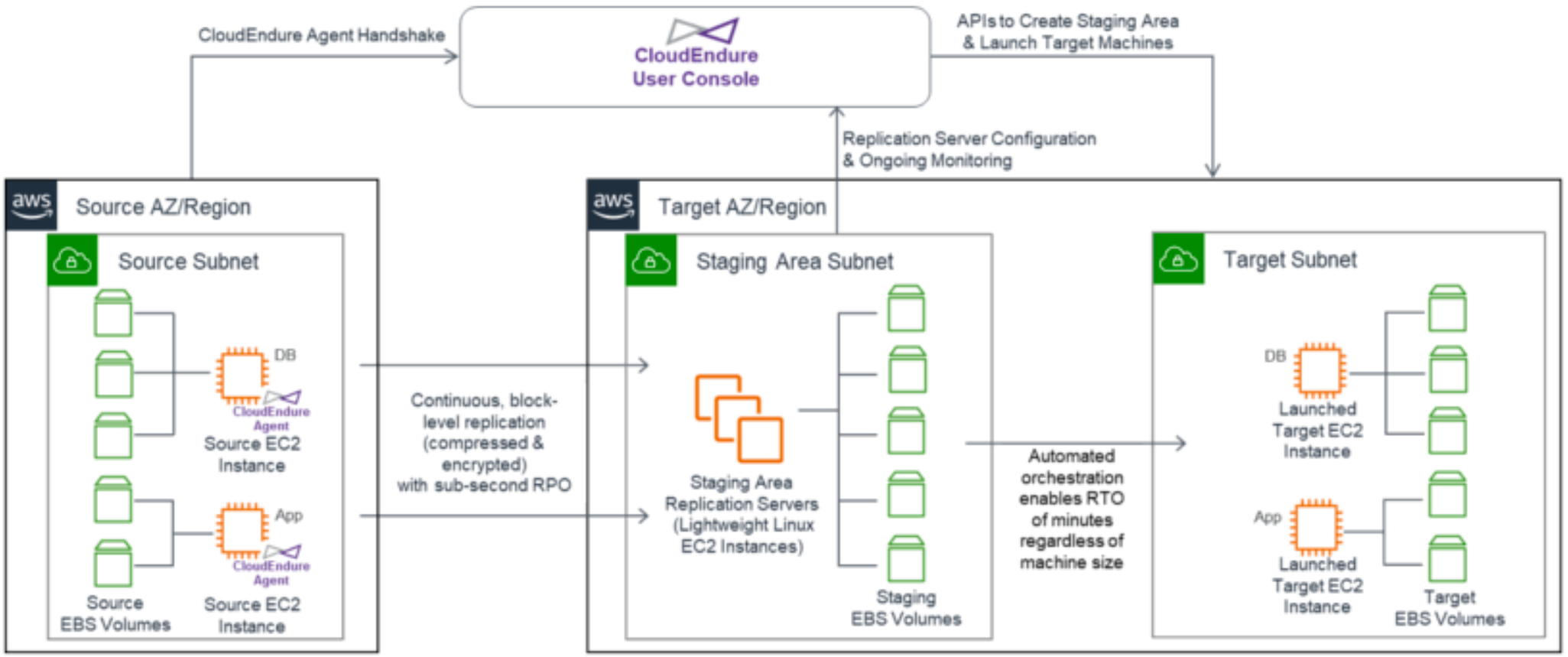
2. The instance is replaced automatically by the ELB.

3. The instance gets terminated automatically by the ELB.

4. **The ELB stops sending traffic to the instance that failed its health check.**



RPO in hours	RPO in minutes	RPO in seconds	RPO in seconds/None
RTO in 24 hours	RTO in hours	RTO in minutes	RTO in seconds


CloudEndure Agent Handshake
CloudEndure
User Console
APIs to Create Staging Area
& Launch Target Machines
Replication Server Configuration
& Ongoing Monitoring
aws
Source AZ/Region
Source Subnet
DB
CloudEndure
Agent
Source EC2
Instance
App
CloudEndure
Agent
Source
EBS Volumes
Source EC2
Instance
Continuous, block-
level replication
(compressed &
encrypted)
with sub-second RPO
aws
Target AZ/Region
Staging Area Subnet
Staging Area
Replication Servers
(Lightweight Linux
EC2 Instances)
Staging
EBS Volumes
Automated
orchestration
enables RTO
of minutes
regardless of
machine size
Target Subnet
DB
Launched
Target EC2
Instance
App
Launched
Target EC2
Instance
Target
EBS Volumes

Design Resilient Architectures

Design Cost-Optimized Architectures

Sustainability Architectures

Design Performant Architectures

Operationally Excellent Architectures

Specify Secure Applications

Well Architected Framework

Cost Optimization

Domain 4: Design Cost-Optimized Architectures

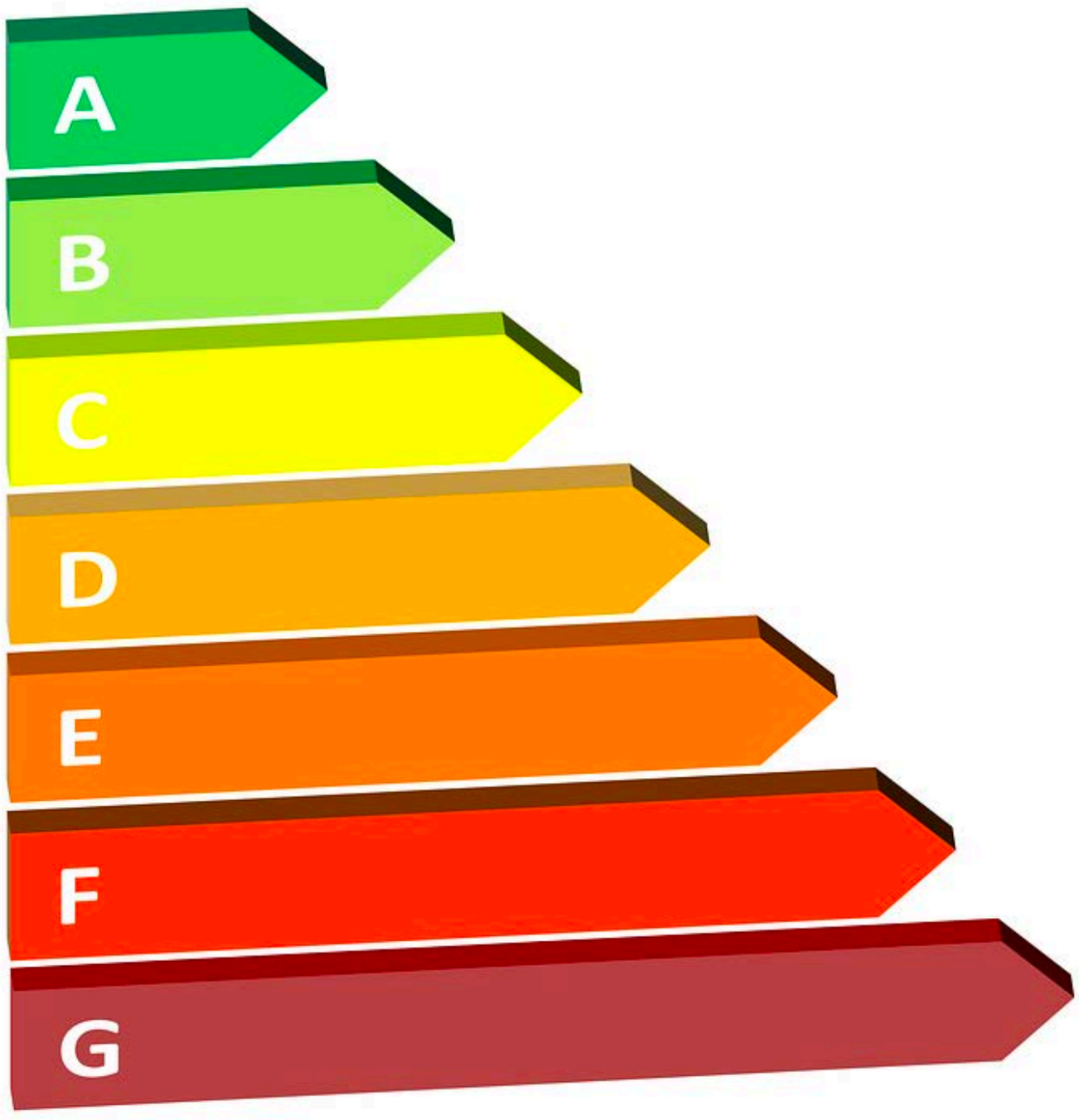
- 4.1 Determine how to design cost-optimized storage.

- 4.2 Determine how to design cost-optimized compute.

Adopt a consumption model

 # users

Measure overall efficiency



Service Clouds

Managed Datacenter

Fundamental Pricing Characteristics

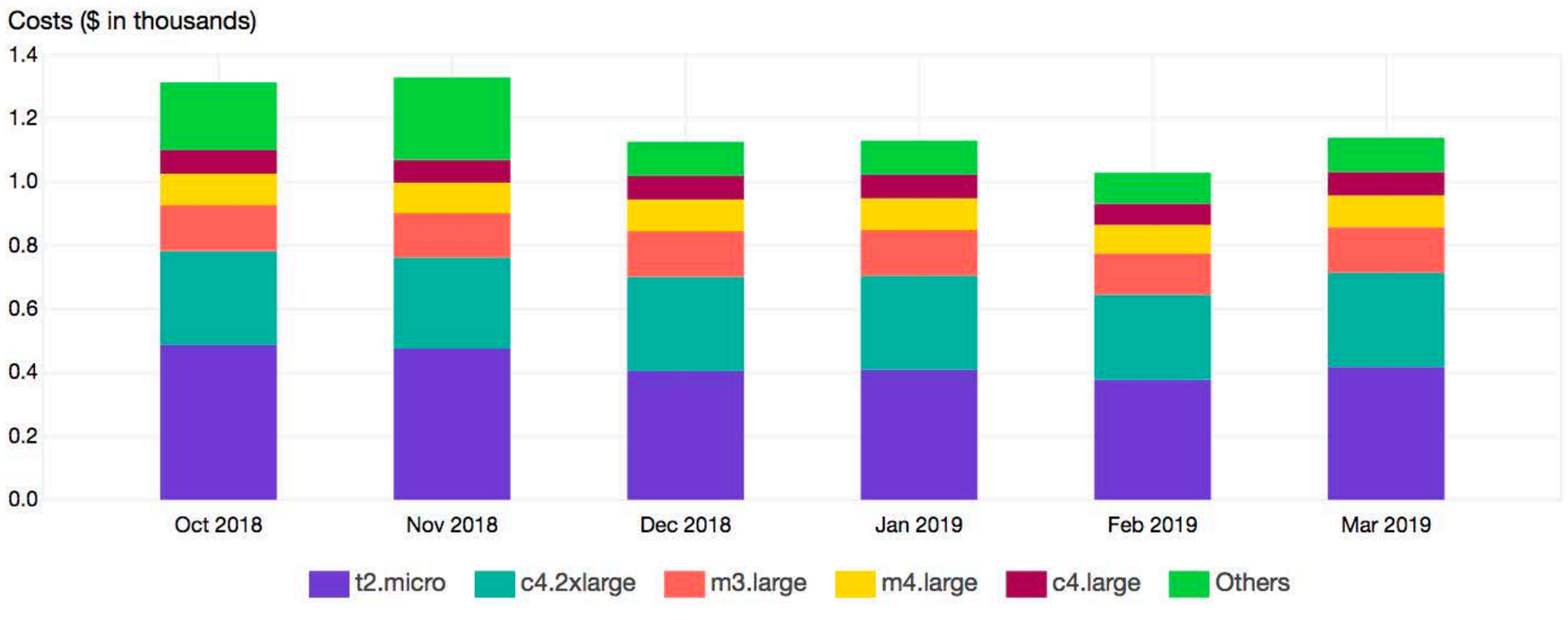
- Compute
- Storage
- Data transfer

AWS Cost Explorer

Monthly costs by service

Last 6 Months | Monthly | Stack

Group by: Instance Type × | Service | Linked Account | Region | Usage Type | Tag | API Operation | Availability Zone | More



Download CSV

Instance Type	Oct 1, 2018	Nov 1, 2018	Dec 1, 2018	Jan 1, 2019
Total cost ($)	1,312.71	1,328.54	1,125.99	1,129.65
t2.micro ($)	486.75	475.89	405.63	409.27
c4.2xlarge ($)	296.11	286.56	296.11	296.11

FILTERS — CLEAR ALL

Service — Include only
EC2-Instances × 1
Linked Account — Include all
Region — Include all
Instance Type — Include all
Usage Type — Include all
Usage Type Group — Include all
Tag — Include All
API Operation — Include all
Charge Type — Include all

More filters

ADVANCED OPTIONS

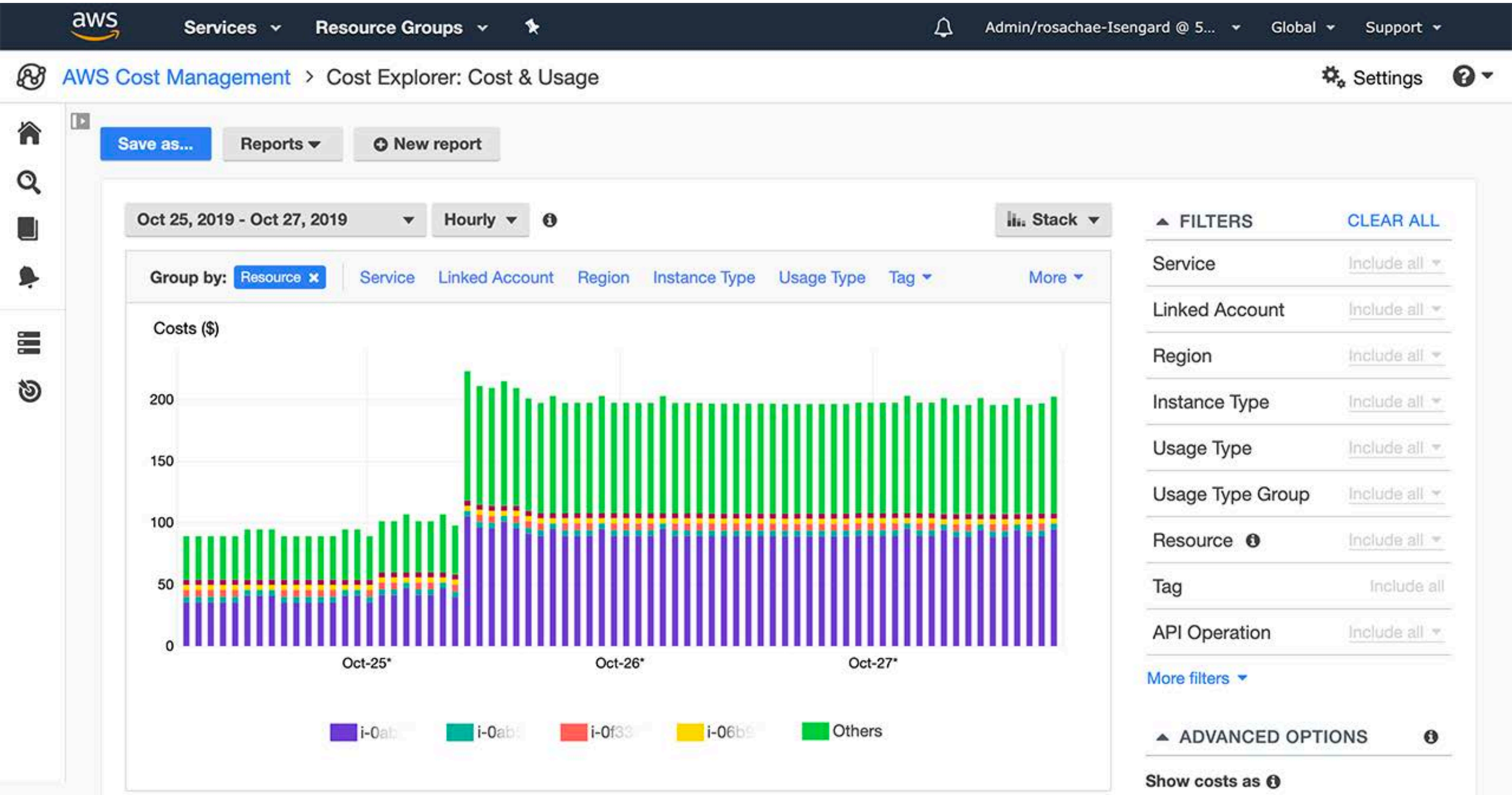
Show costs as

Unblended costs

Include costs related to

☐ Show only untagged resources

Hourly and Resource Level Granularity



Savings Plans

Recommendation options

Savings Plans type	Savings Plans term	Payment option	Based on the past
◉ Compute	○ 1-year	◉ All upfront	○ 7 days
○ EC2 Instance	◉ 3-year	○ Partial upfront	○ 30 days
		○ No upfront	◉ 60 days

Recommendation: Purchase a Compute Savings Plan at a commitment of $2.40/hour

You could save an estimated $1,173 monthly by purchasing the recommended Compute Savings Plan.

Based on your past **60 days** of usage, we recommend purchasing a Savings Plan with a commitment of **$2.40/hour** for a **3-year term**. With this commitment, we project that you could save an average of **$1.61/hour** - representing a **40%** savings compared to On-Demand. To account for variable usage patterns, this recommendation maximizes your savings by leaving an average **$0.04/hour** of On-Demand spend.

Before recommended purchase	After recommended purchase (based on your past 60 days of usage)	
Monthly On-Demand spend ⓘ	Estimated monthly spend ⓘ	Estimated monthly savings ⓘ
$2,955 ($4.05/hour)	**$1,782** ($2.44/hour)	**$1,173** ($1.61/hour)
Based on your On-Demand spend over the past 60 days	Your recommended $2.40/hour Savings Plans commitment + an average $0.04/hour of On-Demand spend	40% monthly savings over On-Demand $2,955 - $1,782 = $1,173

This recommendation examines your usage over the past 60 days (including your existing Savings Plans and EC2 Reserved Instances) and calculates what your costs would have been had you purchased the recommended Savings Plans. See applicable rates for Savings Plans here. To generate this recommendation, AWS simulates your bill for different commitment amounts and recommends the commitment amount that provides the greatest estimated savings. Learn more

Recommended Compute Savings Plans

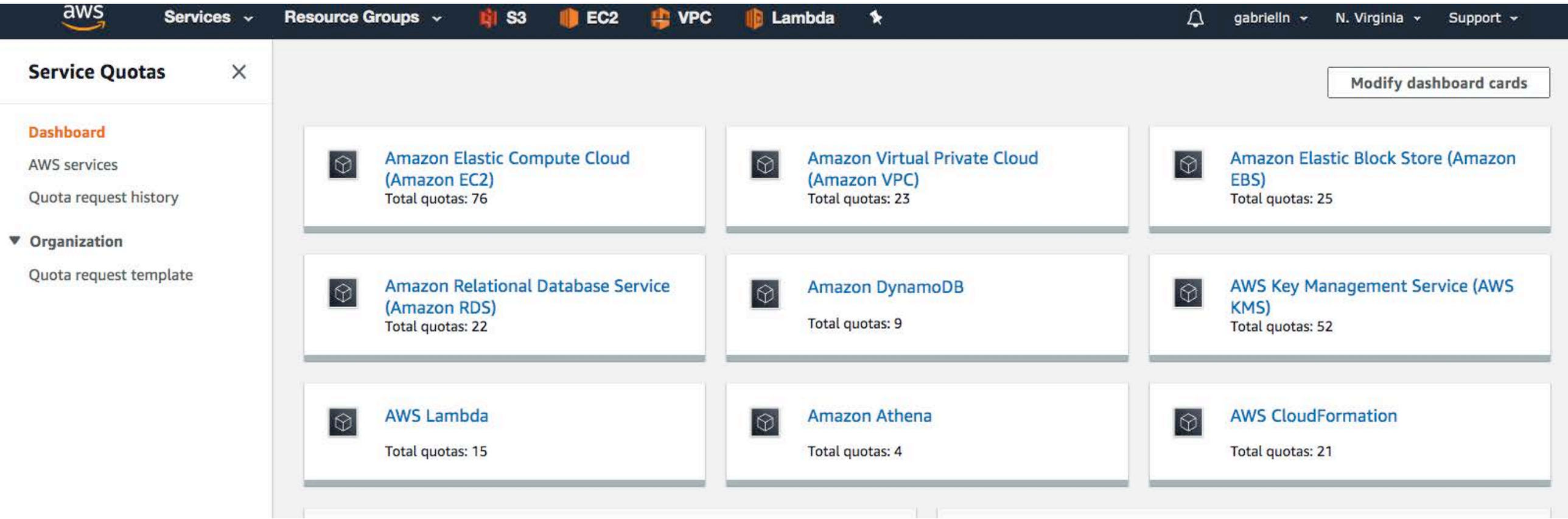
Download CSV | Add selected Savings Plan(s) to cart

✖	Term	Payment option	Recommended commitment	Estimated hourly savings ⬍
☑	3-year	All upfront	$2.40/hour	$1.61 (40%)

*Average hourly spend and minimum hourly spend based on your current on-demand spend for the given instance family

Why Service Quotas/Limits are important?

Service Quotas/Limits



API Request Limits and Allocations

To maintain optimum performance and ensure that the Lightning Platform API is available to all our customers, Salesforce balances transaction loads by imposing two types of limits:

- Concurrent API Request Limits
- Total API Request Allocations

When a call exceeds a request limit, an error is returned.

Concurrent API Request Limits

The following table lists the limits for various types of orgs for concurrent requests (calls) with a duration of 20 seconds or longer.

Org Type	Limit
Developer Edition and Trial orgs	5
Production orgs and Sandboxes	25

Total API Request Allocations

The following table lists the limits for the total API requests (calls) per 24-hour period for an org.

Salesforce Edition	API Calls Per License Type Per 24-Hour Period	Total Calls Per 24-Hour Period
Developer Edition	N/A	15,000
• Enterprise Edition • Professional Edition with API access	• Salesforce: 1,000 • Salesforce Platform: 1,000 • Lightning Platform - One App: 200	100,000 + (number of licenses x calls per license type) + purchased API Call Add-Ons

https://developer.salesforce.com/docs/atlas.en-us.salesforce_app_limits_cheatsheet.meta/salesforce_app_limits_cheatsheet/salesforce_app_limits_platform_apexgov.htm

Amazon EC2

- Reserved Instances
    - Standard RIs
    - Convertible RIs
    - Scheduled RIs
- Spot Instances (Stock market price) -
    - Hibernate instance if price goes up and restart again
    - Spot block - 4 hrs block

Question

- You receive a Spot Instance at a bid of $0.05/hr. After 30 minutes, the Spot Price increases to $0.06/hr and your Spot Instance is terminated by AWS. What was the total EC2 compute cost of running your Spot Instance?

1. $0.05
2. $0.06
3. $0.025
4. $0.02
5. $0.00

Answer

- You receive a Spot Instance at a bid of $0.05/hr. After 30 minutes, the Spot Price increases to $0.06/hr and your Spot Instance is terminated by AWS. What was the total EC2 compute cost of running your Spot Instance?

1. $0.05
2. $0.06
3. $0.025
4. $0.02
5. **$0.00**

Amazon S3 Pricing

- Storage Class
- Storage
- Requests
- Data transfer

S3 Storage

	Storage pricing
S3 Standard - General purpose storage for any type of data, typically used for frequently accessed data	
First 50 TB / Month	$0.023 per GB
Next 450 TB / Month	$0.022 per GB
Over 500 TB / Month	$0.021 per GB
S3 Intelligent - Tiering * - Automatic cost savings for data with unknown or changing access patterns	
Frequent Access Tier, First 50 TB / Month	$0.023 per GB
Frequent Access Tier, Next 450 TB / Month	$0.022 per GB
Frequent Access Tier, Over 500 TB / Month	$0.021 per GB
Infrequent Access Tier, All Storage / Month	$0.0125 per GB
Monitoring and Automation, All Storage / Month	$0.0025 per 1,000 objects
S3 Standard - Infrequent Access * - For long lived but infrequently accessed data that needs millisecond access	
All Storage / Month	$0.0125 per GB
S3 One Zone - Infrequent Access * - For re-createable infrequently accessed data that needs millisecond access	
All Storage / Month	$0.01 per GB
S3 Glacier ** - For long-term backups and archives with retrieval option from 1 minute to 12 hours	
All Storage / Month	$0.004 per GB
S3 Glacier Deep Archive ** - For long-term data archiving that is accessed once or twice in a year and can be restored within 12 hours	

S3 Data Retrieval

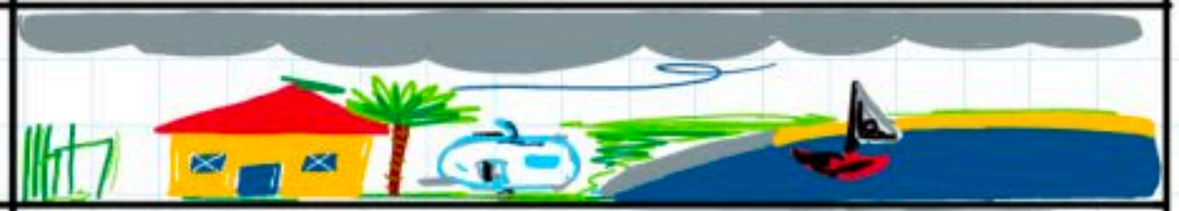
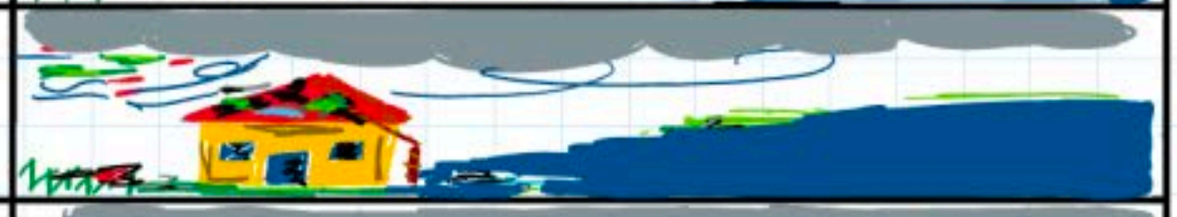
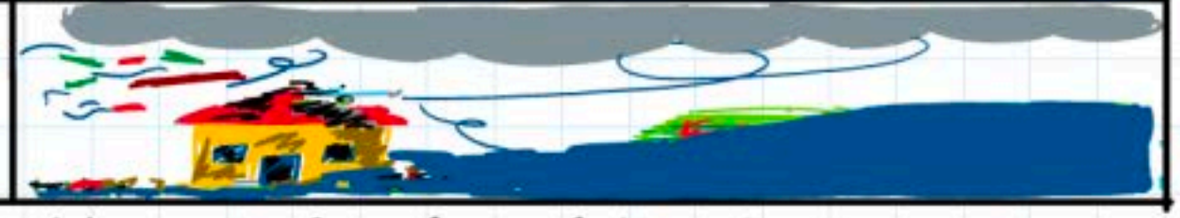
	PUT, COPY, POST, LIST requests (per 1,000 requests)	GET, SELECT, and all other requests (per 1,000 requests)	Lifecycle Transition requests (per 1,000 requests)	Data Retrieval requests (per 1,000 requests)	Data retrievals (per GB)
S3 Standard	$0.005	$0.0004	n/a	n/a	n/a
S3 Intelligent - Tiering	$0.005	$0.0004	$0.01	n/a	n/a
S3 Standard - Infrequent Access*	$0.01	$0.001	$0.01	n/a	$0.01
S3 One Zone - Infrequent Access*	$0.01	$0.001	$0.01	n/a	$0.01
S3 Glacier **	$0.05	$0.0004	$0.05	See below	See below
Expedited	n/a	n/a	n/a	$10.00	$0.03
Standard	n/a	n/a	n/a	$0.05	$0.01
Bulk	n/a	n/a	n/a	$0.025	$0.0025
Provisioned Capacity Unit ***	n/a	n/a	n/a	n/a	$100.00 per unit

Transfers

Data Transfer IN To Amazon S3 From Internet	
All data transfer in	$0.00 per GB

Data Transfer OUT From Amazon S3 To Internet	
Up to 1 GB / Month	$0.00 per GB
Next 9.999 TB / Month	$0.09 per GB
Next 40 TB / Month	$0.085 per GB
Next 100 TB / Month	$0.07 per GB
Greater than 150 TB / Month	$0.05 per GB

Data Transfer OUT From Amazon S3 To	
CloudFront	$0.00 per GB
US East (N. Virginia)	$0.01 per GB
Asia Pacific (Singapore)	$0.02 per GB
Asia Pacific (Sydney)	$0.02 per GB
US West (Los Angeles)	$0.02 per GB
Canada (Central)	$0.02 per GB
Europe (London)	$0.02 per GB

CATEGORY	WIND GUSTS (km/h)	EFFECTS
1	90-125	
2	125-164	
3	165-224	
4	225-279	
5	280 +	

SEVERE

* Adapted from the Bureau of Meteorology's Tropical Cyclone Category System.

https://www.abc.net.au/news/2019-02-20/category-system/10831006?nw=0

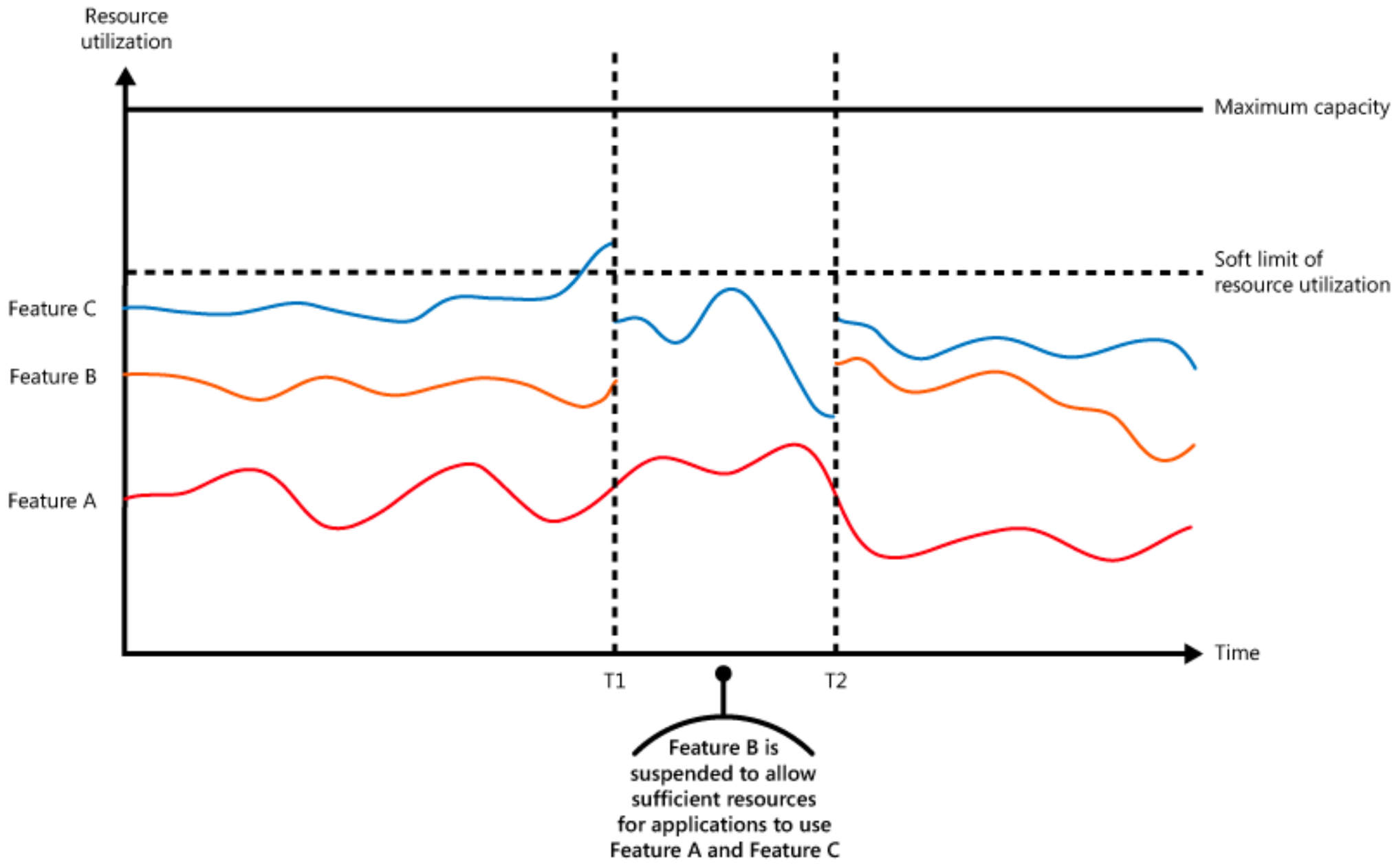
What if the load is high?

What if the load is high?

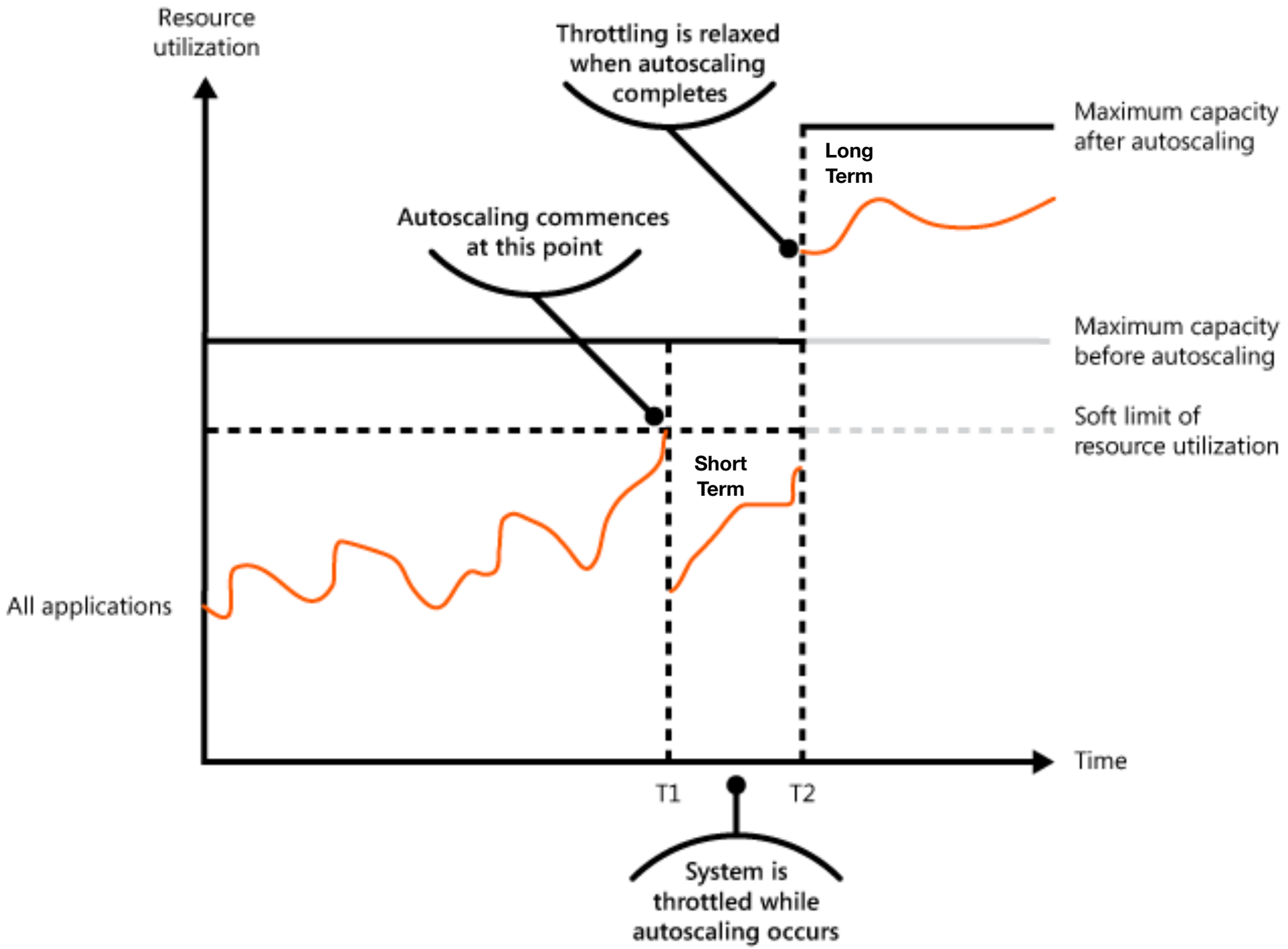
- Essential services

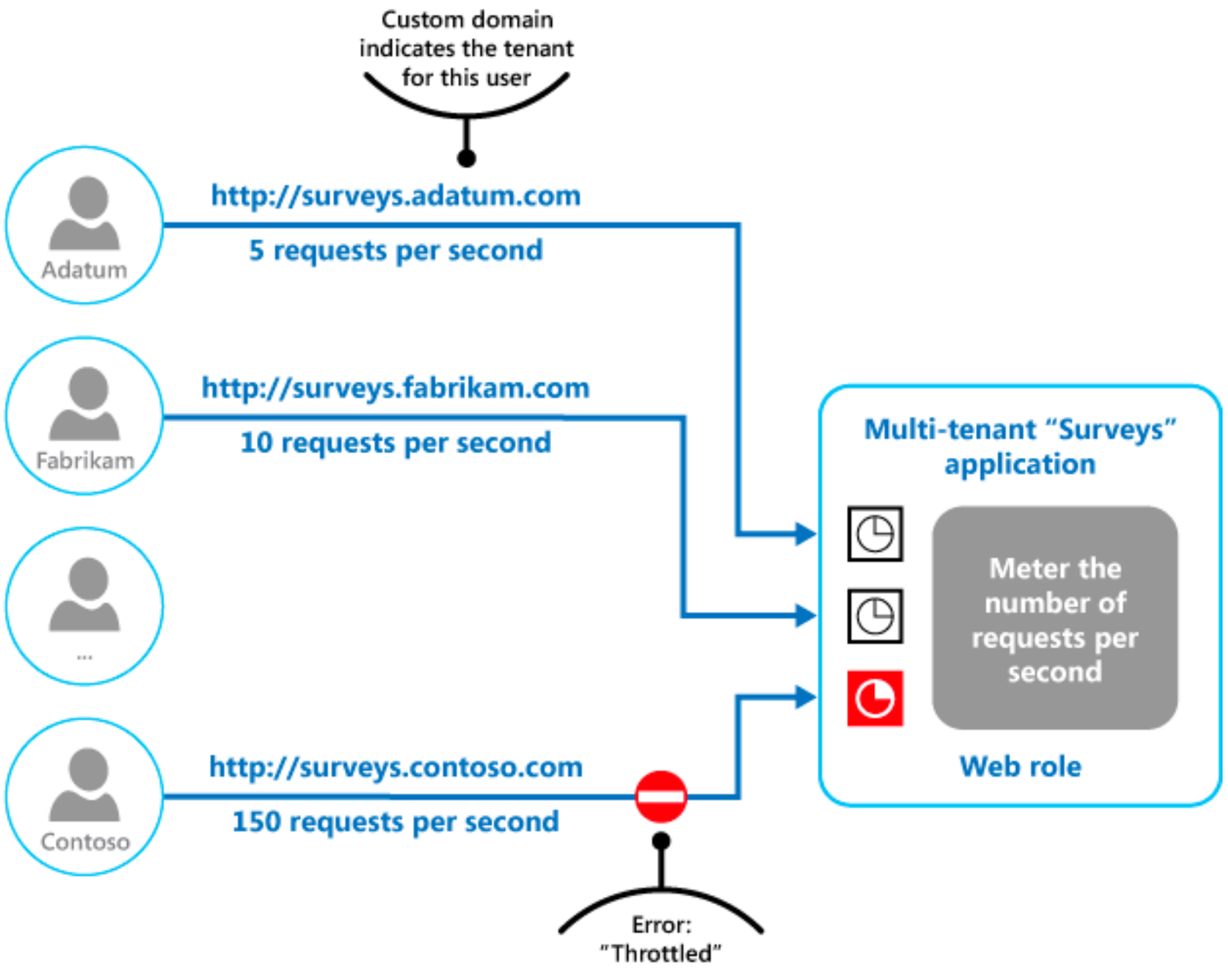
- Non-Essential services

Throttling pattern

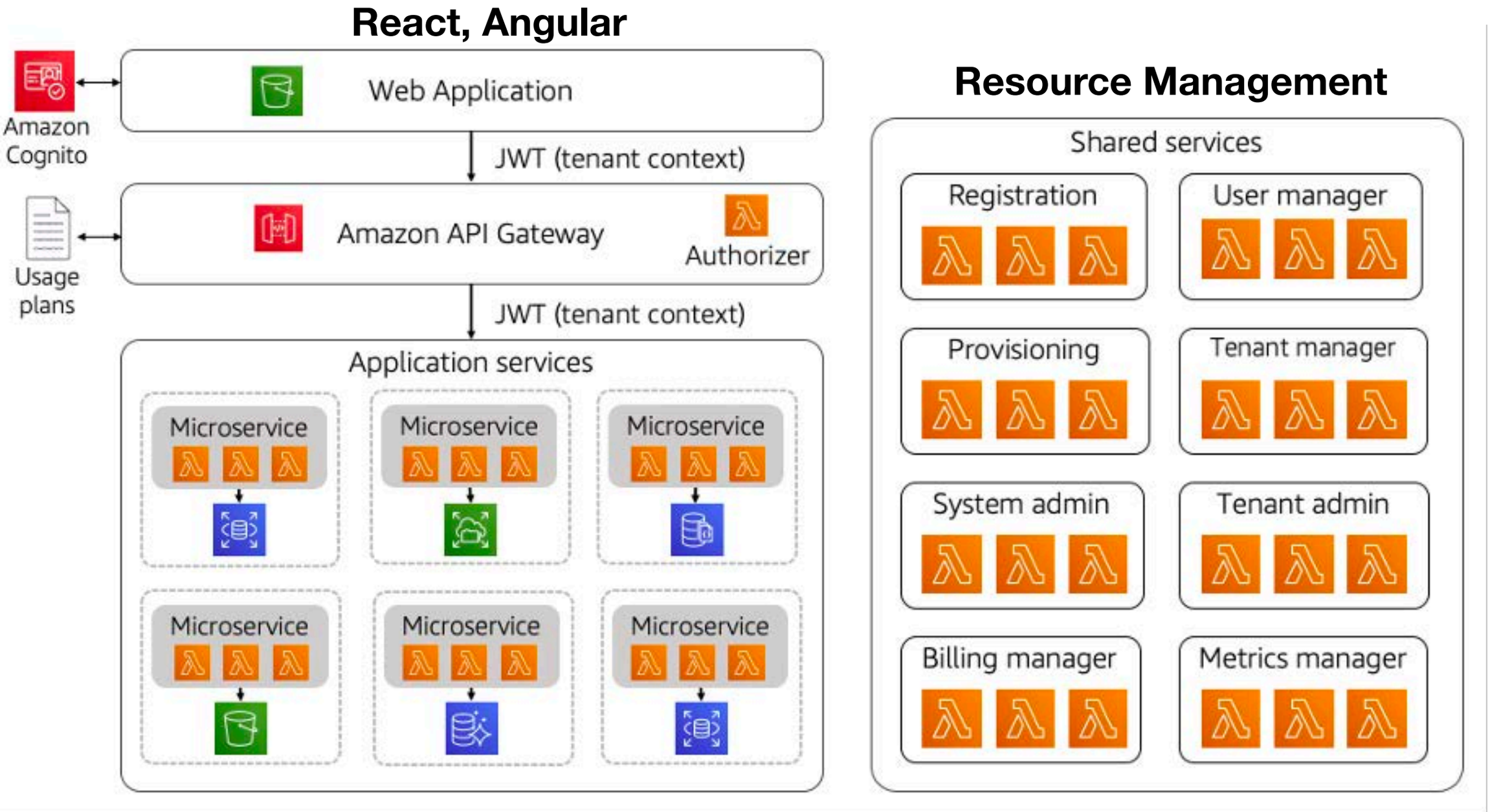


https://docs.microsoft.com/en-us/azure/architecture/patterns/throttling


Resource utilization
Throttling is relaxed when autoscaling completes
Maximum capacity after autoscaling
Long Term
Autoscaling commences at this point
Maximum capacity before autoscaling
Soft limit of resource utilization
Short Term
All applications
Time
T1
T2
System is throttled while autoscaling occurs


Custom domain indicates the tenant for this user
Adatum
http://surveys.adatum.com
5 requests per second
Fabrikam
http://surveys.fabrikam.com
10 requests per second
...
Contoso
http://surveys.contoso.com
150 requests per second
Error: "Throttled"
Multi-tenant "Surveys" application
Meter the number of requests per second
Web role

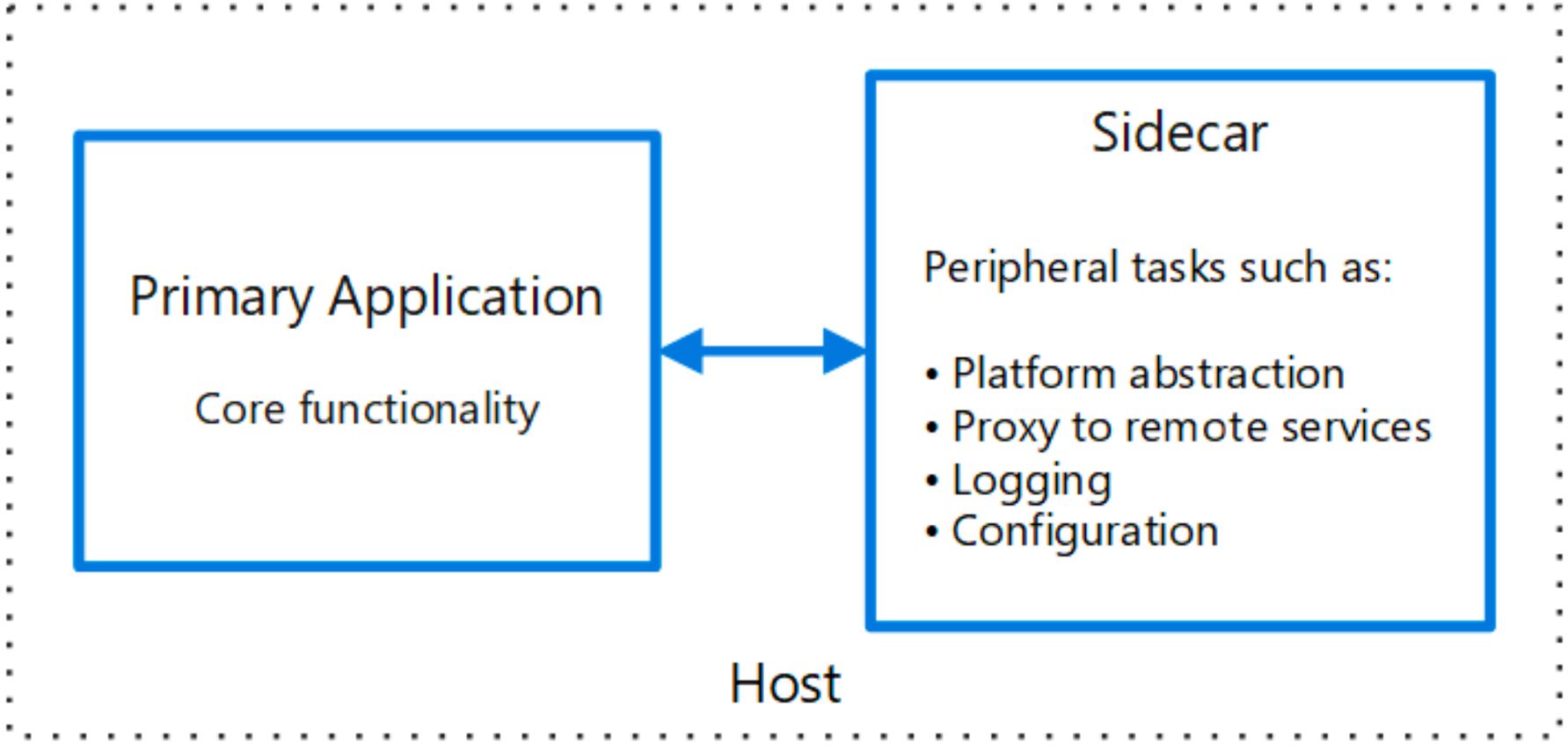
Serverless SaaS



compliance, noisy neighbor, isolation, and performance

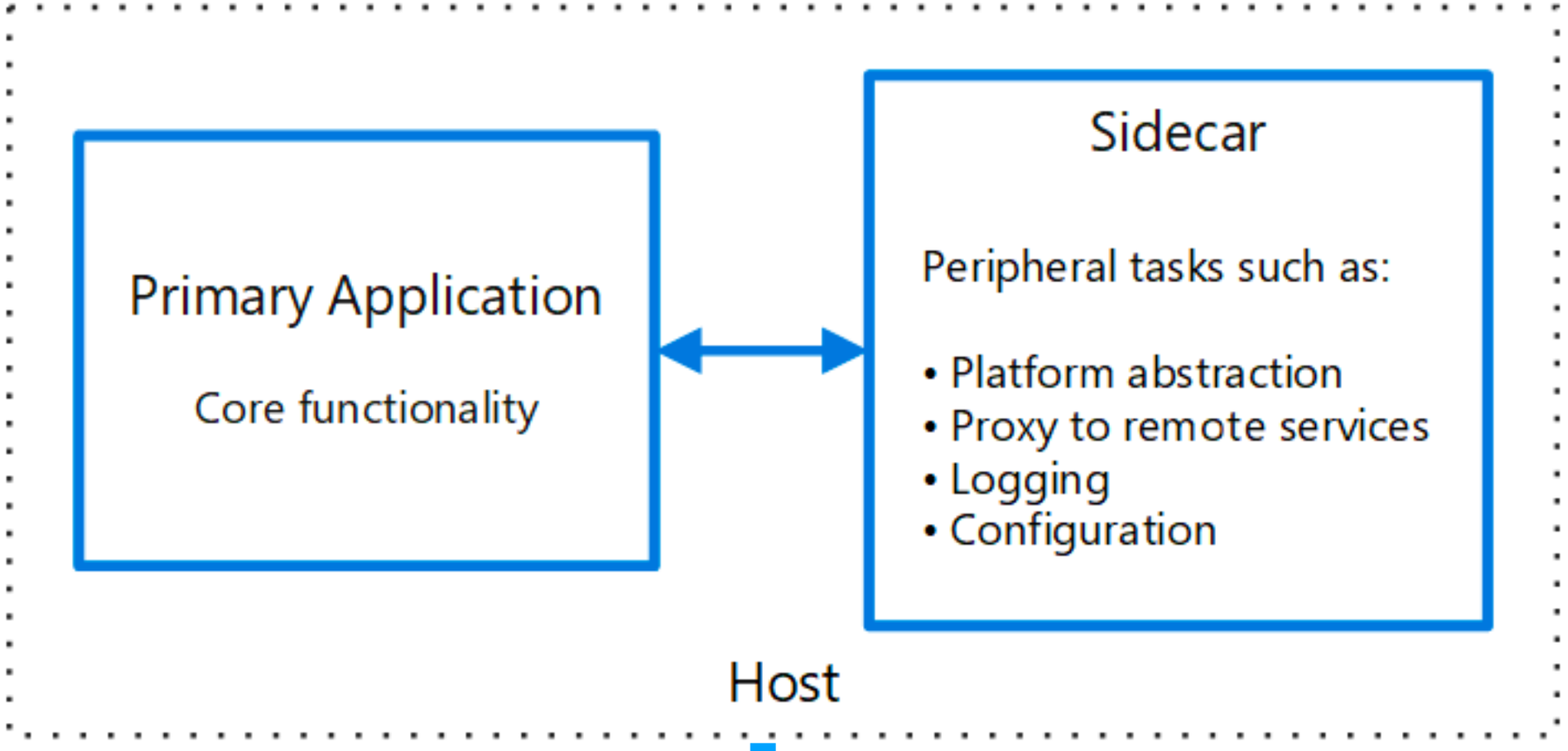
https://aws.amazon.com/blogs/containers/using-sidecar-injection-on-amazon-eks-with-aws-app-mesh/

Sidecar pattern



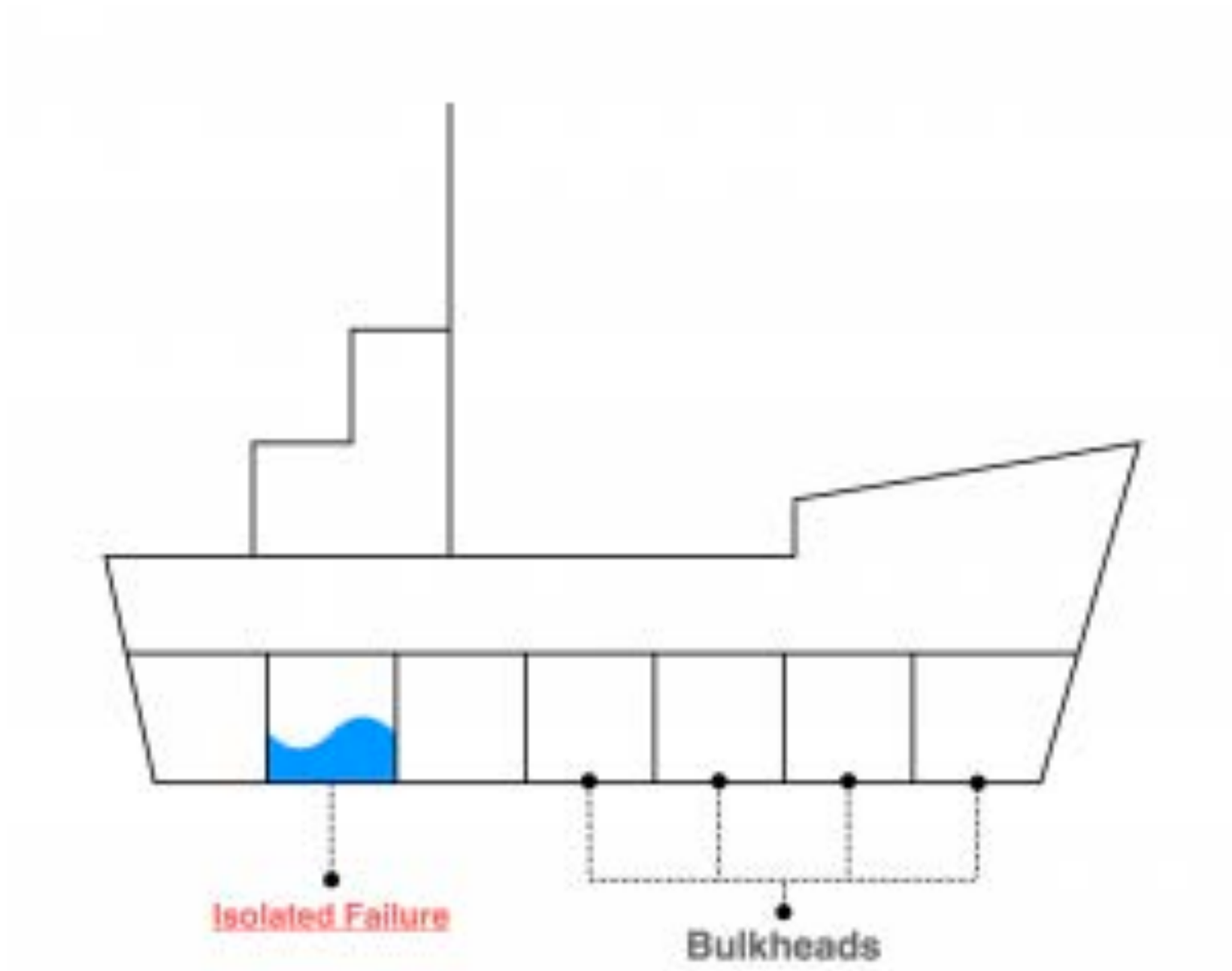
- How inter process communication takes place?
- Build Libraries for multiple applications

Sidecar pattern

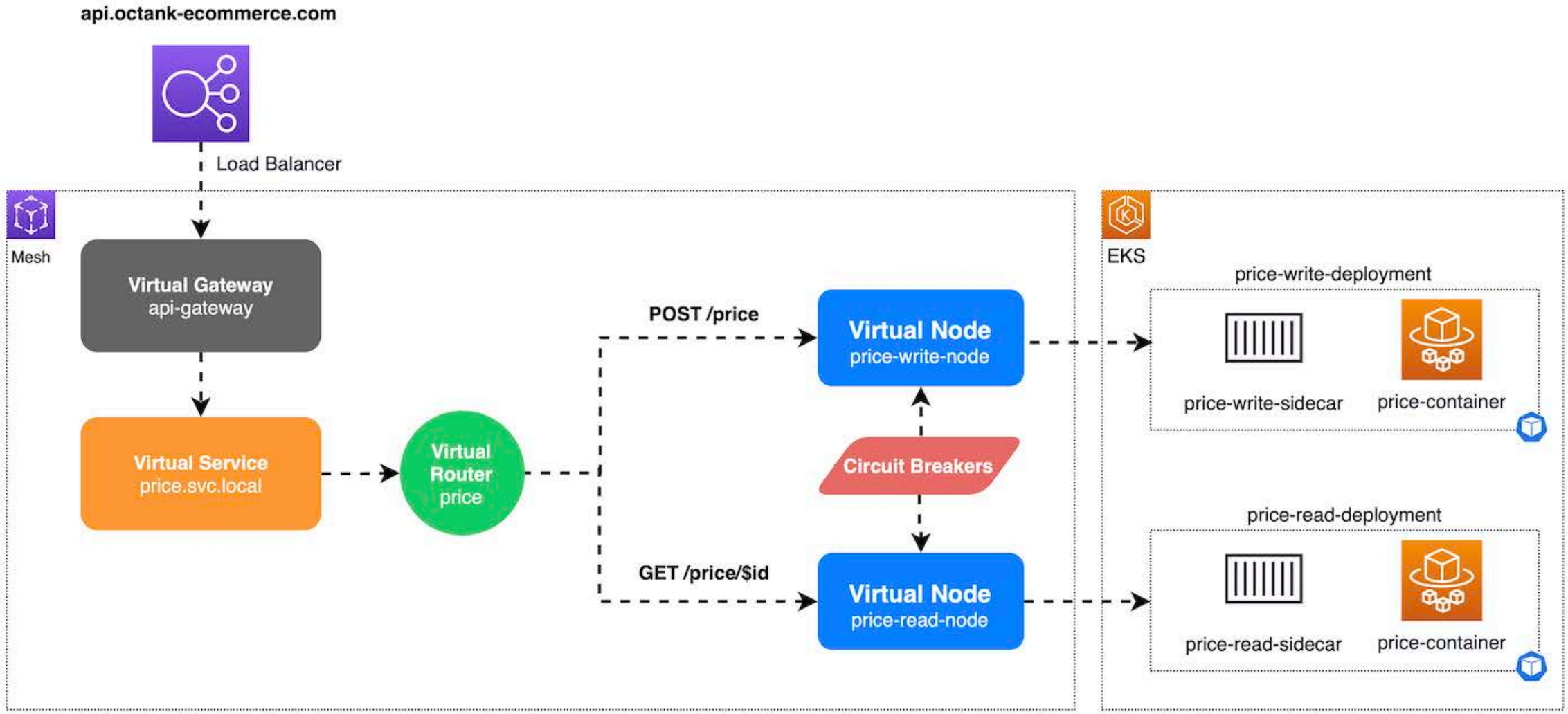


- Heterogeneous apps
- Different Organizations

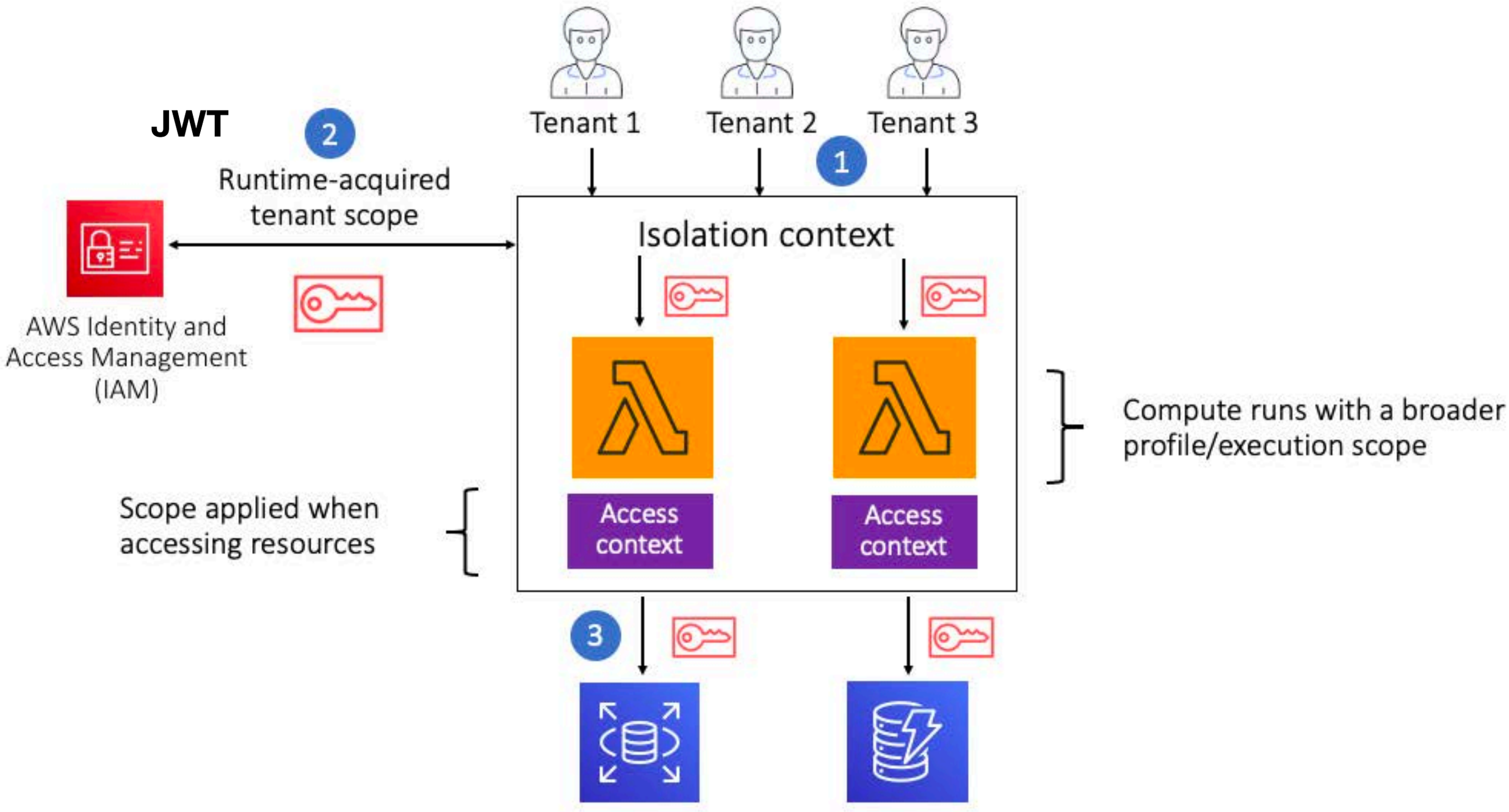
- Communication Latency
- Small apps

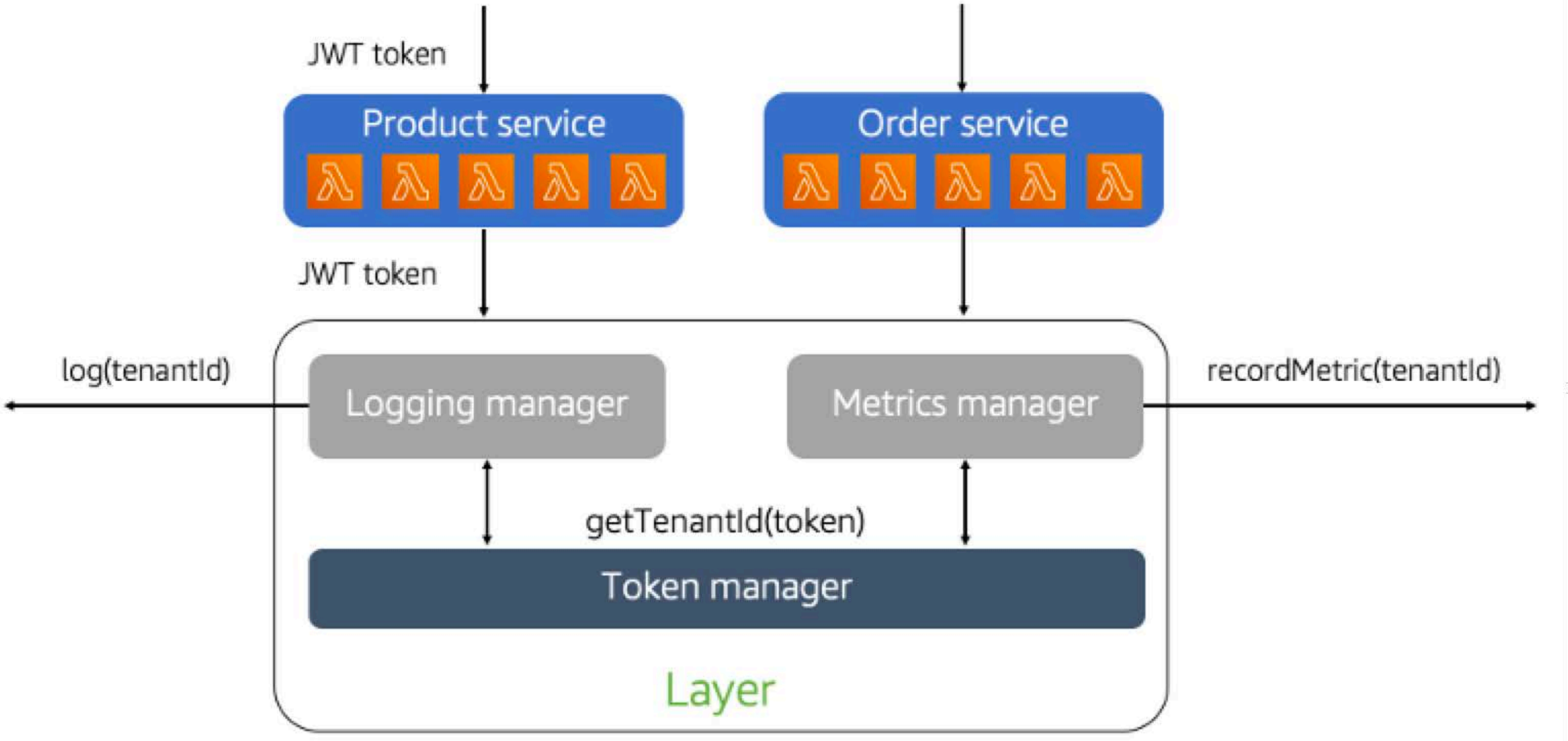


Can you give some example?

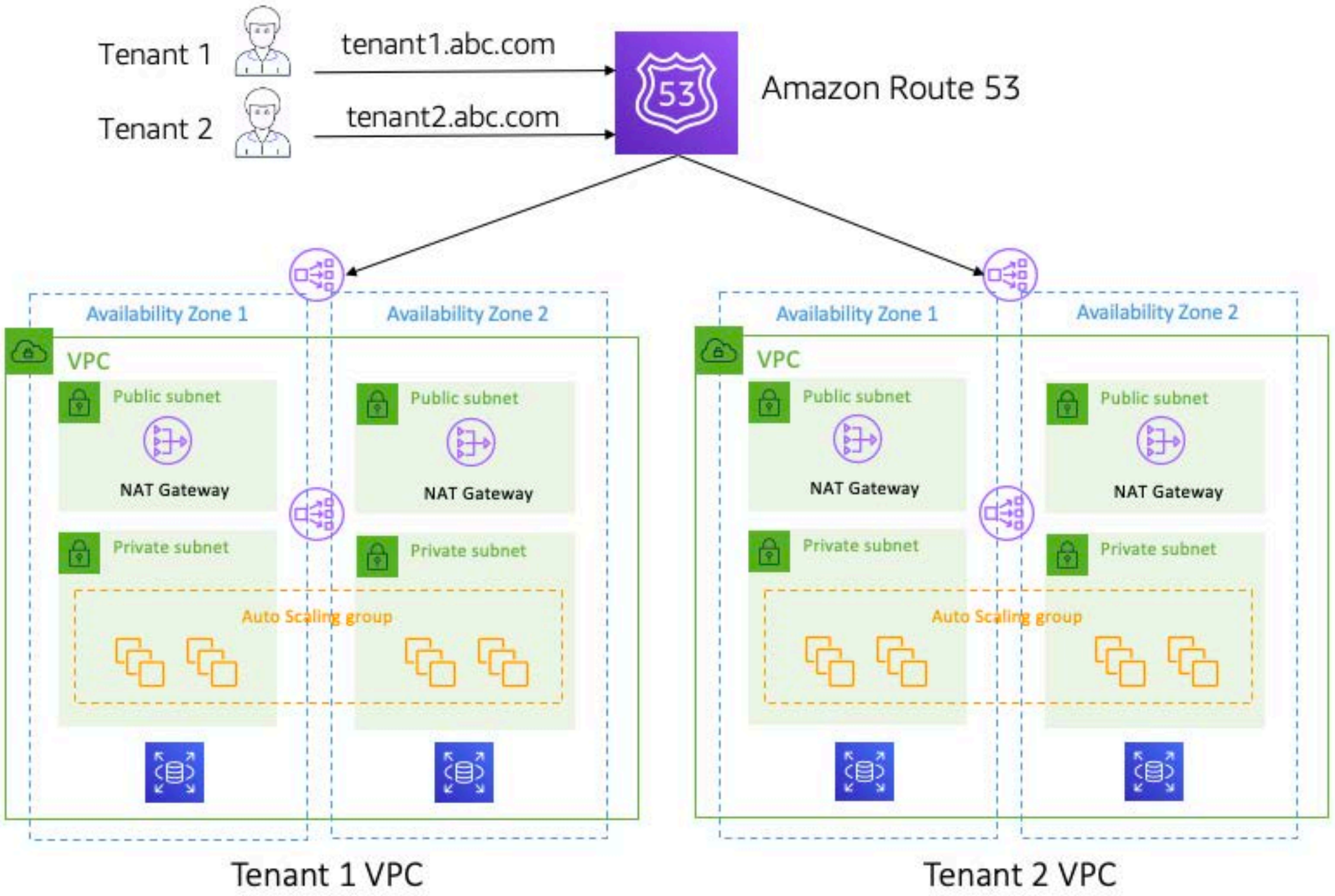

api.octank-ecommerce.com
Load Balancer
Mesh
Virtual Gateway
api-gateway
Virtual Service
price.svc.local
Virtual
Router
price
POST /price
Virtual Node
price-write-node
Circuit Breakers
GET /price/$id
Virtual Node
price-read-node
EKS
price-write-deployment
price-write-sidecar
price-container
price-read-deployment
price-read-sidecar
price-container

Access Management

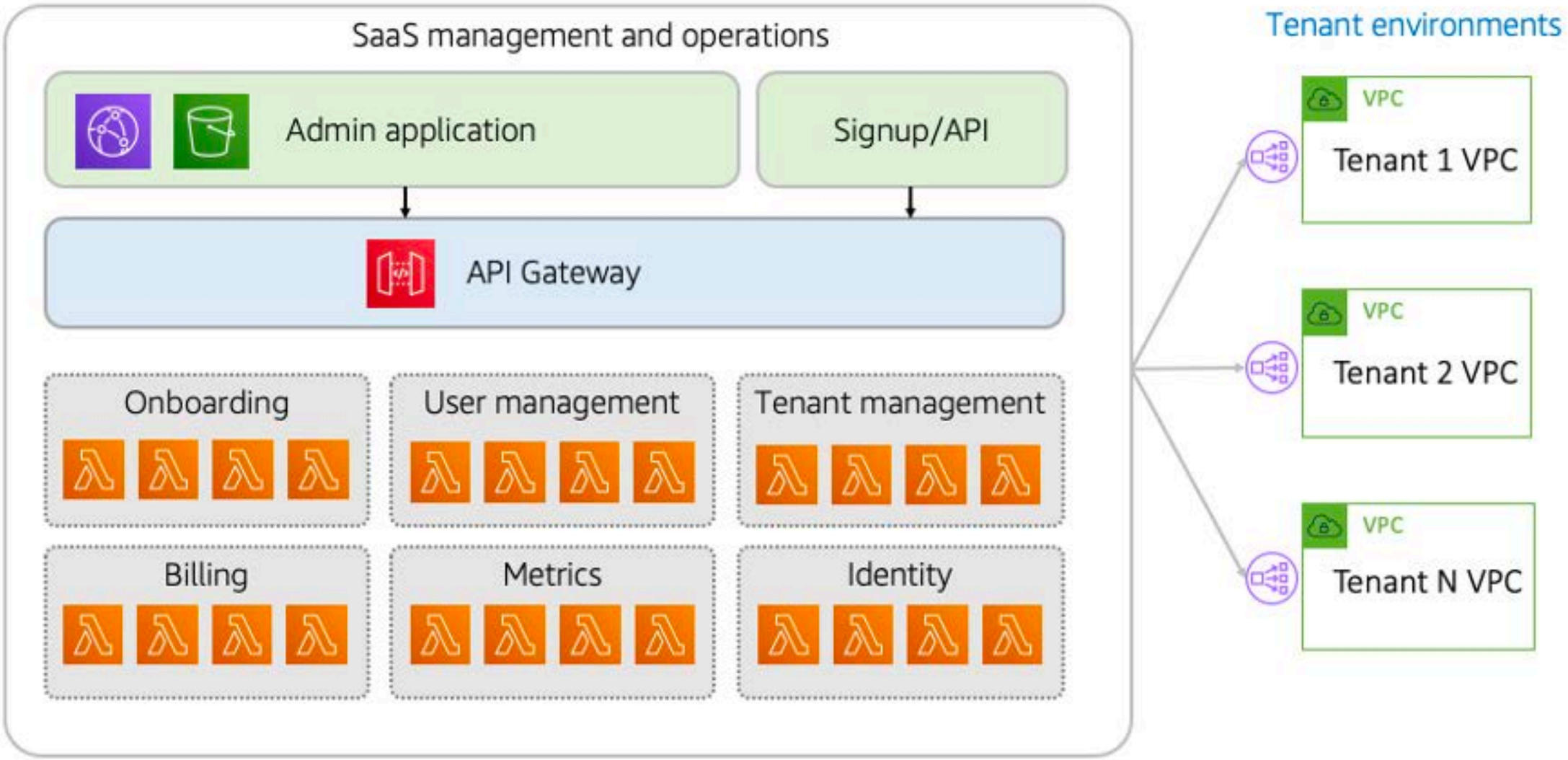



JWT token
Product service
Order service
JWT token
log(tenantId)
Logging manager
Metrics manager
recordMetric(tenantId)
getTenantId(token)
Token manager
Layer

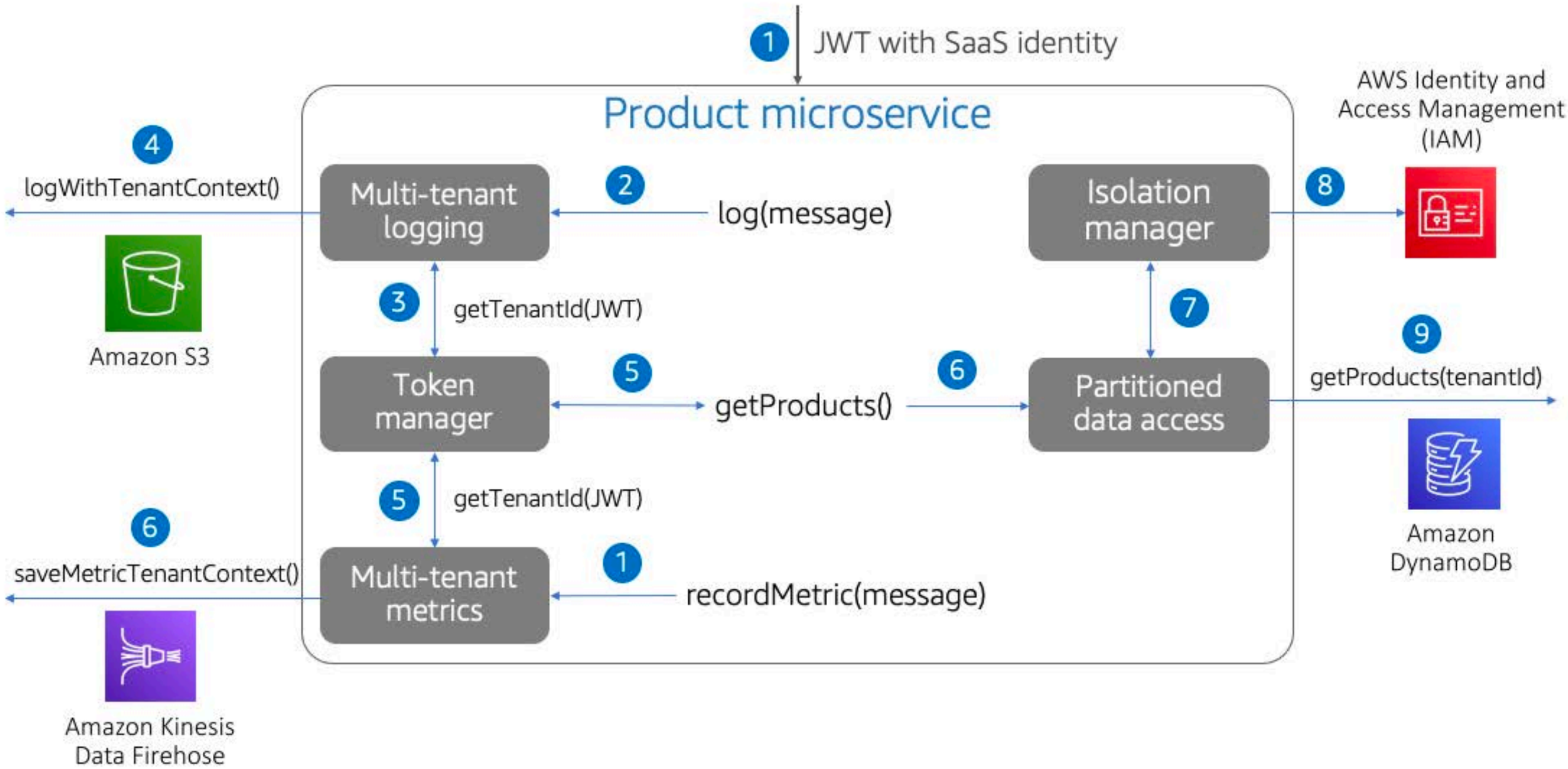
Multi tenant solution



Onboarding and Management



Multi-Tenant Microservices



https://docs.aws.amazon.com/wellarchitected/latest/saas-lens/tenant-insights.html

Amazon EBS Pricing

- Volumes
- Input/output operations per seconds (IOPS)
- Snapshots
- Data transfer

Design Resilient Architectures

Design Cost-Optimized Architectures

Sustainability Architectures

Design Performant Architectures

Operationally Excellent Architectures

Specify Secure Applications

Well Architected Framework

Customer is responsible for
sustainability **in** the cloud

Data Design & Usage

Software Application Design

Platform Deployments and Scaling

Data Storage | Code Efficiency | Utilization & Scaling

Servers | Cooling | Water | Waste

AWS Global Infrastructure

Data Centers

Electricity Supply

Building Materials

Customer
AWS

AWS is responsible for
sustainability **of** the cloud

https://sustainability.google/commitments/circular-economy/#

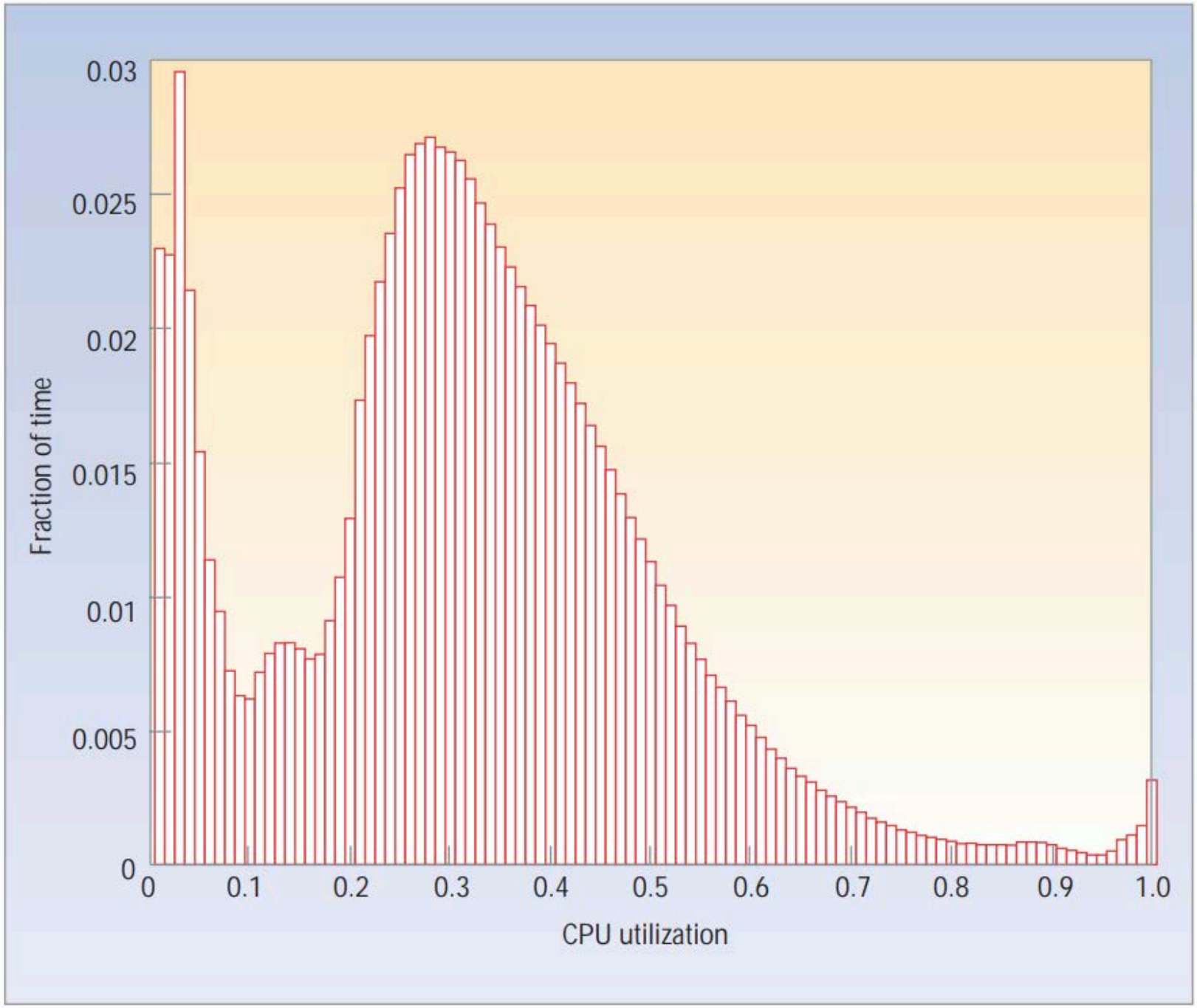


Figure 1. Average CPU utilization of more than 5,000 servers during a six-month period. Servers are rarely completely idle and seldom operate near their maximum utilization, instead operating most of the time at between 10 and 50 percent of their maximum utilization levels.

https://static.googleusercontent.com/media/research.google.com/en/us/pubs/archive/33387.pdf

compute resources should be maximized, on average 70-80%

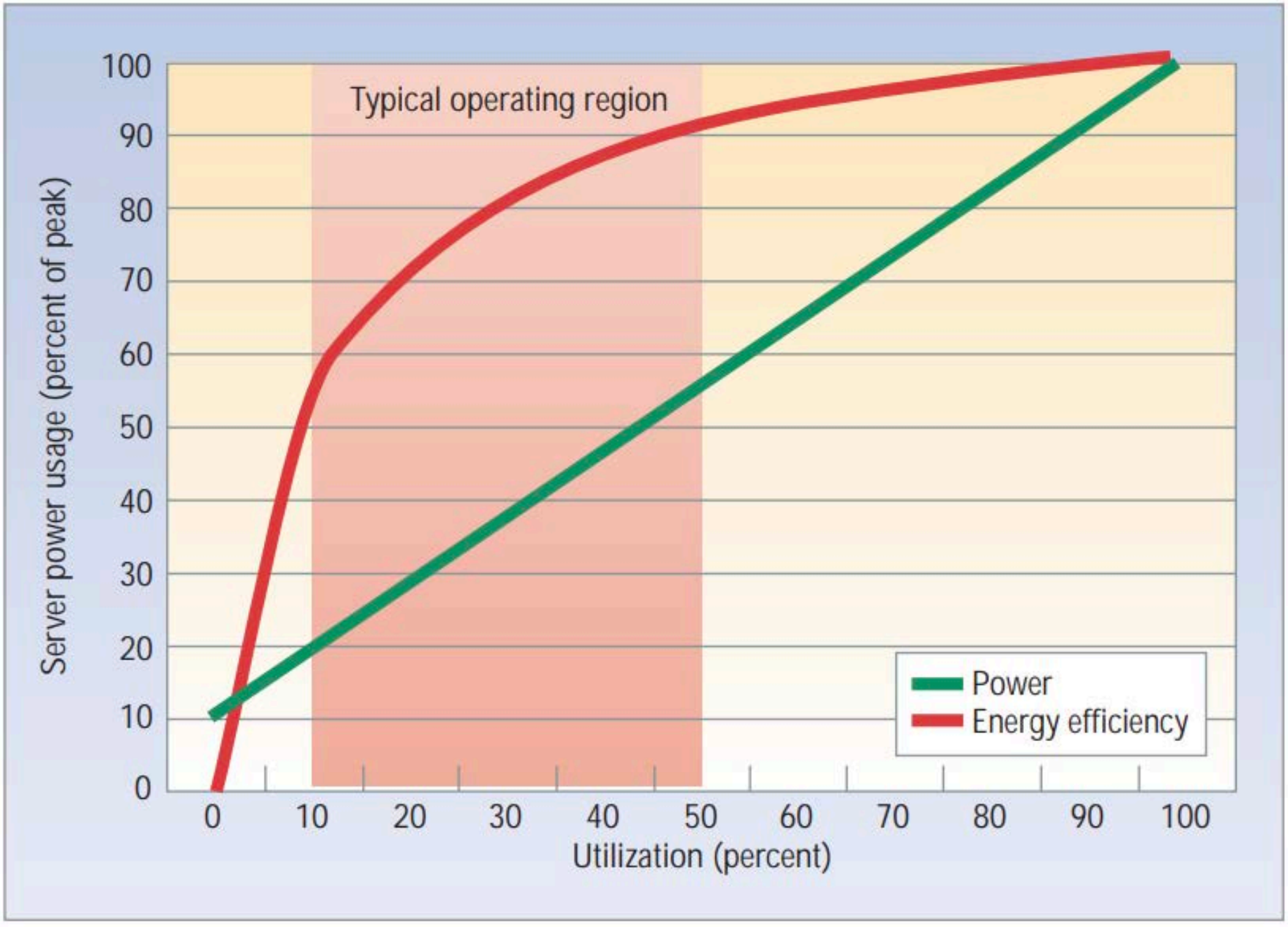
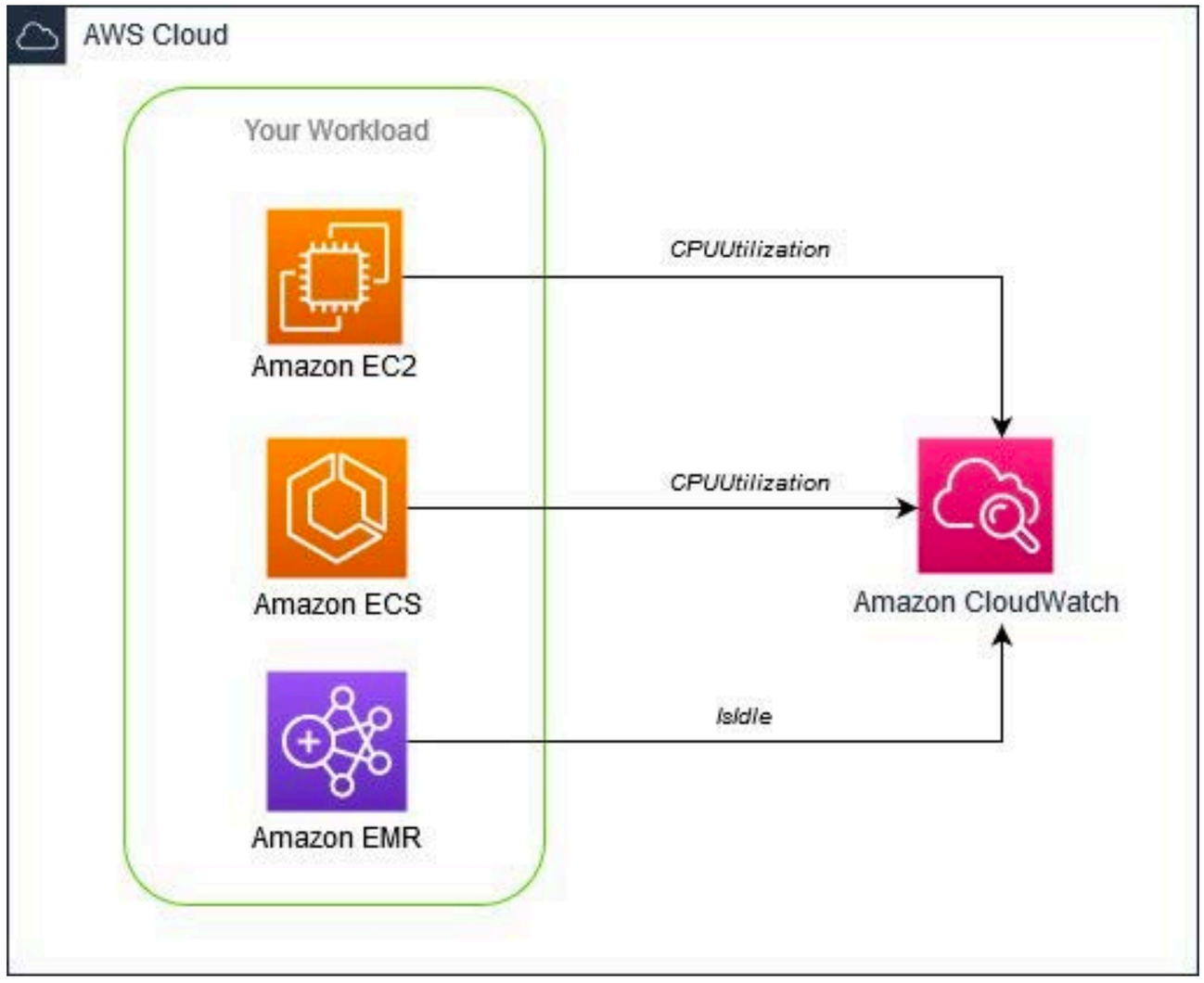
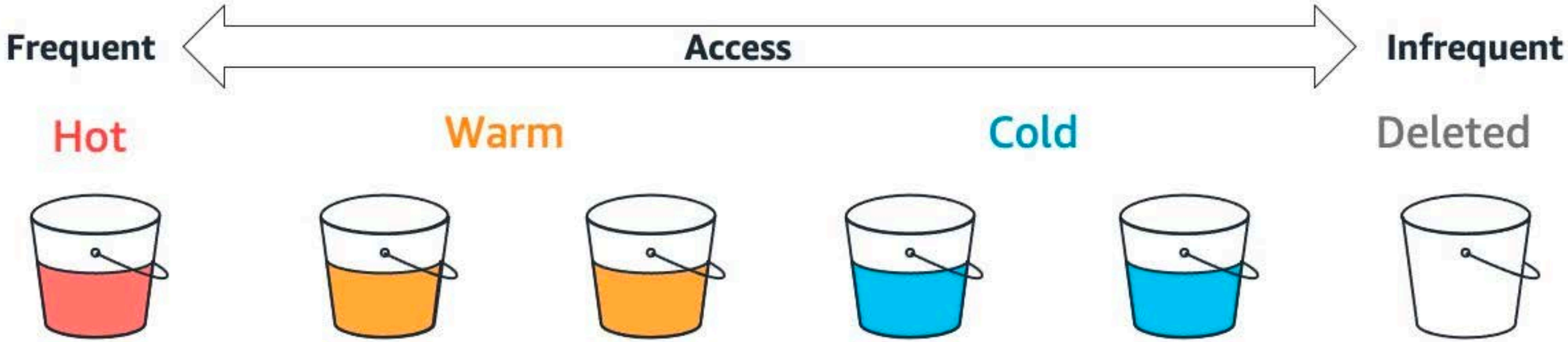


Figure 4. Power usage and energy efficiency in a more energy-proportional server. This server has a power efficiency of more than 80 percent of its peak value for utilizations of 30 percent and above, with efficiency remaining above 50 percent for utilization levels as low as 10 percent.

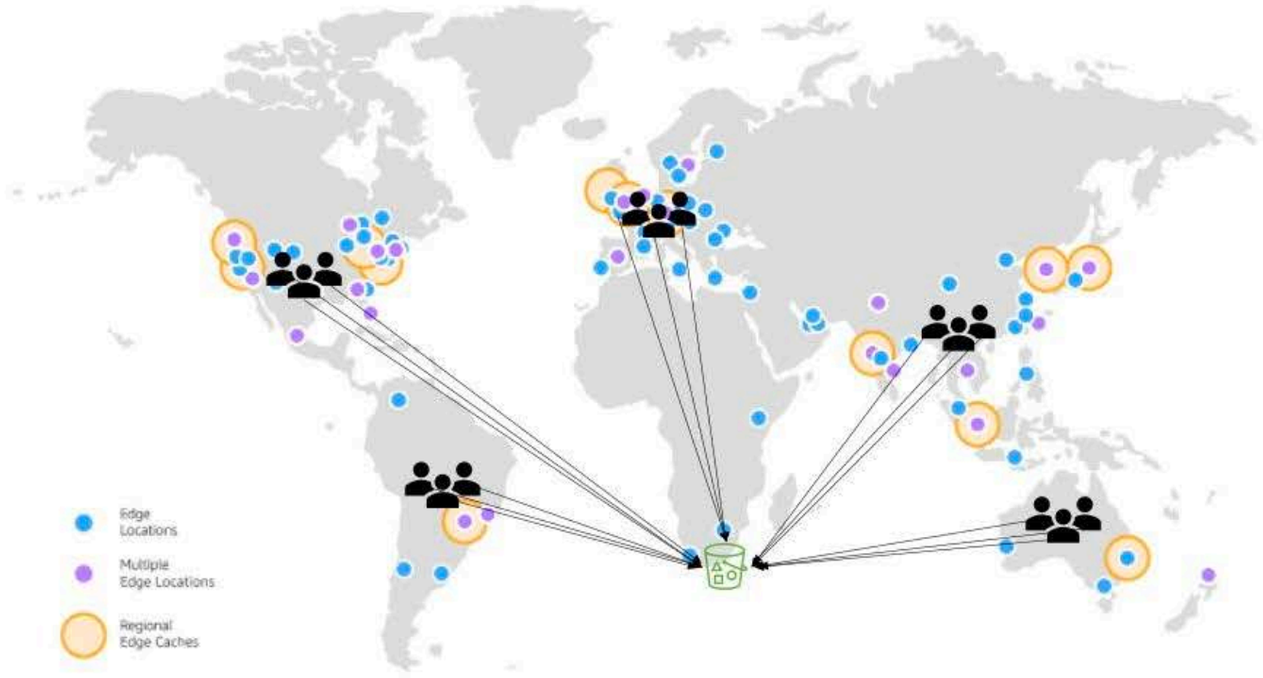

AWS Cloud
Your Workload
Amazon EC2
Amazon ECS
Amazon EMR
CPUUtilization
CPUUtilization
IsIdle
Amazon CloudWatch

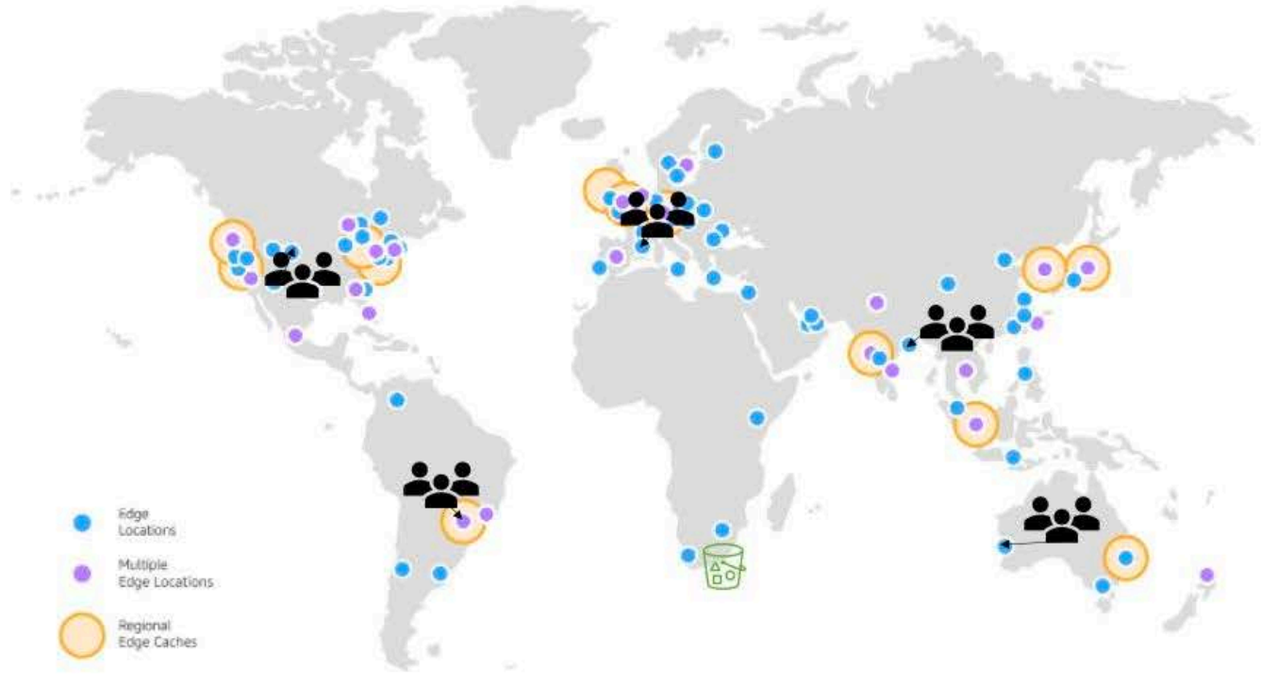


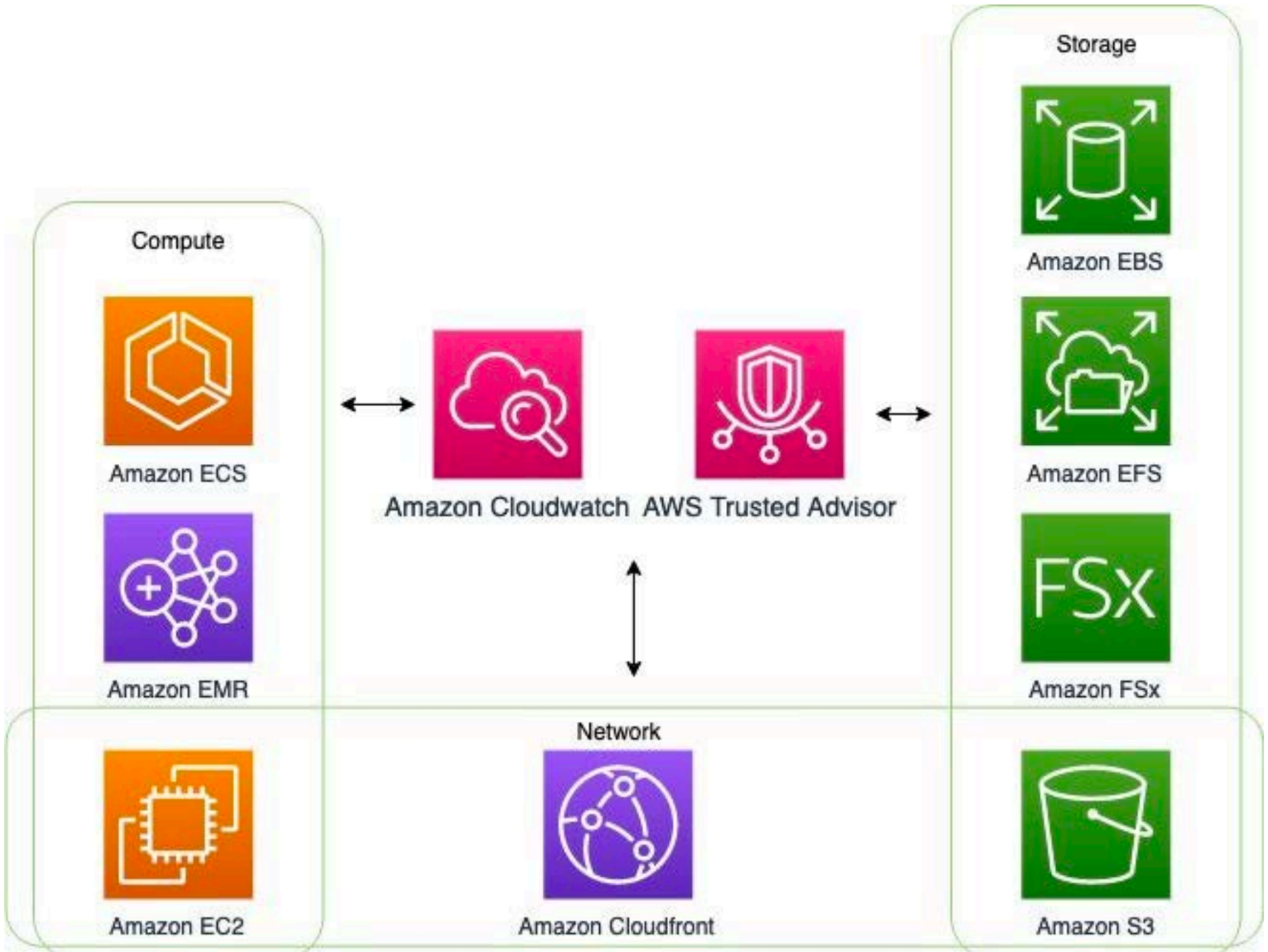
S3 Standard	S3 Standard-IA	S3 One Zone-IA	S3 Glacier	S3 Glacier Deep Archive	
Active data	Inactive data	Inactive data	Archive data	Archive data	Useless data
Millisecond access	Millisecond access	Millisecond access	Minutes to hours	Hours access	
≥ 3 AZ	≥ 3 AZ	1 AZ	≥ 3 AZ	≥ 3 AZ	0 AZ

https://aws.amazon.com/blogs/architecture/optimizing-your-aws-infrastructure-for-sustainability-part-ii-storage/

S3 Access


Edge
Locations
Multiple
Edge Locations
Regional
Edge Caches


Edge
Locations
Multiple
Edge Locations
Regional
Edge Caches


Compute
Amazon ECS
Amazon EMR
Amazon EC2
Amazon Cloudwatch
AWS Trusted Advisor
Network
Amazon Cloudfront
Storage
Amazon EBS
Amazon EFS
FSx
Amazon FSx
Amazon S3

Useful Calculations

- x Million users * y KB = xy GB
    - example: 1M users * a documents of 100KB per day = 100GB per day.
- x Million users * y MB = xy TB
    - example: 200M users * a short video of 2MB per day = 400TB per day.

https://matthewdbill.medium.com/back-of-envelope-calculations-cheat-sheet-d6758d276b05

Objects

- File: 100 KB
- Web Page: 100 KB (not including images)
- Picture: 200 KB
- Short Posted Video: 2MB
- Steaming Video: 50MB per minute
- Long/Lat: 8B

Lengths

- Maximum URL Size: ~2000
- ASCII charset: 128
- Unicode charset: 143, 859

Requests per second

Month	1 Billion	1 Million	1 Thousand
Day	32M	32K	32
Hour	1.3M	1.3K	1.30
Minute	22K	22.00	0.02
Second	400	0.4	0.0004
Day	**1 Billion**	**1 Million**	**1 Thousand**
Hour	42M	42K	42
Minute	700K	700	0.7
Second	12K	12	0.01

Shutterfly - Photo upload

- 100M photos (200KB) are uploaded daily to a server.
- 100 (number of millions) * 12 (the number per second for 1M) = 1200 uploads a second.
- 1200 (uploads) * 200KB (size of photo) = 240MB per second.
- The web servers will need to handle a network bandwidth of 240MB per second.

Usage-Users

- Facebook: 2.27B | YouTube: 2B | Instagram: 1B
- Pinterest: 332M | Twitter: 330M | Onedrive: 250M
- TikTok: 3.7M

similarweb

Rankings Data Solutions Pricing Resources Explore

Login Get started

Official Measure of the Digital World

Try it now. Similarweb is the fastest, easiest way to discover what's really happening online.

Analyze any website or app

Usage-Visits

similarweb Rankings Data Solutions Pricing Resources Explore Login Get started

Analyze any website or app

Rank	Website	Category	Change	Avg. Visit Duration	Pages / Visit	Bounce Rate
1	google.com	Computers Electronics and Technology > Search Engines	=	00:11:17	8.55	28.33%
2	youtube.com	Arts & Entertainment > TV Movies and Streaming	=	00:21:13	11.80	20.23%
3	facebook.com	Computers Electronics and Technology > Social Networks and Online Communities	=	00:10:03	8.46	32.73%
4	twitter.com	Computers Electronics and Technology > Social Networks and Online Communities	=	00:10:25	9.80	31.97%
5	instagram.com	Computers Electronics and Technology > Social Networks and Online Communities	=	00:07:51	11.06	34.86%
6	baidu.com	Computers Electronics and Technology > Search Engines	=	00:05:48	7.98	20.76%
7	wikipedia.org	Reference Materials > Dictionaries and Encyclopedias	-1	00:03:56	3.11	56.86%
8	yandex.ru	Computers Electronics and Technology > Search Engines	+1	00:11:24	9.36	22.10%
9	yahoo.com	News and Media	=	00:07:43	5.75	34.73%
10	xvideos.com	Adult	=	00:10:01	9.33	20.86%
11	whatsapp.com	Computers Electronics and Technology > Social Networks and Online Communities	+1	00:03:26	1.57	72.10%

Usage-Visits

- Facebook: 26.12B | Twitter: 6.34B | Pinterest: 1.32B
- Spotify: 293M | Ikea: 233M | Nike: 110M
- Argos: 54M | John Lewis: 37M |Superdry: 3.5M
- Virgin Money: 1.8M | Aviva: 1.61M

Requests per second

- SQL Databases
  - Storage: 60TB
  - Connections: 30K
  - Requests: 10-25K per second
  - Writes: 5k-10k per seconds

- Redis Cache
  - Storage: 300 GB
  - Connections: 10k
  - Requests: 100k per second

https://redis.io/topics/benchmarks

https://redis.io/topics/clients#:~:text=In%20Redis%202.4%20there%20was,conf.

Requests per second

- No-SQL
  - 20k-50k RPS
  - 10k-25k WPS
- Web Servers
  - Requests: 5–10k requests per second
- Queues/Streams
  - Requests: 1000–3000 requests/s (FIFO)
  - Throughput: 1MB-50MB/s (Write) / 2MB-100MB/s (Read)

https://cloud.google.com/pubsub/quotas

https://docs.aws.amazon.com/streams/latest/dev/service-sizes-and-limits.html

https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/quotas-messages.html

Two hosts running at 30% utilization are less efficient than one host running at 60%

KPIs

Resource	Example KPIs	Improvement goals
Compute	vCPU minutes per transaction	Maximize utilization of provisioned resources
Storage	GB per transaction	Reduce total provisioned
Network	GB transferred per transaction or packets transferred per transaction	Reduce total transferred and transferred distance

https://docs.aws.amazon.com/wellarchitected/latest/sustainability-pillar/evaluate-specific-improvements.html

Estimate improvement

Resource	Example KPIs	Improvement goals
Compute	% reduction of vCPUs minutes per transaction	Maximize utilization
Storage	% reduction GB per transaction	Reduce total provisioned
Network	% reduction of GB transferred per transaction or packets transferred per transaction	Reduce total transferred and transferred distance

Use the minimum amount of hardware to meet your needs

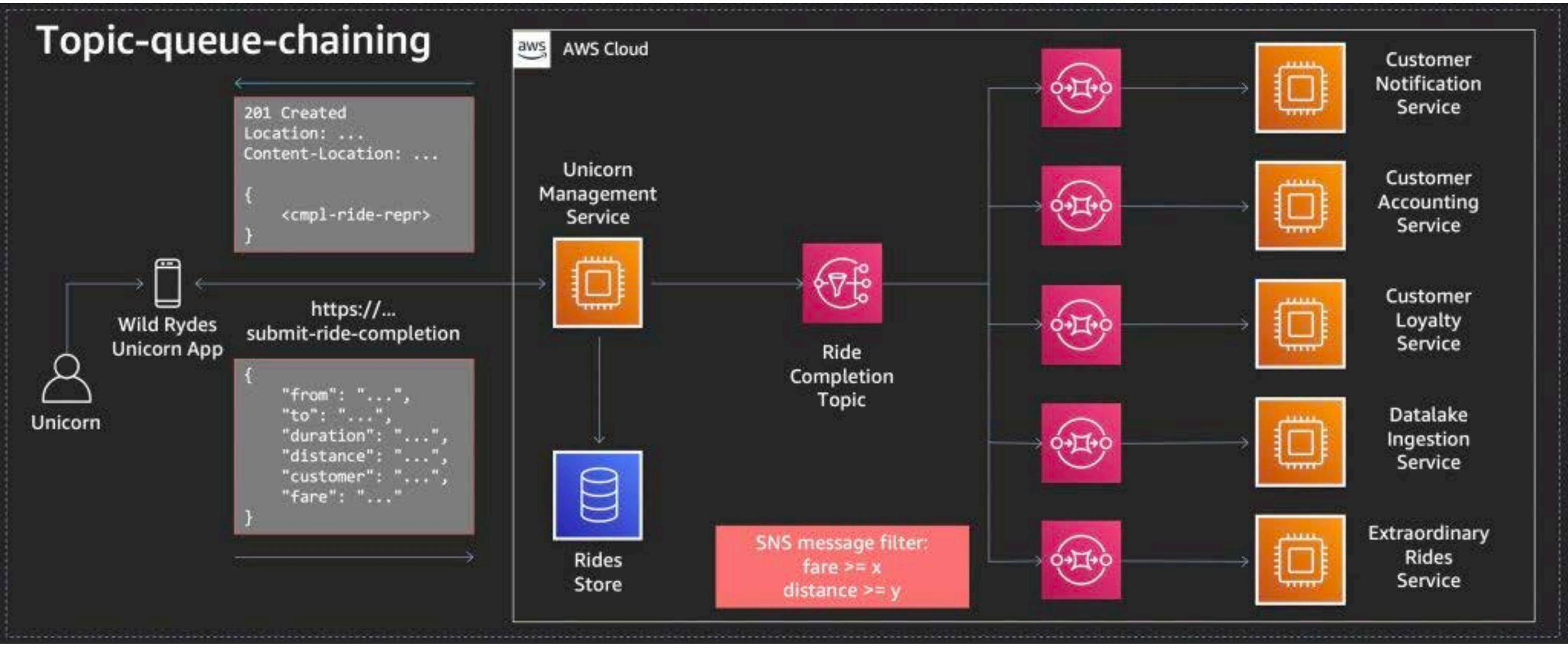
Estimate how many servers we need to serve read requests considering each server is capable of handling 7 requests per second. We have to serve a total traffic of 46 Million per day.

- 7 requests per second
- 7*60 = 420 requests per minute
- 420*60 requests per hour = 400 * 60 = 24000 per hour = 24K per hour
- 24K * 24 requests per day ≈ 30K*20 = 600K requests per day per server
- We want to handle 46 Million requests per day which means 46M/600K = approx 45M/500K = 45M/0.5M = 45*2 = 90 servers.

Use technologies that best support your data access and storage patterns

What if there is a spike?

Topic-queue-chaining pattern - flatten peak loads



Remove or refactor workload components with low or no use

Optimize areas of code that consume the most time or resources

- Monitor performance as a function of resource
- code profiler
- Replace algorithms with more efficient versions


ORDER
CHAOS

Principles of Chaos Engineering

I. Plan an Experiment

Create a hypothesis. What could go wrong?

II. Contain the Blast Radius

Execute the smallest test that will teach you something.

III. Scale or squash

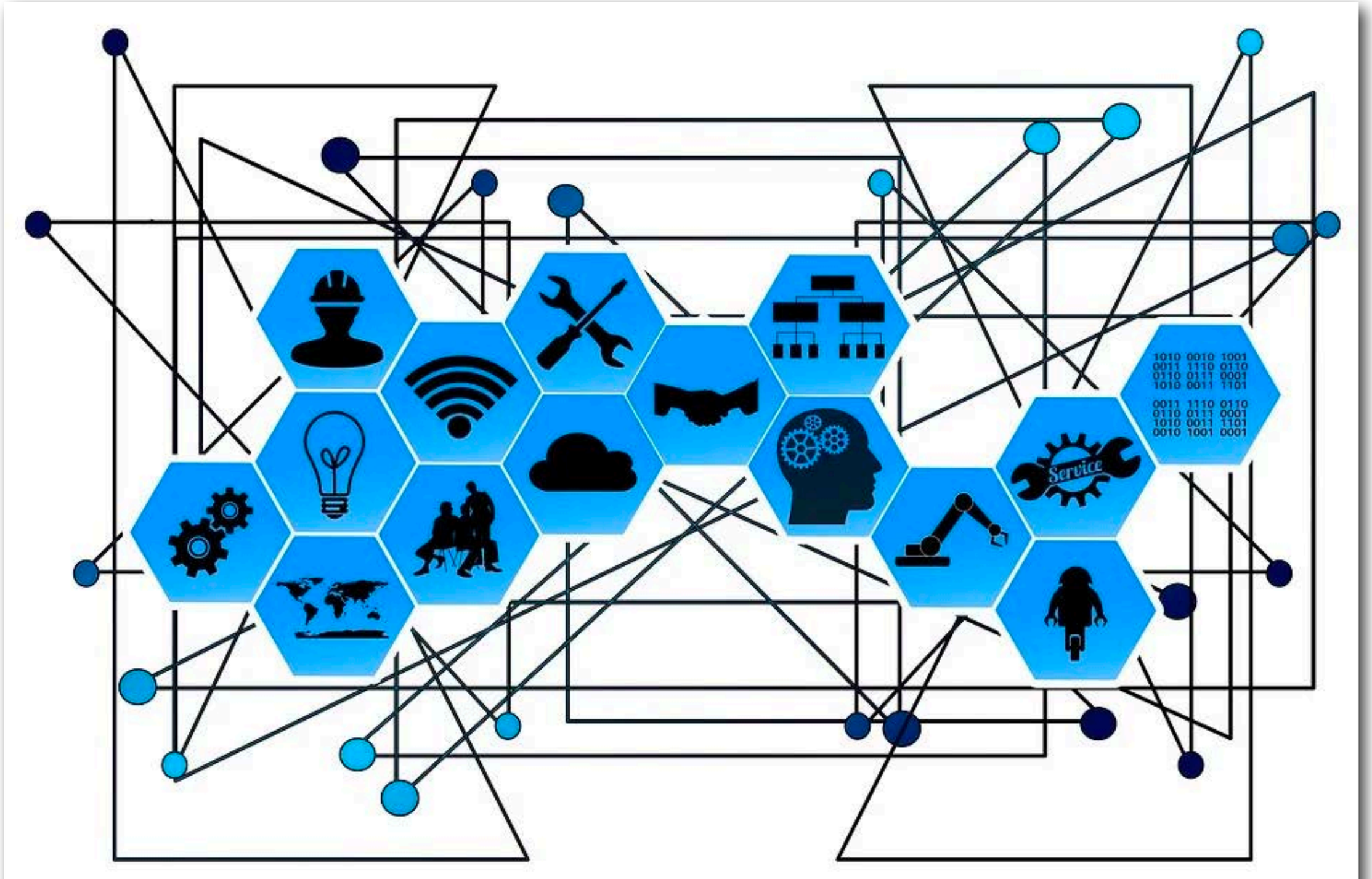
Find an issue? Job well done. Otherwise increase the blast radius until you're at full scale.

https://principlesofchaos.org/

Eight Fallacies of Distributed Computing

https://en.wikipedia.org/wiki/Fallacies_of_distributed_computing

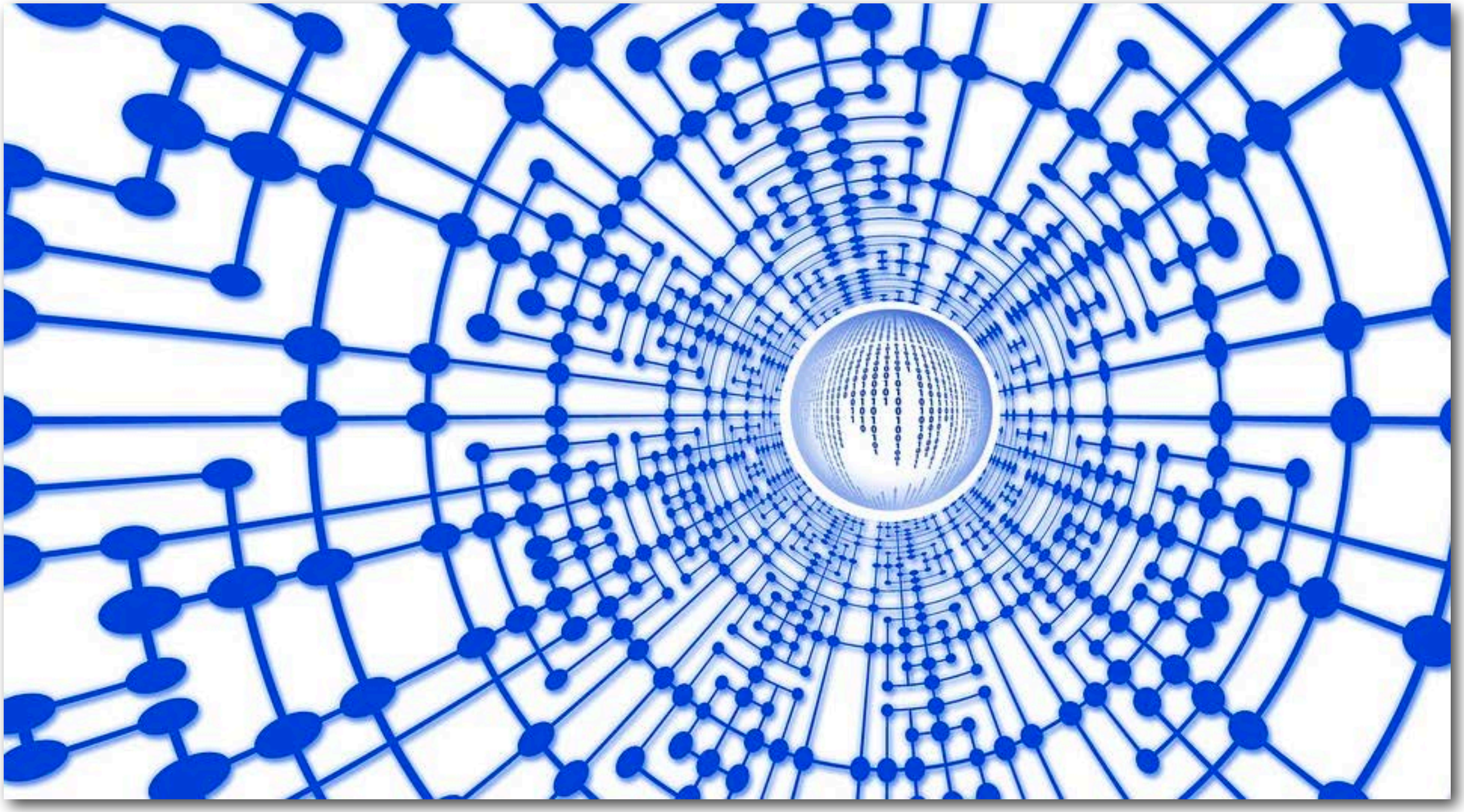
1. Network is Reliable



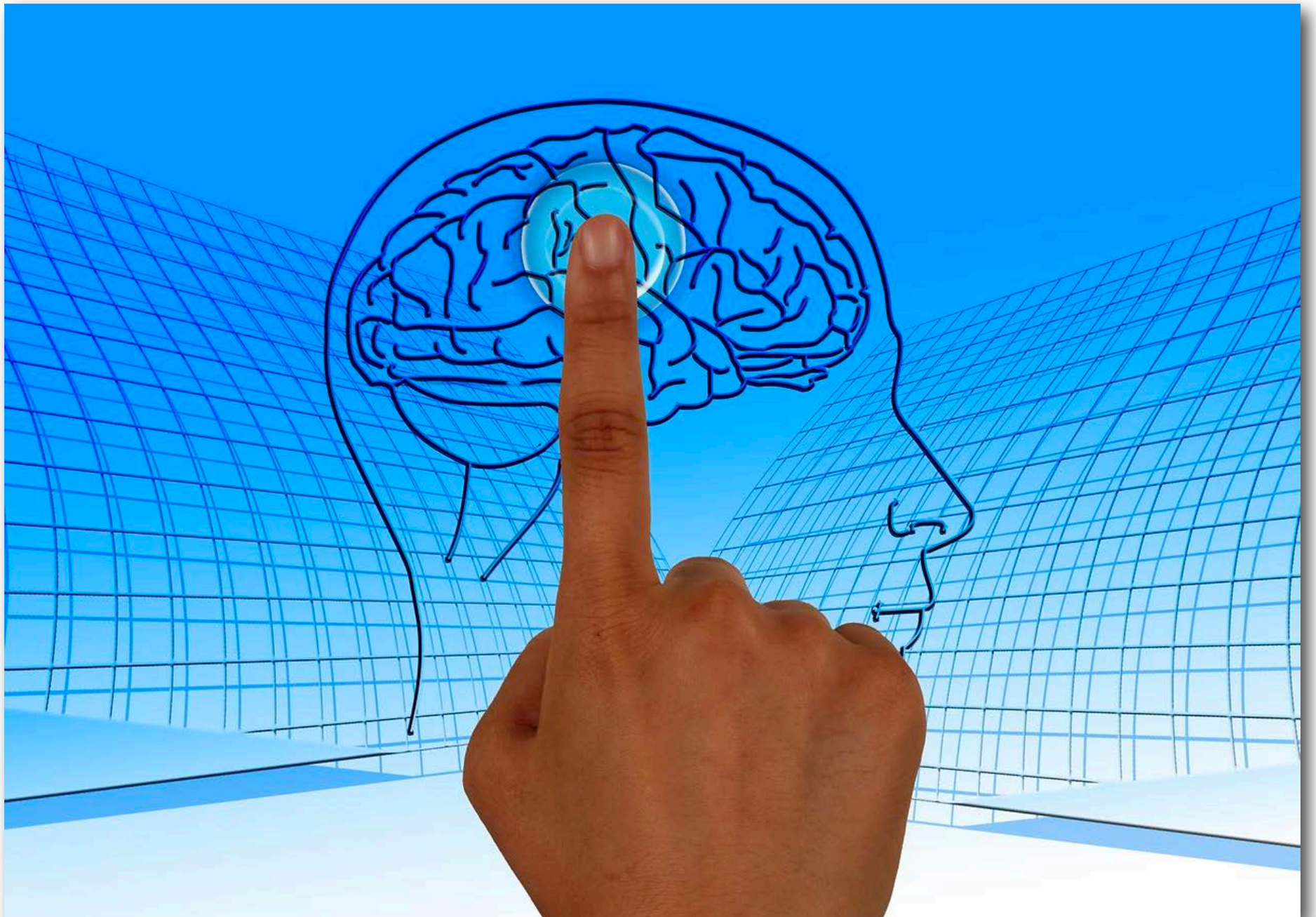
Automatic Retry

Sharding

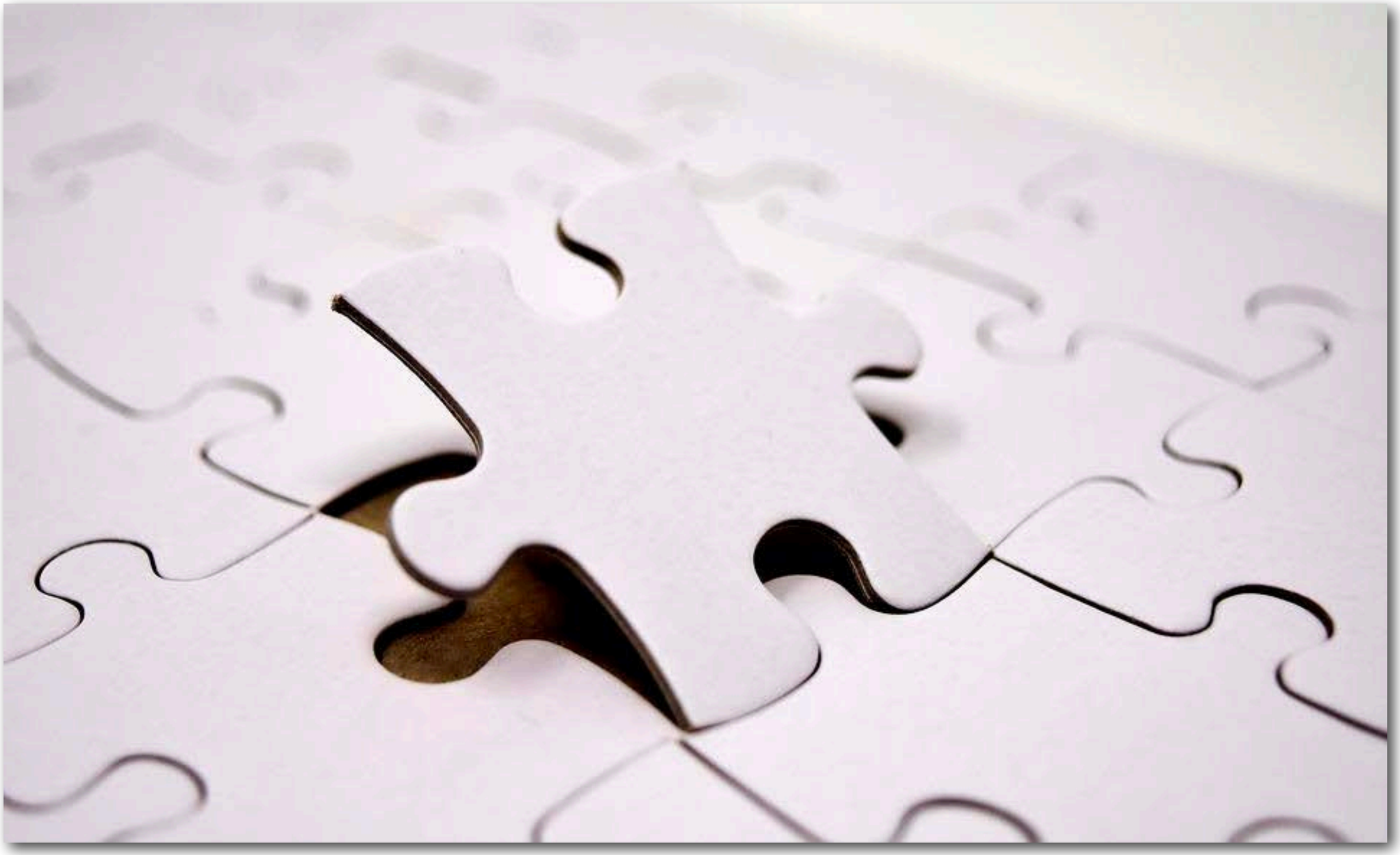
2. Latency is Zero



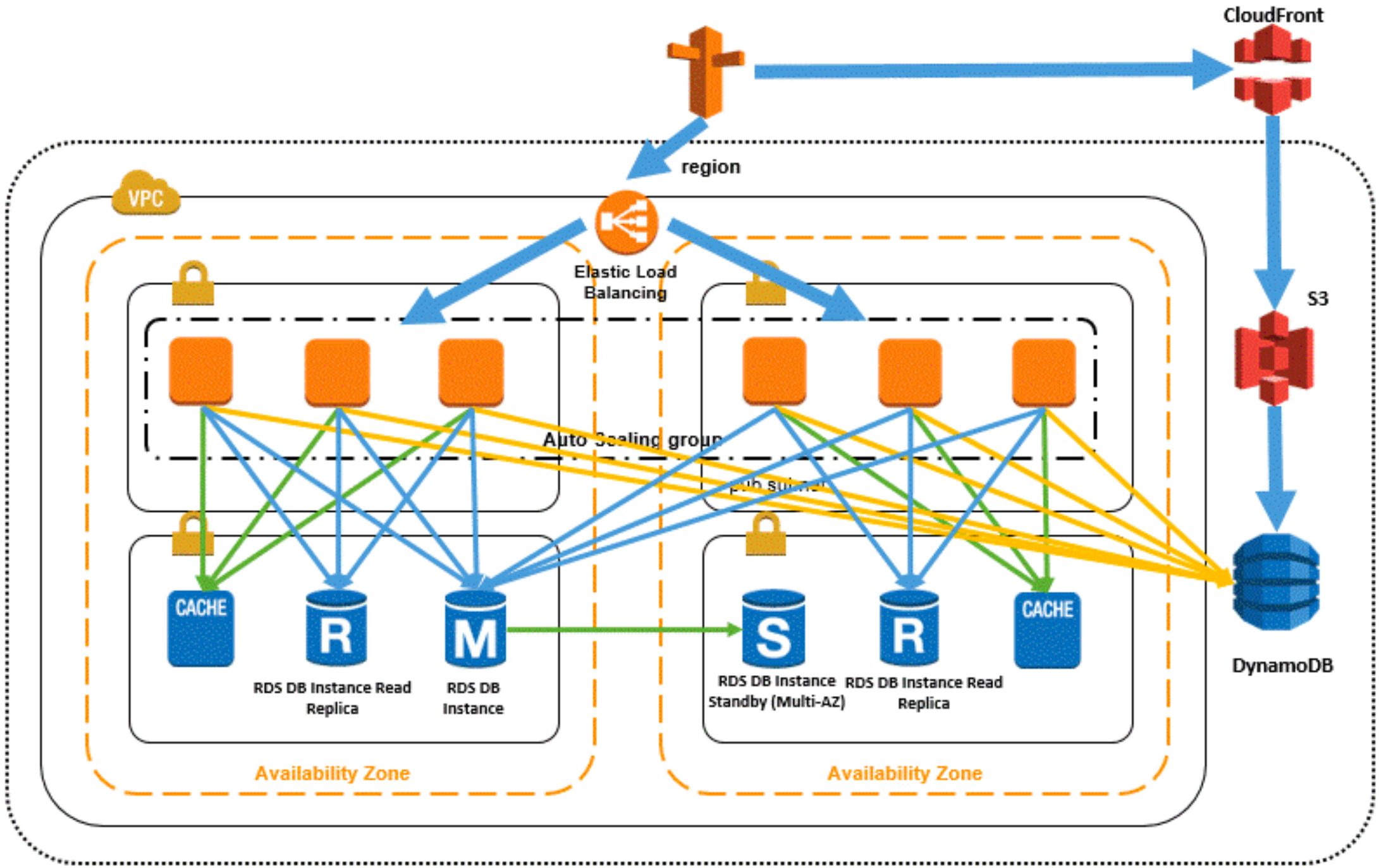
Caching



Bulk Requests

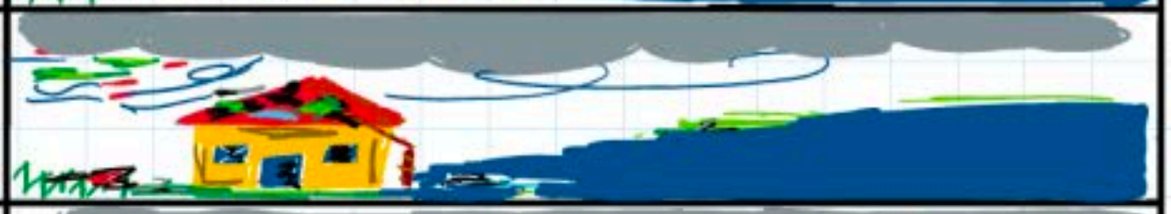
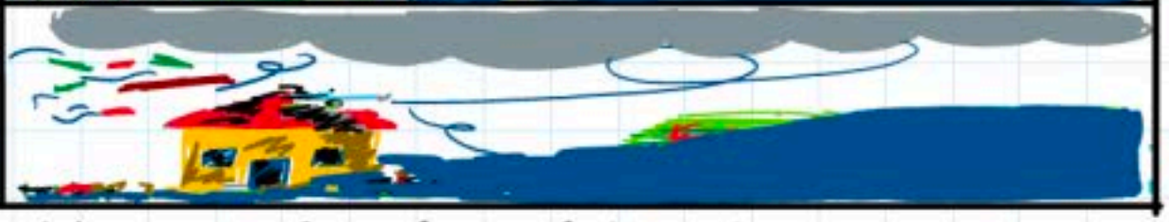


Multi AZ Deployments near Client



3. Bandwidth is Infinite

Hurricane Pattern

CATEGORY	WIND GUSTS (km/h)	EFFECTS
1	90-125	
2	125-164	
3	165-224	
4	225-279	
5	280 +	

SEVERE

* Adapted from the Bureau of Meteorology's Tropical Cyclone Category System.

https://www.abc.net.au/news/2019-02-20/category-system/10831006?nw=0

Smaller Payloads

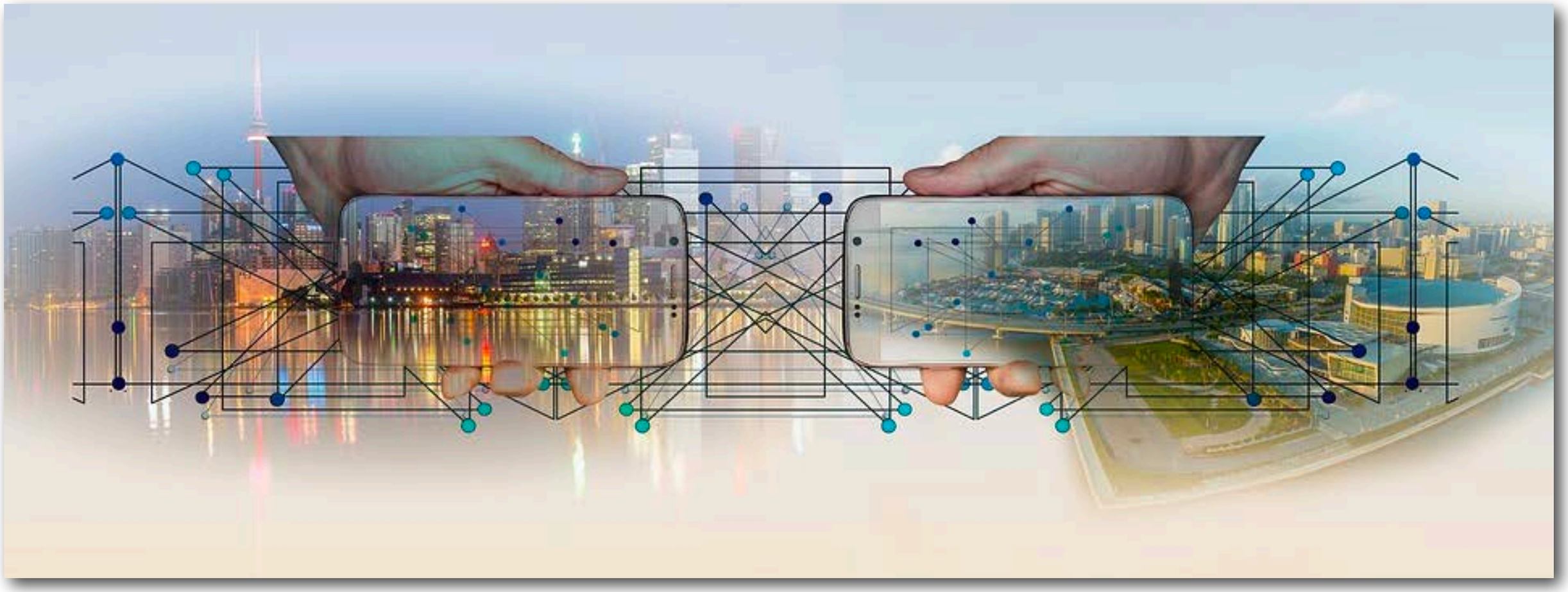
4. Network is Secure

Firewall

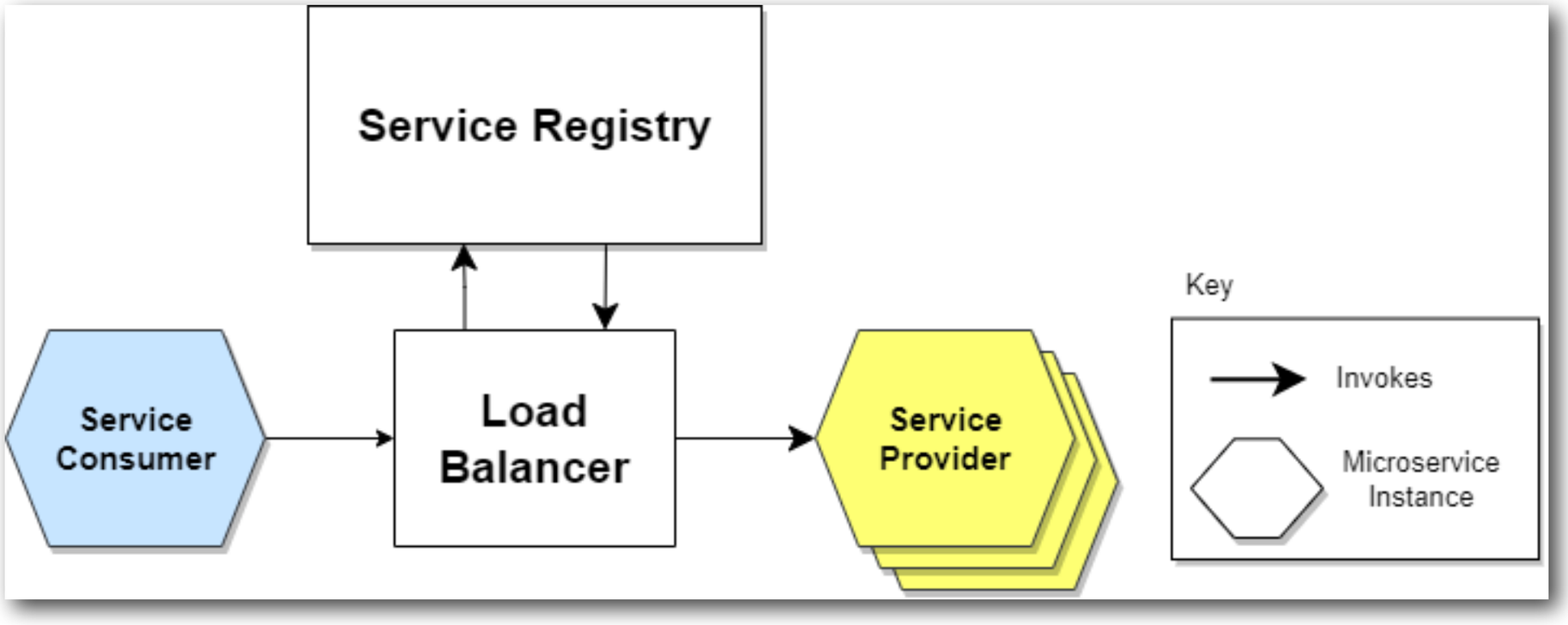
Encryption

Authentication

5. Topology never change



Service Provider



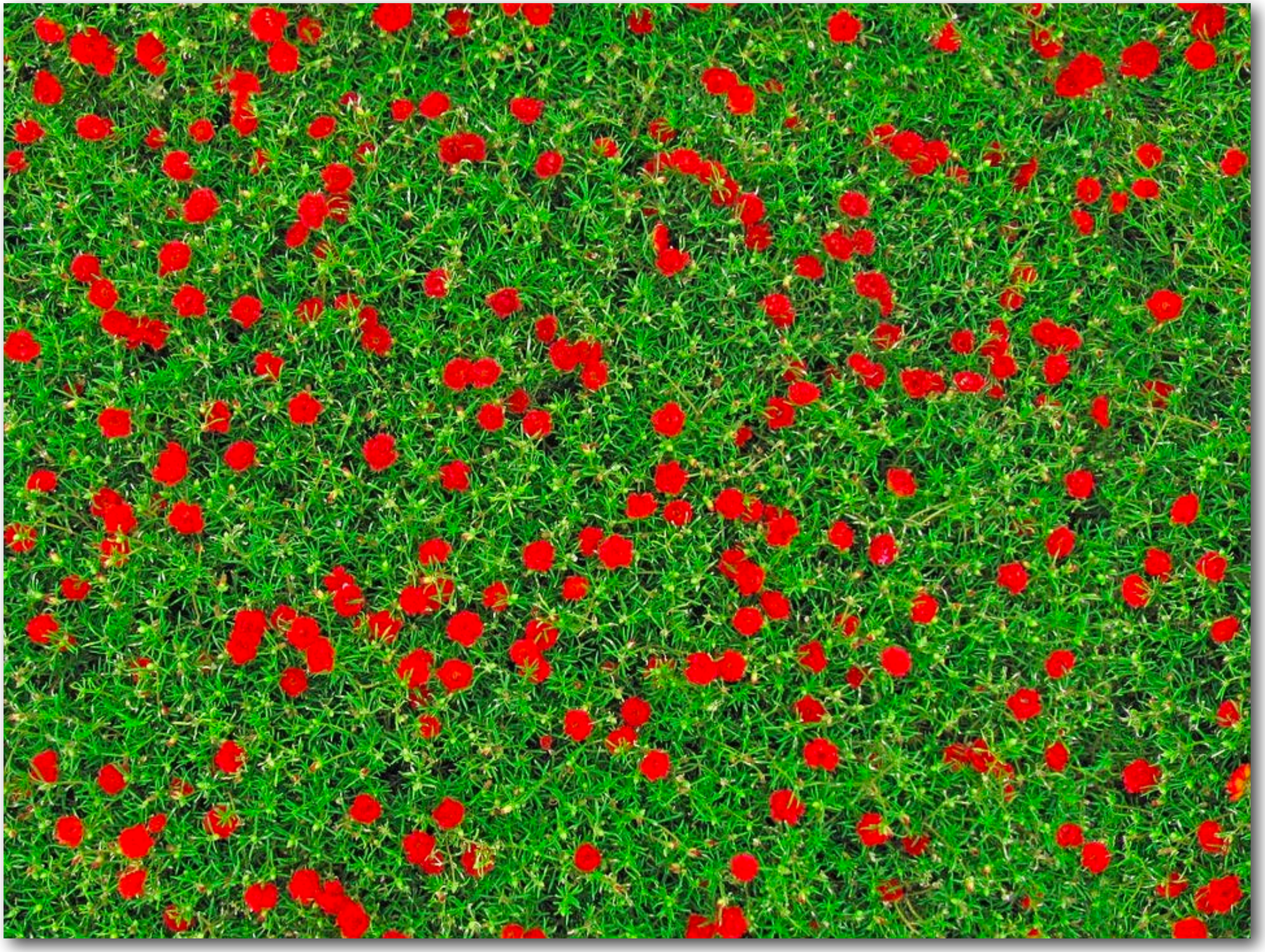
https://www.baeldung.com/cs/service-discovery-microservices

6. There is one Administrator

7. Transport Cost is Zero

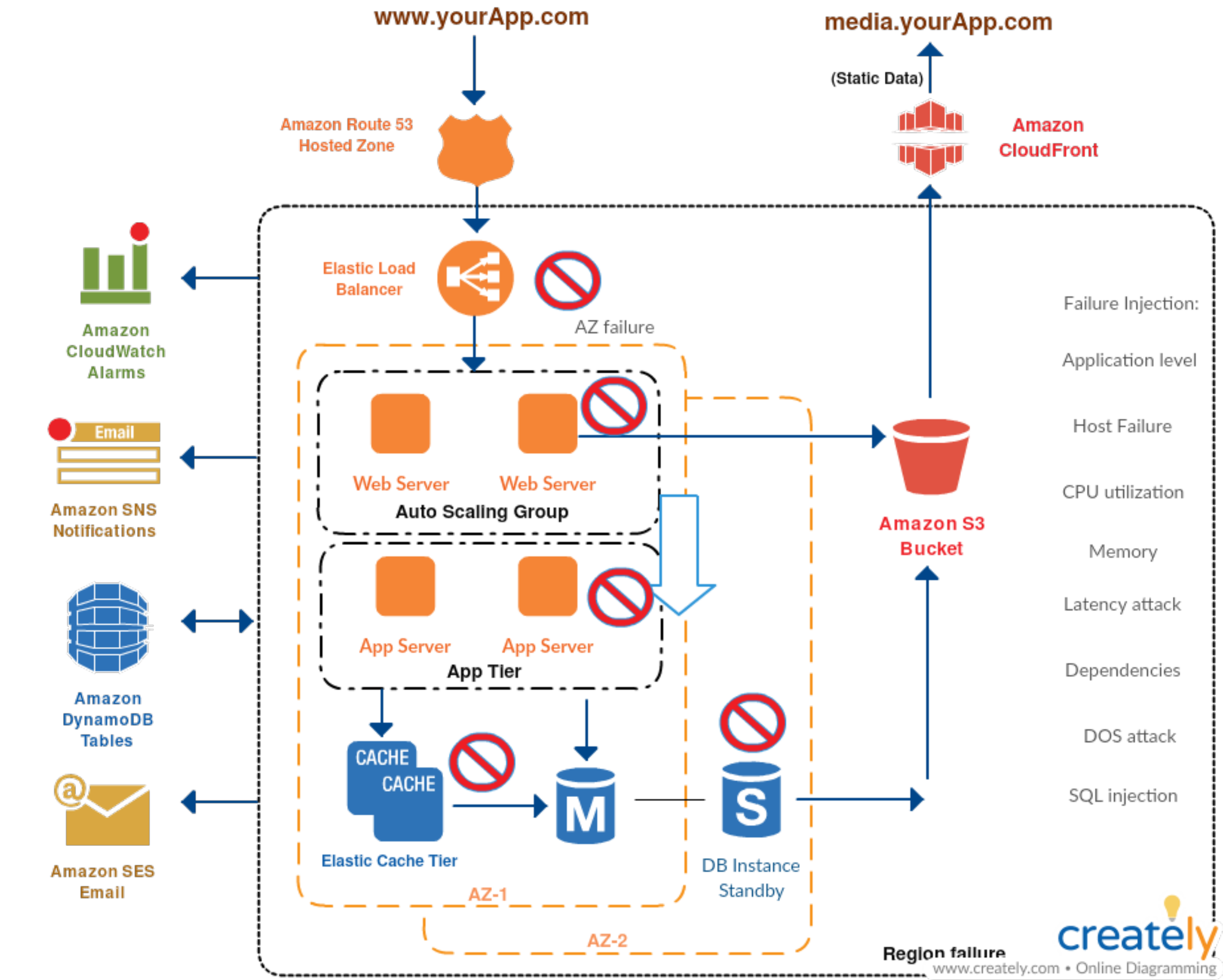
Cost calculation

8. Network is homogenous



Circuit Breaker

Chaos Engineering



Design Resilient Architectures

Design Cost-Optimized Architectures

Sustainability Architectures

Design Performant Architectures

Operationally Excellent Architectures

Specify Secure Applications

Well Architected Framework

https://wellarchitectedlabs.com/

wellarchitectedlabs.com

AWS
Well-Architected Labs

Search...

- Operational Excellence
- Security
- Reliability
- Performance Efficiency
- Cost Optimization
- Well-Architected Tool

Introduction

The Well-Architected framework has been developed to help cloud architects build the most secure, high-performing, resilient, and efficient infrastructure possible for their applications. This framework provides a consistent approach for customers and partners to evaluate architectures, and provides guidance to help implement designs that will scale with your application needs over time.

This repository contains documentation and code in the format of hands-on labs to help you learn, measure, and build using architectural best practices. The labs are categorized into levels, where 100 is introductory, 200/300 is intermediate and 400 is advanced.

Prerequisites:

An AWS account that you are able to use for testing, that is not used for production or other purposes. NOTE: You will be billed for any applicable AWS resources used if you complete this lab that are not covered in the AWS Free Tier.

Thanks

Rohit Bhardwaj

Hands-on Senior Architect, Salesforce

Founder: ProductiveCloudInnovation.com
Twitter: rbhardwaj1
LinkedIn: www.linkedin.com/in/rohit-bhardwaj-cloud

tinyurl.com/DesigningWellArchitected

https://www.productivecloudinnovation.com/lessons