Architecting Good Cloud Solutions - Well Architected Framework (WAF)

ECE 4150

Vijay Madisetti

Spring 2024

Cloud Solutions Requirements

AVAILABILITY

DATA MANAGEMENT

MANAGEMENT

MONITORING

MESSAGING

PERFORMANCE

SCALABILITY

SECURITY

How We Learn Technology and Create New Technology

technical knowledge. Bloom classified learning in the classroom into the following levels.

- Knowledge: Student learns terminology, facts, and definitions, including benefits of applying the technology under study.
- Comprehension: Student can make use of ideas and material without seeing their full implication. Extrapolation to new situations is possible in limited context.
- Application: Student can apply knowledge to practical cases through the use of tools.
- Analysis: Student can break down the components of a system, and can identify hierarchies and relationships between elements. Organizational structures and assumptions (unstated) can be recognized.
- Synthesis: Student is able to synthesize a system from start, using decomposition methods or otherwise. This include ability to produce a plan to design and implement the system, and a mechanism to verify that the plan works and will achieve objectives.
- Evaluation: Student can evaluate, compare, critique, and judge various alternative solutions and improve upon the product.

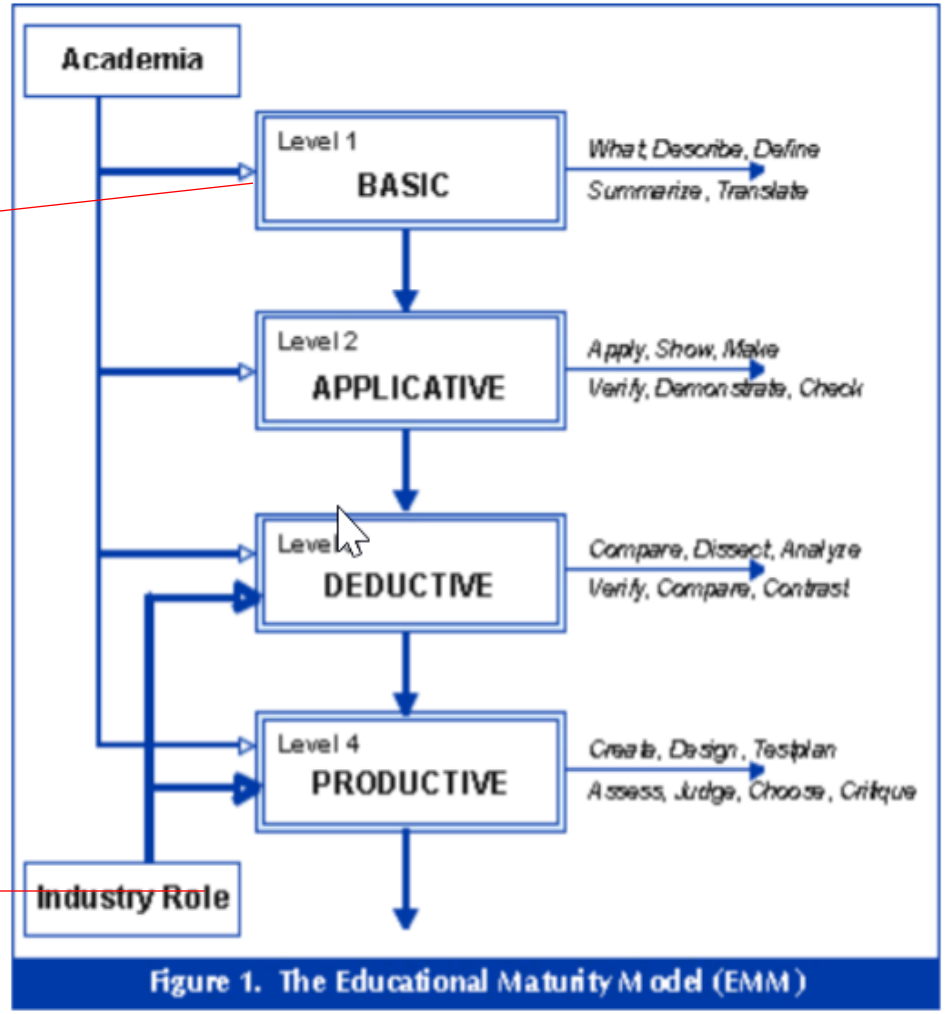
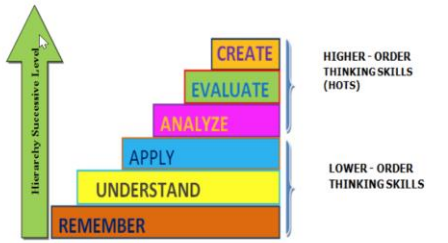


Figure 1. The Educational Maturity Model (EMM)

3. Educational Maturity Model (EMM)



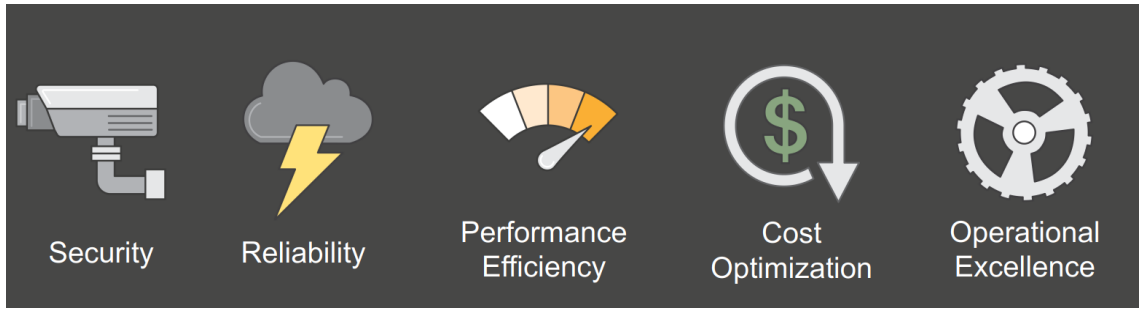
BLOOM'S TAXONOMY OF LEARNING OUTCOMES

..... Moving Towards Perfection in Engineering

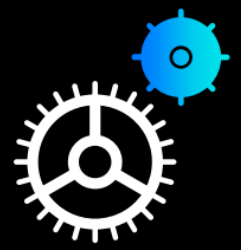
What is WAF?

Well-Architected Framework

https://aws.amazon.com/well-architected/



Pillars of AWS Well-Architected



Operational excellence

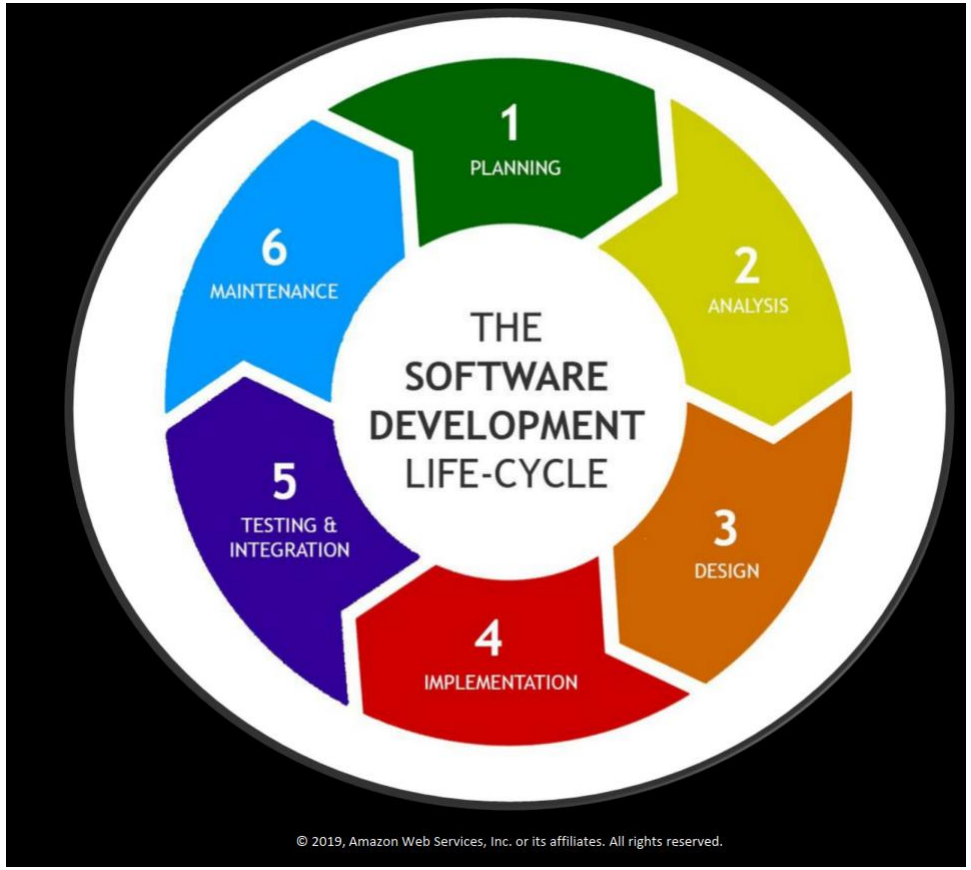
Security

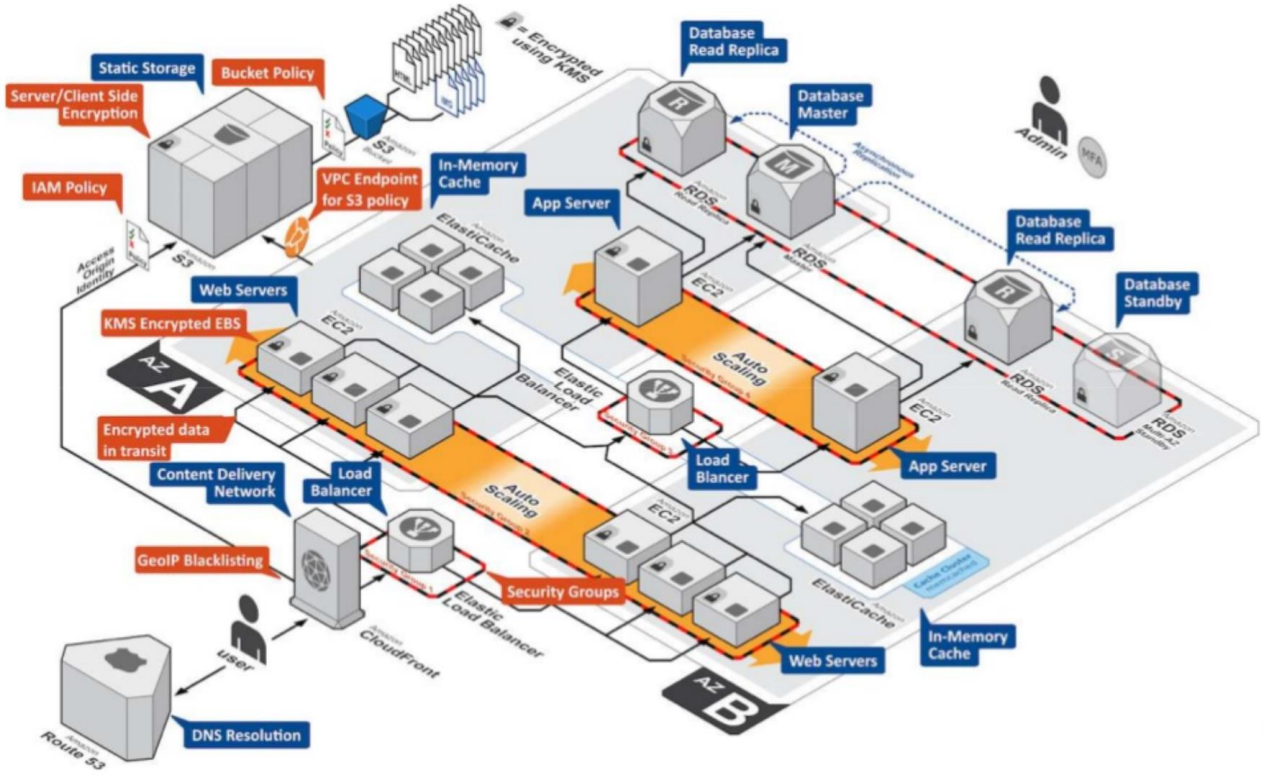
Reliability

Performance efficiency

Cost optimization

aws SUMMIT


1
PLANNING
2
ANALYSIS
3
DESIGN
4
IMPLEMENTATION
5
TESTING & INTEGRATION
6
MAINTENANCE
THE SOFTWARE DEVELOPMENT LIFE-CYCLE
© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.


= Encrypted using KMS
Static Storage
Bucket Policy
Server/Client Side Encryption
IAM Policy
VPC Endpoint for S3 policy
In-Memory Cache
App Server
Database Read Replica
Database Master
Admin
Database Read Replica
Database Standby
Web Servers
KMS Encrypted EBS
Encrypted data in transit
Content Delivery Network
Load Balancer
GeoIP Blacklisting
Security Groups
Load Blancer
App Server
In-Memory Cache
Web Servers
DNS Resolution
user
Route 53
CloudFront
Elastic Load Balancer
ElastiCache
Auto Scaling
EC2
RDS
S3
AZ A
AZ B

General Design Principles

- Stop guessing your capacity needs
- Test systems at production scale
- Automate to make architectural experimentation easier
- Allow for evolutionary architectures
- Build data-driven architectures
- Improve through game days

amazon web services

Design Principles for Security

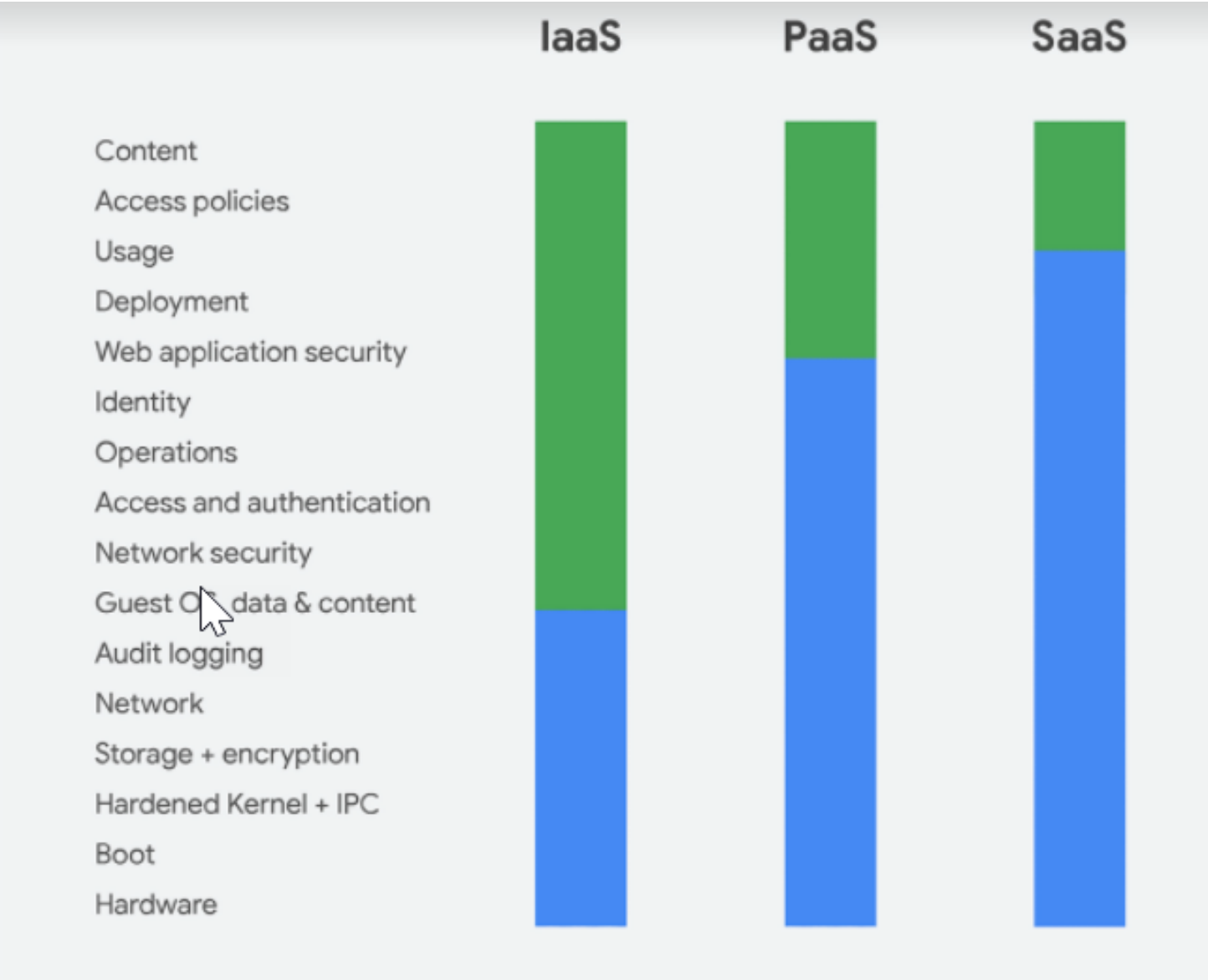
Apply security at all layers

Enable traceability

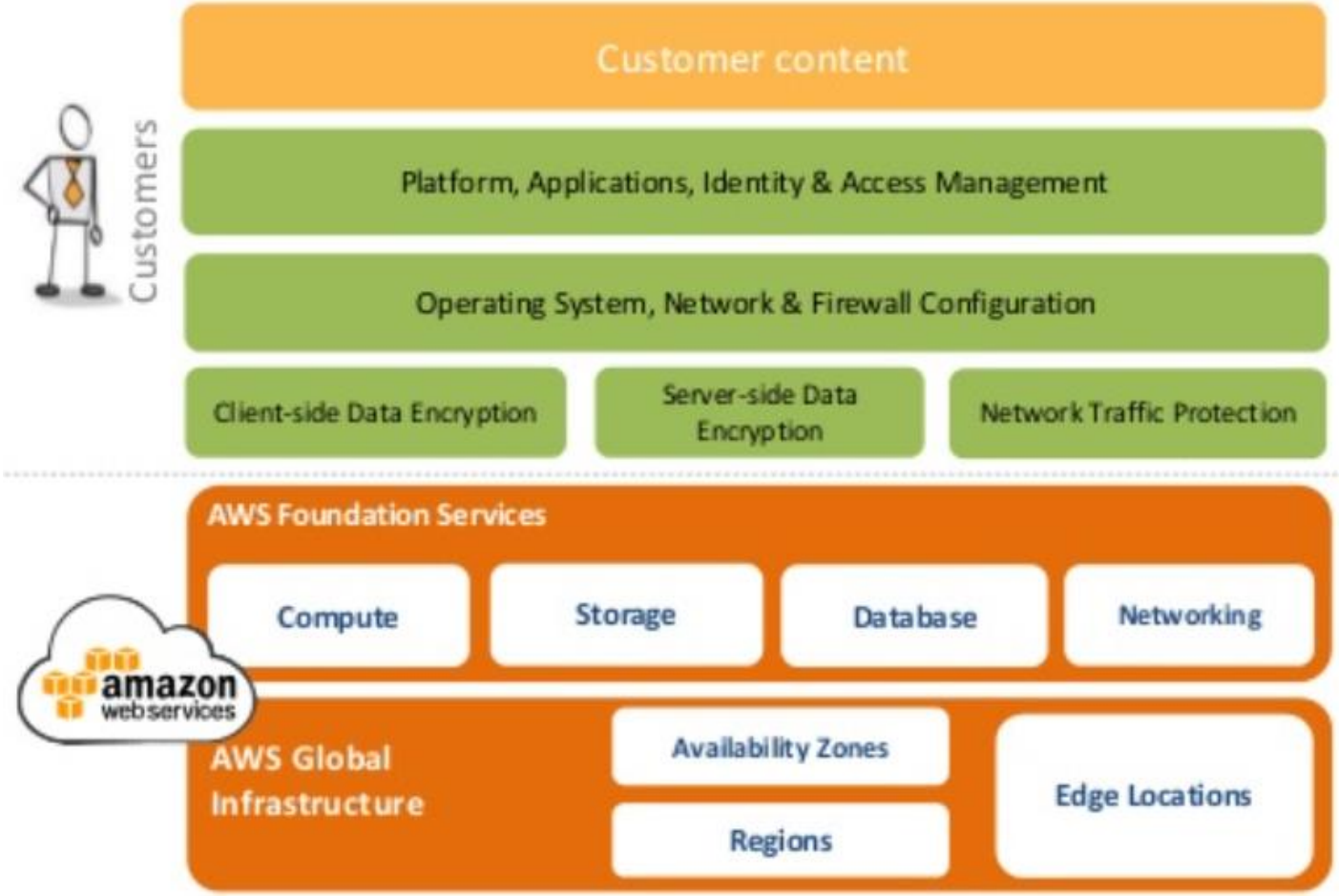
Implement a principle of least privilege

Focus on securing your system

Automate security best practices


IaaS
PaaS
SaaS
Content
Access policies
Usage
Deployment
Web application security
Identity
Operations
Access and authentication
Network security
Guest OS data & content
Audit logging
Network
Storage + encryption
Hardened Kernel + IPC
Boot
Hardware

Shared Responsibility between AWS and our customers



Customers are responsible for their security IN the Cloud

AWS is responsible for the security OF the Cloud

Design Principles for Operational Excellence

- Align Operations Processes to Business Objectives
- Perform Operations with Code
- Make Regular, Small, Incremental Changes
- Test for Responses to Unexpected Events
- Learn from Operational Events and Failures
- Keep Operations Procedures Current

amazon web services

Key Services for Operational Excellence

AWS CloudFormation

Areas	Key Services
Preparation	AWS Developer Tools, AWS CloudFormation, AWS Config, Lambda, RunCommand, Batch
Operations	AWS Developer Tools, AWS CloudFormation, AWS Config, AWS CloudTrail, Amazon CloudWatch, Lambda, RunCommand, Batch
Responses	AWS Developer Tools, AWS CloudFormation, AWS Config, AWS CloudTrail, Amazon CloudWatch, Lambda, RunCommand, Batch

Design Principles for Cost Optimization

Adopt a consumption model

Benefit from economies of scale

Stop spending money on data center operations

Analyze and attribute expenditure

Use managed services to reduce cost of ownership

amazon web services

Key Services for Cost Optimization

Cost Allocation Tags

Areas	Key Services
Cost-effective resources	Reserved Instances; AWS Trusted Advisor
Matched supply and demand	Auto Scaling
Expenditure awareness	Amazon CloudWatch; Amazon SNS
Optimizing over time	AWS Blog & What's New

Design Principles for Performance Efficiency

Democratize advanced technologies

Go global in minutes

Use serverless architectures

Experiment more often

Mechanical sympathy

amazon web services

Key Services for Performance Efficiency

Amazon CloudWatch

Areas	Key Services
Selection	Amazon EBS, Auto Scaling, Amazon S3, Amazon Glacier, Amazon RDS, Amazon DynamoDB
Review	AWS CloudFormation, AWS Blog
Monitoring	Amazon CloudWatch, AWS Lambda
Trade-Off	Amazon CloudFront, Amazon Elasticache, AWSSnowball

Design Principles for Reliability

Test recovery procedures

Automatically recover from failure

Scale horizontally to increase aggregate system availability

Stop guessing capacity

Manage change in automation

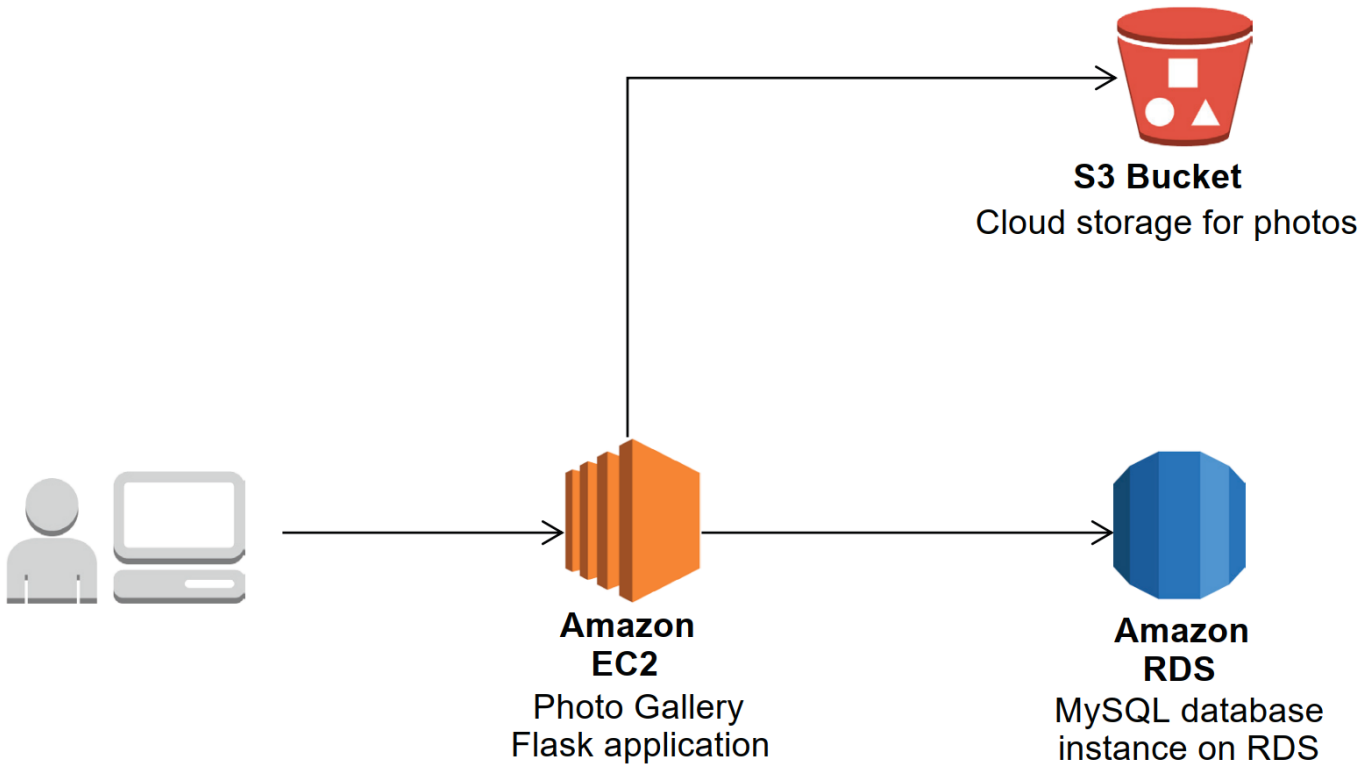
amazon web services

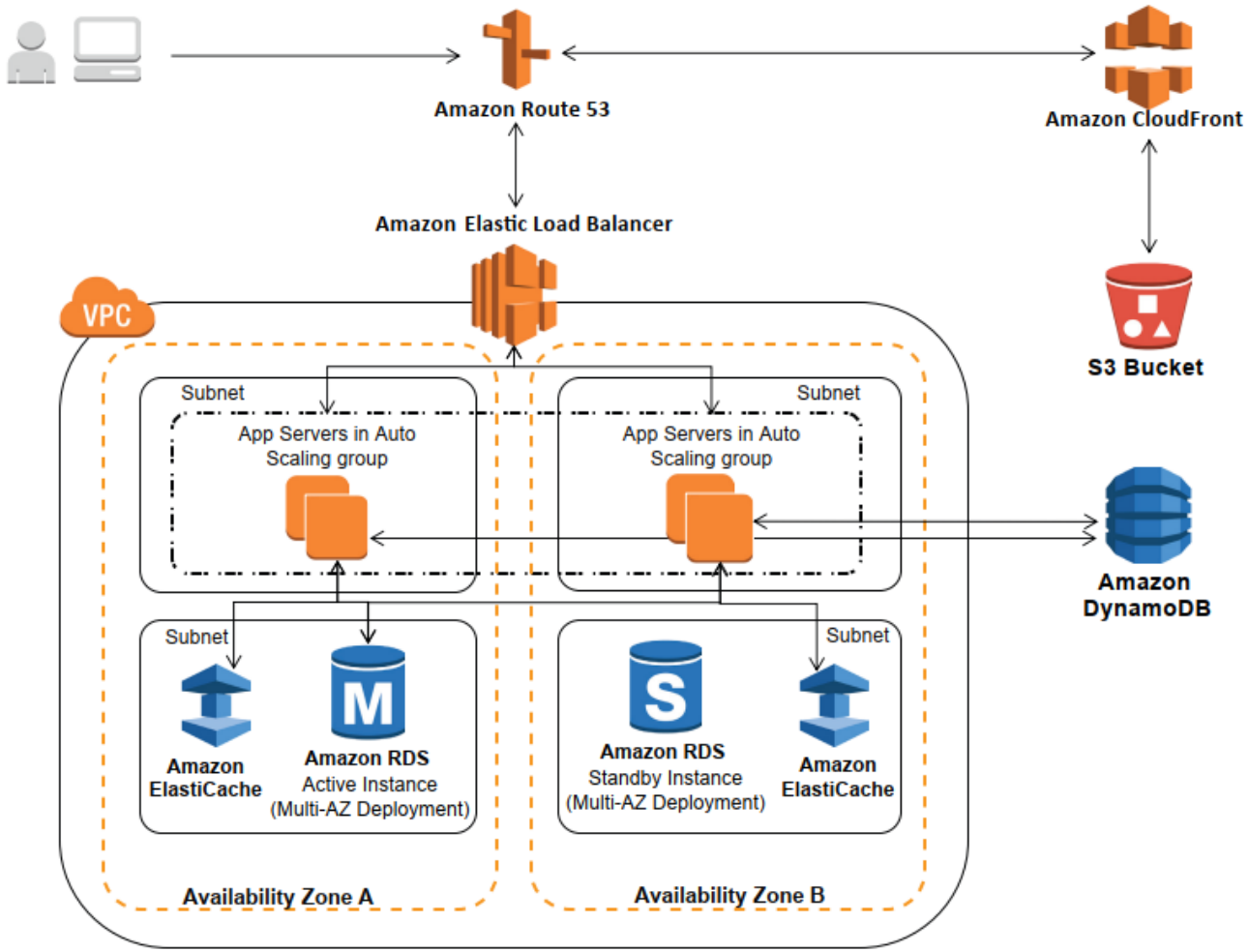
Key Services for Reliability

Amazon CloudWatch

Areas	Key Services
Foundations	AWS IAM, Amazon VPC
Change management	AWS CloudTrail, AWS Config
Failure management	AWS CloudFormation

Lab 2 Architecture – Non-Production Deployments



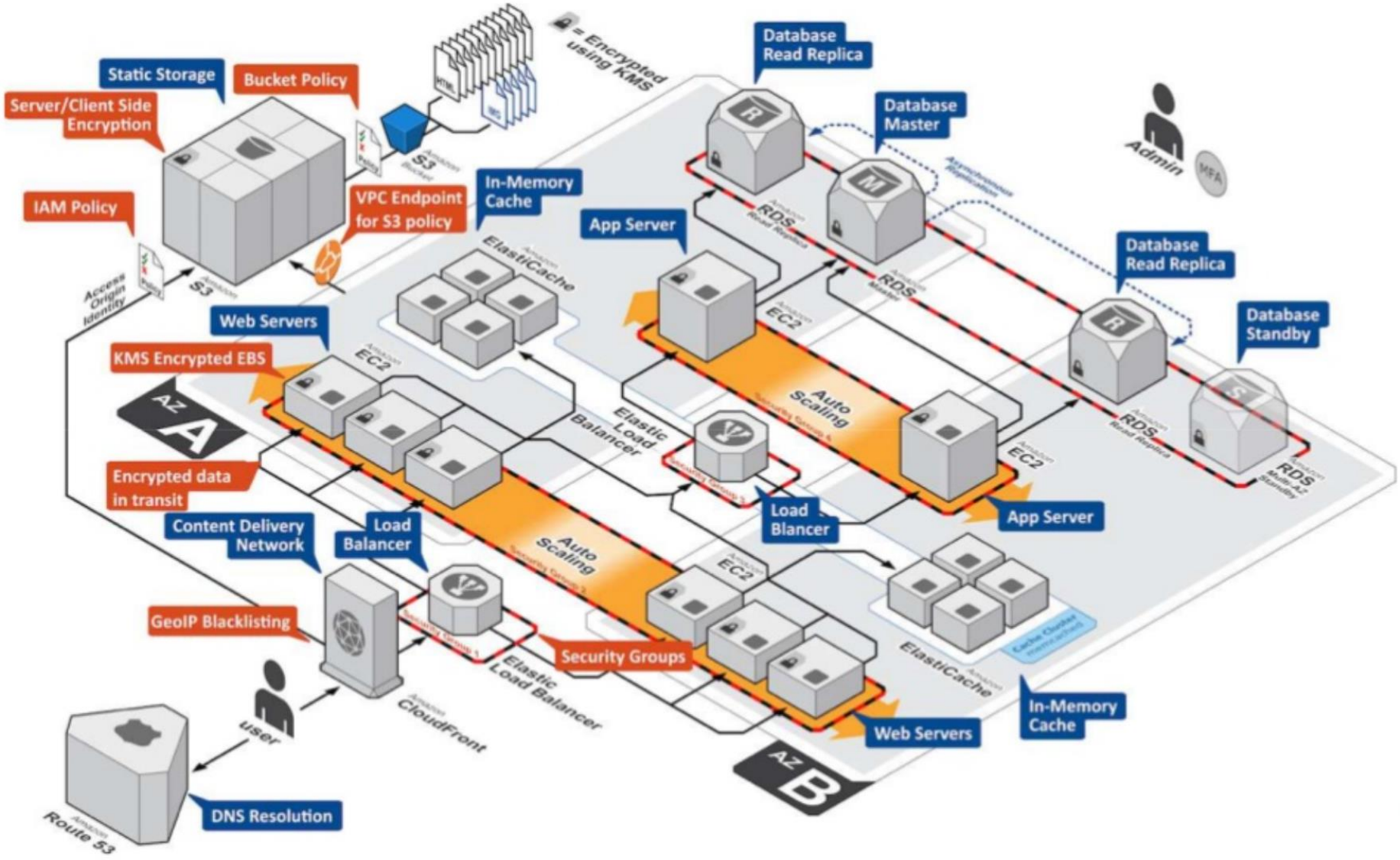


Lab 2 – Production Deployment

Changes between two implementations - *Lab v. Deployment*

- Single points of failures have been replaced by introducing redundancy in the application and database tiers.
- To provide high availability, resources have been added in two separate availability zones.
- A Multi-AZ deployment has been set up for the database using Amazon RDS with a standby instance to provide high availability and automatic failover.
- Auto Scaling has been configured for the application tier to scale up and scale down the number of EC2 instances based on demand.
- Application has been made stateless, and a separate NoSQL database (DynamoDB) has been used for storing state. The stateless application can be scaled horizontally as the requests can be serviced by any of the available application servers.

- Caching environment has been set up using ElastiCache to improve the response time of the application.
- A content delivery network (CDN) has been enabled with CloudFront to deliver dynamic and static content using a global network of edge locations.
- A Virtual Private Cloud topology has been set up using Amazon VPC to isolate parts of the infrastructure through the use of subnets, security groups, and routing controls.
- Load balancing has been set up using Amazon ELB to distribute inbound requests between the instances in the application tier.
- Amazon Route 53 is used to serve DNS requests for the domain and enable latency or geolocation routing.


= Encrypted using KMS
Static Storage
Bucket Policy
Server/Client Side Encryption
IAM Policy
VPC Endpoint for S3 policy
In-Memory Cache
App Server
Database Read Replica
Database Master
Admin
MFA
Database Read Replica
Database Standby
Access Origin Identity
Web Servers
KMS Encrypted EBS
AZ A
Encrypted data in transit
Content Delivery Network
Load Balancer
GeoIP Blacklisting
Security Groups
Elastic Load Balancer
Load Blancer
App Server
In-Memory Cache
Web Servers
AZ B
user
DNS Resolution
Route 53
CloudFront
S3
ElastiCache
EC2
Auto Scaling
RDS

How to achieve WAF?

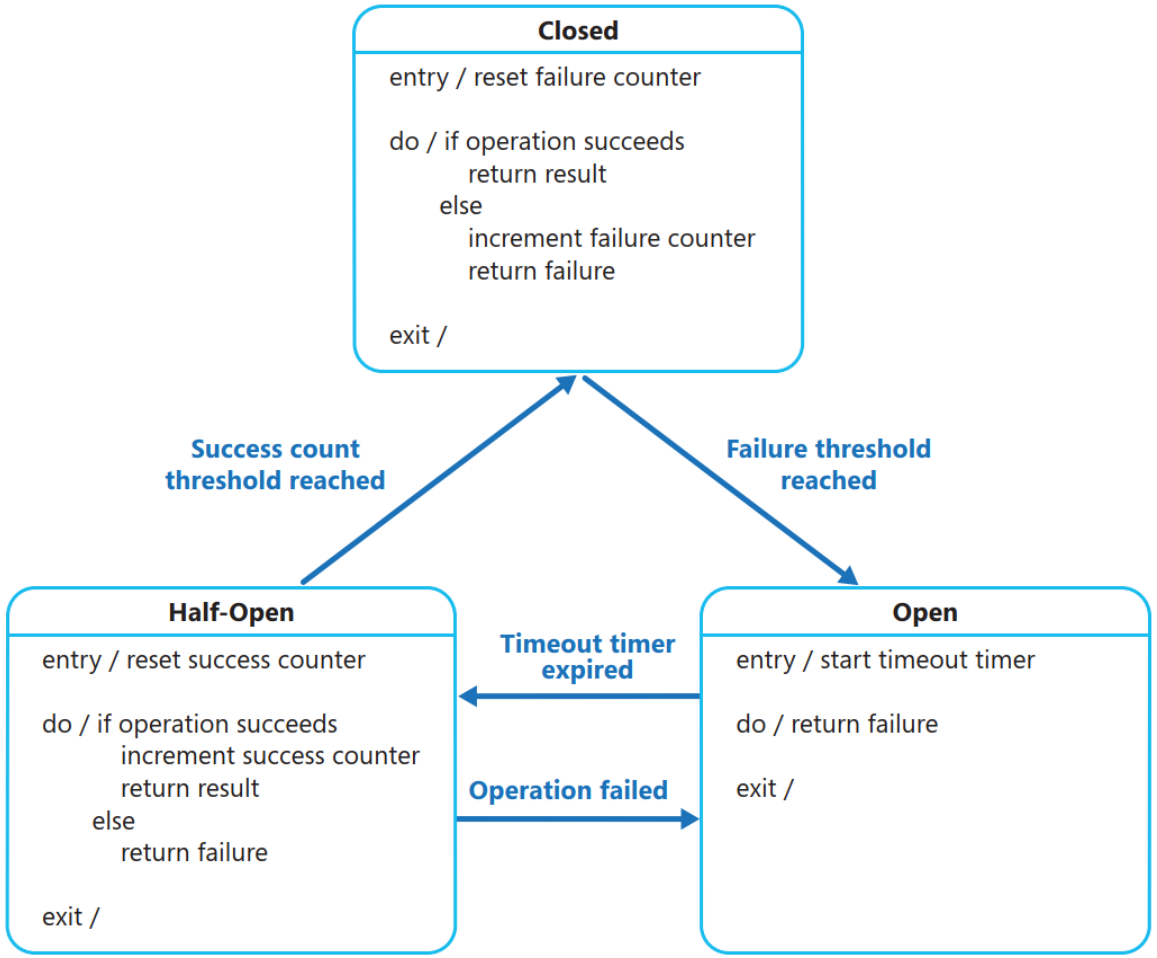
Use design “templates” or “patterns” that industry has accepted as “best practice”.

Design Models and Patterns for Cloud

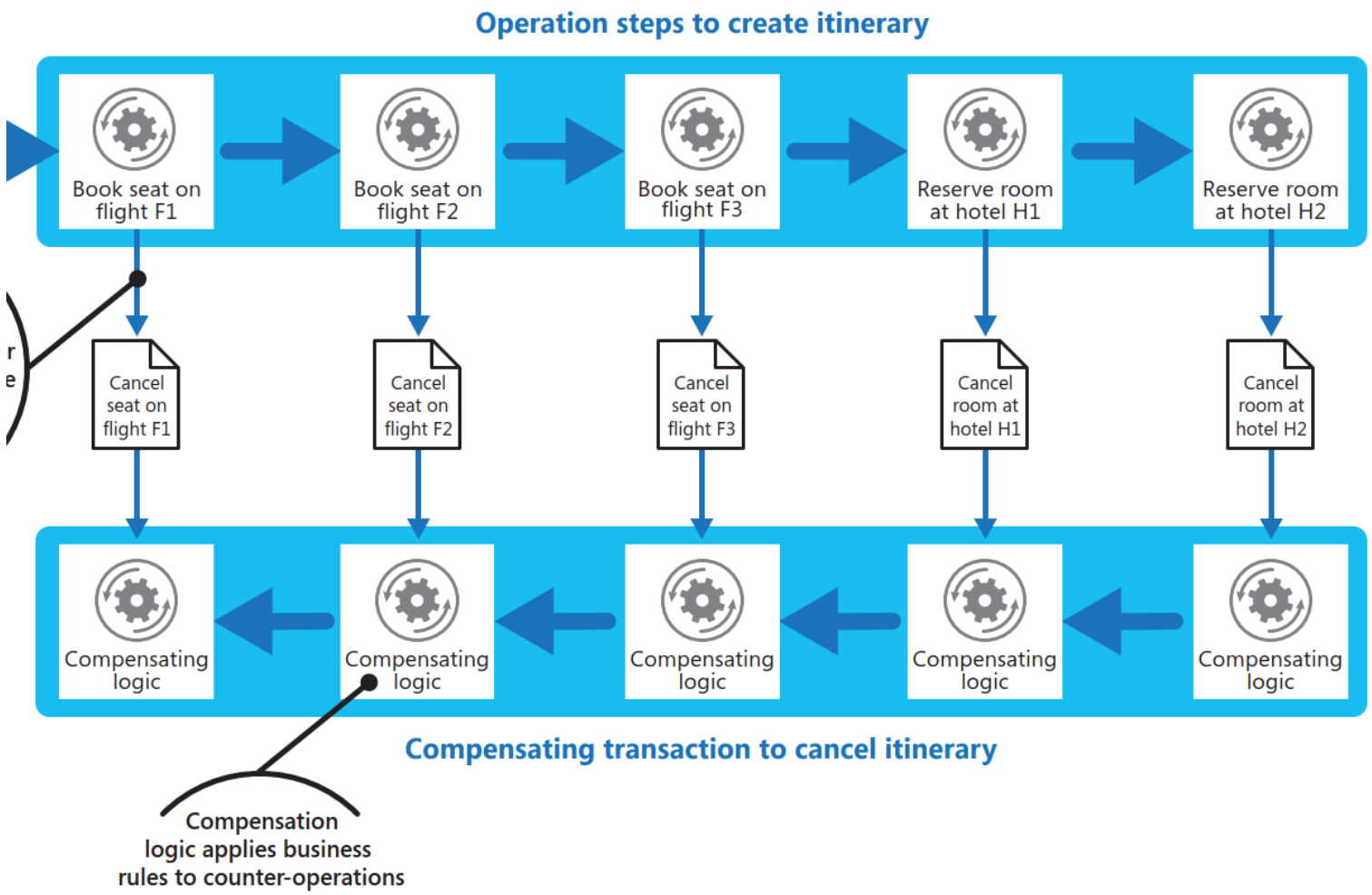
- Circuit Breaker
- Compensating Transaction
- Leader Election
- Retry
- Scheduler Agent Supervisor
- Cache Aside
- Competing Consumers
- CQRS – Command Query Responsibility Segregation
- Event Sourcing

Design Patterns & Models (Contd)

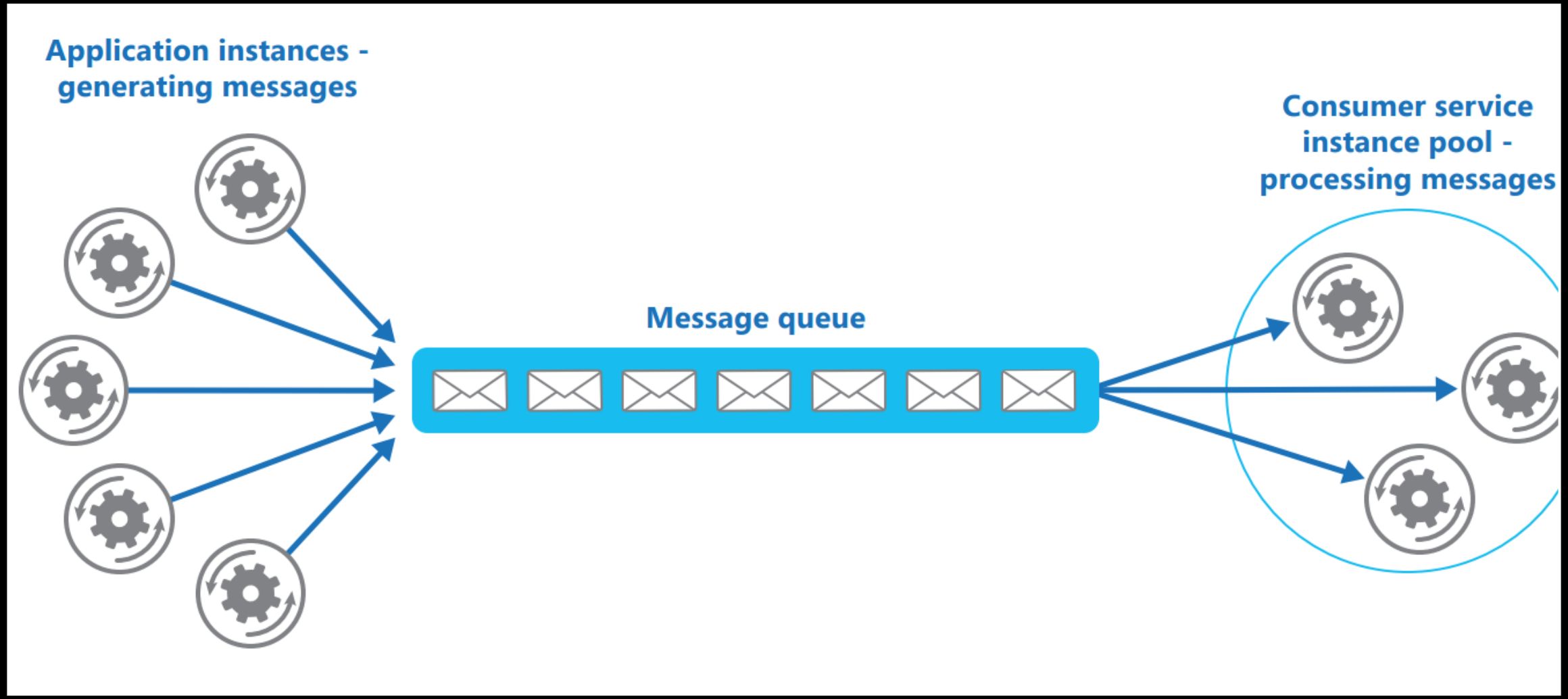
- External Configuration Store
- Federated Identity
- Gatekeeper
- Health Endpoint Monitoring
- Index Table
- Materialized View
- Pipes and Filters
- Priority Queues
- Queue-Based Load Leveling
- Sharding
- Static Content Hosting



Circuit Breaker Pattern

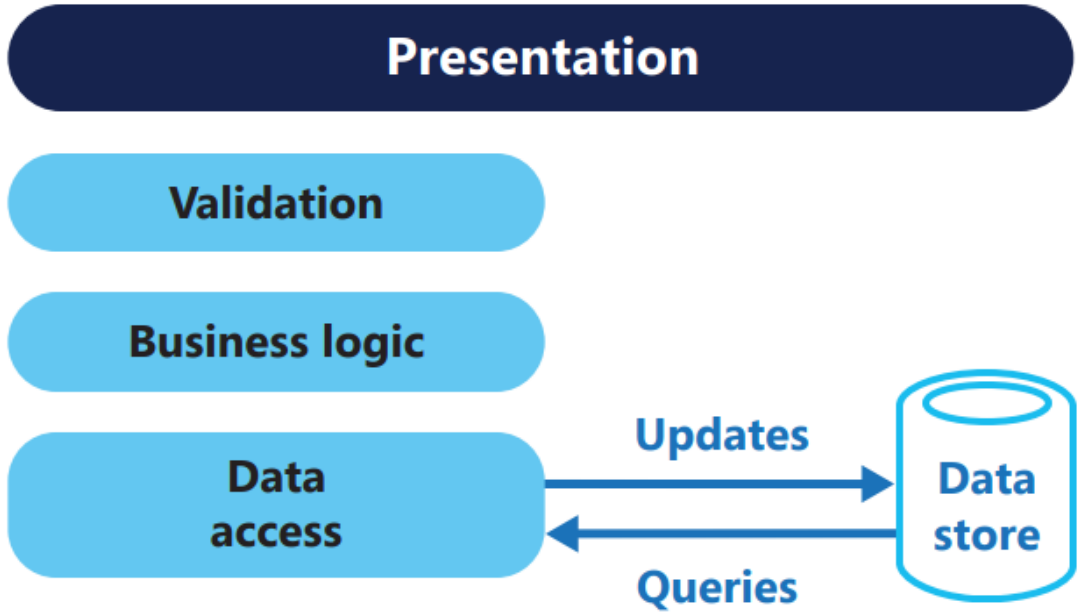


Compensating Transaction Pattern

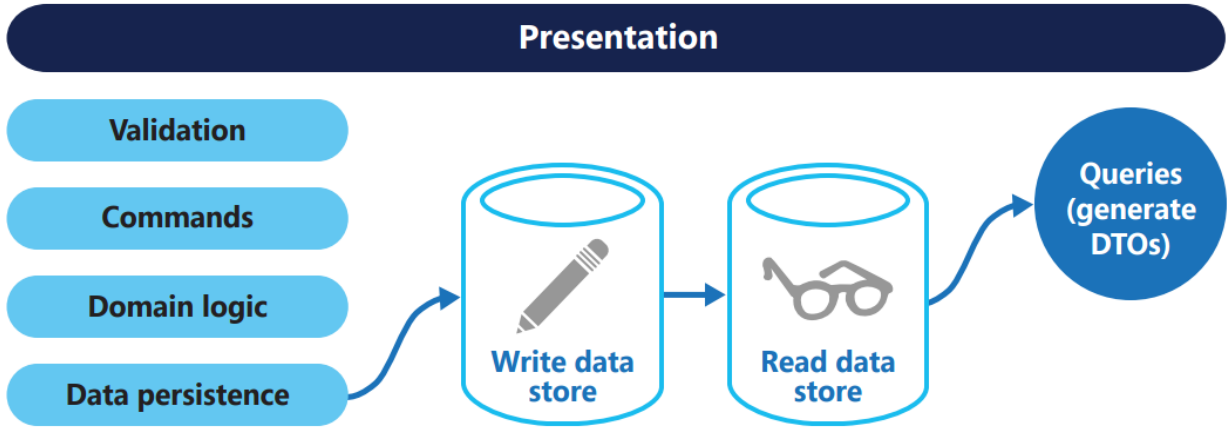


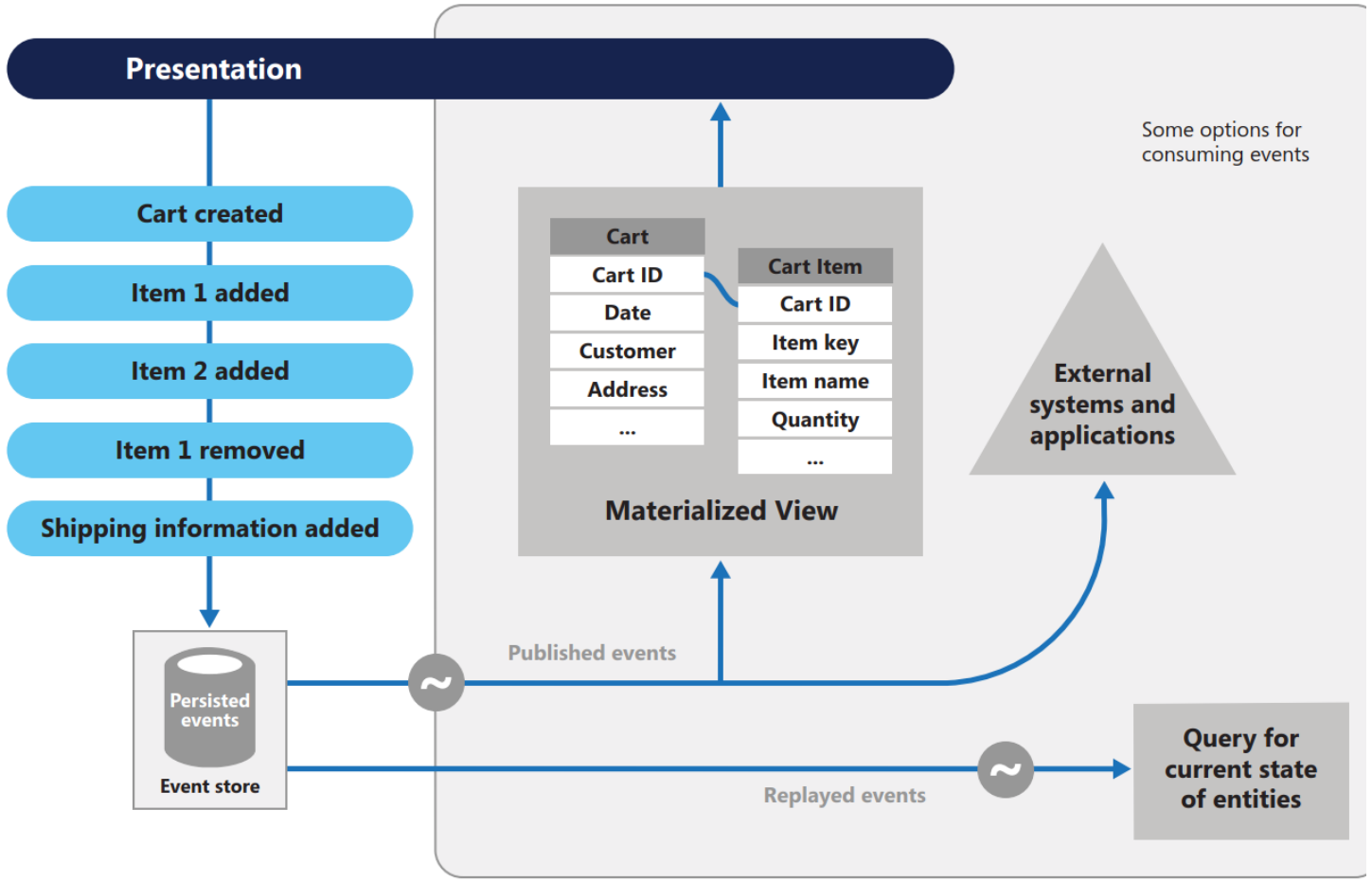
Competing Consumers Pattern

- **CQRS – Command and Query Responsibility Segregation**



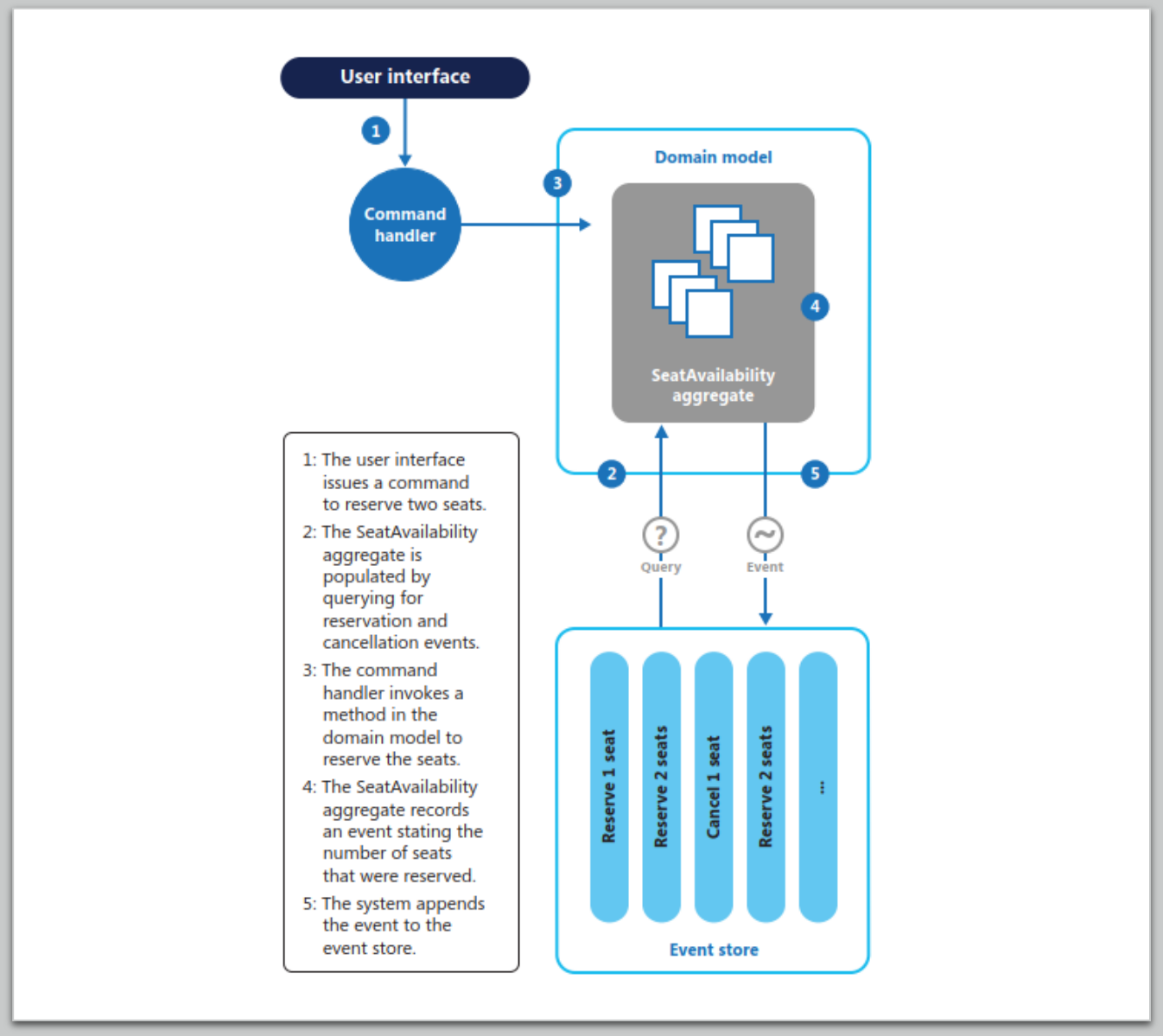
- **CQRS – Command and Query Responsibility Segregation**



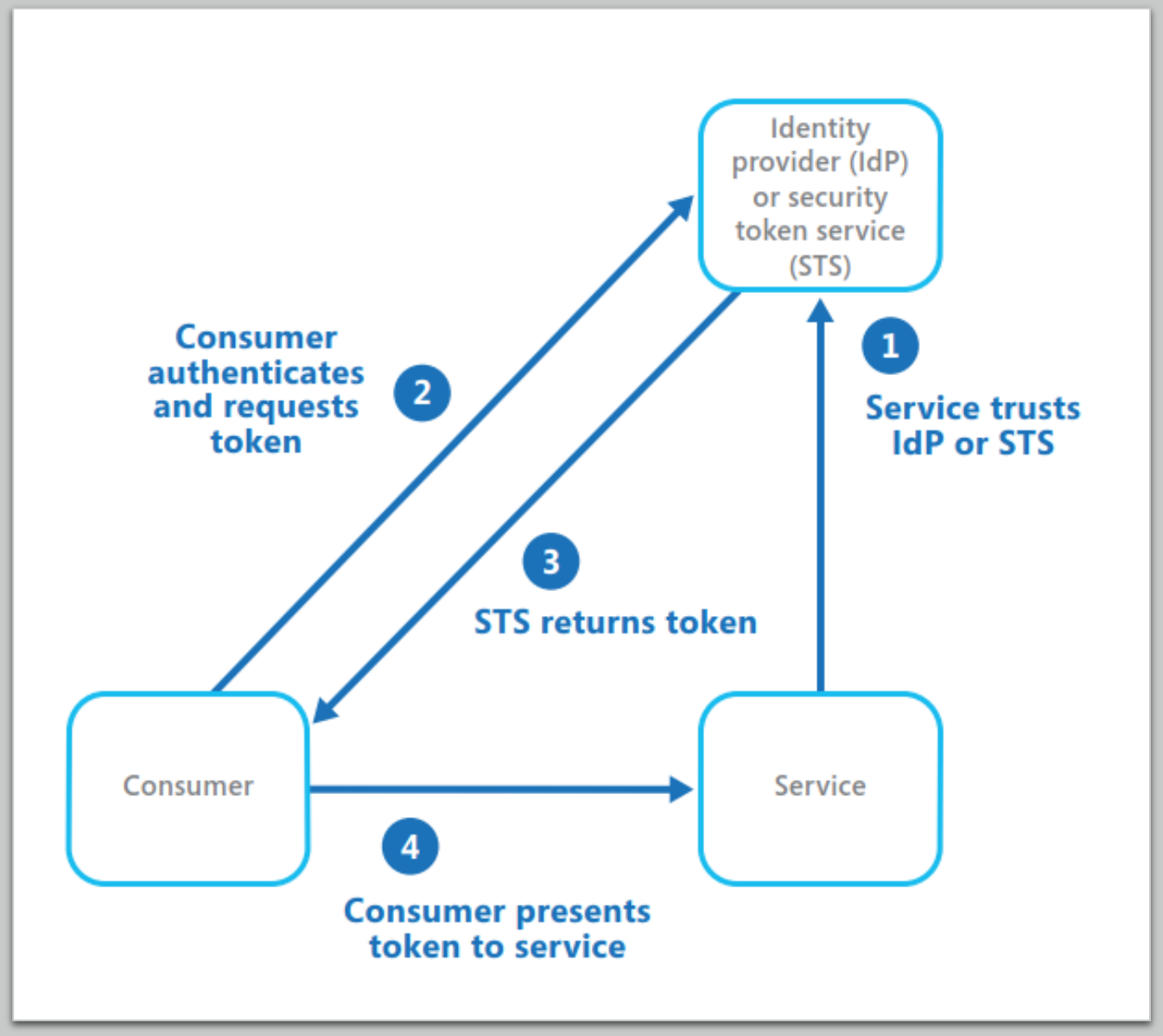


Event Sourcing Pattern

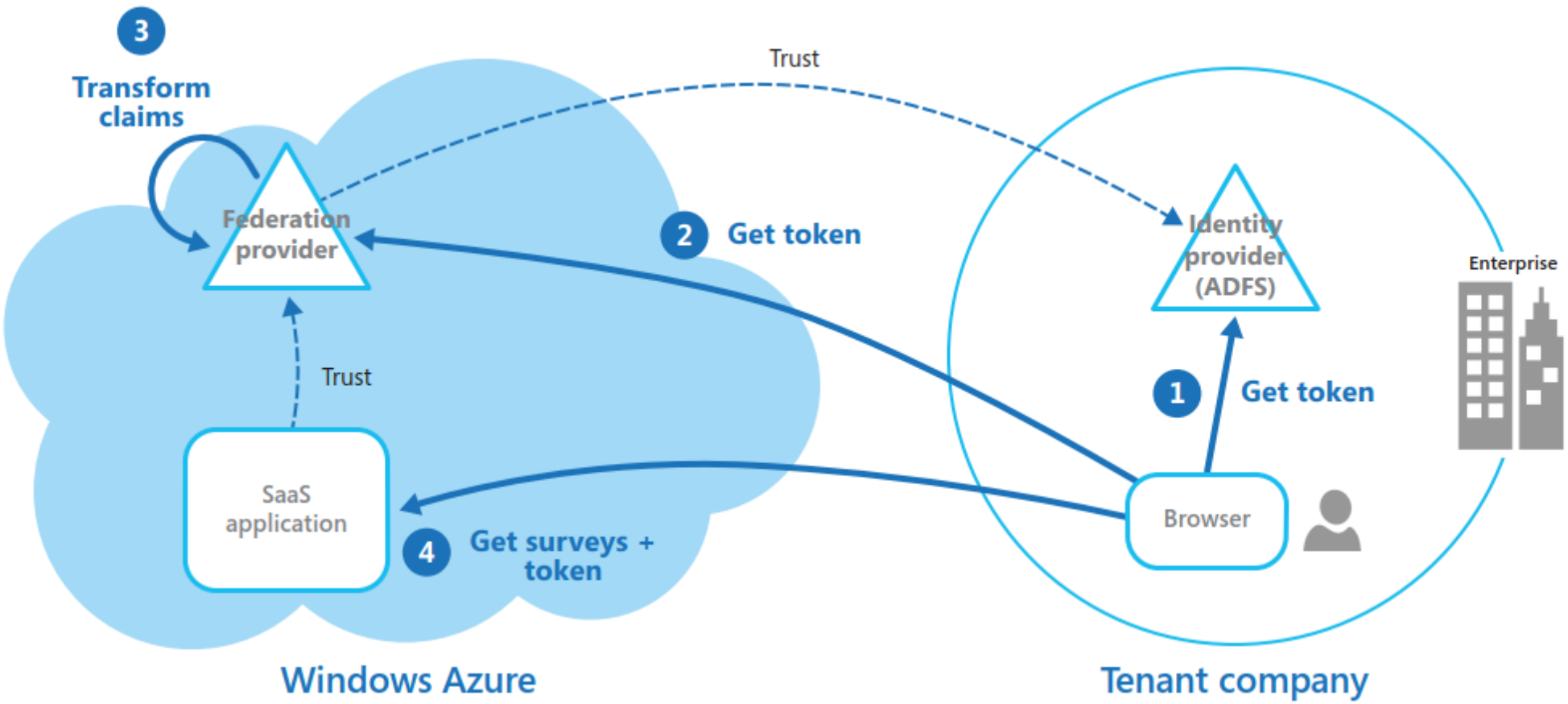
Event Sourcing Pattern (Contd…)



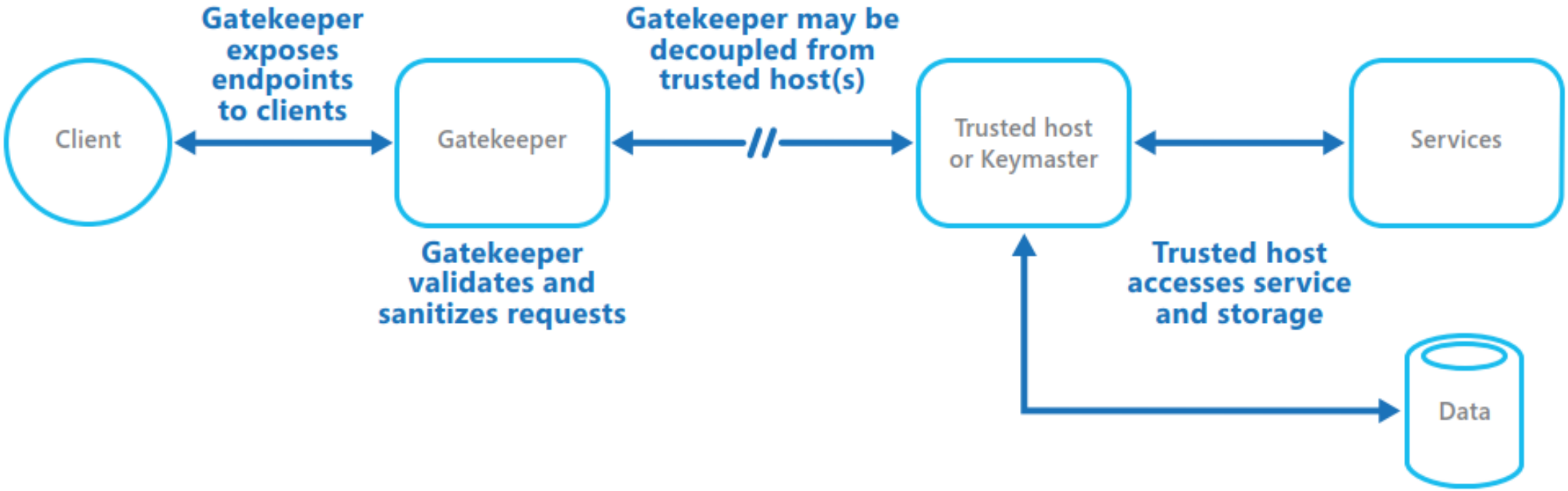
Federated Identity Pattern



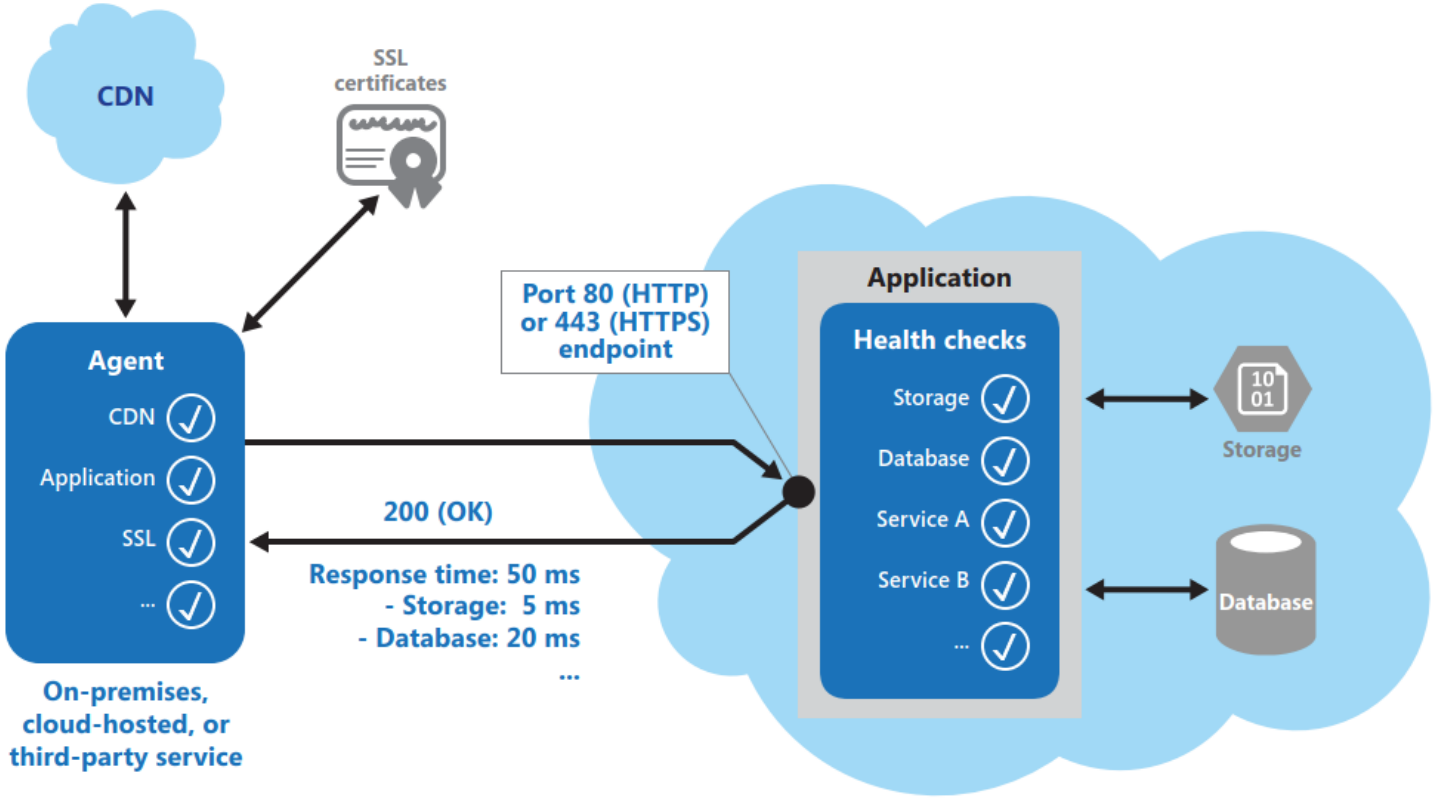
Federated Identity Pattern



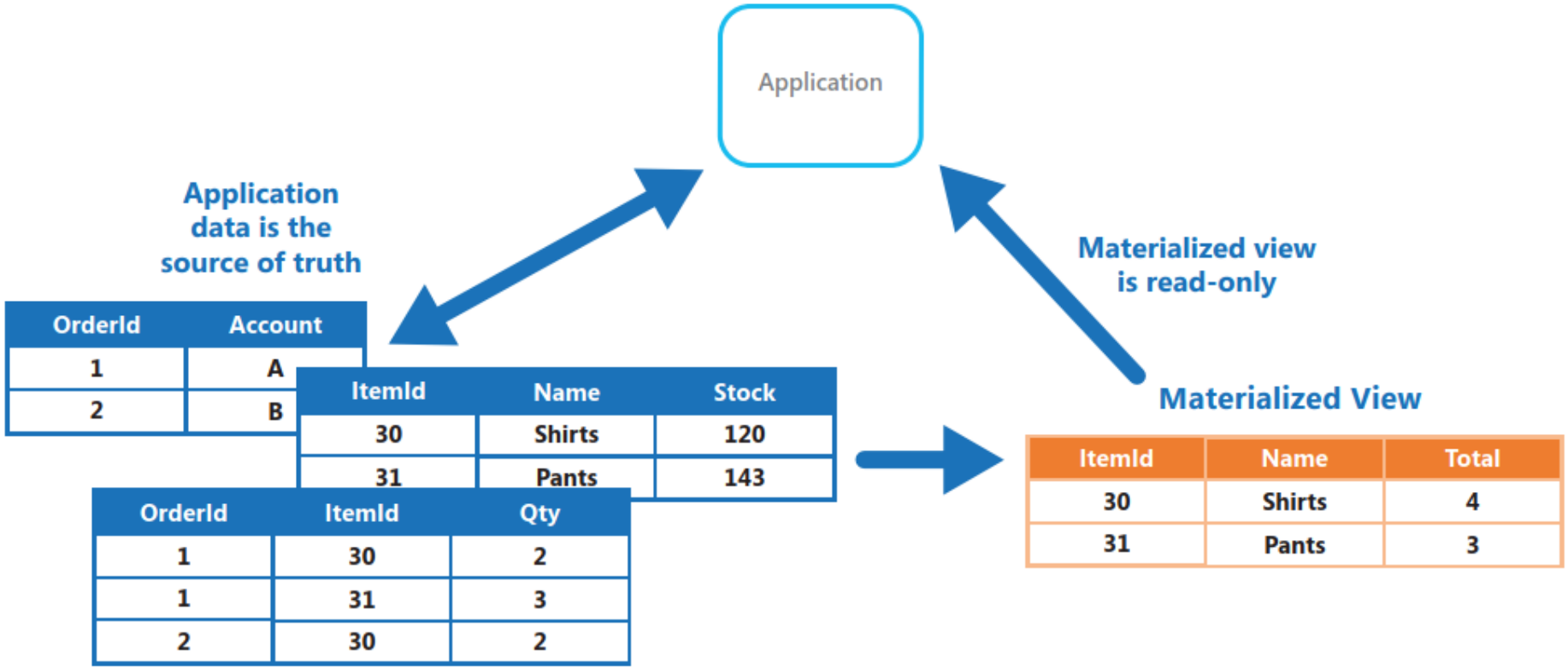
Gatekeeper Pattern



Health Endpoint Pattern



Materialized View Pattern



Materialized View Pattern – Contd.

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

Summary

- We covered a few patterns for “best practice” cloud deployment.
- We will cover a few more in the next class