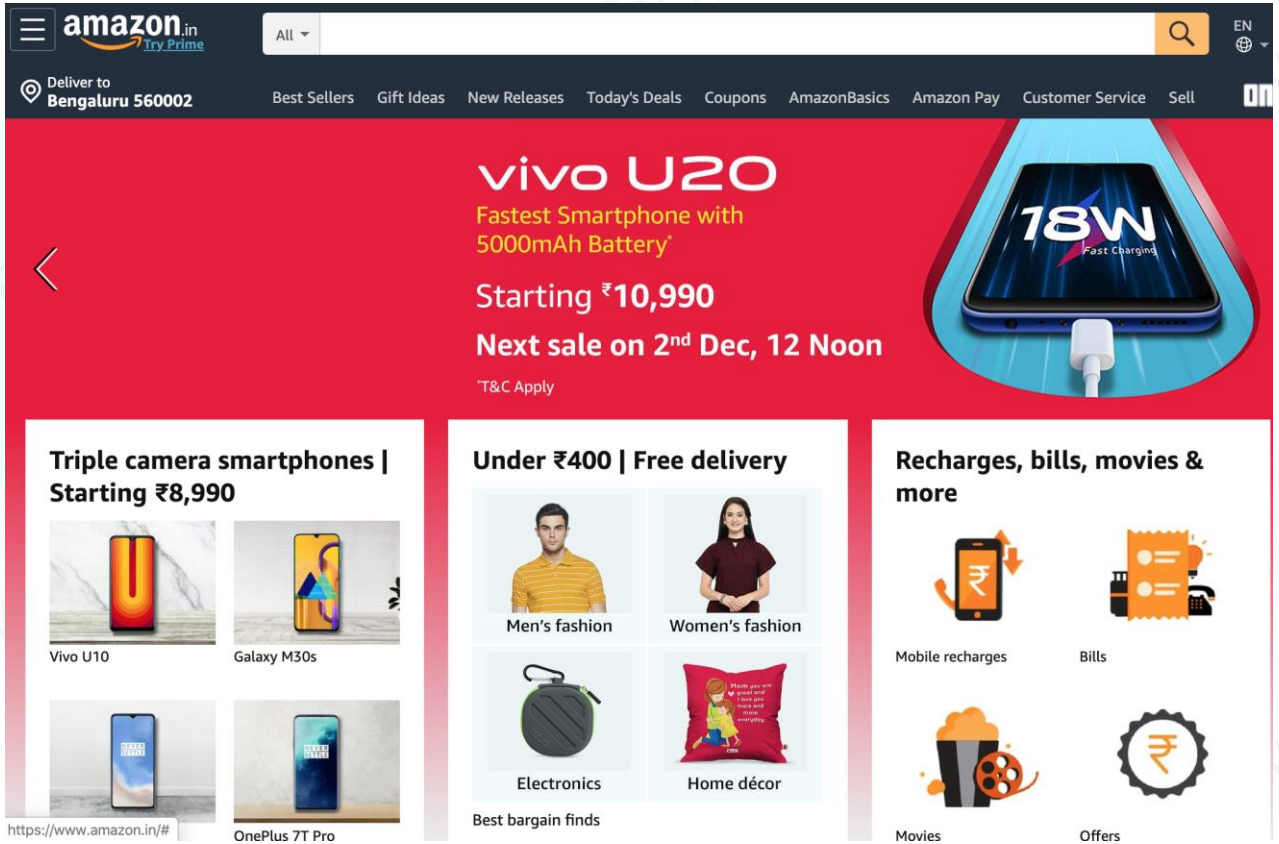
Developer To Architect

Moving from Application Design/Architecture to System Architecture

NewTechWays

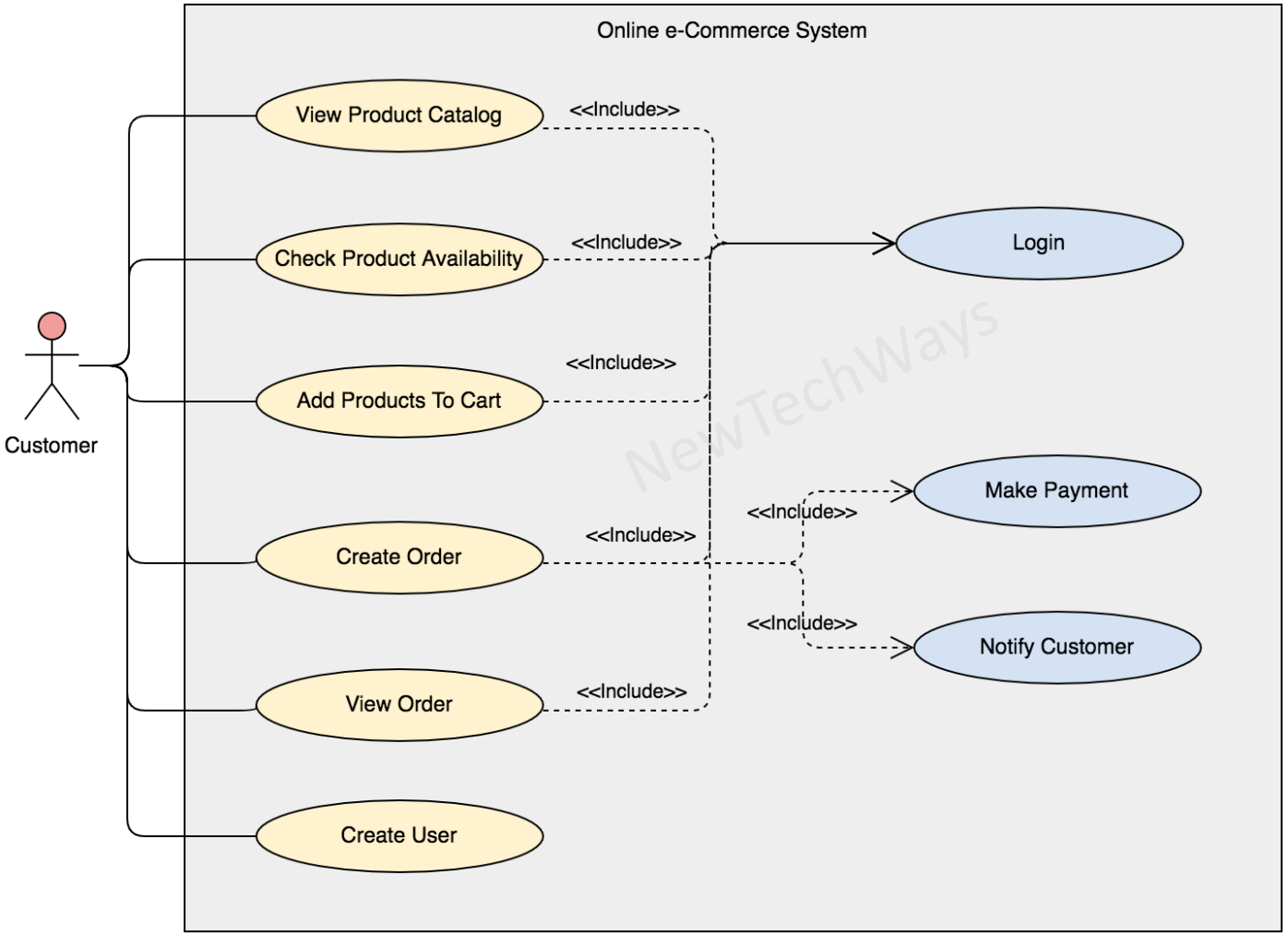
Our Objective

- Assume ‘Architecting’ a big e-commerce system as our objective

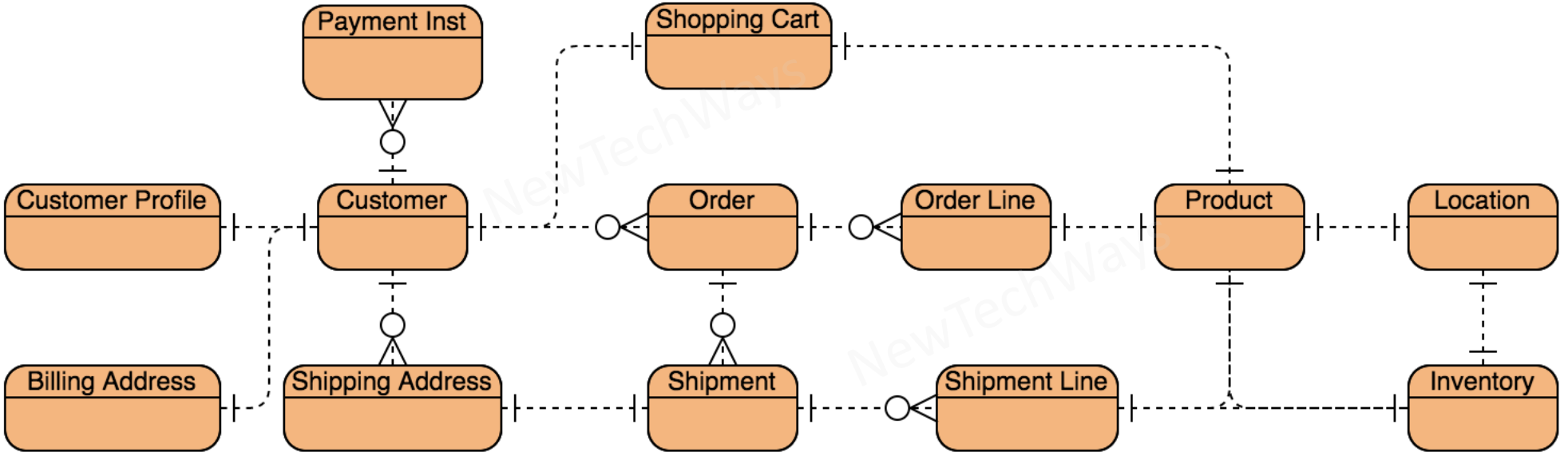


Application Architecture

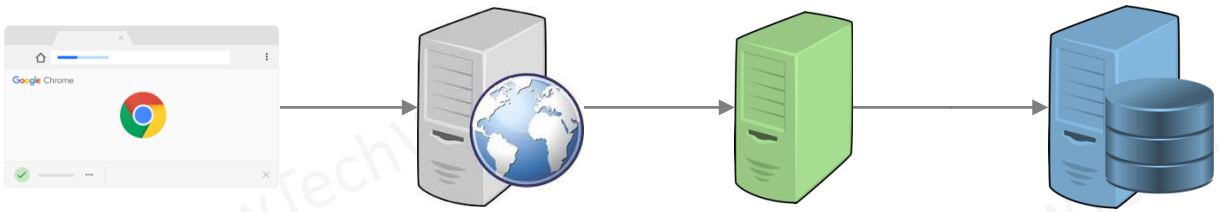
Use Case Model

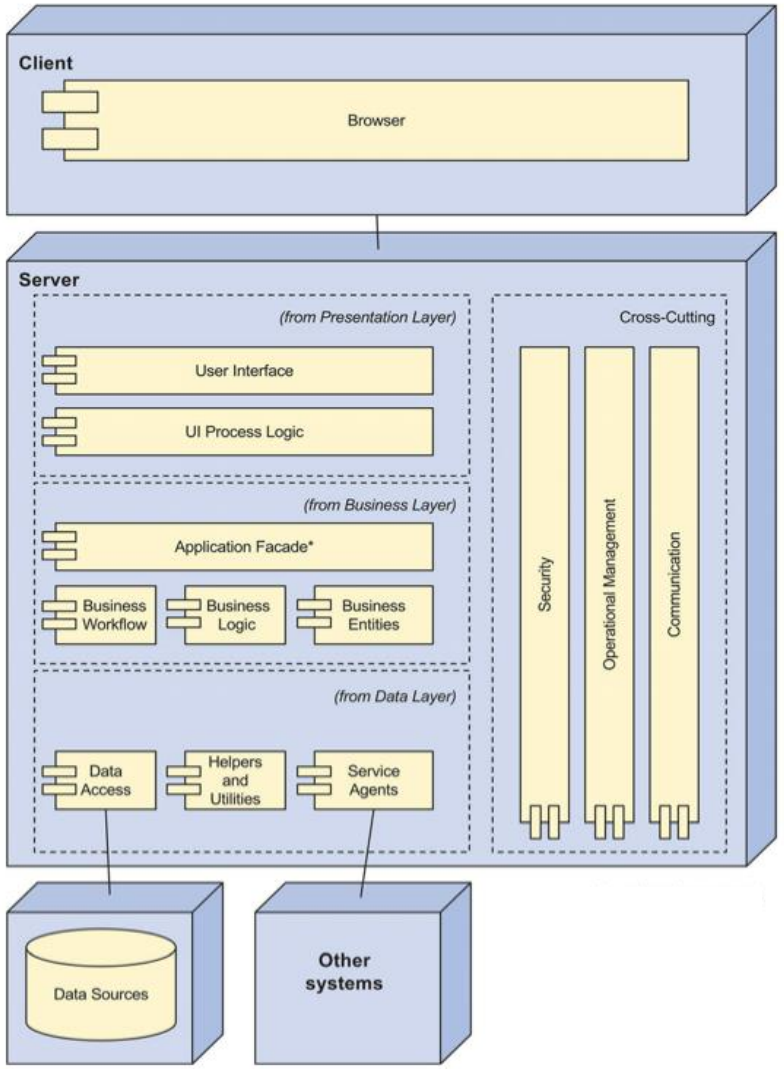


Domain Model



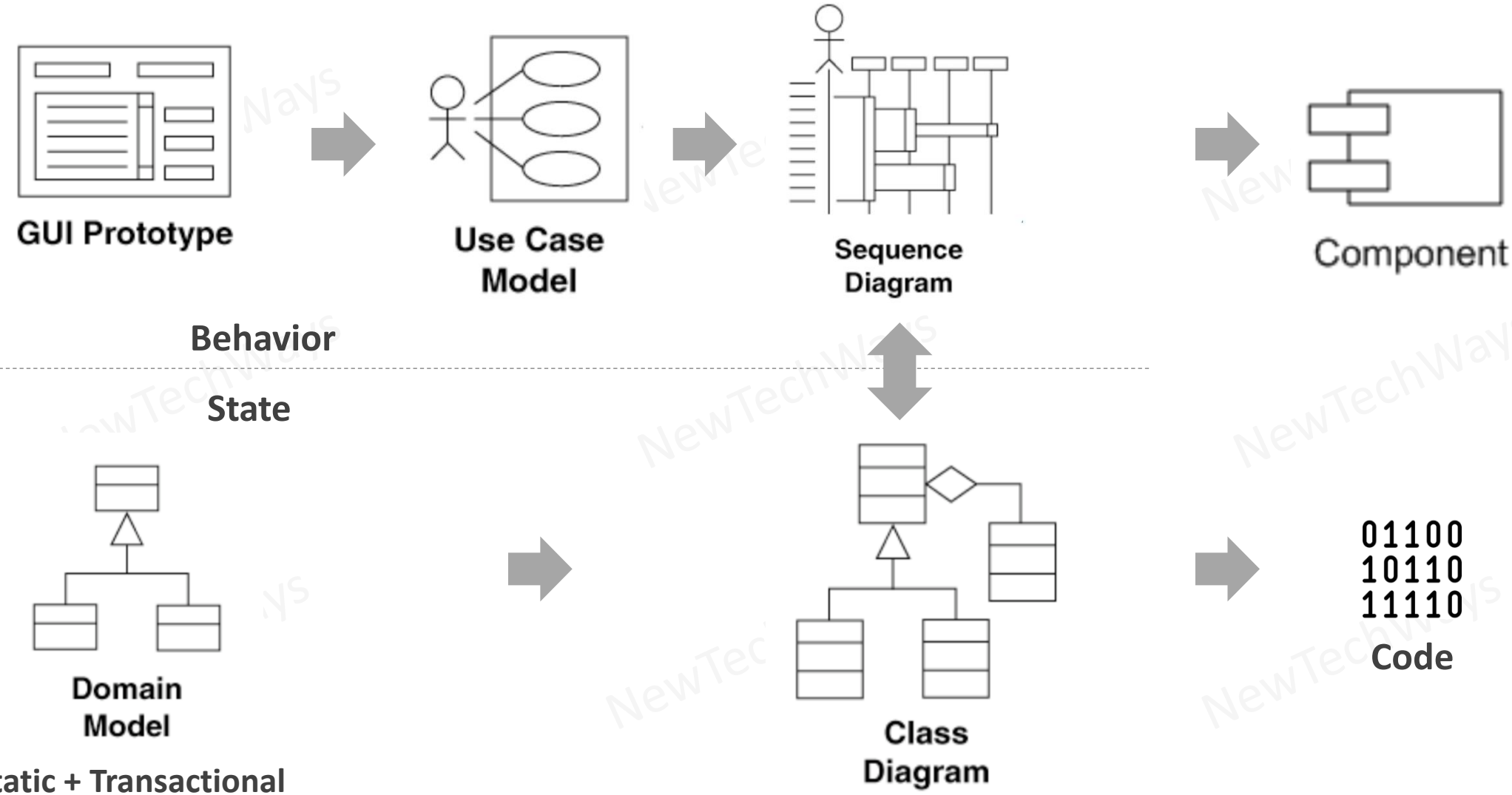
Component Model





NewTechWays

Low Level Design



What About?

High Availability

Response Latency

Global Customers

System & Data Security

File storage for Catalog Images

Mobile Support

Unstructured Data Storage & Analytics

Cloud Deployment

NewTechWays

Going Beyond

Application Architecture To System Architecture

Application Architecture Vs System Architecture

System level challenges
surface up in large scale systems

Scale, Reliability, Security, Deployment are
biggest concerns for a large-scale system

Latency Requirements

- UI page load in < 250ms
  - across the globe
- Customer transactions < 3 sec

- Latency is not an option
  - 10 years ago, Amazon found that every 100ms of latency cost them 1% in sales
  - Now Akamai study shows that every 100-millisecond delay in website load time can hurt sales by 6%
  - Google found an extra .5 seconds in search page generation time dropped traffic by 20%
  - A stockbroker can lose millions if their trading platform is 5ms behind its competition

As page load time goes from:

1s to 3s the probability of bounce **increases 32%**

1s to 5s the probability of bounce **increases 90%**

1s to 6s the probability of bounce **increases 106%**

1s to 10s the probability of bounce **increases 123%**

Scalability Requirements

- 10 million requests/day
- 1K to 100K simultaneous users
- Global customer base
- 100 million products
- Transaction data for last 5 years
- Petabytes of log data

- For 10 M requests with average response size of 10KB
  - Data Outflow = 10M x 10KB = 100 GB
- For 100 M orders/year with average order size of 10 KB, five years data
  - Data storage = 5 x 100M x 10KB = 5 TB
- For 100 M products with average product description/image size of 1 MB
  - Data Storage = 100M x 1MB = 100 TB
- Logs generated and archived
  - Data Storage => in Petabytes
- Buying customers in 100 countries

Availability & Reliability Requirements

- Unavailability results in
  - Business loss
  - Reputation loss
- 99.95% Availability
  - Maximum cumulative disruption of 4 hours 22 minutes in a year
- 99.999999999% Durability for storage systems
  - Data once stored is practically never lost
- Disaster Recovery
  - Operations to continue even if a region goes down due to a natural calamity

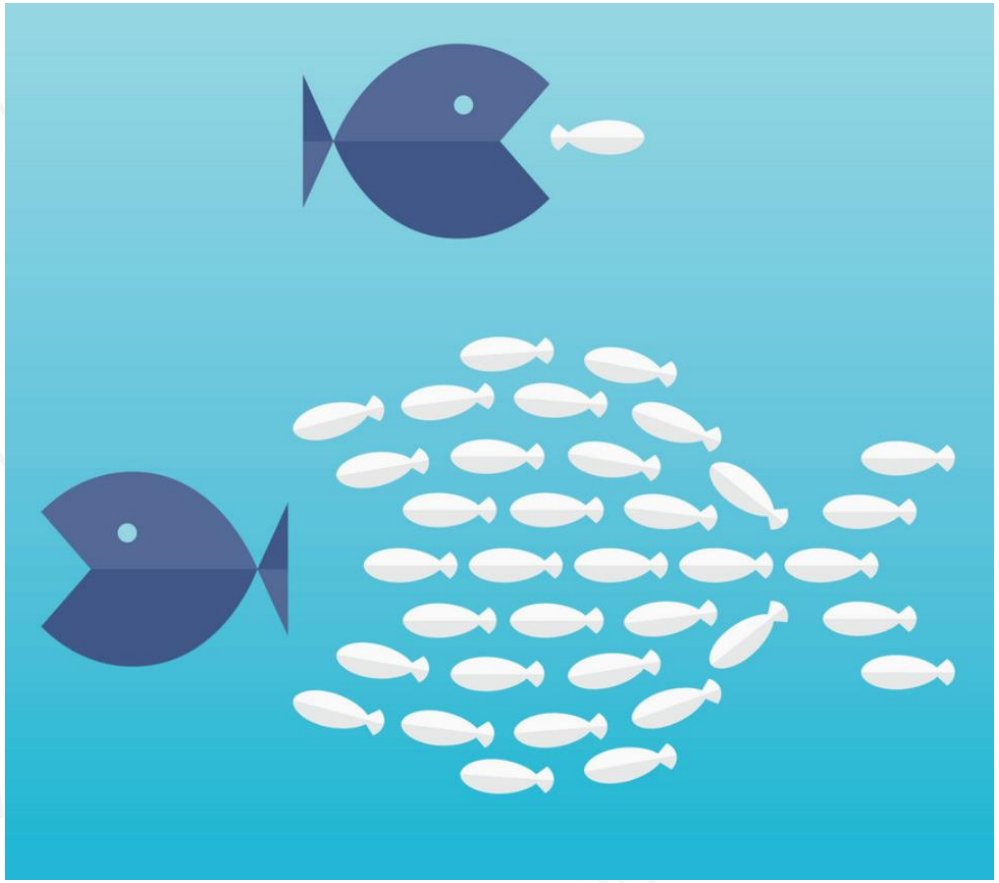
Security Requirements

- Infrastructure protection
  - Network access
  - System access
  - Service access
- Data Protection
  - Data sensitivity classification
  - Protect data at rest
  - Protect data on wire
  - Data backup & replication
- Identity & Access Management
  - Authentication, Authorization
  - Role based access

Designing System Architecture

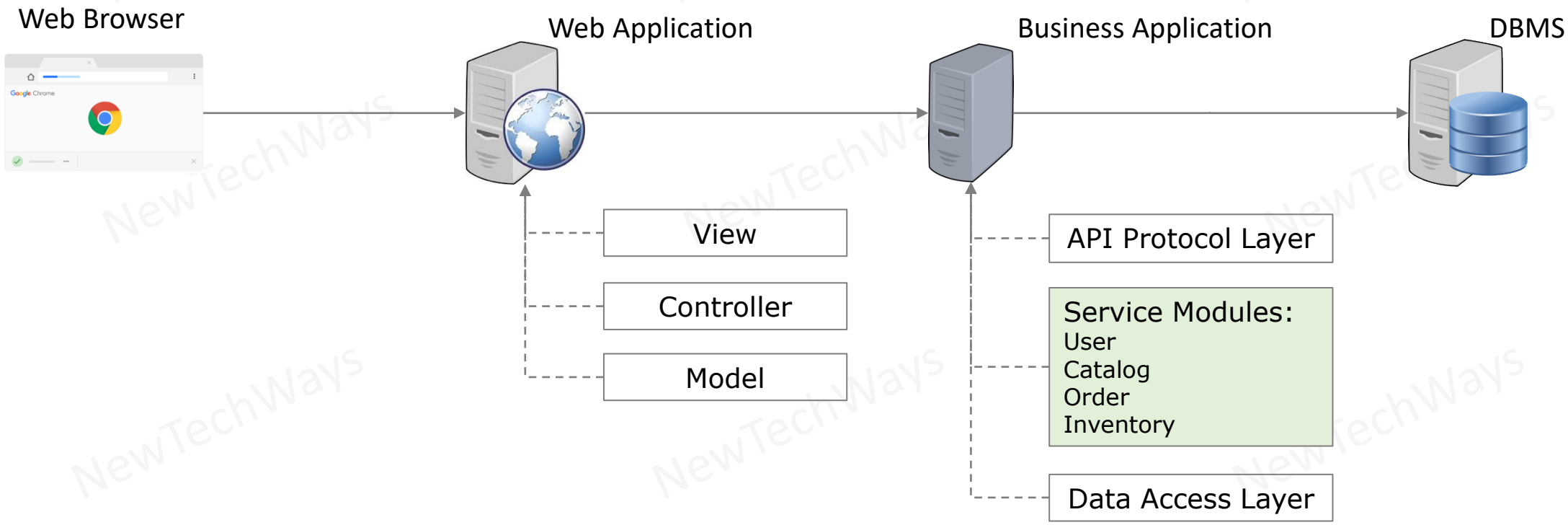
Scalability Principle

- Monolith is an anti-pattern for Scalability
- Scalability goes up with
  - Decentralization
    - More specialized workers – Services
    - More workers – Instances, Processors, Threads
  - Independence
    - Multiple workers are as good as a single worker if they can't work independently
    - They must work concurrently to maximum extent
    - Independence is impeded by
      - Shared resources
      - Shared mutable data



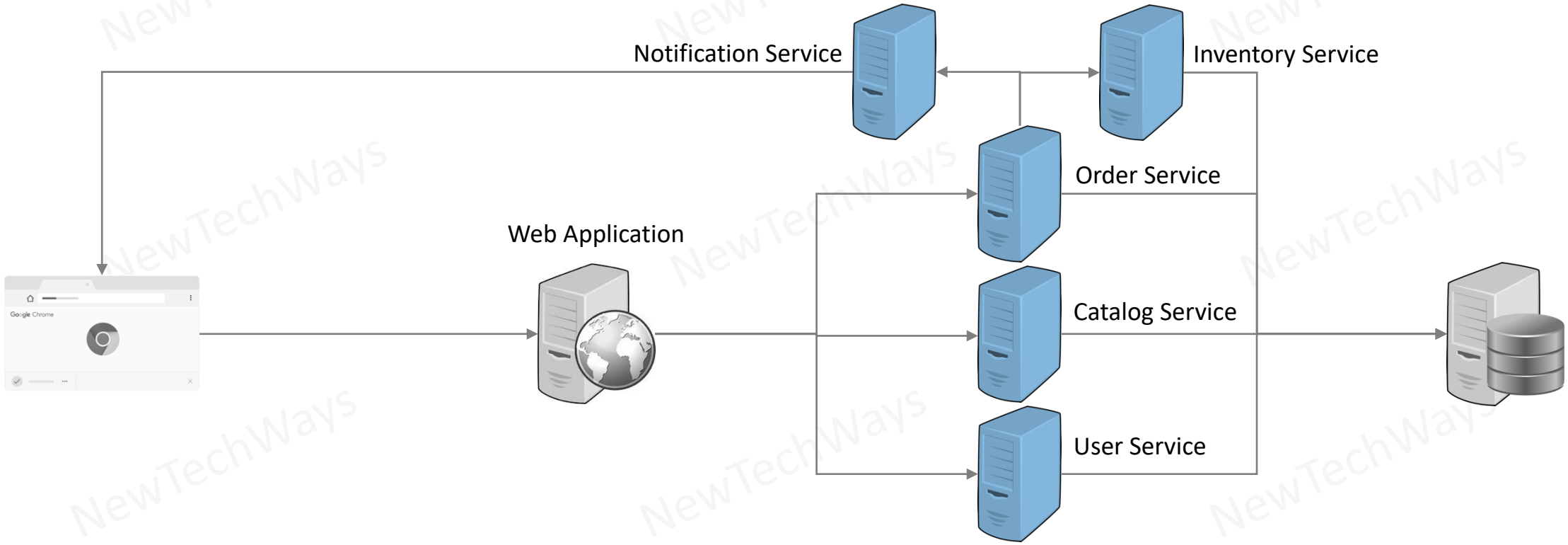
Modularity

- Scalable architecture starts with modularity
  - Provides the foundation for breaking a system function/service into more specialized functions/services



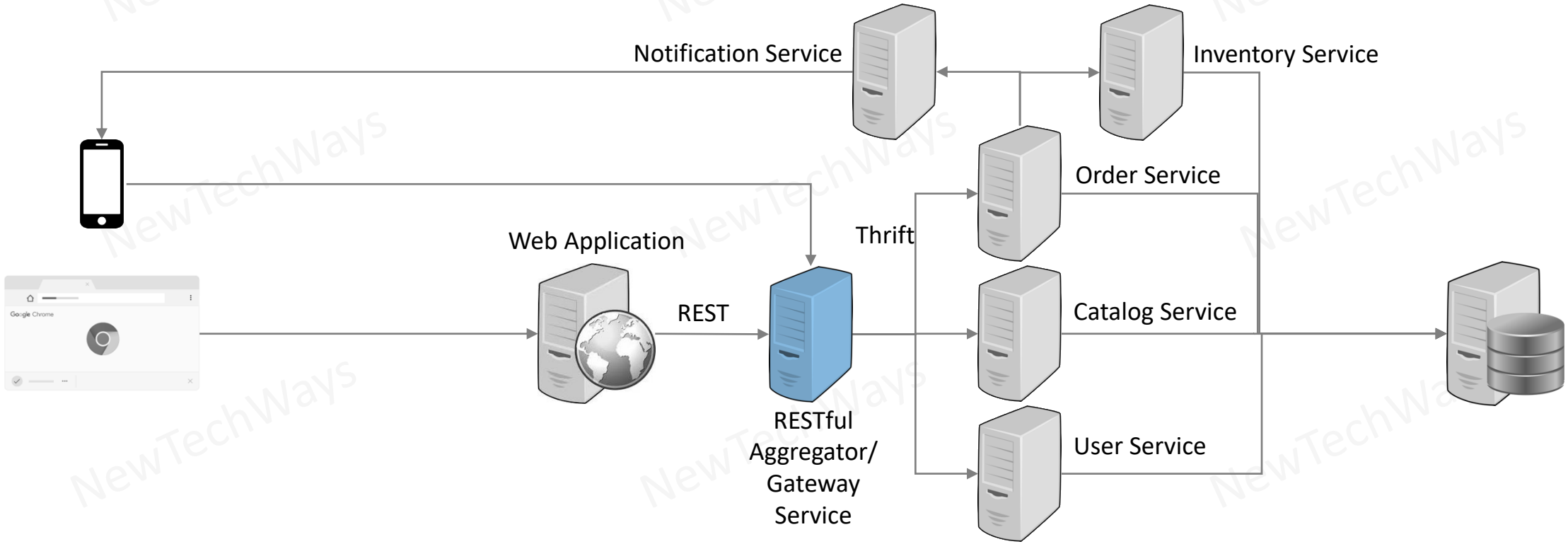
Specialized Services – WebServices

- Services can be scaled differently e.g., Number of instances



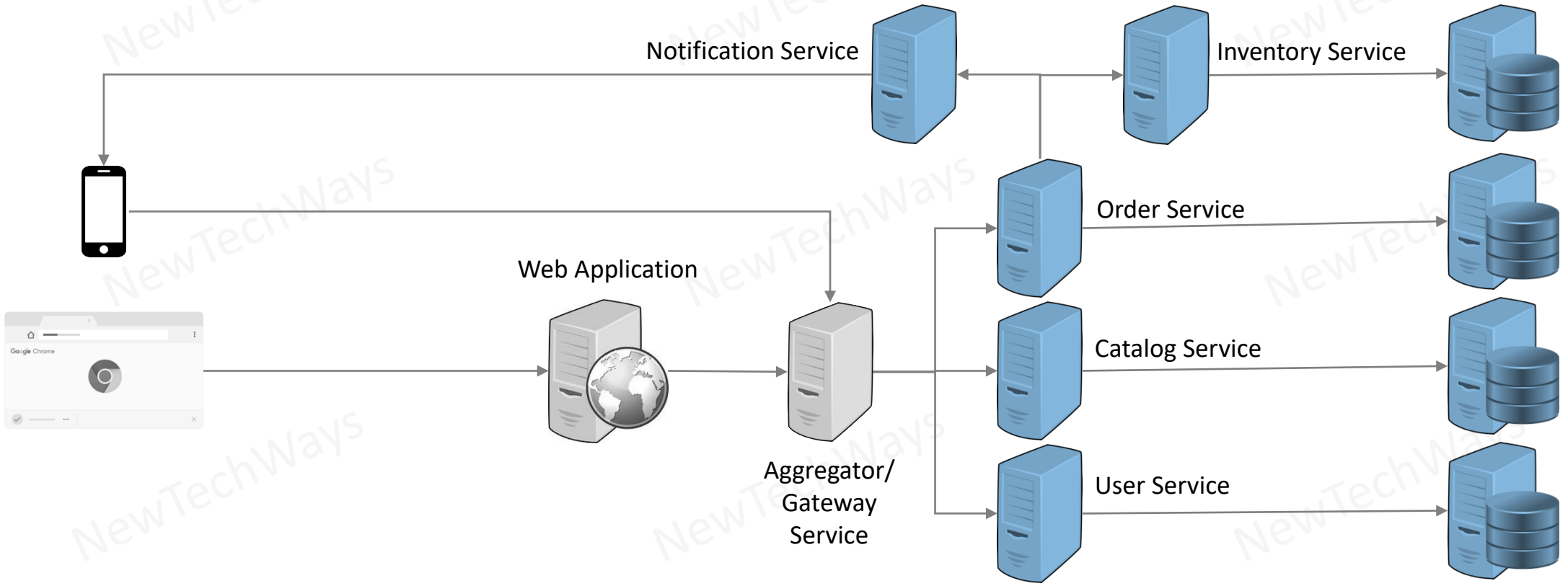
Aggregator Service & RESTful API – Mobile Support

- REST for external interface interoperability & Mobile Support
- RPC for internal communication performance



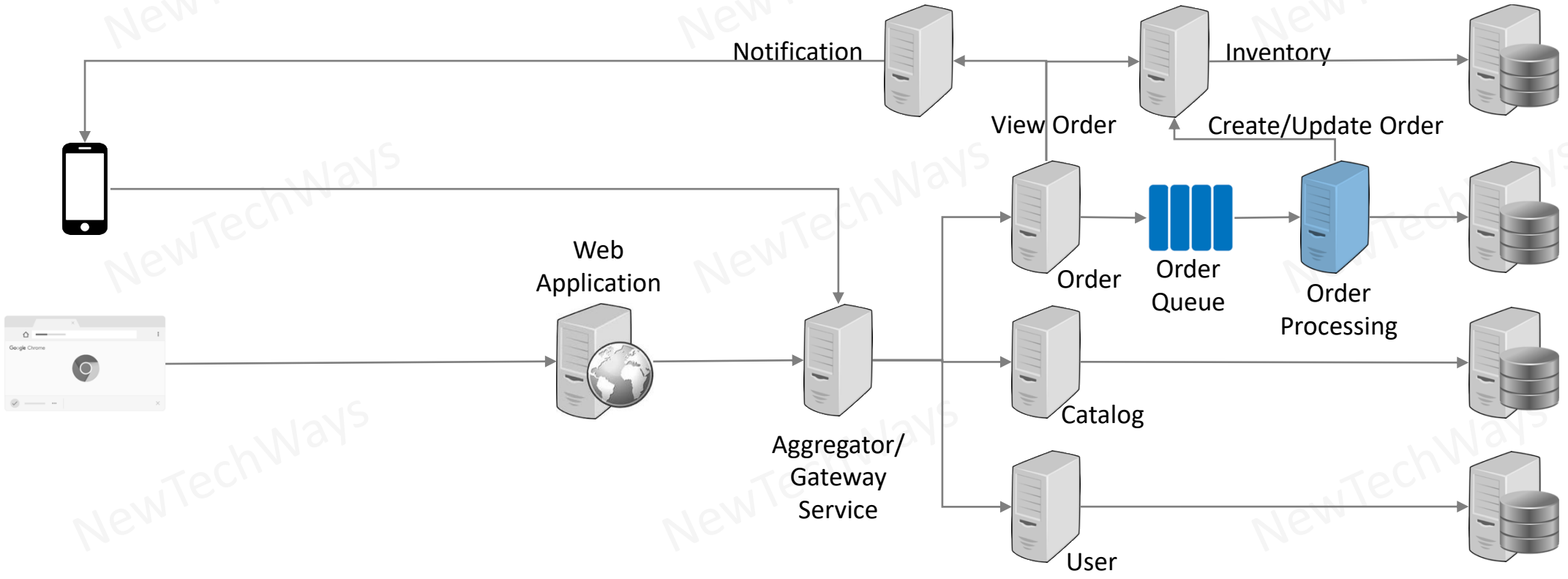
Independent Services – MicroServices

- Micro-Services can be scaled differently and deployed independently



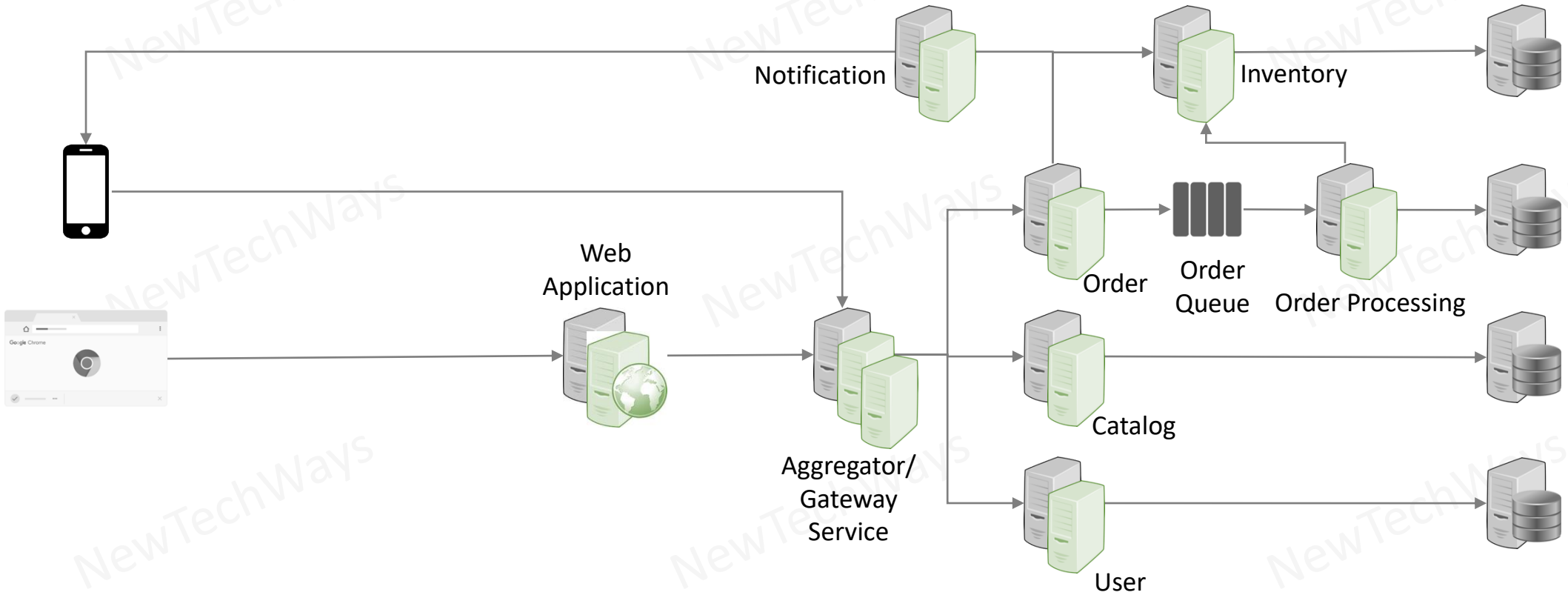
Asynchronous Services

- Updates can be done asynchronously to buffer peak loads and to reduce response latency



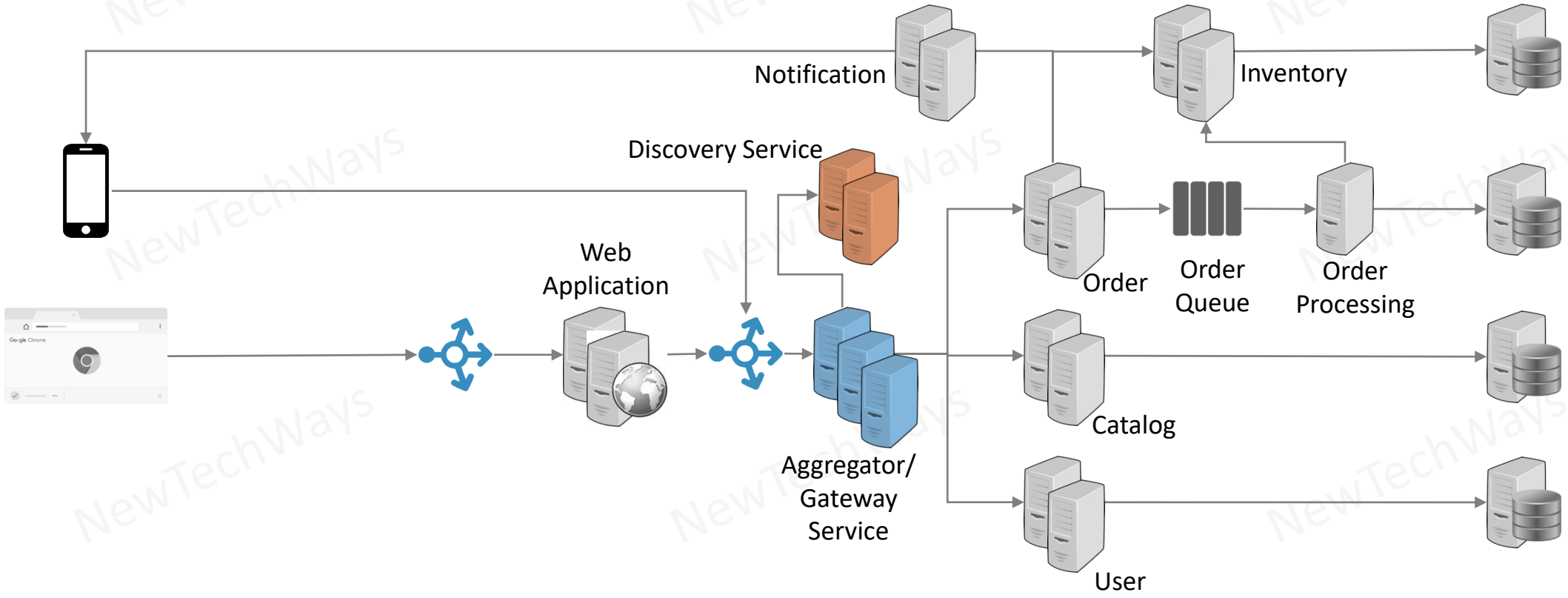
Stateless Replication

- Replication provides more computation power



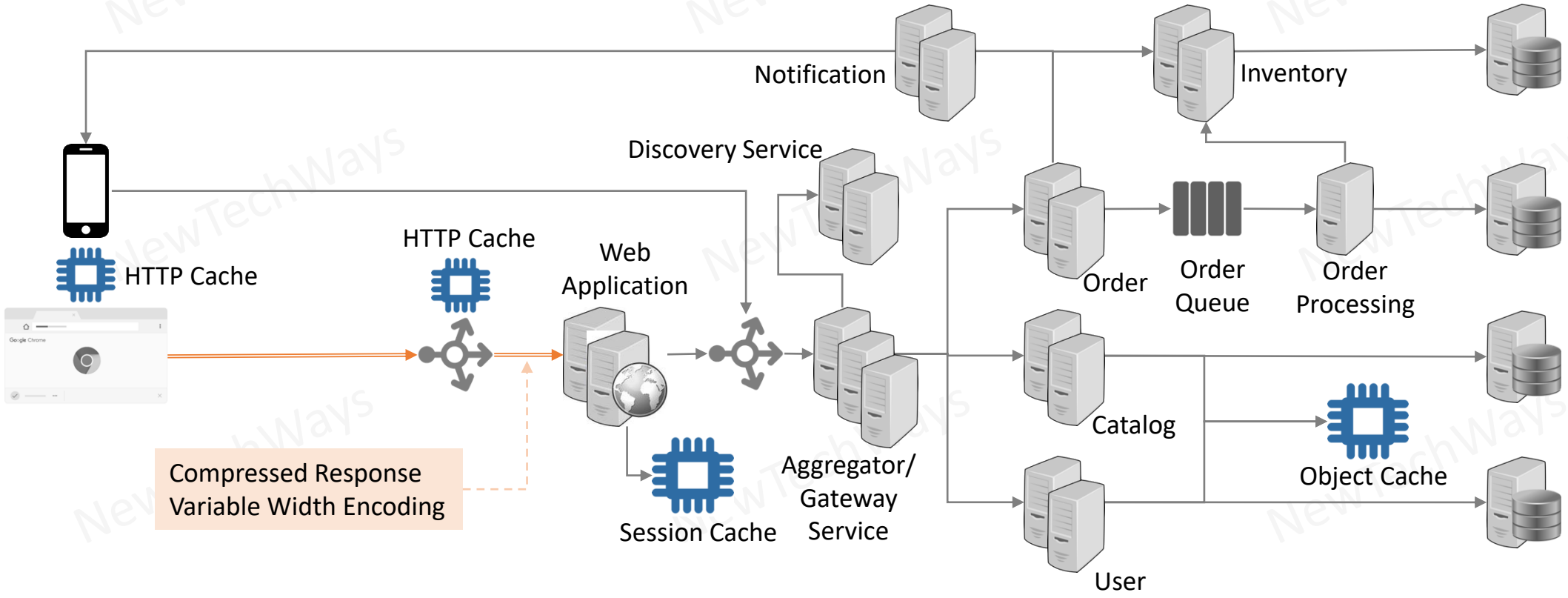
Load Balancing & Discovery

- Load Balancers for routing and load balancing
- Discovery services for naming and tracking healthy services



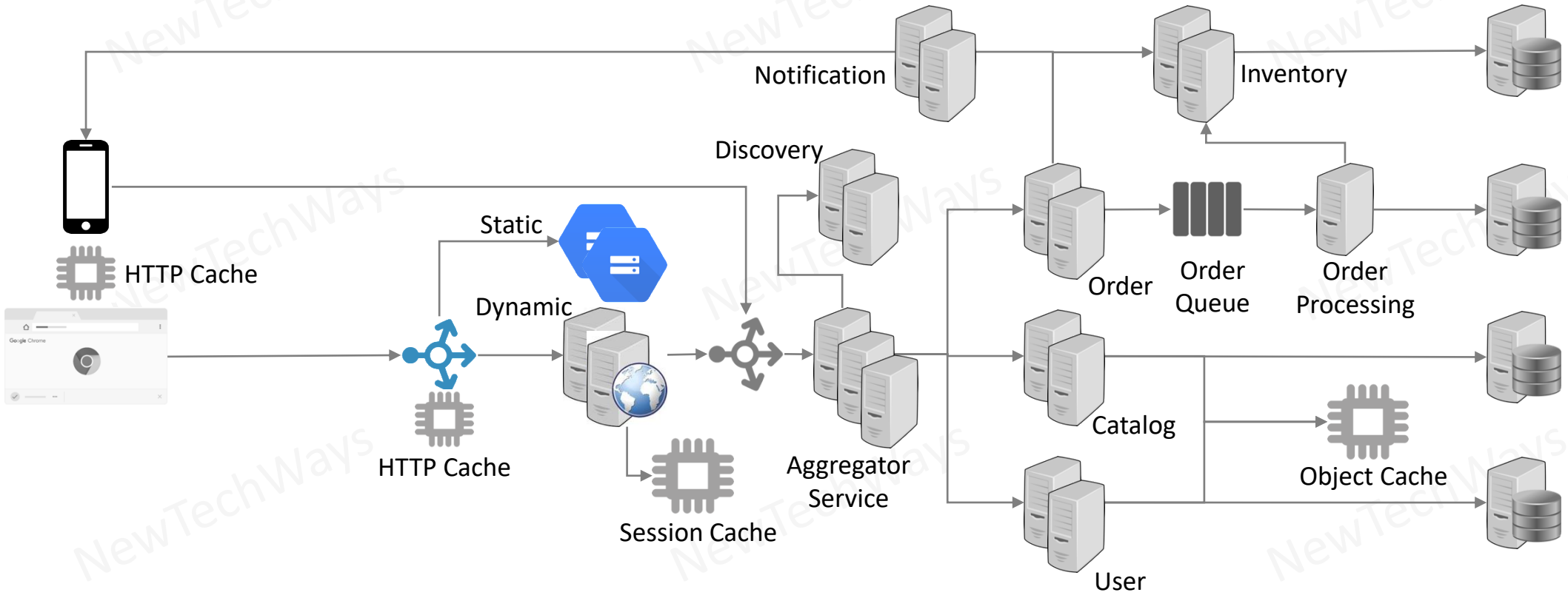
Caching & Compression

- Caching and compression for reducing response latency
- Caching can also reduce the backend load



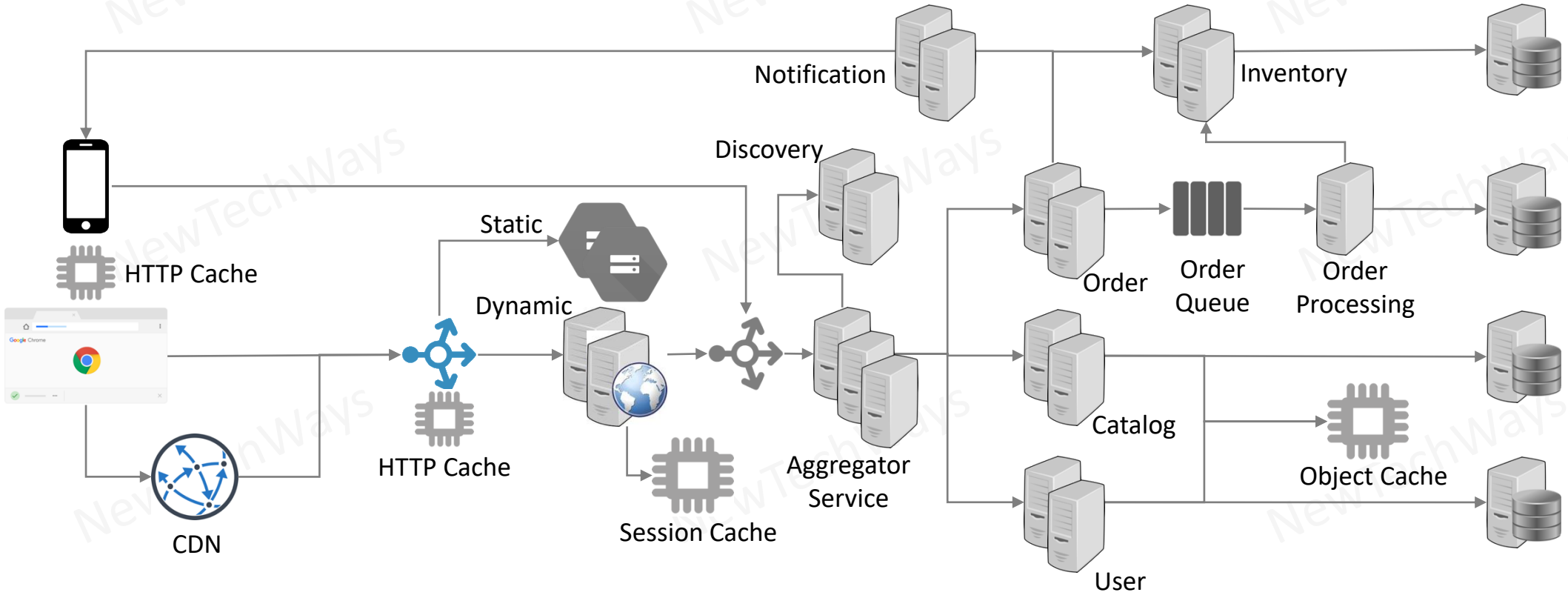
Serving Static Data

- High throughput Cloud Object Stores for serving static content



CDN & Edge Proxy

- CDN for global distribution of static content
- Edge Proxies for reducing latency through Early Termination, SSL Termination



Think About It!

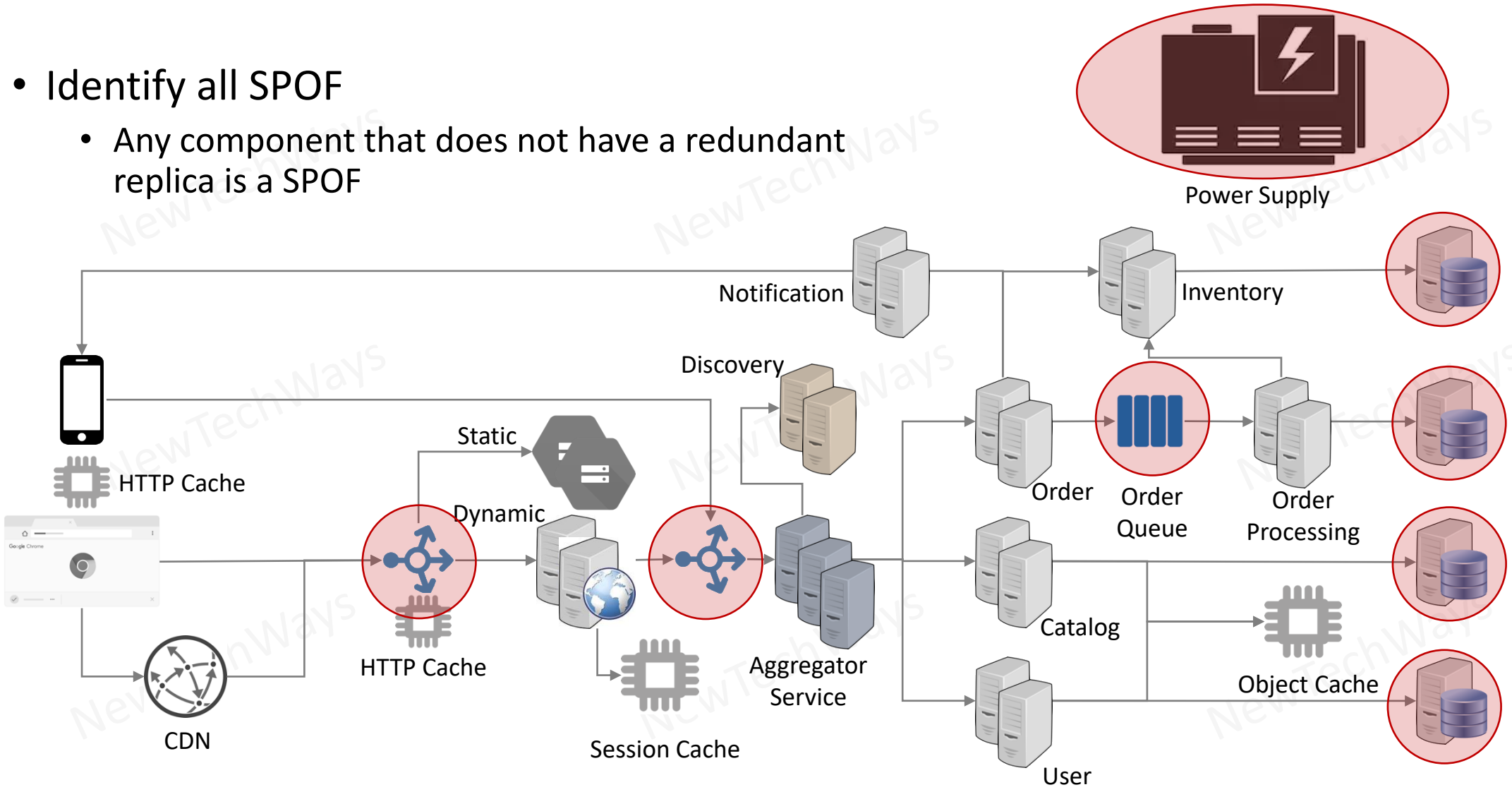
- User Session Management
- Load Balancer
  - Routing & Sessions
  - Types – Software/Hardware, External/Internal
  - Policies
- Balancing Latency & Throughput
- RDBMS Scaling Limitations
- NoSQL Vs RDBMS
- Transaction Concurrency
- Eventual Consistency
- Routing for Global Scalability

Reliability Principle

- Reliability
  - Normal functioning even in the presence of faults
- Availability
  - Always available even in the presence of faults
- Reliability and Availability are achieved mainly through Fault Tolerance
- Fault Tolerance requires
  - Provisioning Redundancy
    - Stateless & Stateful
    - Active, Passive, Cold
  - Fault Detection Mechanisms
    - Health-checks, heart-beats
  - Recovery or Failover
    - Stateless & Stateful

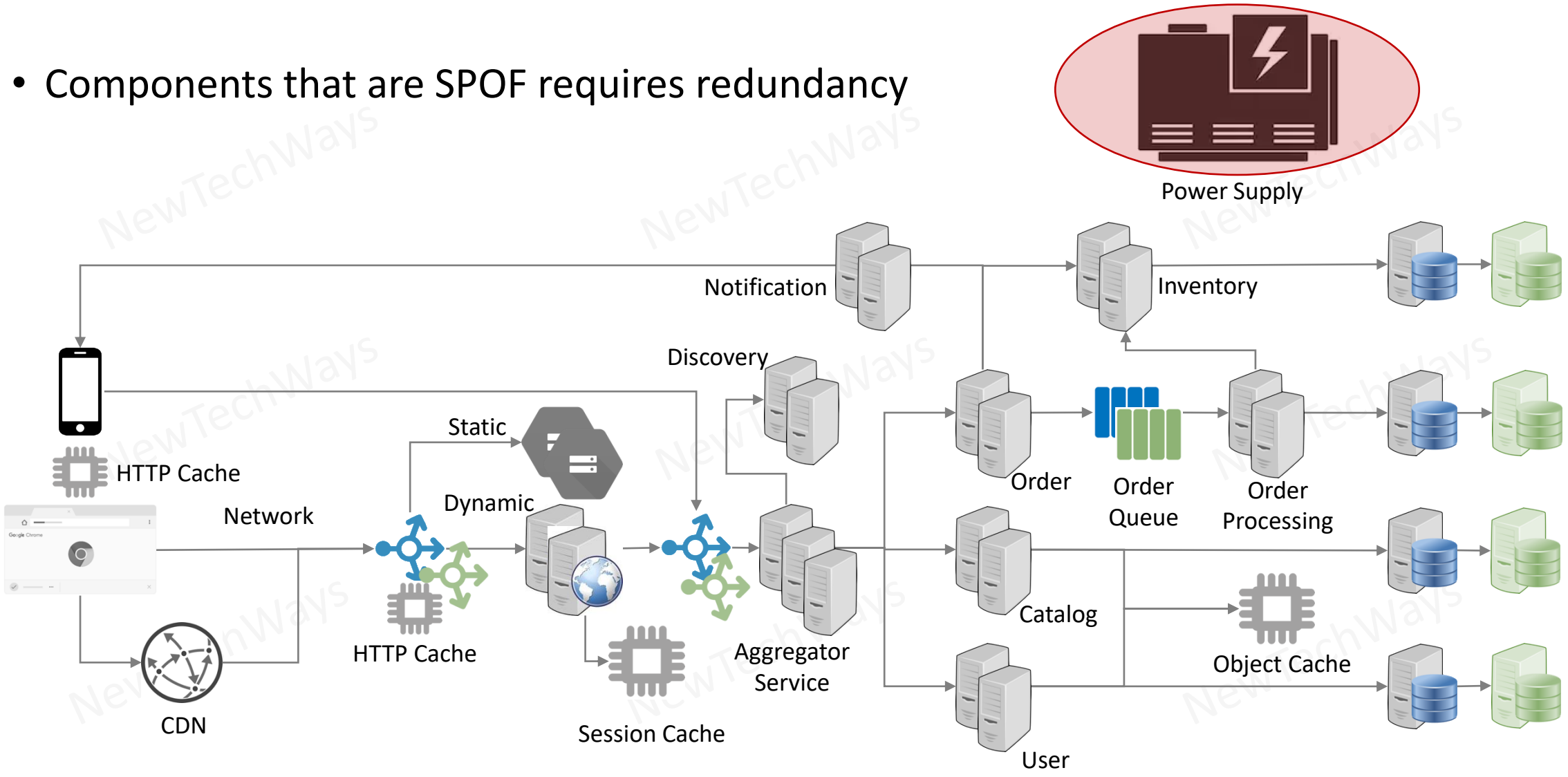
Single Point Of Failures

- Identify all SPOF
  - Any component that does not have a redundant replica is a SPOF



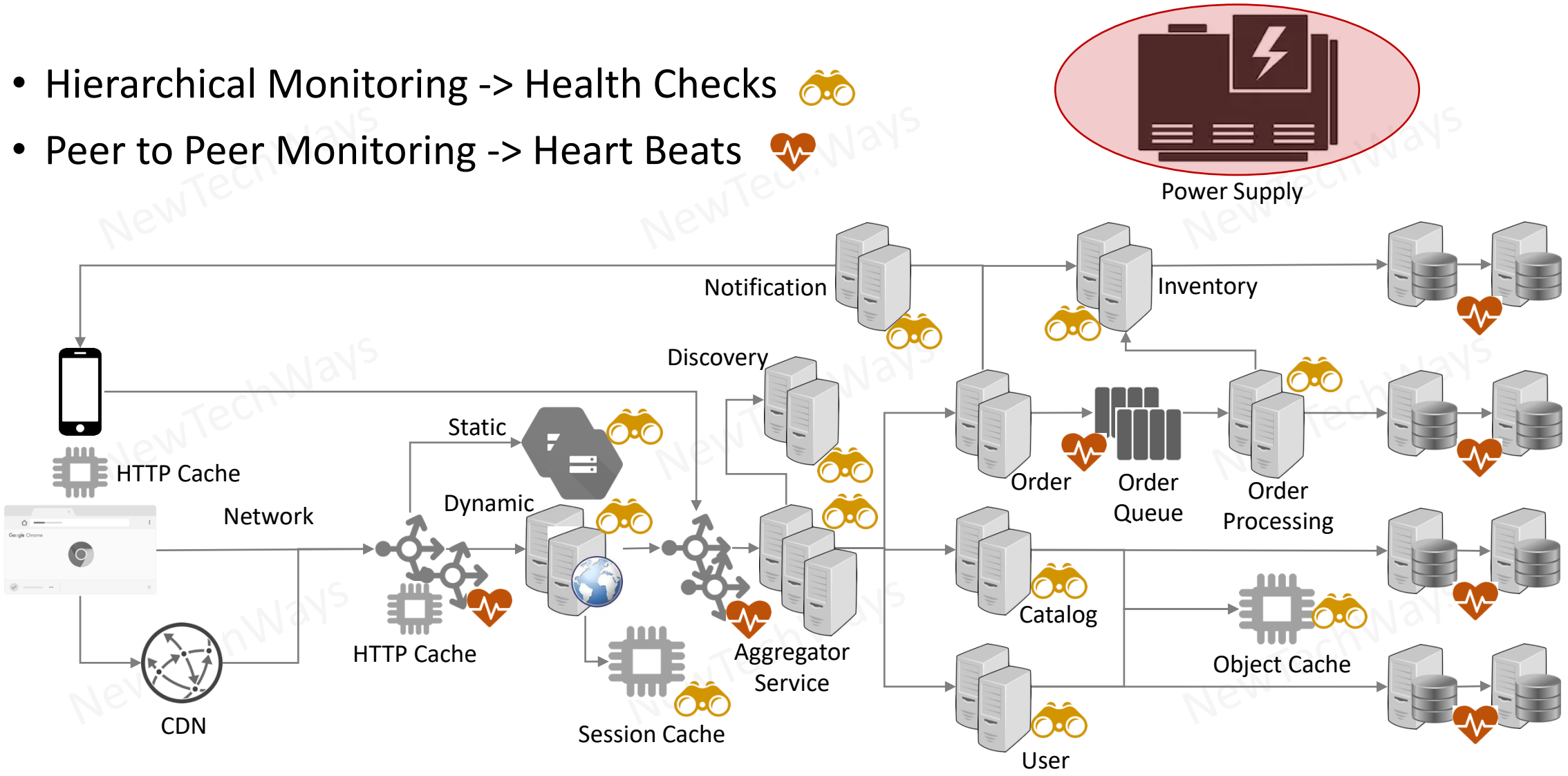
Redundancy

- Components that are SPOF requires redundancy



Monitoring For Fault Detection

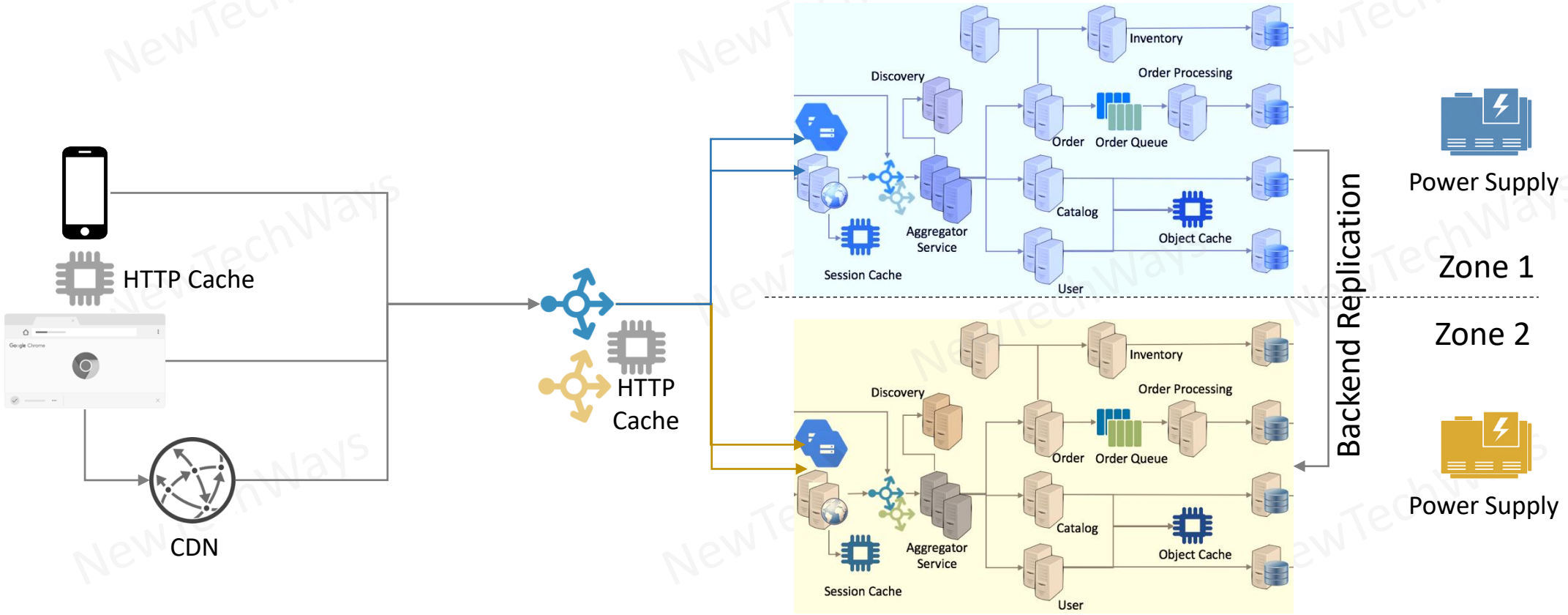
- Hierarchical Monitoring -> Health Checks
- Peer to Peer Monitoring -> Heart Beats



NewTechWays

Zonal Redundancy

- Zonal redundancy for fault isolation



Think About It!

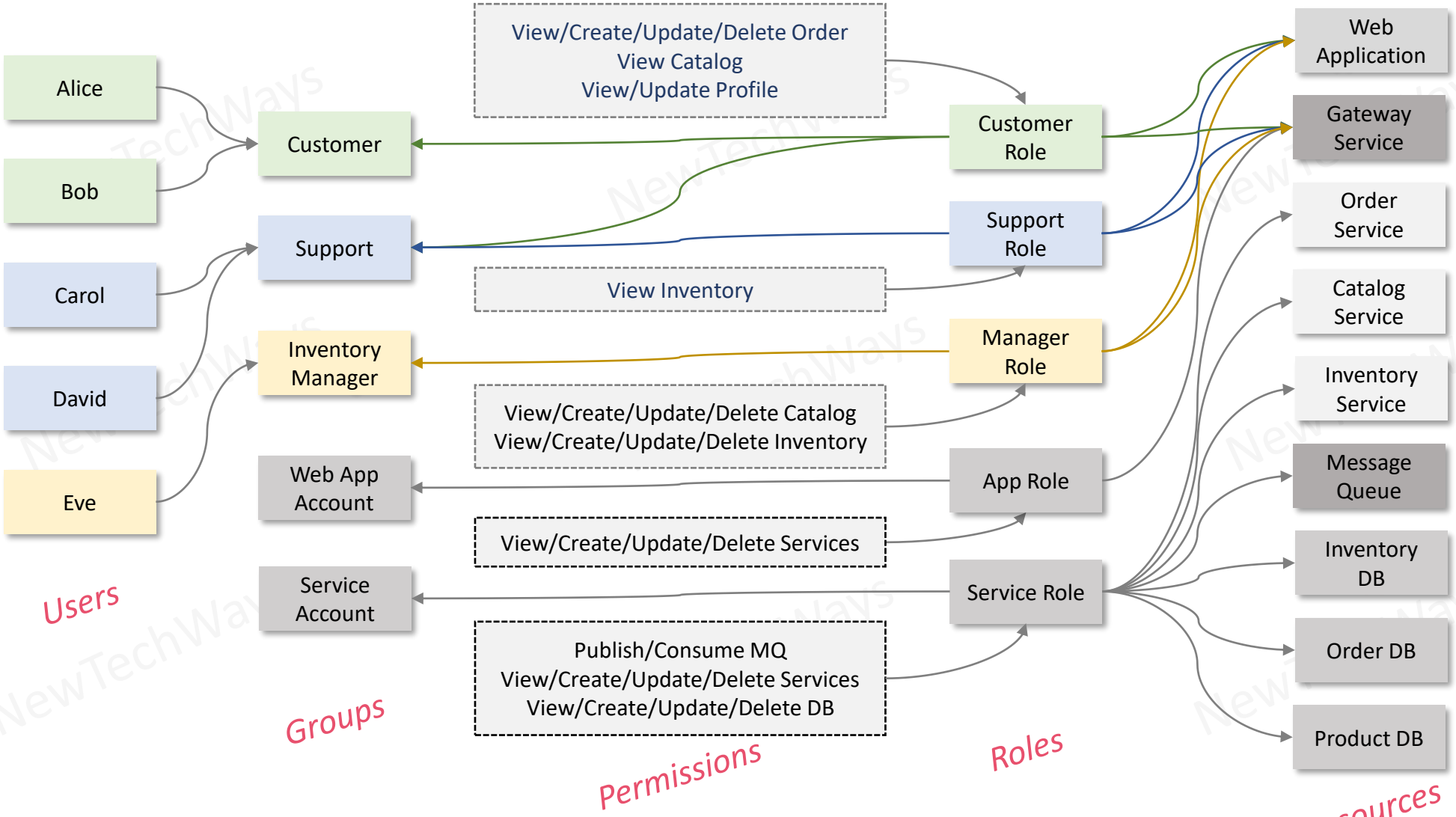
- Zonal Redundancy & Routing
- Regional Redundancy & Routing
- Redundancy & Recovery
  - Stateless Components
  - Stateful Components
- Automated Recovery
  - Detecting Faults
  - Triggering Failover
- Reliability Practices
  - Retries & Idempotency
  - Graceful handling of errors
  - Controlled service degradation
  - Load Shedding
  - Backpressure

Security Principles

- Least Privilege
  - Minimalistic user access rights just enough to perform a task
- Minimize attack surface area
  - Reduce entry points by authorizing every API request or Activity
    - Allow only minimum set of roles for each API
- Defense In Depth
  - Multiple layered resource access controls
- Separation of Duties
  - Different entities should have different roles
- Fail Secure
  - Failure should not expose more than required information through error messages or logs
- Weakest Link
  - Any system is as strong as its weakest link

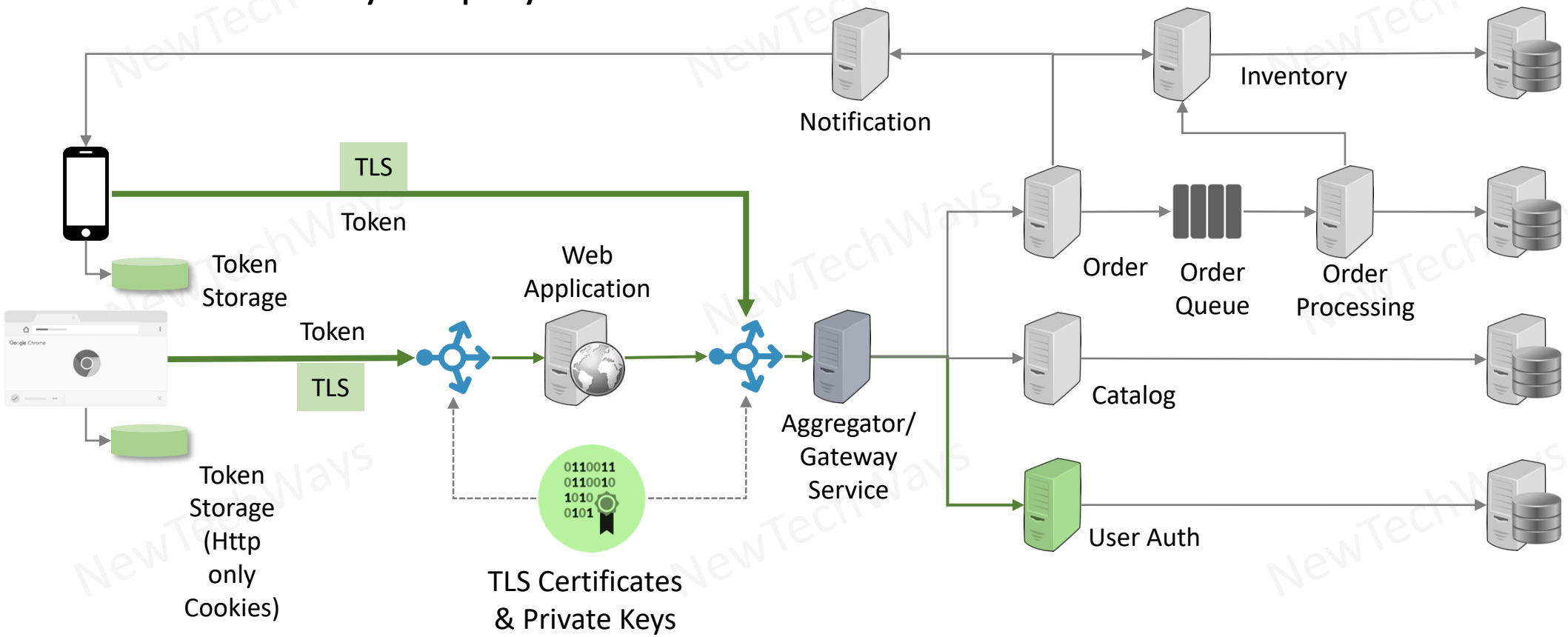
OWASP Top 10 - 2017
A1:2017-Injection
A2:2017-Broken Authentication
A3:2017-Sensitive Data Exposure
A4:2017-XML External Entities (XXE) [NEW]
A5:2017-Broken Access Control [Merged]
A6:2017-Security Misconfiguration
A7:2017-Cross-Site Scripting (XSS)
A8:2017-Insecure Deserialization [NEW, Community]
A9:2017-Using Components with Known Vulnerabilities
A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]

Role Based Access Control

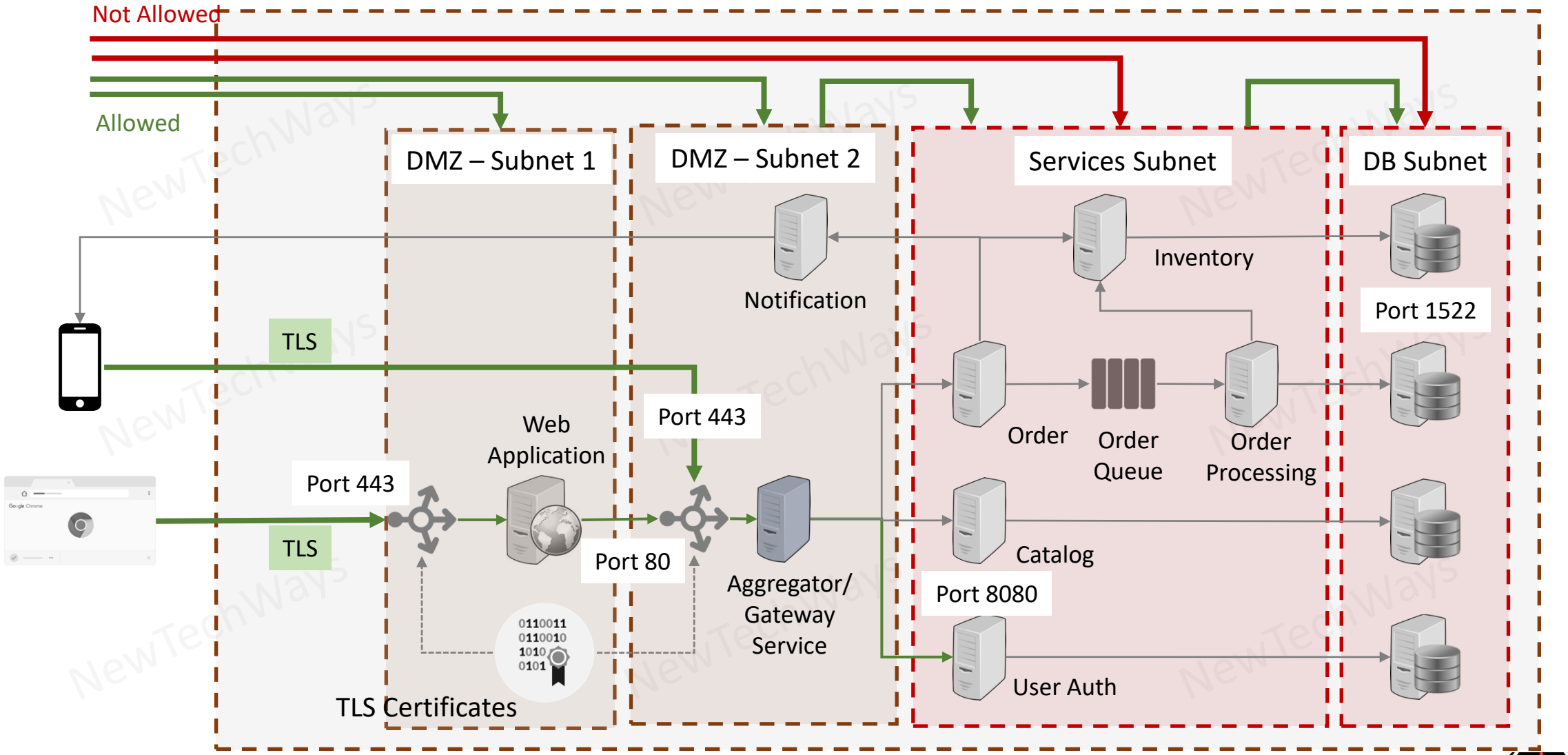


Authentication & Authorization

- Token based authentication over HTTPS
- Certificates & keys deployed on external load balancers



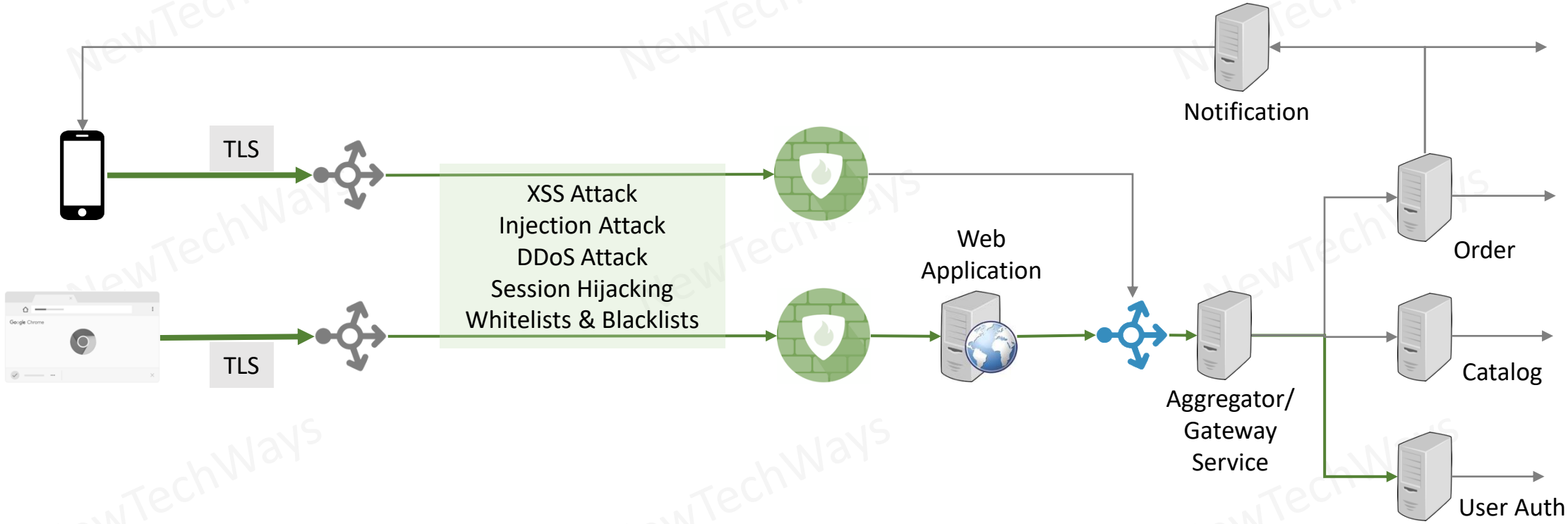
Network Security



NewTechWays

Web Application Firewalls

- Multiple security related attacks can be prevented by inspecting requests for common attacks and vulnerabilities by using Web Application Firewalls



Think About It!

- Security Token Storage
- Stateless & Stateful Authentication
- Third-Party Authorization with OAuth2
- Single Sign On
- TLS Termination
- Common Vulnerabilities
- Stored Data Encryption
- Key Management

That's Just The Beginning

Further challenge lies in the detail

And we have sorted it out for you

If you are an Architect or a Developer

And you want to bridge the gap between

A Great Developer and a True Architect

You should get started with

'NewTechWays' course

'Developer To Architect'

And leapfrog your career

Thank You!

NewTechWays

Academy of Software Architecture

Visit us at

https://www.newtechways.com