FORTINET | USA

XPERTS

SUMMIT 2024

Helping you create a digitally secure future.

Architecture of Modern HTTP-Based Applications

Module 03

Agenda

- Web Application Architecture
- Web Application Delivery
- Best Practices and Recommendations

Web Application Architecture

Sub header

Network Design

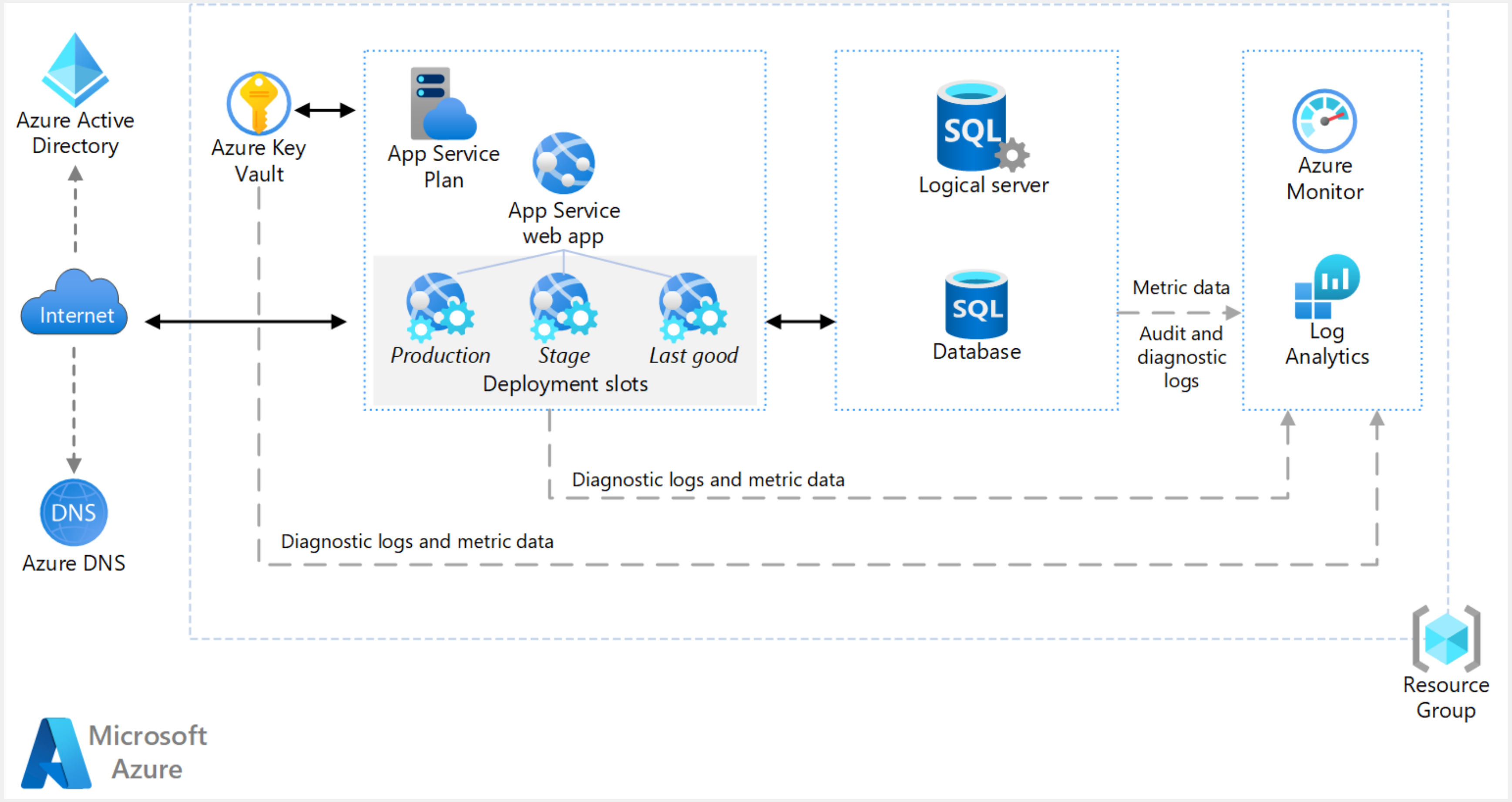
- Ensures data efficiency and security.
- Frameworks typically address topology, networking hardware, and protocols.
- A well-designed network ensures high availability, scalability, security, and optimal performance of the web applications.
- Integral for both on-premises and cloud solutions.

- Use Cases:
  - Large enterprise web applications distributed across multiple data centers.
  - Cloud services requiring robust network infrastructure for global accessibility and performance.
  - Global enterprise network setups directing traffic efficiently across continents.

Network Design

- Pros
  - Efficient data flow.
  - Reduced latency and improved speed.
  - Enhanced security and easier compliance with data standards.
- Cons
  - Complexity in setup and management.
  - Higher initial investment in sophisticated networking resources.

Network Design



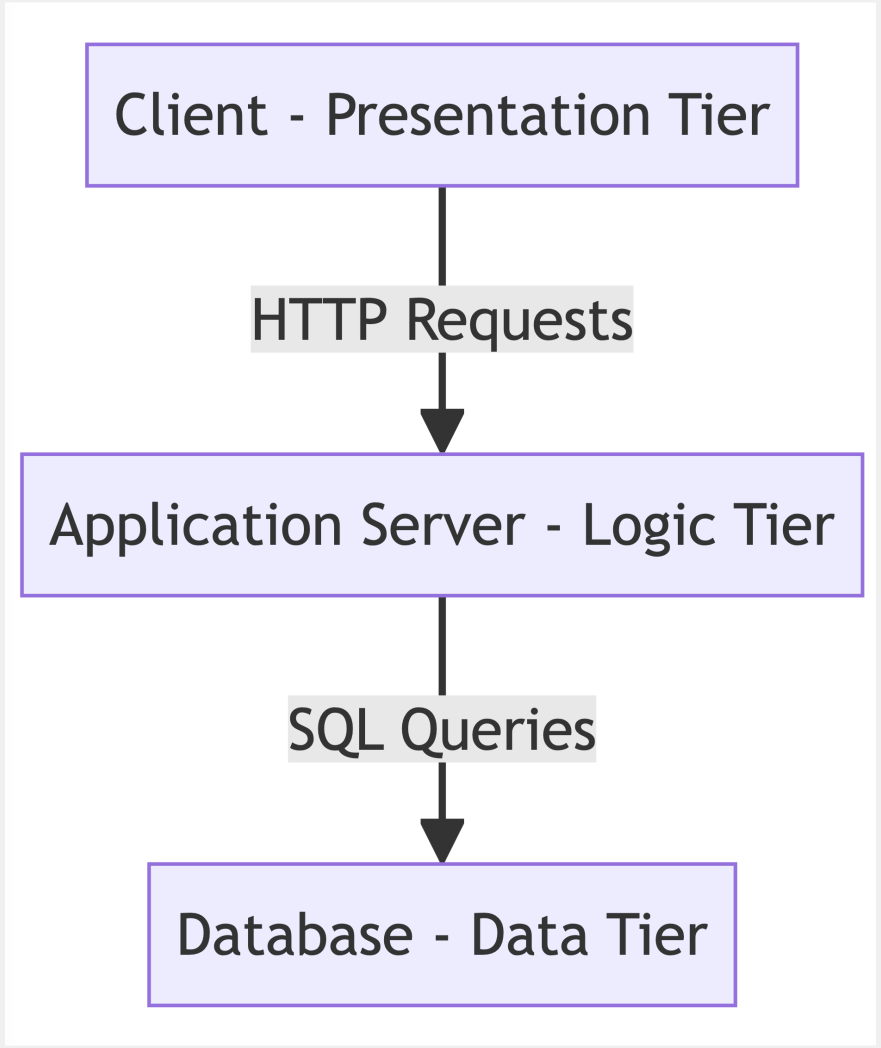
Tiering Deployments

- Divides applications into multiple layers such as presentation, logic, and data management.
- Allows for independent scaling and development of each tier.
- Tiering promotes separation of concerns, scalability, and maintainability, where each layer can be developed and updated independently.
- Enhances security by isolating critical database functions.
- Use Cases:
  - E-commerce platforms where user interface, business logic, and database management are distinctly separated.
  - Enterprise applications requiring robust security mechanisms between user data and business processes.

Tiering Deployments

- Pros
  - Enhanced security – separate layers restrict direct access to databases from the client side.
  - Improved scalability – each layer can be scaled independently based on demand.
- Cons
  - Increased complexity in deployment.
  - Potential performance overhead due to inter-tier communication.

Tiering Deployments



Web Application Delivery

Sub header

Content Delivery Network (CDN)

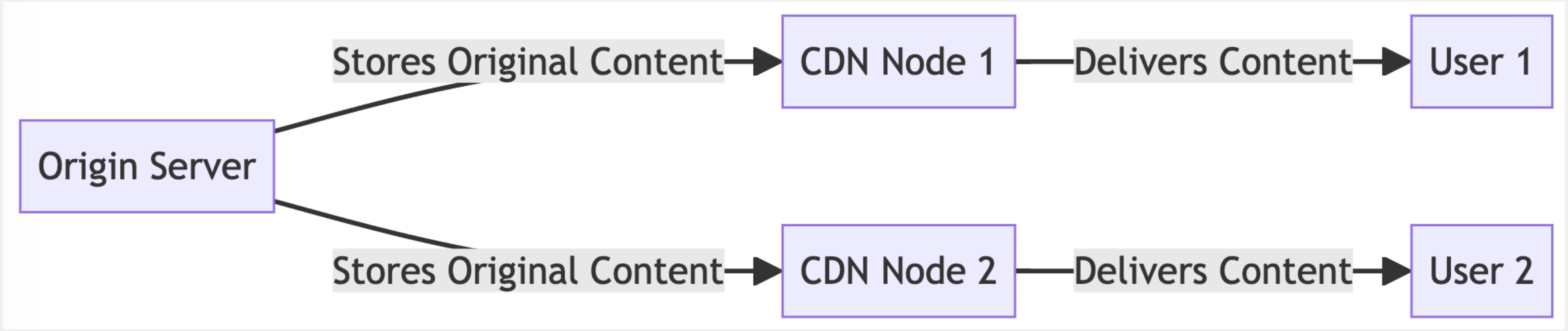
- A CDN is a network of servers distributed geographically to deliver internet content more rapidly to users by caching static content at edge locations closer to the user.
- Distributes copies of content at various network points to maximize bandwidth for content access.
- Reduces website load times.
- Use Cases:
  - Streaming platforms delivering video content worldwide.
  - Websites with international traffic looking to reduce latency.
  - Preventing DDoS attacks / High load spikes (Black Friday/Christmas)
  - Online retail businesses aiming for high-speed content delivery to enhance customer shopping experience.

Content Delivery Network (CDN)

- Pros
  - Reduced bandwidth costs.
  - Improved global availability and faster loading times.
  - Enhanced tolerance to spikes in traffic.

- Cons
  - Costs can be prohibitive for small to medium-sized enterprises.
  - Potential issues with stale content if not properly configured.

Content Delivery Network (CDN)



Serverless

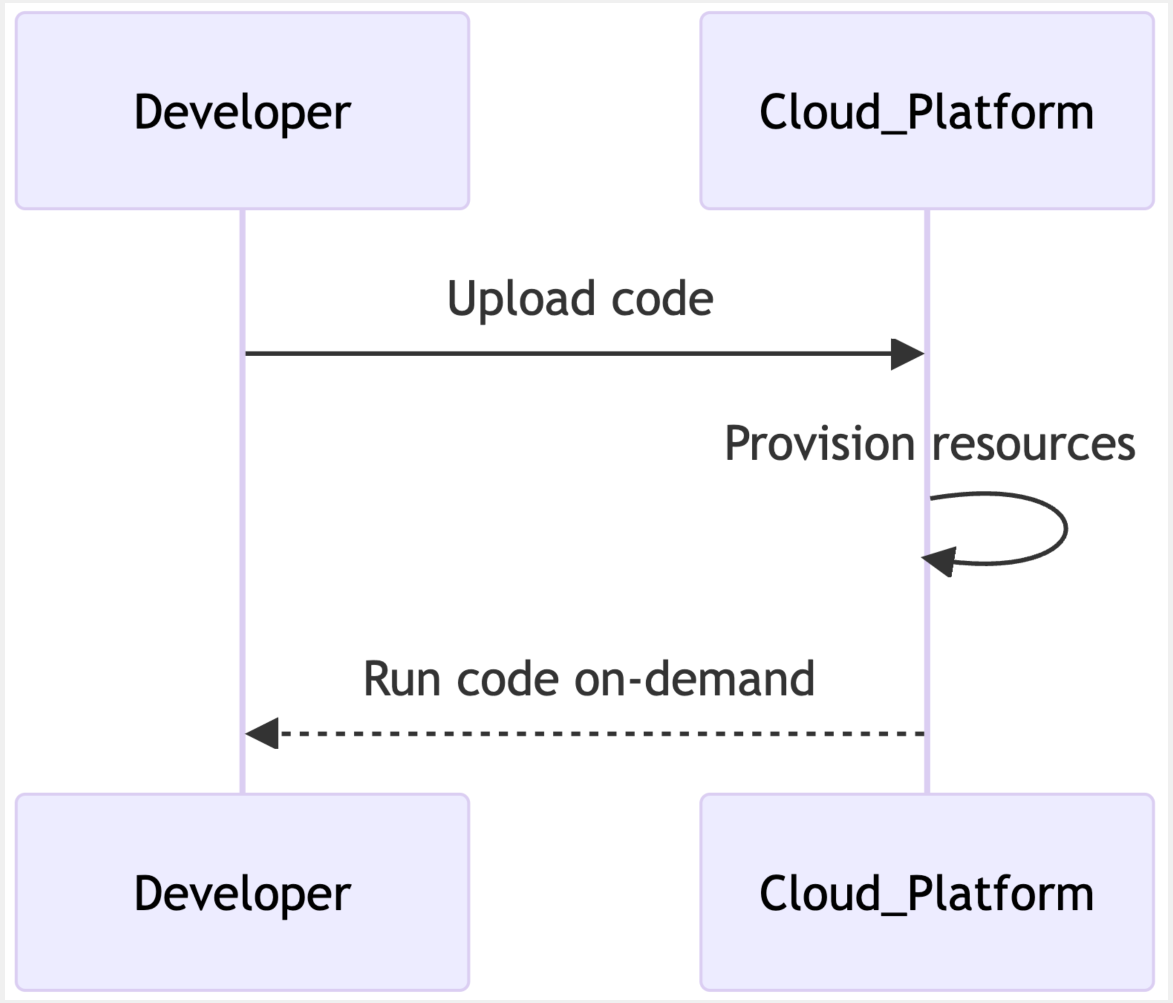
- Cost-effective for sporadic demand patterns.
- Reduces developer overhead for server maintenance.
- Allows developers to build applications without managing servers. The cloud provider automatically handles the server allocation and scaling.
- Serverless architectures reduce operational complexities in deploying code and can significantly lower costs associated with backend servers.
- Use Cases:
  - Dynamic scaling applications for event-driven architectures like IoT.
  - Event-driven applications responding to online customer actions.
  - Applications requiring high availability without constant server use.

Serverless

- Pros
  - Eliminates the need for ongoing server management and maintenance.
  - Costs are based on actual use, not reserved capacity.

- Cons
  - Performance issues like cold starts.
  - Limited control over backend environment and potential vendor lock-in.

Serverless


Developer
Cloud_Platform
Upload code
Provision resources
Run code on-demand
Developer
Cloud_Platform

Load Balancing

- Spreads incoming network traffic across multiple servers to ensure reliability and high availability.
- Essential for maintaining high performance and availability in applications with variable traffic and heavy user loads.
- Prevents any single server from becoming a bottleneck.
- Improves application responsiveness.

- Use Cases:
  - High-traffic websites like news portals during high-impact events.
  - Cloud applications and services with elastic scalability needs.
  - Corporate websites during product launches or promotional events.

Load Balancing

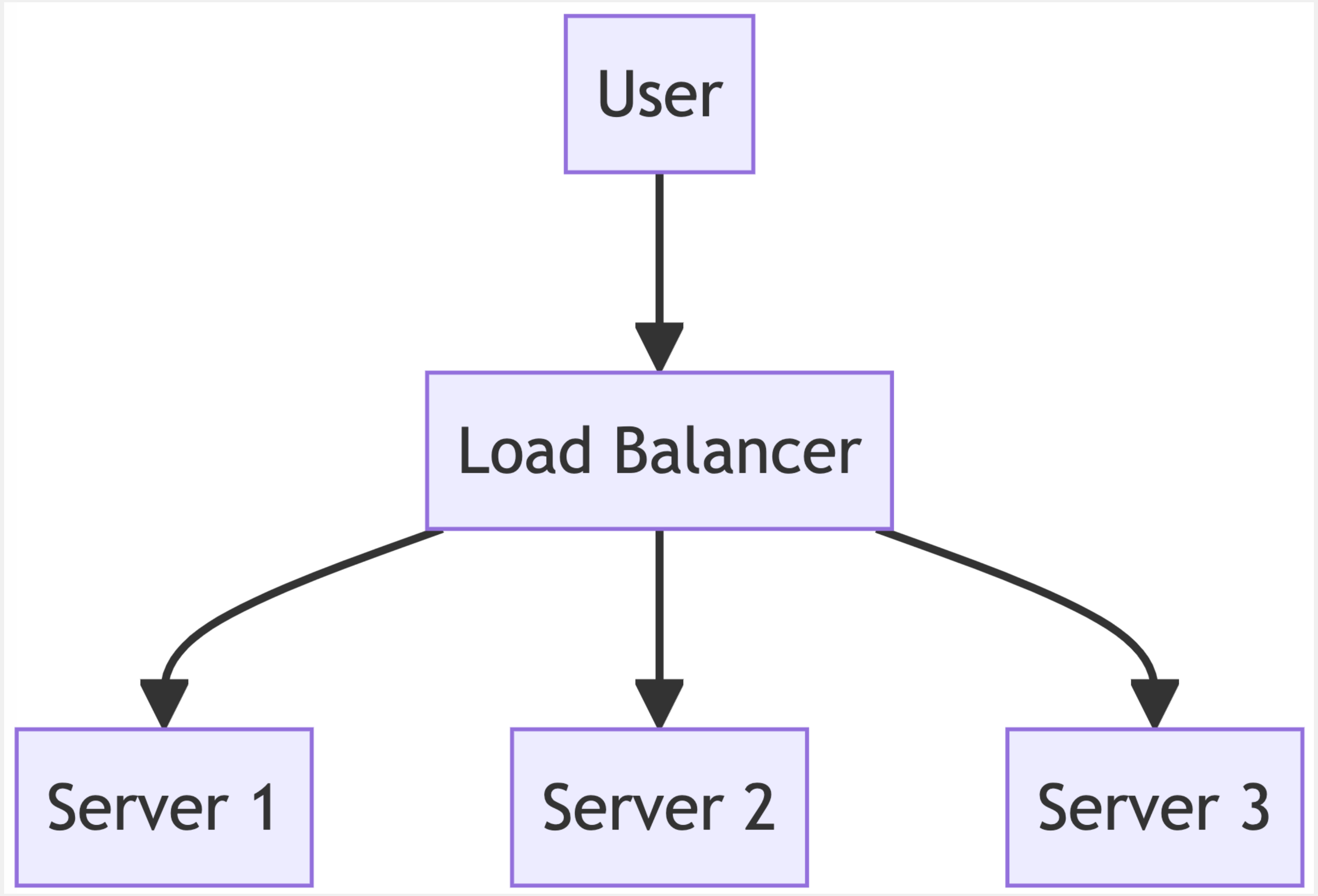
- Methods:
  - Static load balancing
    - Round-robin method
    - Weighted round-robin method
    - IP hash method
  - Dynamic load balancing
    - Least connection method
    - Weighted least connection method
    - Least response time method
    - Resource-based method

Load Balancing

- Pros
  - Better fault tolerance.
  - Scalability and flexibility in resource management.

- Cons
  - Can introduce complexity in the configuration.
  - Occasionally needs manual intervention for optimal performance.

Load Balancing



Best Practices and Recommendations

Sub header

Security Best Practices

- Essential Security Practices
  - Layered Security
  - Regular Updates and Patch Management
  - Authentication and Authorization
- Security Technologies to Consider
  - Web Application Firewalls (WAFs
  - Encryption - Use HTTPS to secure data in transit and employ encryption for sensitive data at rest.
  - DDoS Protection

Performance Optimization Strategies

- Key Performance Practices
  - Content Delivery Networks (CDNs)
  - Caching Mechanisms
  - Load Balancing Configuration
- Performance Tools and Techniques
  - Minimization and Compression
  - Asynchronous Loading
  - Serverless Architectures

High Availability and Scalability

- High Availability Fundamentals
  - Redundancy / Cloud / Hybrid
  - Health Checks
- Scalability Practices
  - Elastic Infrastructure
  - Stateless Application Design
  - Containerization
- Implementing Scalability and Availability
  - Database Replication
  - Microservices Architecture

FORTINET®