



## Episode 2: Architectural Thinking





## Who is a **Solutions Architect**?

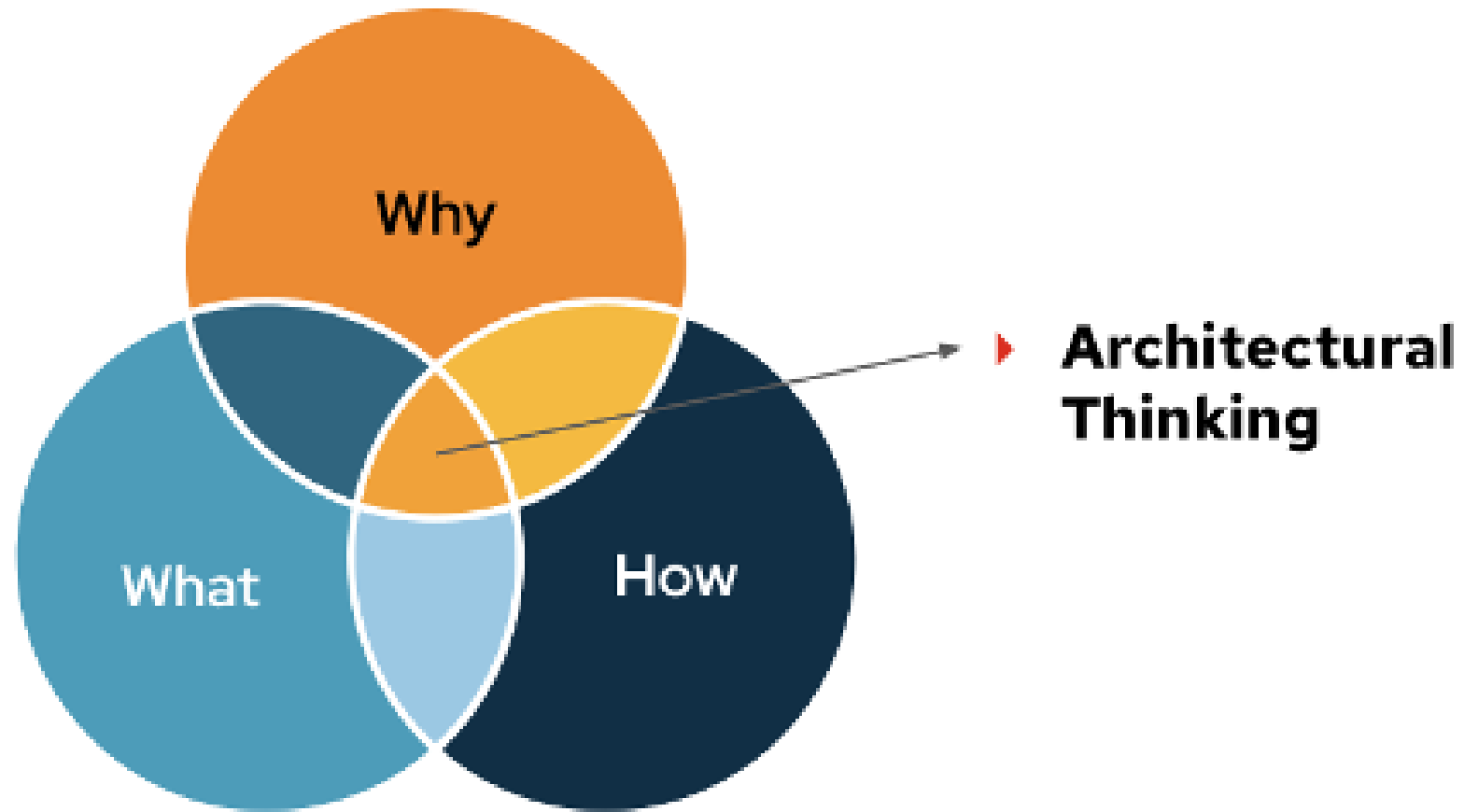
“A Solutions Architect is the **bridge** between **business problems** and **technical solutions**, ensuring systems are scalable, secure, and aligned with business goals.”

# Architectural Thinking:

“An approach to designing solutions that looks at the **big picture, balances trade-offs,** and aligns technology decisions with business goals.”



## Architectural Thinking principles



# Online Food Delivery Platform (Talabat / Uber Eats)



# WHY

## Business Goal:

- توصيل الطعام بسرعة وجودة عالية.

## Key Drivers: تجربة مستخدم ممتازة،

- تقليل وقت التوصيل، زيادة حجم الطلبات.

## Architect's Thinking:

- balance speed (performance) & reliability.

# WHAT - 4E Activities

---

## Explore

- Gather requirements: Mobile app for customers, dashboard for restaurants, driver tracking system.

## Evaluate

- Which database to choose? **SQL (consistency)** vs **NoSQL** (flexibility & speed).
- Cloud vs On-Premises infrastructure?

## Elaborate

- Design architecture: Mobile App → API Gateway → Microservices → Database.
- Integration with Payment Gateway + Maps API.

## Execute

- Build an MVP (Minimum Viable Product).
- Perform Load Testing.
- Deploy on Azure or AWS.

# HOW - Requirements

## Functional Requirements: (What the system should do)

- The user can place an order.
- The driver can receive and track the order.
- The restaurant can update the order status.

## Non-Functional Requirements (What the system should do) (Qualities):

- **Scalability:** The system must handle thousands of orders during peak hours.
- **Availability:** The service should run 24/7 without downtime.
- **Performance:** Pages should load in less than 2 seconds.
- **Resilience:** In case of a server failure, the system should automatically reroute traffic.

# System Qualities (Key NFRs)

## ◆ SCALABILITY:

Ability to increase or decrease resources (Servers, Storage, Databases) automatically based on demand.

- Example: Auto-scaling in Azure VM Scale Sets.

## ◆ Availability:

Ensuring the system is continuously running without downtime.

- Example: 99.99% SLA with Azure Availability Zones.

## ◆ Performance:

System responsiveness and efficiency even with a large number of users.

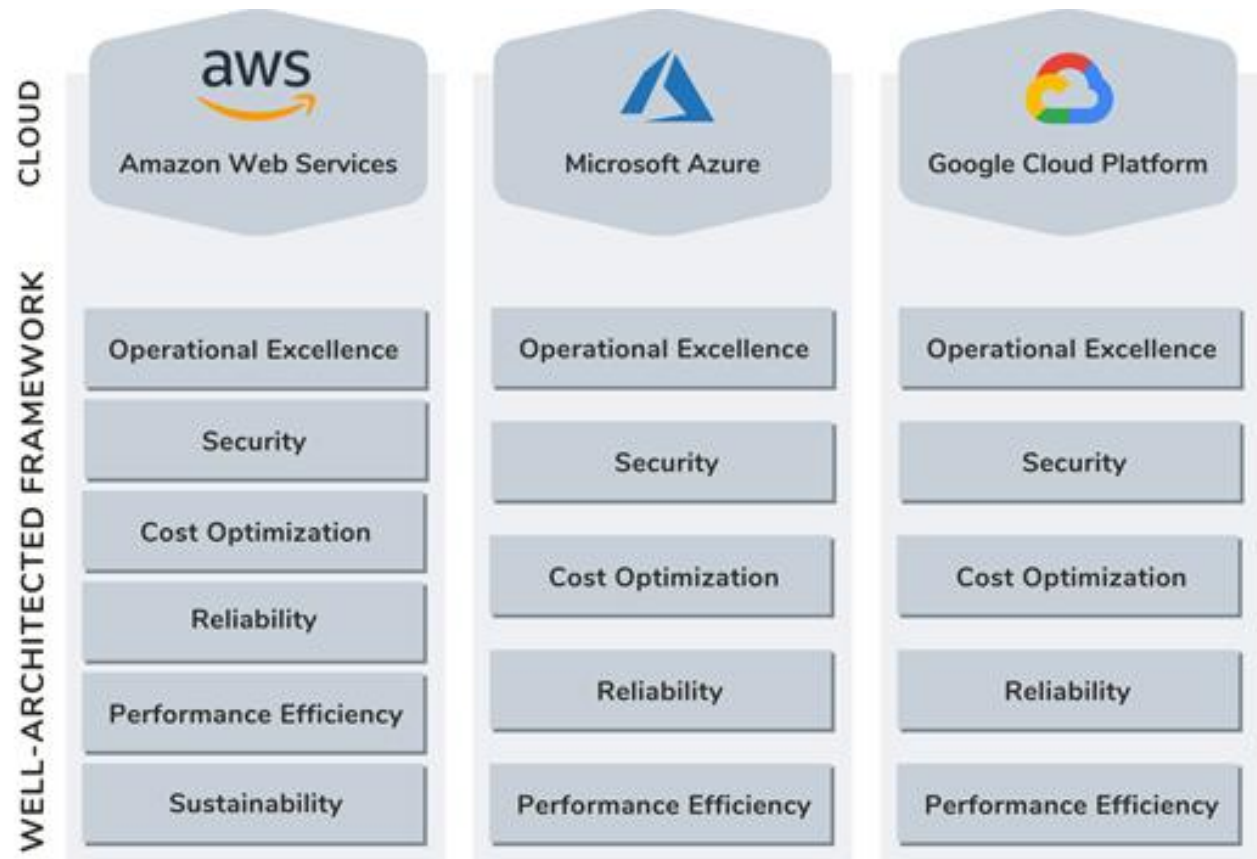
- Example: Caching + CDN to improve response time.

## ◆ Resilience:

The ability of the system to continue operating even if part of it fails.

- Example: Geo-Redundancy + Disaster Recovery.

# well- architected framework



## Architect's Mindset

---

**“Built for Speed.  
Designed for  
Scale.”**

# Trade-offs

- Architects balance competing priorities
- Choosing one quality often impacts another



# Trade-offs

## Monolith vs. Microservices



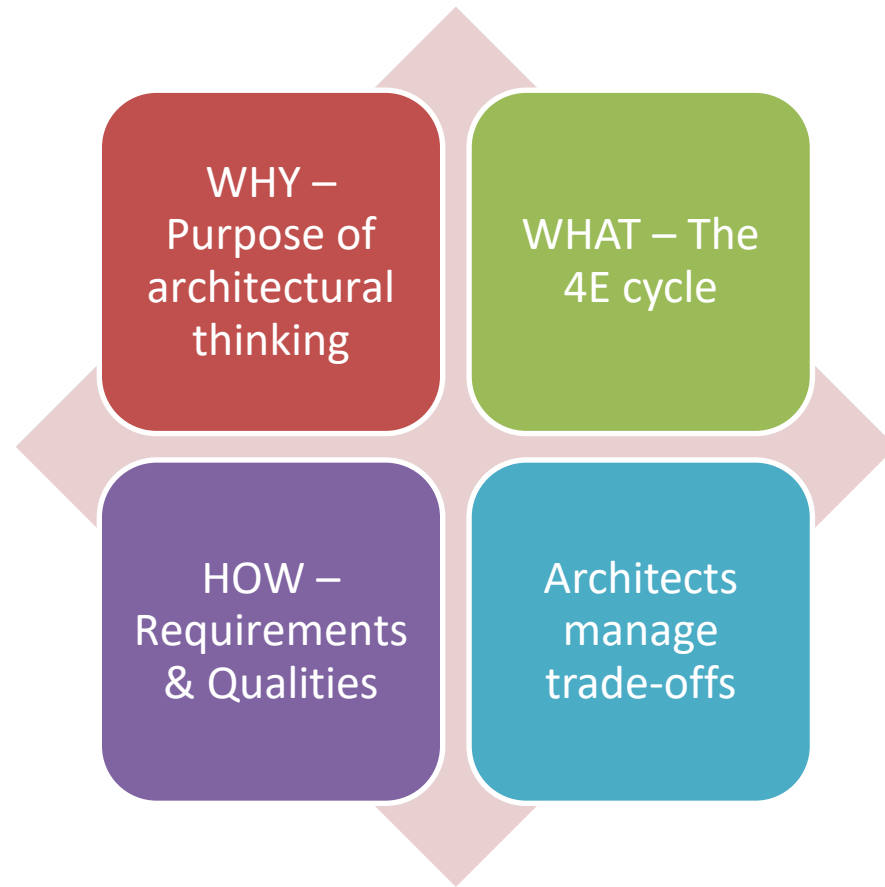
### Monolith•

- أسهل وأسرع في البداية. ✓
- Debugging وإدارة الـ Codebase أوضح. ✓
- صعب يتوسع مع تزايد المستخدمين. ✗
- أي تغيير صغير ممكن يكسر النظام كله. ✗

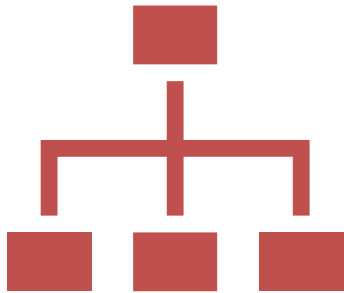
### Microservices•

- مرونة في التوسع ( scaling كل خدمة لوحدها زي Payment أو Orders). ✓
- فرق مختلفة تشتغل بالتوازي. ✓
- تعقيد عالي في الشبكات والـ Communication (API Gateway, Service Mesh). ✗
- محتاج Observability قوي (Logging, Monitoring). ✗

# Summary



# Closing



Think like an architect – structured, balanced, and value-driven



You are shaping systems to meet real business needs.

Thank you!



**Emad Adel**

Multi-Cloud Solutions Architect  
Microsoft Certified Trainer