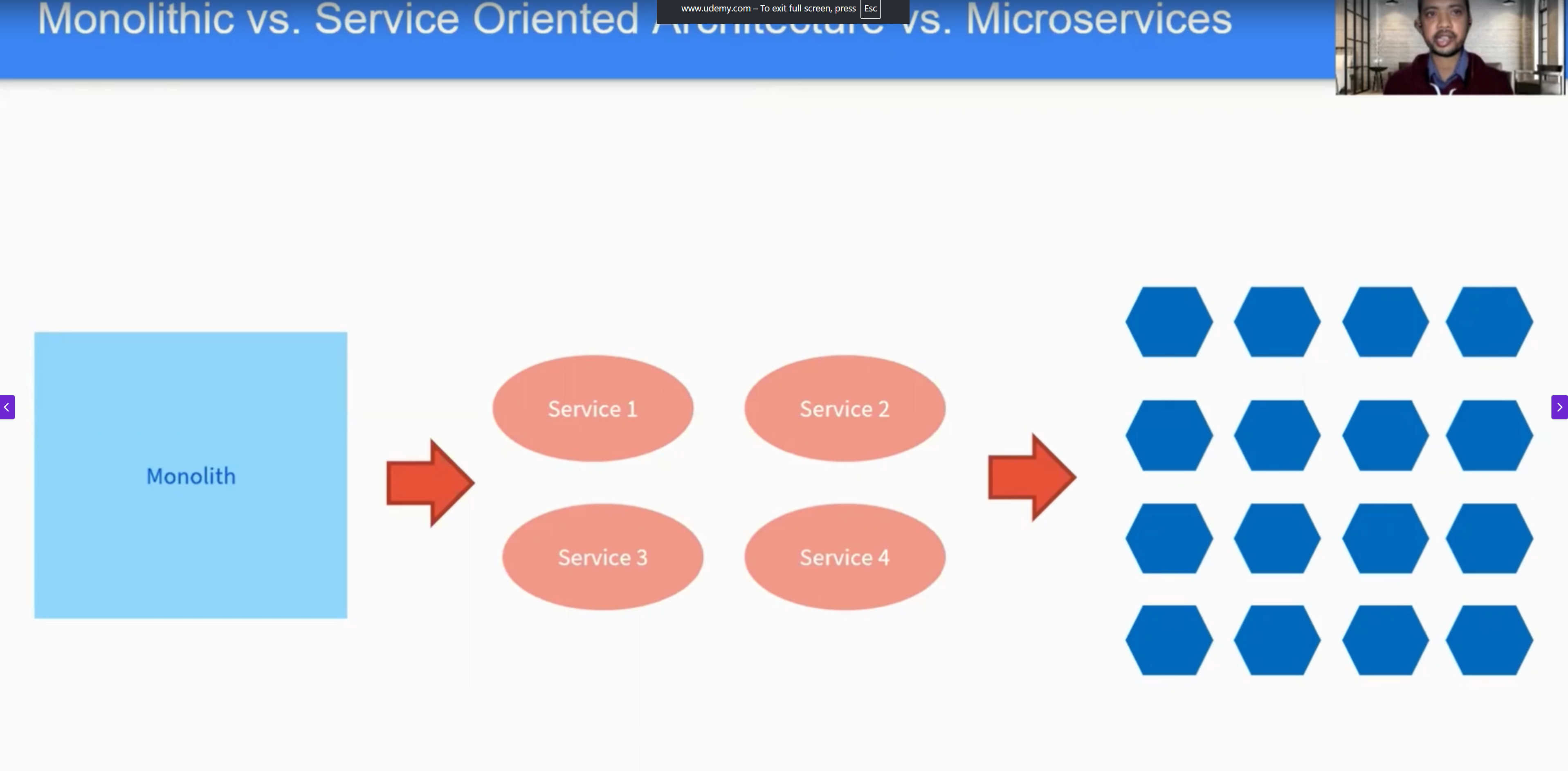
* 
* **Service-Oriented Architecture (SOA)** is a design pattern where software components (called "services") are loosely coupled and communicate with each other over a network using standardized protocols (typically via SOAP, XML, or HTTP). Each service in SOA is designed to perform a specific business function and can be reused across different applications.
* **🔹 1. What is SOA (Service-Oriented Architecture)?**
* **Definition**: SOA is a design pattern where business logic is split into individual **services** that are orchestrated and reused across systems.
* **Core Concepts**:
* Services are independent and loosely coupled
* Services interact via well-defined interfaces
* Emphasis on **interoperability** (especially in enterprise systems)
* Often uses **SOAP/XML**, **WSDL**, **ESB** (Enterprise Service Bus)
* **🔹 2. What is Monolithic Architecture?**
* **Definition**: A monolith is a single unified codebase that contains all the application’s modules.
* **Characteristics**:
* All features and functions are tightly integrated
* Single codebase and database
* Easier to develop initially but hard to scale/maintain
* Changes in one area can affect the whole system
* **🔹 3. What is Microservices Architecture?**
* **Definition**: Microservices break the application into **small, independently deployable services**, each responsible for a specific functionality.
* **Core Concepts**:
* Lightweight protocols like **REST/JSON** or **gRPC**
* Independent deployment and scalability
* Decentralized data management
* Follows **DevOps**, **CI/CD**, and containerization (Docker, Kubernetes)
* **🔸 4. Key Differences Between SOA, Monolith, and Microservices**

| * **Aspect** | * **Monolithic** | * **SOA** | * **Microservices** |
| --- | --- | --- | --- |
| * **Structure** | * Single unit | * Modular, services via ESB | * Small, independent services |
| * **Technology** | * One stack | * Interoperable (SOAP, WSDL, XML) | * Lightweight (REST, JSON, gRPC) |
| * **Scalability** | * Difficult | * Moderate | * High (each service scales individually) |
| * **Deployment** | * All together | * Complex due to central ESB | * Independently |
| * **Coupling** | * Tightly coupled | * Loosely coupled | * Highly loosely coupled |
| * **Reuse** | * Hard to reuse | * Reuse of enterprise-level services | * Code-level reuse is discouraged |
| * **Communication** | * Function calls | * ESB (SOAP/XML) | * HTTP/gRPC, message brokers |
| * **Best For** | * Simple, small applications | * Large enterprise systems | * Agile, scalable cloud-native apps |

**🔄 How SOA Works in This Example:**

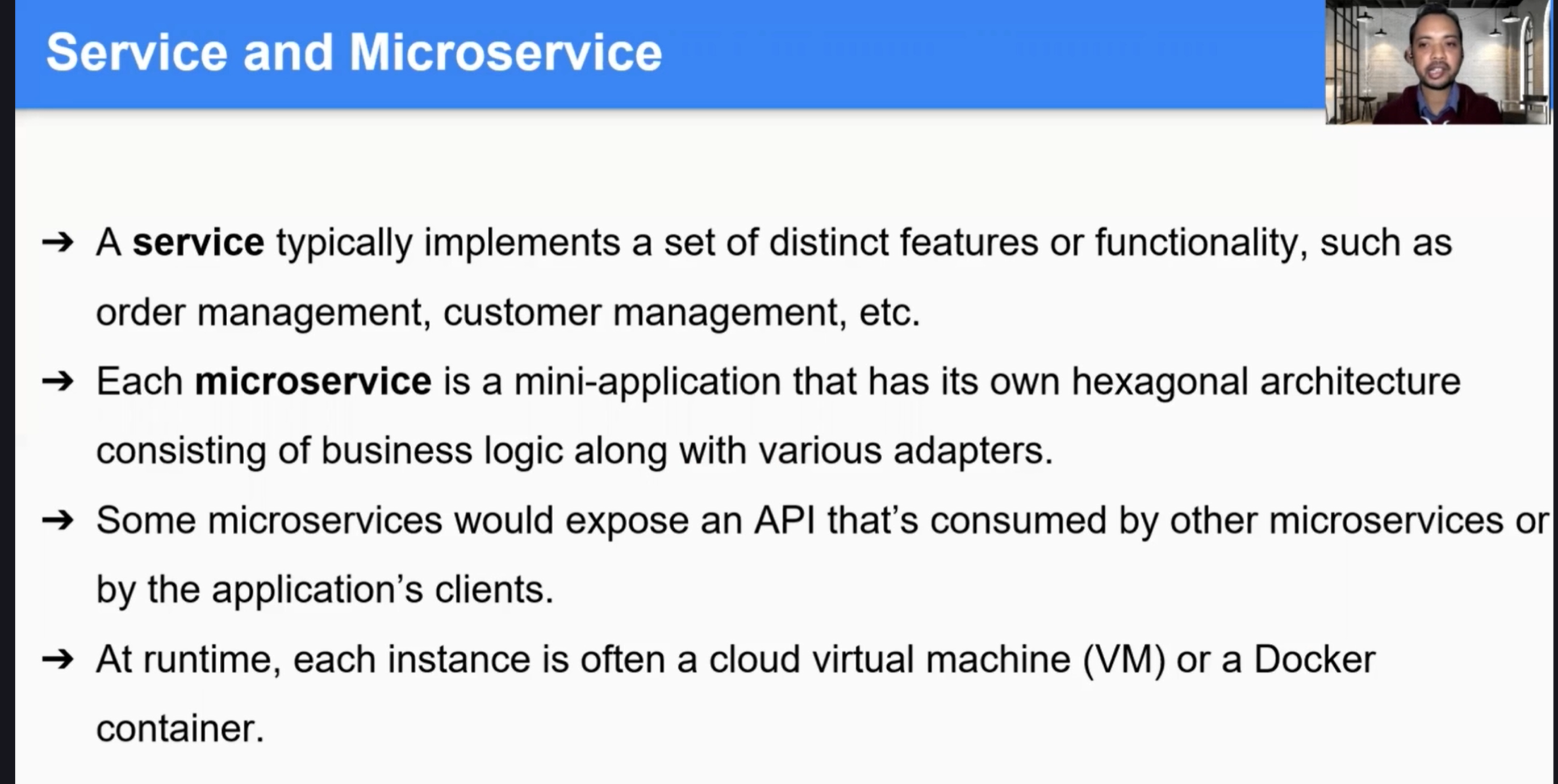
Each of the above modules is an **independent service** exposed over the network via **SOAP/Web Services** using **WSDL** (Web Service Description Language).

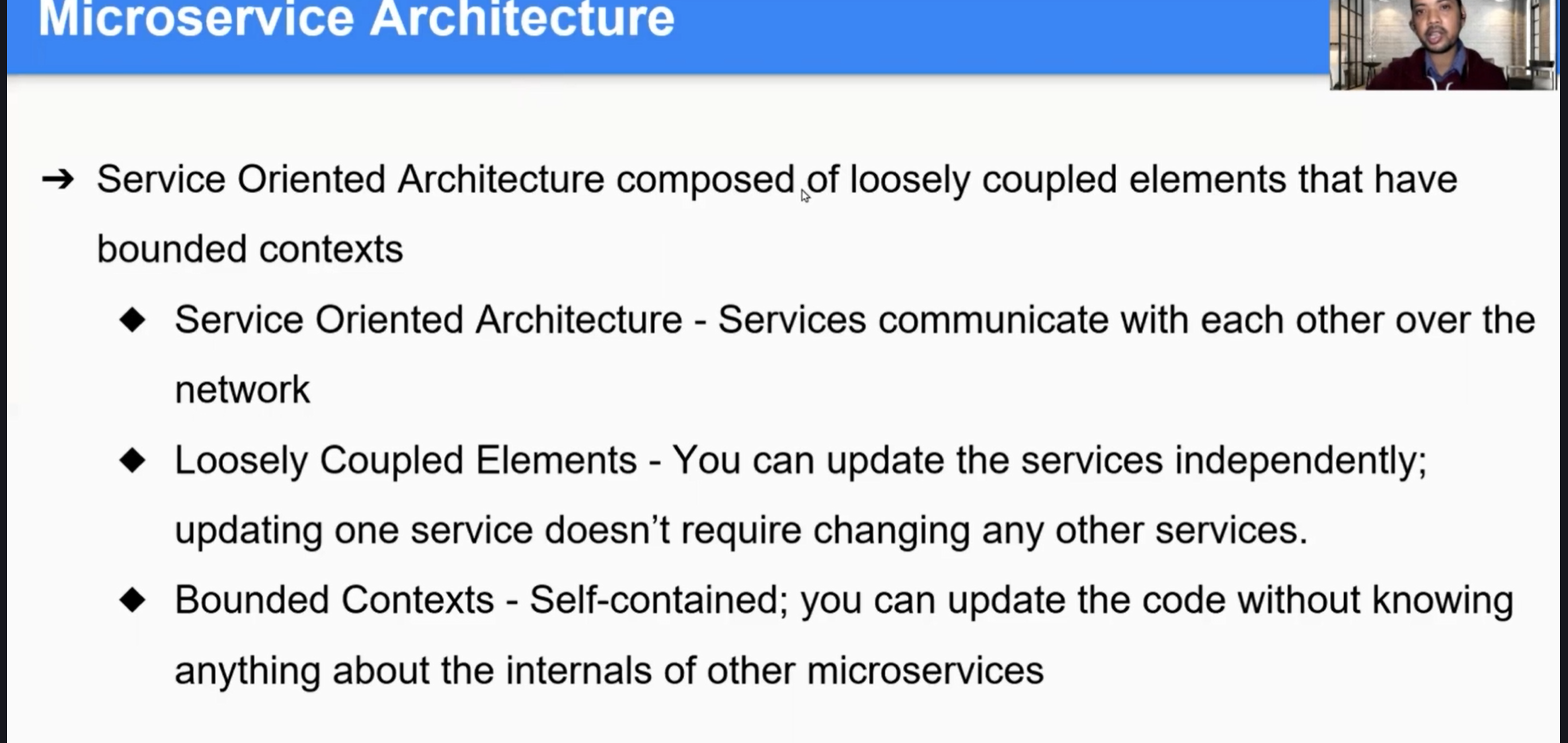
These services interact via a central component, often an **Enterprise Service Bus (ESB)**, which:

* Coordinates requests between services
* Transforms data formats
* Handles security, logging, and error handling

**📌 Scenario: Customer transfers money**

1. **User logs in** – Authentication Service validates credentials.
2. **User initiates transfer** – Funds Transfer Service checks balance and processes the transfer.
3. **Transaction complete** – Notification Service sends SMS/email confirmation.
4. **All services** exchange messages using **SOAP over HTTP** with **XML**.





* Microservice Architecture itself a Service Oriented Architecture
* In SOA if we Changning any service we might need to change another dependent service