

## Lab 8: Deployment View & Quality Attribute Analysis (ATAM)

This final lab shifts focus from coding to **architecture documentation** and **quality attribute evaluation**. Students will design the physical deployment of the Microservices Architecture and use a simplified Architecture Trade-off Analysis Method (ATAM) to evaluate how the chosen architecture meets key Non-Functional Requirements (NFRs).

---

### Objectives

1. Create a **UML Deployment Diagram** to visualize the physical setup of the Microservices (Deployment View).
  2. Conduct a simplified **Architecture Trade-off Analysis Method (ATAM)** focusing on **Scalability** and **Availability**.
  3. Compare the Monolithic (Layered) vs. Microservices architecture concerning the chosen Quality Attributes.
- 

### Technology & Tool Installation

This lab is primarily documentation and diagramming.

| Tool   | Purpose   | Installation/Setup Guide                                     |
|--|---|--|
| <b>draw.io<br/>(Diagrams.net)</b>              | Creating the UML Deployment Diagram and conceptual visualization. | Access online via a web browser.                             |
| <b>Google Docs/Word</b>                        | Documentation of the ATAM results and comparison matrix.          | Standard word processing software.                           |
| <b>Microservices Setup<br/>(from Labs 5-7)</b> | Conceptual input (mental model) for component deployment.         | N/A (No coding, but understanding the previous labs is key). |

---

### Activity Practice 1: UML Deployment Diagram

**Goal:** Model the physical allocation of software components (nodes) to execution environments (hardware/containers).

## Step-by-Step Instructions (Using draw.io)

1. **Identify Nodes (Physical Environments):** In UML, a node represents a computational resource (server, container, device). We'll define nodes based on common cloud deployment patterns.
  - **Action:** Create three main nodes/components using the "Node" shape (a 3D box):
    - **Client Device:** Represents the user's browser/mobile app.
    - **Load Balancer (Nginx/Cloud LB):** The entry point to distribute traffic.
    - **Application Cluster (Kubernetes/VMs):** The execution environment for the backend services.
2. **Place Artifacts (Services):** An artifact represents a physical piece of code (e.g., JAR file, Docker image). Place the following artifacts within the **Application Cluster** node:
  - **API Gateway Artifact** (from Lab 6)
  - **Product Service Artifact** (from Lab 5)
  - **Order Service Artifact** (conceptual)
  - **Notification Service Artifact** (from Lab 7)
  - **Message Broker Artifact** (RabbitMQ/Kafka)
3. **Place Data Stores:**
  - **Action:** Place a database symbol (e.g., PostgreSQL DB) and connect it to the **Product Service Artifact** and **Order Service Artifact**. *Crucially, show that each Microservice has its own, separate database instance.*
4. **Draw Associations (Communication):** Use dashed arrows to show network communication.
  - **Action:** Draw associations:
    - **Client Device** \$\rightarrow\$ **Load Balancer** (HTTP/HTTPS)
    - **Load Balancer** \$\rightarrow\$ **API Gateway Artifact** (Routes traffic)
    - **API Gateway Artifact** \$\rightarrow\$ **Product Service Artifact** (Internal HTTP/REST)

- **Order Service Artifact**  $\rightarrow$  **Message Broker Artifact**  
(AMQP/Queue Protocol)
  - **Message Broker Artifact**  $\rightarrow$  **Notification Service Artifact**  
(AMQP/Queue Protocol)
- 

### Activity Practice 2: Quality Attribute Analysis (Simplified ATAM)

**Goal:** Evaluate the architectural decisions based on key NFRs (**Scalability** and **Availability**) using a structured method.

#### Step-by-Step Instructions

1. **Define Scenarios (Test Cases):** For each Quality Attribute, define a scenario that challenges the architecture.
  - **Scalability Scenario (SS1):** "During a 5-minute Black Friday promotion, the system must handle a sudden **10x spike in concurrent users** placing items in their carts and viewing product details."
  - **Availability Scenario (AS1):** "The **Notification Service** fails completely for 1 hour due to a deployment error. The system must still be able to **successfully accept and process new orders.**"
2. **Evaluate Architectures against Scenarios:** Analyze how the **Monolithic (Layered)** vs. **Microservices** approaches would handle the defined scenarios.

| Quality Attribute | Scenario                         | Monolithic (Layered) Approach  | Microservices Approach  |
|-------------------|----------------------------------|--|---|
| Scalability       | SS1 (10x User Spike)             | <b>Response:</b> Must scale the <i>entire</i> application instance (Database, UI, Logic) even if only the Product Catalog needs extra capacity. This is inefficient. | <b>Response:</b> Can scale only the <b>Product Service</b> and <b>Cart Service</b> instances independently. The Database can be sharded/replicated specifically for high-read services. <b>Efficient scaling.</b> |
| Availability      | AS1 (Notification Service Fails) | <b>Response:</b> If the Notification logic is tightly coupled within   | <b>Response:</b> Due to the <b>Event-Driven Architecture (Lab 7)</b> , the Order Service places the event in the  |

| Quality Attribute | Scenario | Monolithic (Layered) Approach  | Microservices Approach  |
|-------------------|----------|--|---|
|                   |          | the Monolith's main process, the <i>entire transaction</i> might fail, or at least be slowed, reducing overall availability. | <b>Message Broker.</b> The Notification Service failure has <b>zero impact</b> on the Order Service's ability to complete the order. <b>High fault isolation.</b> |

3. **Identify Trade-offs:** Based on the evaluation, explicitly state the main architectural trade-offs.

- **Trade-off:** Microservices provide superior **Scalability** and **Availability** (Fault Isolation) but introduce significant **Complexity** in deployment (need for Kubernetes/Docker, API Gateway, Message Broker) and **operational overhead**. The Monolith is simpler but sacrifices resilience.
- 

### Documentation & Submission

Prepare the final documentation artifacts for this lab.

1. **UML Deployment Diagram:** The diagram created in Practice 1, clearly showing the nodes, artifacts, and communication links.
  2. **ATAM Analysis Table:** The completed table from Practice 2 comparing the two architectures against the defined Scenarios (SS1 and AS1) and justifying the findings.
  3. **Trade-off Statement:** A concise paragraph summarizing the architectural trade-offs identified.
- 

This concludes the 8-lab series, successfully guiding the student through the full lifecycle from requirements and monolithic design to modern microservices, EDA, and architectural quality analysis.