## 1. Deployment View (Textual Description)

Since the system is deployed in a local simulation environment using PHP Native Servers and Docker, the physical nodes and their execution environments are defined as follows:

### 1.1. Client Tier

- **Node:** Client Device (Web Browser).
- **Responsibility:** Renders the user interface and sends HTTP requests.
- **Protocol:** Communicates via HTTP/REST.

### 1.2. Application Tier (Microservices)

- **Node 1: API Gateway**
  - **Execution Environment:** PHP Built-in Server.
  - **Port:** 5000.
  - **Component:** gateway.php.
  - **Responsibility:** Entry point, Security Check (Token Validation), and Routing.
- **Node 2: Product Service**
  - **Execution Environment:** PHP Built-in Server.
  - **Port:** 5001.
  - **Component:** api.php.
  - **Responsibility:** Core business logic for managing product data.
- **Node 3: Notification Worker**
  - **Execution Environment:** PHP CLI (Background Process).
  - **Component:** consumer.php.
  - **Responsibility:** Asynchronous email processing.

### 1.3. Infrastructure Tier

- **Node 4: Message Broker**
  - **Software:** RabbitMQ.
  - **Port:** 5672 (AMQP).
  - **Responsibility:** Buffering and routing events between Product Service and Notification Worker.
- **Node 5: Database Server**
  - **Software:** MySQL / MariaDB.

- **Port:** 3306.
- **Responsibility:** Persistent storage for the webbandocu database.

---

## 2. ATAM Analysis (Architecture Trade-off Analysis Method)

We evaluated the architecture based on two key scenarios: **Scalability (SS1)** and **Availability (AS1)**.

### 2.1. Comparison Matrix

| Quality Attribute | Scenario Description | Monolithic Architecture (Lab 3) | Microservices Architecture (Lab 8) | Evaluation |
|---|---|---|---|---|
| **Scalability** | **Scenario SS1 (High Traffic):** During a flash sale event, traffic to the "View Product" feature increases by 500%, while the "Checkout" traffic remains normal. | **Inefficient:** To handle the load, we must scale the entire application (User, Order, Product modules together). This consumes excessive CPU/RAM for modules that are not under load. | **Highly Efficient:** We can spin up multiple instances of the Product Service (e.g., on ports 5002, 5003) independently. The Gateway balances the load. Resources are allocated precisely where needed. | **Microservices Wins** |

| | | | | |
|---|---|---|---|---|
| **Availability** | **Scenario AS1 (Service Failure):** The third-party Email Service (SMTP) becomes unresponsive, or the email sending code crashes due to a bug. | **System Failure:** In a synchronous monolith, the user waits for the email to be sent. If it fails or times out, the entire "Place Order" transaction may roll back or hang, preventing sales. | **Fault Tolerant:** Thanks to the Event-Driven Architecture (Lab 7), the Order is placed immediately. The email task is queued in RabbitMQ. Even if the Notification Worker crashes, the order is safe, and emails are sent later. | **Microservices Wins** |

### 3. Trade-off Analysis Statement

While the Microservices architecture demonstrates superior performance in the scenarios above, it introduces a significant **Trade-off**.

**Trade-off: Operational Complexity vs. Scalability**

- **The Cost:** The transition to Microservices has drastically increased the system's complexity. Instead of managing a single PHP application, we now manage:
    - Multiple independent processes (gateway.php, api.php, consumer.php).
    - Multiple network ports and inter-service communication rules.
    - Additional infrastructure components (RabbitMQ).

    - Debugging is more difficult as a single request spans multiple nodes (Distributed Tracing is required).
- **The Benefit:** In exchange for this complexity, we gain the ability to scale specific parts of the system and ensure that a failure in one module (e.g., Notifications) does not crash the entire business process (e.g., Ordering).

**Conclusion:** For a growing E-commerce platform like ShopSphere, this trade-off is justified. The cost of complexity is outweighed by the business value of high availability and scalability.