Exam Subject: Deployment of a Multi-Tenant ERP Infrastructure

Module: ERP, odoo 19

Objective: Deploy and manage two independent Odoo 19 Community instances for two different companies (Company A and Company B) using Docker Compose and Portainer.

1. Context

A consulting firm needs to host Odoo for two distinct clients on a single Linux server. To ensure isolation and ease of management, you are tasked with using containerization.

2. Technical Requirements

Students must fulfill the following architecture requirements:

- **Platform:** A Linux Server (Ubuntu/Debian) with Docker and Portainer installed.
- **Orchestration:** Each company must have its own `docker-compose.yml` stack.
- **Isolation:**
  - Each Odoo instance must have its own dedicated PostgreSQL database container.
  - Data must be persistent using Docker Volumes.
- **Networking:**
  - **Company A** must be accessible via port **8069**.
  - **Company B** must be accessible via port **9069**.

3. Tasks to Perform

Task 1: Environment Preparation

- Install Docker and Docker Compose.
- Deploy **Portainer CE** as the management GUI.

Task 2: Infrastructure as Code (Docker Compose)

Create two separate stacks in Portainer. Each stack must include:

1. An `odoo:19.0` image.
2. A `postgres:15` image.
3. Environment variables for database credentials.
4. Mapped volumes for `/var/lib/odoo` and `/var/lib/postgresql/data`.

Task 3: Configuration & Customization

- Login to **Company A** and **Company B**.
- Install the "Sales" module on Instance A and the "CRM" module on Instance B to prove they are independent.
- Configure a different company logo for each instance.

Task 4: Monitoring (Optional/Bonus)

- Show logs of the containers via the Portainer interface.
- Demonstrate how to restart a single instance without affecting the other.

4. Deliverables

Students must submit a technical report containing:

1. The two `docker-compose.yml` files used.
2. Screenshots of the **Portainer Stacks** dashboard showing all 4 containers running.
3. Screenshots of the two different Odoo instances accessed via different ports in a web browser.
4. A brief explanation of why we use volumes in this architecture.

5. Evaluation Criteria

Criteria	Weight
Connectivity: Both instances are reachable on the assigned ports.	30%
Data Persistence: If containers are deleted and recreated, data remains.	25%
Configuration: Correct use of environment variables and naming conventions.	20%
Security: Proper password management for PostgreSQL (no default 'admin').	15%
Report Quality: Clarity and technical accuracy.	10%

- **Implement a Reverse Proxy (Nginx/Traefik):** Instead of using ports 8069 and 9069, they must use domain names like `companyA.local` and `companyB.local` on port 80.

- **Resource Limitation:** Set CPU and RAM limits for each stack within the Docker Compose file to prevent one company from crashing the server.