**In-Class Assessment 1**

**(a) Three Primary Cloud Service Models**

1. **Infrastructure as a Service (IaaS)**IaaS provides virtualized computing resources over the internet, including servers, storage, networking, and virtual machines. Users manage operating systems, applications, and data while the cloud provider maintains the physical infrastructure.*Example in software development*: A development team using Amazon EC2 to provision virtual servers for hosting a custom application. They can configure the OS, install development tools, and scale server capacity based on project needs without managing physical hardware.
2. **Platform as a Service (PaaS)**PaaS offers a complete development and deployment environment in the cloud, including tools, frameworks, and runtime environments. It abstracts infrastructure management, allowing developers to focus on coding and application logic.*Example in software development*: A team using Google App Engine to build and deploy a Python web application. The platform handles server management, database scaling, and deployment pipelines, enabling developers to focus solely on writing code.
3. **Software as a Service (SaaS)**SaaS delivers fully functional software applications over the internet on a subscription basis. Users access the software via browsers or APIs without installing or maintaining it locally.*Example in software development*: A team using GitHub (a SaaS tool) for version control. They collaborate on code repositories, track changes, and manage project workflows through a web interface without hosting or updating the platform themselves.

**(b) Docker and Containerization**

**What is Docker?**Docker is an open-source platform that automates the deployment of applications in lightweight, portable containers. Containers package an application and its dependencies (libraries, runtime, configuration) into a single unit, ensuring consistent behavior across different environments.

**Scenario for Using Docker**A development team building a microservices-based e-commerce application with multiple components (e.g., a Node.js frontend, Python backend API, and PostgreSQL database). Each component has specific dependencies and configuration requirements.

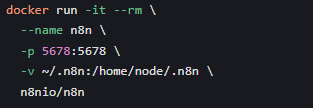
**How Containerization Contributes**

* **Consistency**: Containers ensure the application runs identically on developers' local machines, testing environments, and production servers, eliminating "it works on my machine" issues.
* **Isolation**: Each microservice runs in its own container, preventing conflicts between dependencies (e.g., different Node.js versions for frontend and backend).
* **Scalability**: Containers can be quickly replicated to scale individual services (e.g., increasing API containers during high traffic).
* **Simplified Deployment**: Containers are portable and can be deployed to any cloud provider or on-premises infrastructure that supports Docker, streamlining the CI/CD pipeline.

**(c) Deploying n8n with Docker**

**Step-by-Step Deployment**To deploy n8n (an open-source workflow automation tool) using Docker:

1. **Run the Docker Command**:Open a terminal and execute:



1. **Access n8n**:Navigate to http://127.0.0.1:5678 in a browser. You’ll see the n8n workflow editor interface.

**Command Explanation**

* docker run: Creates and starts a new container from an image.
* -it: Combines -i (interactive mode, keeps stdin open) and -t (allocates a pseudo-TTY), allowing interaction with the container.
* --rm: Automatically removes the container when it stops (cleans up temporary resources).
* --name n8n: Assigns the name "n8n" to the container for easy reference.
* -p 5678:5678: Maps port 5678 on the host machine to port 5678 in the container (n8n’s default port).
* -v ~/.n8n:/home/node/.n8n: Mounts a volume from the host’s ~/.n8n directory to the container’s /home/node/.n8n directory, persisting workflow data even when the container restarts.
* n8nio/n8n: Specifies the Docker image to use (official n8n image from Docker Hub).

**Screenshot Note**After running the command, the n8n interface at http://127.0.0.1:5678 will display a visual workflow editor with drag-and-drop nodes for building automations. A screenshot would show this interface with options to create new workflows or explore templates.