**In-class assessment1**

(a)**Infrasturcture as a service:**

IaaS provides virtualized computing resources over the internet. Users get access to fundamental computing resources like virtual machines, storage, and networks without managing the physical infrastructure.

**Example:**A development team uses AWS EC2 to create multiple virtual machines for different environments (development, testing, staging). They can quickly scale resources during intensive testing phases and only pay for what they use.

**Platform as a service:**

PaaS provides a platform allowing customers to develop, run, and manage applications without dealing with the underlying infrastructure. It includes development tools, database management, and business intelligence services.

**Example:**A team uses Heroku or Google App Engine to deploy their web application. They can focus on writing code while the platform handles scaling, load balancing, and database management automatically.

**Software as a service:**

SaaS delivers software applications over the internet on a subscription basis. Users access the software through web browsers without installation or maintenance.

**Example:**Development teams use GitHub for version control, Slack for communication, and Jira for project management. These tools are accessed as services without any infrastructure management.

(b)Docker is a powerful containerization platform that enables developers to package applications and their dependencies into lightweight, portable containers ensuring consistent execution across different computing environments. In modern software development, a practical scenario for leveraging Docker would be when building a microservices-based e-commerce platform consisting of multiple interdependent services like user authentication, product catalog, shopping cart, and payment processing. Containerization fundamentally transforms the development and deployment process in this scenario by providing isolated environments for each service, eliminating the common "it works on my machine" problem during development, streamlining continuous integration and delivery pipelines through standardized container images, and enabling seamless scaling of individual services based on demand while maintaining environment consistency from local development through testing to production deployment, thereby significantly reducing deployment failures and improving overall development velocity and operational reliability.

(c)Docker command explaination:

docker run -d --name n8n -p 127.0.0.1:5678:5678 -v n8n\_data:/home/node/.n8n n8nio/n8n

--name n8n assigns a recognizable name to the container

-p 127.0.0.1:5678:5678 maps the container's port 5678 to your local machine's same port but only accessible from localhost for security

-v n8n\_data:/home/node/.n8n creates a persistent volume to preserve your n8n data and workflows between container restarts

