A) Describe the three primary cloud service models in cloud computing infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Provide specific examples of how each model can be applied in the context of software development.

1. **Infrastructure as a Service (IaaS)**

   Description: IaaS provides virtualized computing resources over the internet, such as servers, storage, and networking. It gives developers maximum control over infrastructure without requiring them to manage physical hardware.

   Example: A development team can use Amazon EC2 (AWS) to set up development and test environments. They can configure the OS, install dependencies and deploy their code as needed just like on a physical server, but with the flexibility to scale resources quickly.

2. **Platform as a Service (PaaS)**

   Definition: PaaS provides a ready-to-use development and deployment environment with tools, libraries, and frameworks managed by the cloud provider. Developers can focus on coding without worrying about underlying infrastructure or runtime management.

   Example: when developing a microservices-based application, developers can directly push their code and the platform automatically handles scaling, load balancing and monitoring. This accelerates development cycles since teams can focus on coding and testing rather than system administration.

3. **Software as a Service (SaaS)**

   Definition: SaaS delivers ready-to-use applications over the internet. Users access the software through a browser or API without managing infrastructure, platforms, or even the application itself.

   Example: GitHub is a kind of SaaS tools, which support coding, issue tracking and collaboration without requiring the team to manage servers or updates. For example, GitHub provides a complete version control and CI/CD system accessible directly through a web interface.

B) What is Docker? Describe a scenario where you would use containerization technologies such as Docker in software development. How does containerization contribute to the development and deployment process of software in this scenario?

Definition: Docker is a **containerization technology** that packages an application together with everything that needs to run, like libraries, runtime, and configuration, into a lightweight, portable container. These containers are isolated from each other but share the host machine's kernel, which makes them fast to start, resource-efficient and consistent across different environments.

Example: When I want to use a microservices-based e-commerce platform that has been build by others and includes a user authentication service (Python Flask), an order processing service (Java Spring Boot) and a product recommendation service (running a deep learning model with PyTorch). In traditional deployment, each service would require specific dependencies installed directly on my machine. This often causes compatibility issues. For example, I use different Python version from the original one, leading to broken dependencies. Or my server lacks the proper CUDA drivers, preventing the recommendation service from running. However, with Docker, each service can be packaged into its own container image. I only need to run ***docker run*** to start these containers in any environment, without manually configuring dependencies. Using tools like **docker-compose**, the containers can be orchestrated so the entire system runs consistently in development, testing, and production.

How: containerization contributes primarily through consistency and portability. Developers can verify the whole system locally, testers can reproduce the same setup, and production can run the exact same container images. This eliminates environment-related bugs and makes deployment and scaling much faster. For a microservices architecture, Docker ensures each service can be developed, tested, and updated independently, greatly improving the speed and reliability of software development and delivery.

C) Deploy n8n (n8n.io) with Docker and capture a screenshot of http://127.0.0.1:5678. Please explain the docker command in detail.

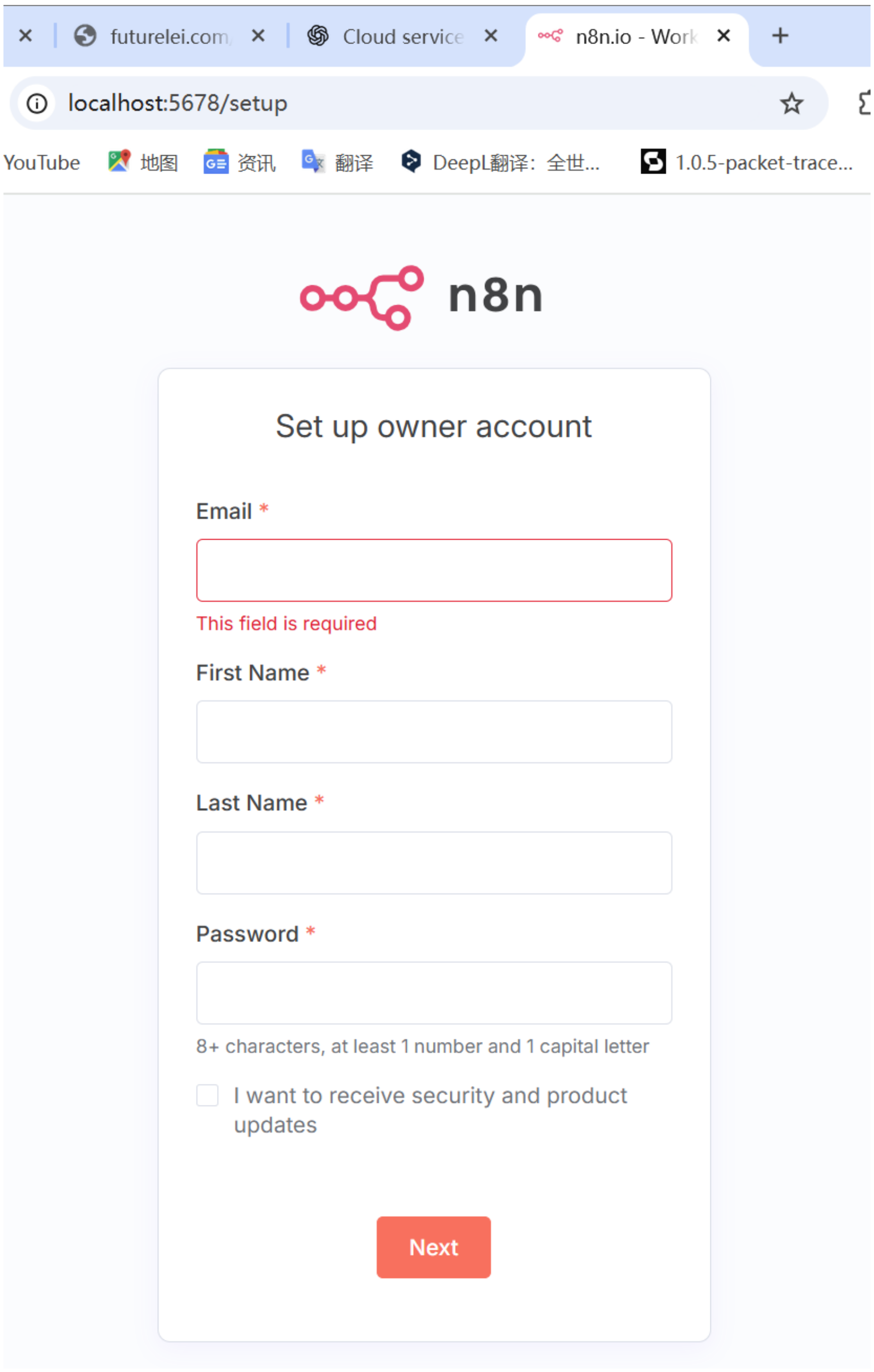
My environment is Docker Desktop and Windows 11. The following is my commands.

a. First, I use ***mkdir E:\ n8n-data*** to create a directory to store the configuration of n8n and work stream data to ensure that I can still use it next time. Because if the data in the container is not mounted to the local disk, all configurations and workflows will be lost once the container is deleted.

b. Then I use this command ***docker run -it --rm -p 5678:5678 -v E:\n8n-data:/home/node/.n8n n8nio/n8n*** to execute the container.

   - ***docker run***: execute a new container instance.
   - ***-it***: in order to see real-time logs in PowerShell and interact if needed.
     -i: keeps STDIN open
     -t: Allocate a pseudo-TTY to the container.
   - ***--rm***: automatically removes the container when it stops. This is to prevent unused containers from piling up, which is handy for development/testing.
   - ***-p 5678:5678***: a map from host port 5678 to container port 5678. So, in this way, http://127.0.0.1:5678 can be open in our own browsers and we can reach the n8n UI running inside the container.
   - ***-v E:\n8n-data:/home/node/.n8n***: mounts a volume between host folder and container folder. Because n8n saves workflows/config inside /home/node/.n8n. By mapping it to E:\n8n-data, we can keep our data safe on Windows even if the container is removed.
   - ***n8nio/n8n***: use the official n8n Docker image that is secure and reliable.

c. Finally, the screenshot of http://127.0.0.1:5678 is shown as follows.
   Here I use localhost to replace 127.0.0.1. It is the same result.



localhost:5678/setup

n8n

Set up owner account

Email *

This field is required

First Name *

Last Name *

Password *

8+ characters, at least 1 number and 1 capital letter

I want to receive security and product updates