(a)

IaaS:

- Description: IaaS provides the fundamental building blocks of computing over the internet. It offers on-demand access to virtualized computing resources: servers (VMs), storage, and networking. With IaaS, you avoid the capital expense and complexity of buying and managing physical servers. You are responsible for managing everything above the infrastructure layer: the operating system, runtime, middleware, and your application.

- Example: Instead of purchasing physical hardware for each developer, a team can use IaaS to create and tear down identical, isolated development and testing environments on demand. This ensures consistency and saves costs.

PaaS:

- Description: PaaS provides a platform allowing customers to develop, run, and manage applications without the complexity of building and maintaining the underlying infrastructure. It's a complete development and deployment environment in the cloud. The provider manages the servers, storage, networking, operating system, middleware, and runtime. You focus solely on developing your application and managing its data.

- Example: Services like AWS Elastic Beanstalk or GitLab CI/CD are PaaS solutions that automate the process of testing, building, and deploying code changes, freeing developers from managing the underlying build servers.

SaaS:

- Description: SaaS delivers a complete, fully functional software application over the internet, on a subscription basis. The cloud provider hosts and manages the entire application, including the infrastructure, platform, and software itself. Users access the application directly through a web browser or a thin client. There is no software to install or maintain.

- Example: Cloud-based IDEs like GitHub Codespaces or AWS Cloud9

allow developers to code directly from a browser with a pre-configured environment, eliminating "it works on my machine" problems.

(b)

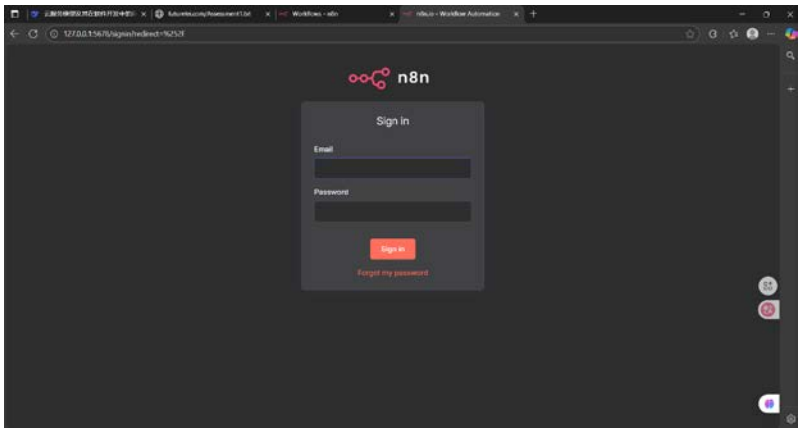
- Docker is a platform that uses containerization technology to make it easier to create, deploy, and run applications. It allows you to package an application with all of its dependencies (libraries, system tools, code, runtime) into a standardized, isolated unit called a container.

- Scenario: Developing and Deploying a Microservices-Based Web Application. Each developer on the team clones the project's code repositories (e.g., from GitHub). Instead of spending hours manually installing Node.js v18, Python 3.9, PostgreSQL, and Redis on their individual machines, they simply run docker-compose up. Within seconds, every developer has the entire application stack running identically on their local machine, regardless of whether they are using macOS, Windows, or Linux.

- Docker provides a standardized, portable, and efficient way to manage the application lifecycle. It decouples the application from the underlying infrastructure, leading to faster development cycles, more reliable testing, and simpler, more robust deployments.

(c)

- screenshot



- detail explanation

1. docker pull n8nio/n8n Pull the n8n Docker image.

2. docker run -d Run the container.

3. docker ps Check whether the container is running.

4. docker logs n8n Check the log of the container