# In-Class Assessment 1

## (a) Three Primary Cloud Service Models

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

* **Description**: Provides virtualized computing resources over the internet, including servers, storage, networking, and virtualization. Users manage the operating systems, applications, and development frameworks.
* **Software Development Example**: A development team uses AWS EC2 instances to create customized development environments. They can provision virtual machines with specific configurations, install required development tools (like IDEs, databases, and testing frameworks), and scale resources up or down based on project needs.

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

* **Description**: Offers a platform allowing customers to develop, run, and manage applications without dealing with the underlying infrastructure. Includes operating systems, middleware, databases, and development tools.
* **Software Development Example**: Developers use Heroku or Google App Engine to deploy web applications. They simply push their code to the platform, which automatically handles scaling, load balancing, and database management, allowing developers to focus solely on writing application code.

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

* **Description**: Delivers software applications over the internet on a subscription basis. Users access the software through web browsers without worrying about installation, maintenance, or underlying infrastructure.
* **Software Development Example**: Development teams use GitHub for version control and collaboration, Slack for team communication, and Jira for project management. These tools are accessed as services without any local installation or maintenance.

## (b) Docker and Containerization Scenario

### 1.What is Docker?

Docker is an open-source platform that enables developers to package applications and their dependencies into lightweight, portable containers. These containers can run consistently across different environments, from development to production.

**Scenario: Microservices Application Developmen**

In a microservices architecture for an e-commerce application, different services (user authentication, product catalog, shopping cart, payment processing) are developed as independent microservices.

### 2.How Containerization Contributes:

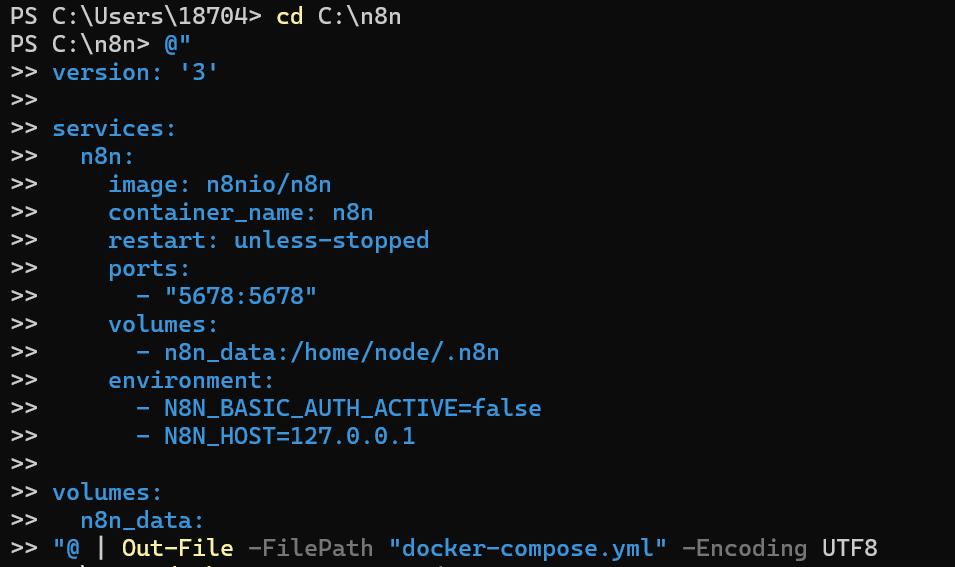
1. **Environment Consistency**: Each microservice runs in its own container with specific dependencies, ensuring identical behavior across development, testing, and production environments.
2. **Isolated Development**: Developers can work on individual services without affecting others. For example, the payment service team can update their container without impacting the user authentication service.
3. **Rapid Deployment**: Using Docker Compose, the entire application stack can be started with a single command, significantly reducing setup time from days to minutes.
4. **Continuous Integration/Deployment**: Containers can be easily integrated into CI/CD pipelines, enabling automated testing and deployment.
5. **Scalability**: Individual services can be scaled independently by running multiple container instances based on demand.

## (c) n8n Deployment with Docker

### 1. Install Docker

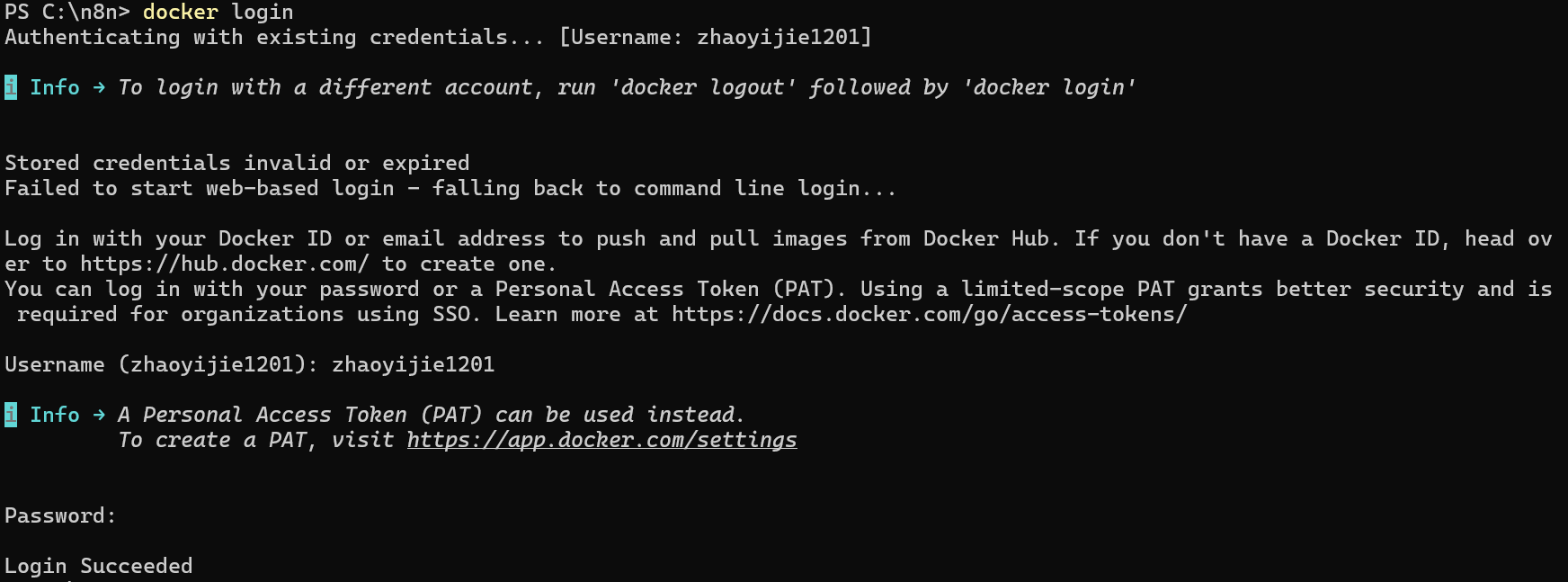
### 2. Deploy using Docker Compose

**1) Create the docker-compose.yml file**



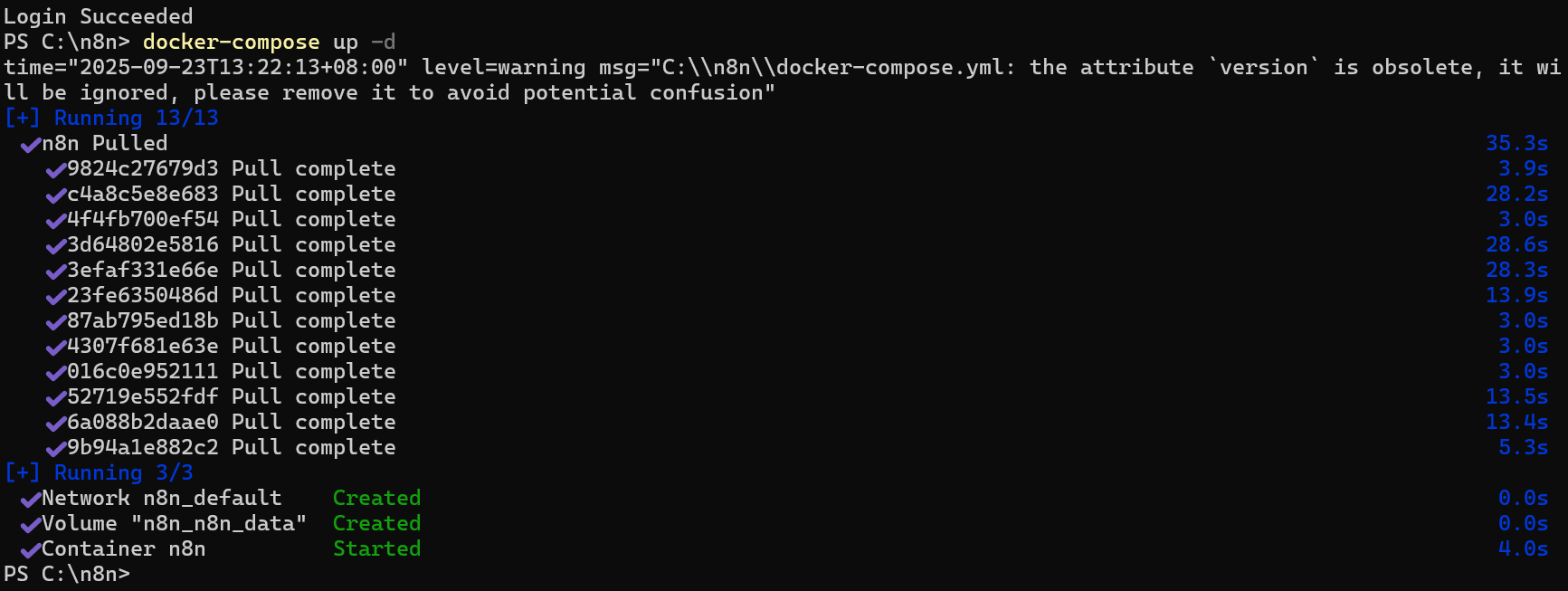
**2) Start n8n using Docker Compose**

* Log in to Docker



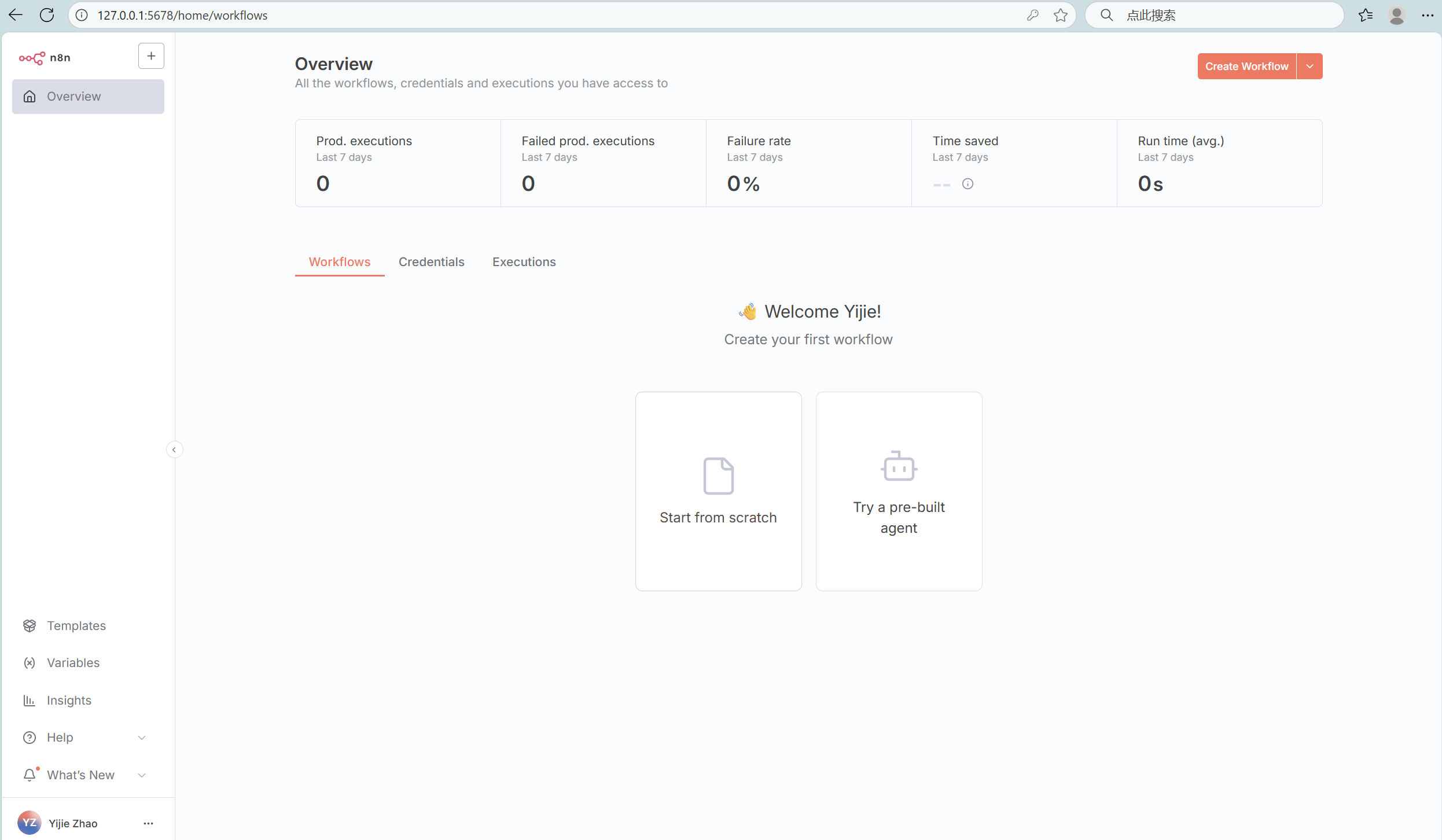
* Start n8n using Docker Compose

**docker-compose up -d**



### 3. Open the browser and visit: <http://127.0.0.1:5678>

* **Visit** [**http://127.0.0.1:5678**](http://127.0.0.1:5678)



* Then, we can view the created n8n in the local Docker software.

