(a) Three Primary Cloud Service Models

1. Infrastructure as a Service (IaaS)

IaaS provides virtualized computing resources such as servers, storage, networking over the internet. Users manage operating systems, middleware, and applications, while the provider manages the physical infrastructure.

We can rents virtual machines on Amazon EC2 or Microsoft Azure Virtual Machines to build and test a custom web application. And we cam install our own operating system, runtime, and development tools.

2. Platform as a Service (PaaS)

PaaS provides a ready-to-use platform such asruntime environment, database, middleware that developers can use to deploy applications without managing infrastructure.

We uses Heroku or Google App Engine to deploy a Node.js web application. The platform automatically handles scaling, patching, and server management. Developers focus only on writing code.

3. Software as a Service (SaaS)

SaaS provides fully functional software applications delivered over the internet on a subscription basis. Users simply use the application without worrying about infrastructure or platform management.

A development team uses GitHub, Jira, or Slack to collaborate, manage projects, and track issues during the software development lifecycle.

(b) Docker and Containerization

What is Docker?

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

Scenario:

Imagine a team is building a Python Flask web application with a PostgreSQL database. On different developer machines, differences in Python versions or OS libraries may cause errors.

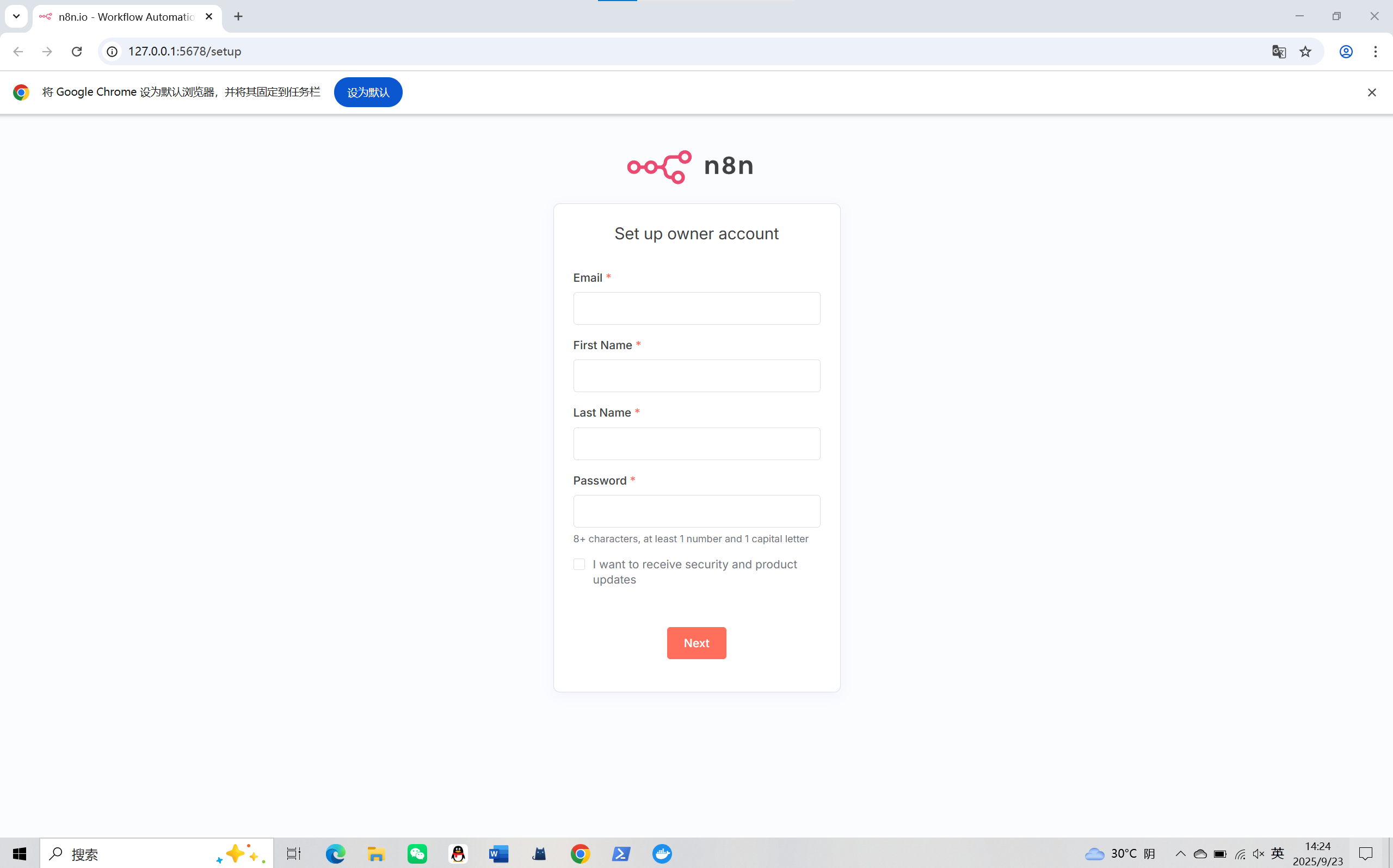
When this team using Docker:

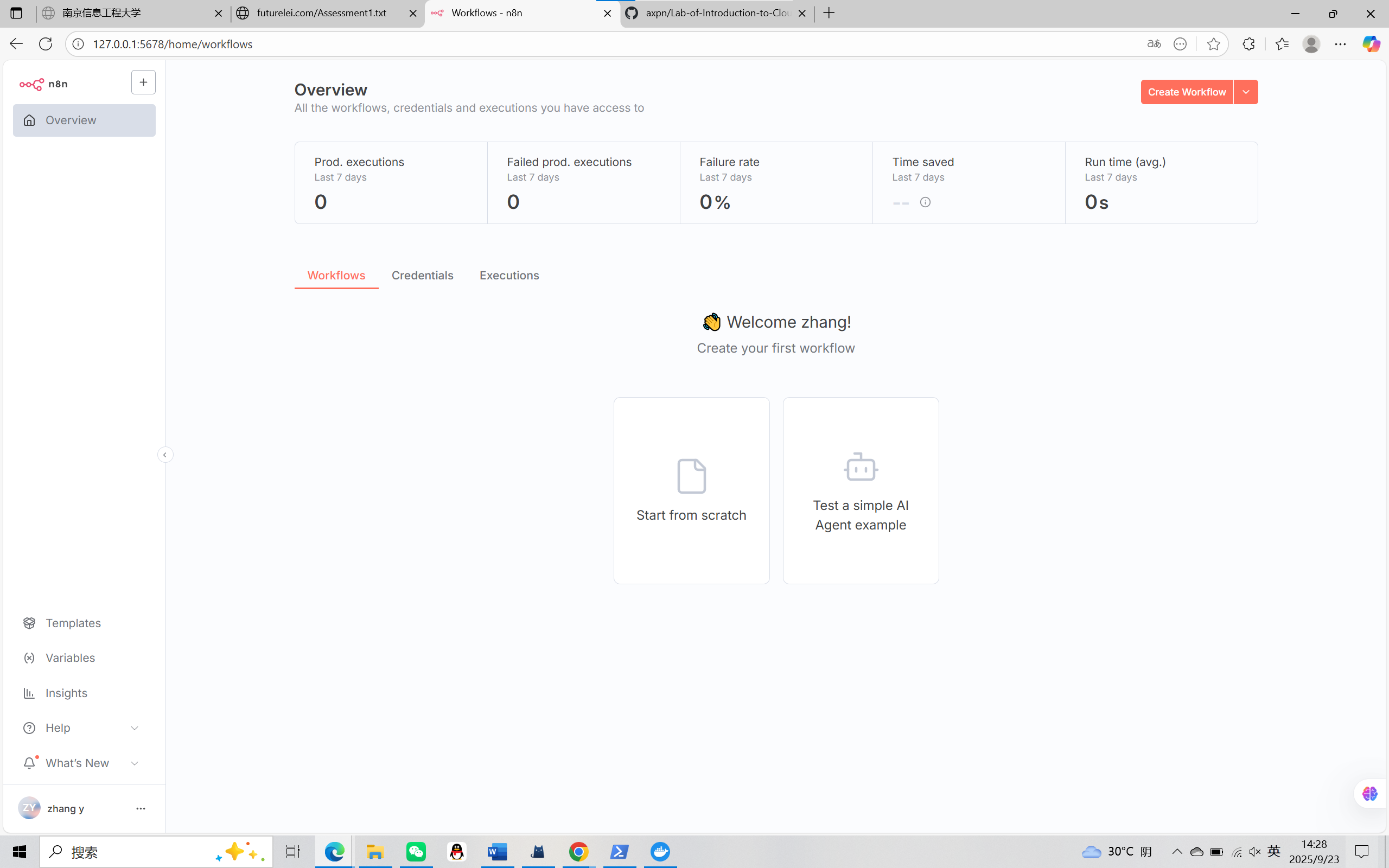
The Flask app runs inside a container with its dependencies defined in a Dockerfile.

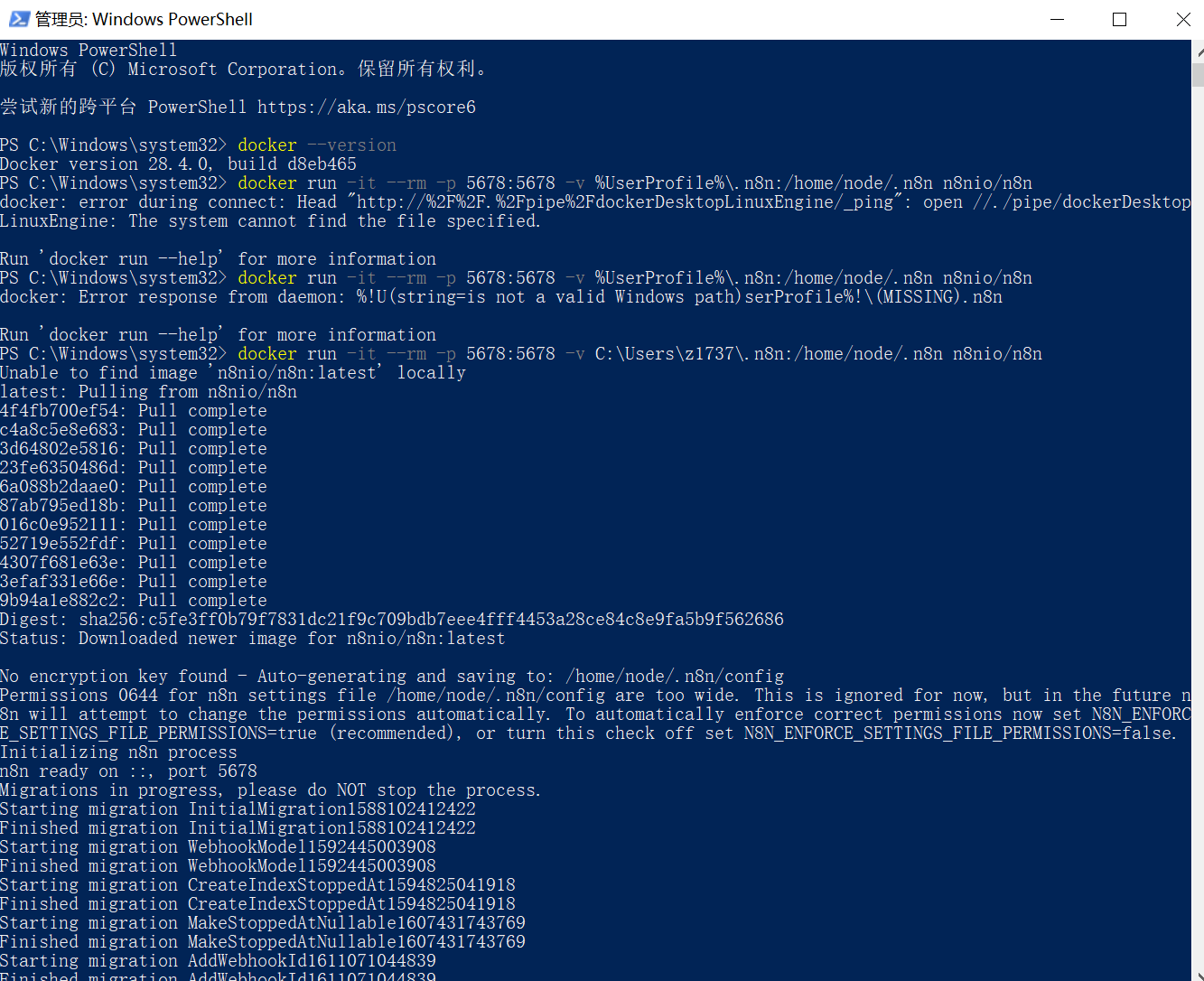
The database runs in a separate PostgreSQL container.

Both are orchestrated using Docker Compose.

(c)







 docker run → Creates and runs a new container.

 -it → Runs the container in interactive mode with a terminal attached.

 --rm → Automatically removes the container when it exits (keeps system clean).

 -p 5678:5678 → Maps port 5678 on the host (your machine) to port 5678 inside the container. This makes n8n accessible at [**http://127.0.0.1:5678**](http://127.0.0.1:5678).

 -v C:\Users\z1737\.n8n:/home/node/.n8n → Mounts a local folder (~/.n8n) into the container for persistent data (workflows, credentials). Without this, data would be lost after restarting.

 n8nio/n8n → The Docker image for n8n (fetched from Docker Hub if not already on the system).