1. Cloud Service Models in Software Development

Infrastructure as a Service (IaaS)

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

Software Development Application:

Development Environments: Spin up temporary VMs for testing different configurations

CI/CD Pipelines: Use cloud compute resources for building and testing applications

Example: AWS EC2 instances for hosting development servers or running automated tests

Platform as a Service (PaaS)

Description: PaaS provides a platform allowing developers to build, deploy, and manage applications without dealing with the underlying infrastructure.

Software Development Application:

Application Deployment: Deploy web applications without managing servers

Database Management: Use managed database services

Example: Heroku for deploying web applications or Google App Engine for scalable app hosting

Software as a Service (SaaS)

Description: SaaS delivers software applications over the internet on a subscription basis, with the provider managing everything from infrastructure to application updates.

Software Development Application:

Development Tools: Use cloud-based IDEs like GitHub Codespaces

Project Management: Tools like Jira for agile development tracking

Example: GitHub for version control and collaboration or Slack for team communication

(b) Docker and Containerization

What is Docker?

Docker is a containerization platform that packages applications and their dependencies into isolated, lightweight containers that can run consistently across different environments.

Scenario: Microservices Development

In a microservices architecture where an e-commerce application is split into separate services (user service, product catalog, payment processing, order management).

How Containerization Contributes:

Consistency: Each service runs in identical environments from development to production

Isolation: Services can be developed, updated, and scaled independently

Rapid Deployment: New versions can be deployed without affecting other services

Resource Efficiency: Multiple containers share the host OS kernel

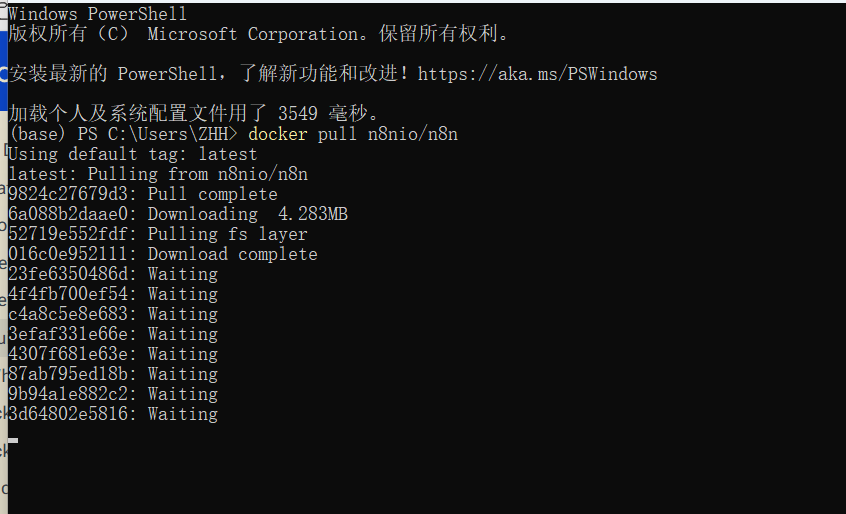
Dependency Management: Each service bundles its specific dependencies

(c) n8n Deployment with Docker

1. Check if any n8n containers exist:

bash

docker ps -a



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

docker run: Creates and starts a new container

-d: Runs the container in detached mode (background)

--name n8n: Assigns the name "n8n" to the container

-p 5678:5678: Maps host port 5678 to container port 5678

First 5678: Host machine port

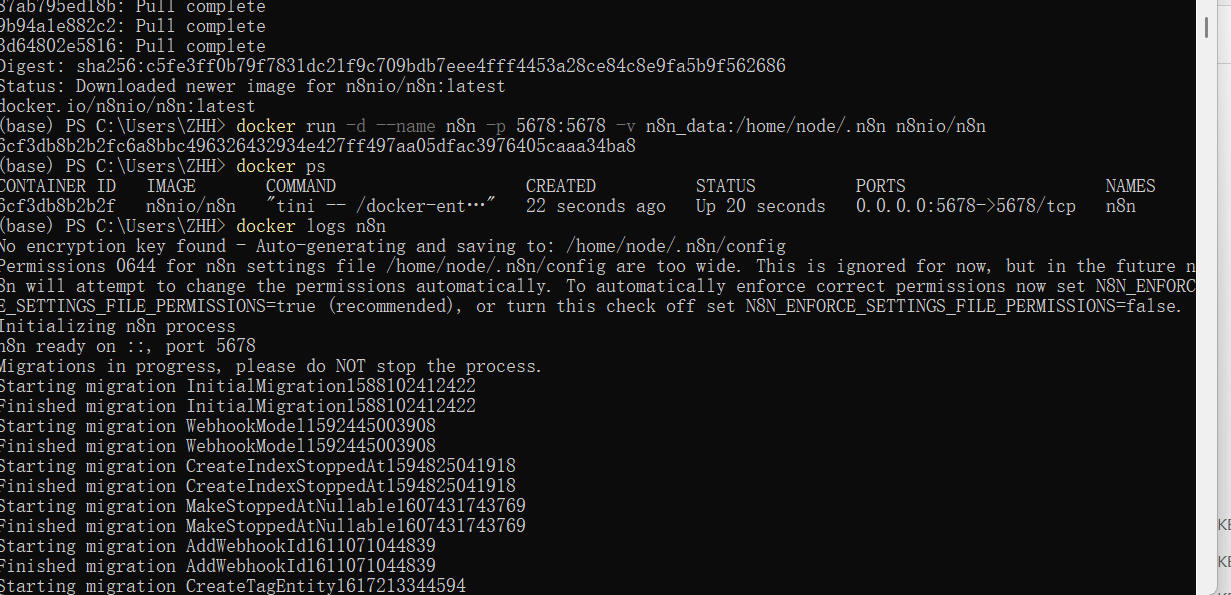
Second 5678: Container internal port

-v n8n\_data:/home/node/.n8n: Creates a volume for data persistence

n8n\_data: Named volume on host system

/home/node/.n8n: Directory inside container where n8n stores data

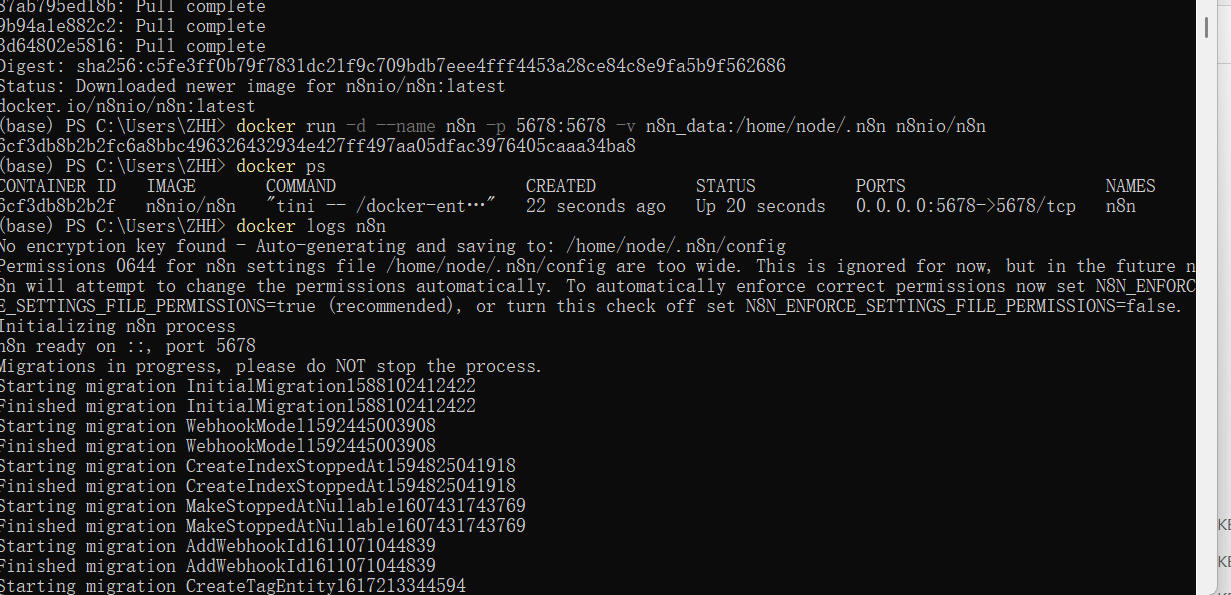
n8nio/n8n: Official n8n Docker image from Docker Hub



4. If port 5678 is already in use:

# Check what's using port 5678netstat -tulpn | grep 5678

# Use a different portdocker run -d --name n8n -p 5679:5678 -v n8n\_data:/home/node/.n8n n8nio/n8n# Then access via <http://127.0.0.1:5679>



5. n8n accessible at http://127.0.0.1:5678

