(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.

\*\*IaaS (Infrastructure as a Service)\*\*

\* \*\*What it is:\*\* Provides on-demand access to fundamental computing resources like compute, storage, and networking. You are responsible for managing the operating system, runtime, middleware, applications, and data.

\* \*\*Typical Vendors/Products:\*\* AWS EC2/ELB/EBS, Azure Virtual Machines, Google Compute Engine, Alibaba Cloud ECS, etc.

\* \*\*Development Use Case Examples:\*\*

\* Setting up an auto-scaling cluster of CI build machines (e.g., using AWS EC2 Auto Scaling Groups + S3 for build artifact caching).

\* Deploying a service on a custom machine image (e.g., an AMI with specific drivers or kernel parameters) for performance tuning and running stress tests/benchmarks.

\* Running training/inference nodes that require specific GPUs (selecting a GPU instance and installing the necessary drivers and CUDA toolkit yourself).

\*\*PaaS (Platform as a Service)\*\*

\* \*\*What it is:\*\* Provides a managed platform/runtime environment. You focus primarily on your code and its dependencies, while the platform handles scaling, health checks, logging, deployment strategies, etc.

\* \*\*Typical Vendors/Products:\*\* Heroku, Azure App Service, Google App Engine, AWS Elastic Beanstalk, Google Cloud Run, etc.

\* \*\*Development Use Case Examples:\*\*

\* Quickly hosting a Node.js/Go backend by simply pushing code via Git or deploying a Docker image; the platform manages horizontal scaling and zero-downtime deployments.

\* Creating temporary preview environments for internal tools (e.g., automatically generating a accessible preview URL for each Pull Request).

\* Using built-in managed services like databases, message queues, or logging to reduce operational overhead.

\*\*SaaS (Software as a Service)\*\*

\* \*\*What it is:\*\* Ready-to-use software applications delivered over the web (browser) or APIs, requiring little to no installation or maintenance.

\* \*\*Typical Vendors/Products:\*\* GitHub/GitLab Cloud, Jira, Notion, Figma, Salesforce, Slack, etc.

\* \*\*Development Use Case Examples:\*\*

\* Using GitHub/GitLab for code hosting, Issue/PR management, and CI/CD pipelines.

\* Integrating error monitoring and APM tools like Sentry or Datadog via an SDK to immediately track production errors and performance metrics.

\* Using Auth0 for unified login and identity management, where developers only need to integrate OAuth/OIDC protocols.

\*\*Summary:\*\* IaaS provides \*\*resources and freedom\*\*, PaaS provides a \*\*managed runtime and operational capabilities\*\*, and SaaS provides \*\*ready-made application functionality\*\*. In development, these are often combined: code collaboration happens on SaaS (GitHub), applications run on PaaS (App Service/Cloud Run), while heavy workloads or specific needs are handled by IaaS (self-managed VMs/GPUs).

(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?

\*\*Docker\*\* is a containerization technology. It packages an application along with its dependencies (runtime, libraries, system tools) into a standardized unit called an image. This image can then be run as a container. Containers share the host system's kernel, leading to fast startup times, smaller footprint, and consistent environments.

\*\*Scenario Example:\*\*

Imagine a microservices application consisting of a frontend (React/Vite), a backend (Node.js + Express), a queue (Redis), and a database (Postgres). The common problem of "it works on my machine" often occurs across development, testing, staging, and production environments due to dependency inconsistencies.

\*\*How Containerization Improves the Workflow:\*\*

\* \*\*Consistent Environments:\*\* Each service has a `Dockerfile` defining its OS dependencies, language runtime, and build steps. Anyone can run `docker compose up` to start the entire stack locally with identical conditions.

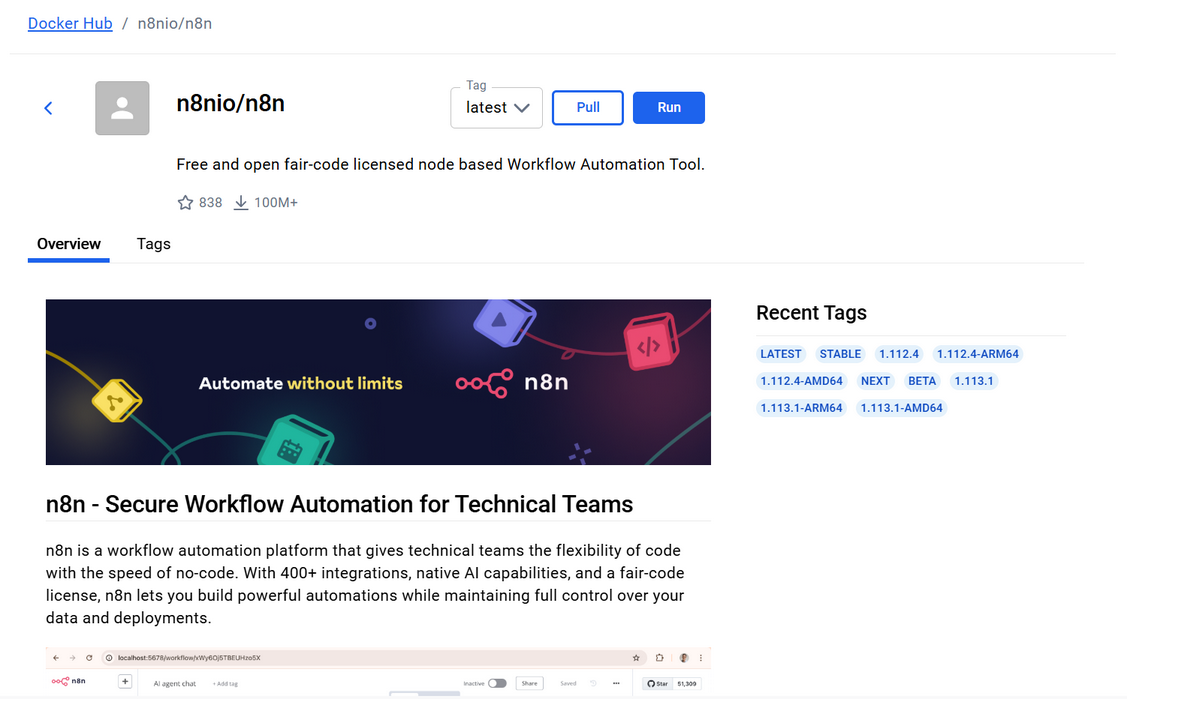
\* \*\*Faster CI/CD Pipelines:\*\* The CI system builds and tests using the same images, eliminating failures caused by environment differences. The build artifact is an immutable image, and deployment simply involves pulling and starting this image.

\* \*\*Reusable Local Orchestration:\*\* A `docker-compose.yml` file can start the entire suite (Postgres, Redis, backend, frontend) with one command, including built-in network configuration, simplifying local integration testing.

\* \*\*Isolation and Security:\*\* Different services run in isolated containers, preventing dependency or port conflicts. This isolation facilitates rolling updates and canary deployments.

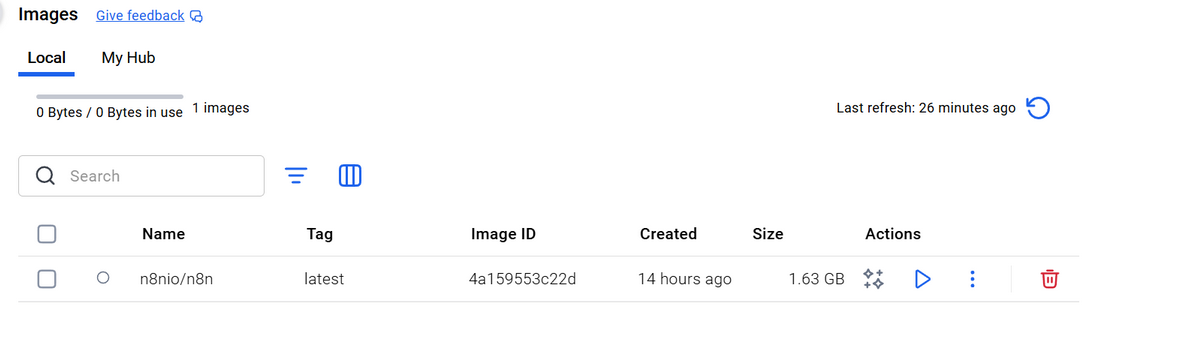
\* \*\*Portability:\*\* The same image runs consistently everywhere – from a developer's laptop to a test server to a Kubernetes cluster in the cloud – streamlining the path from local development to production.

(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.



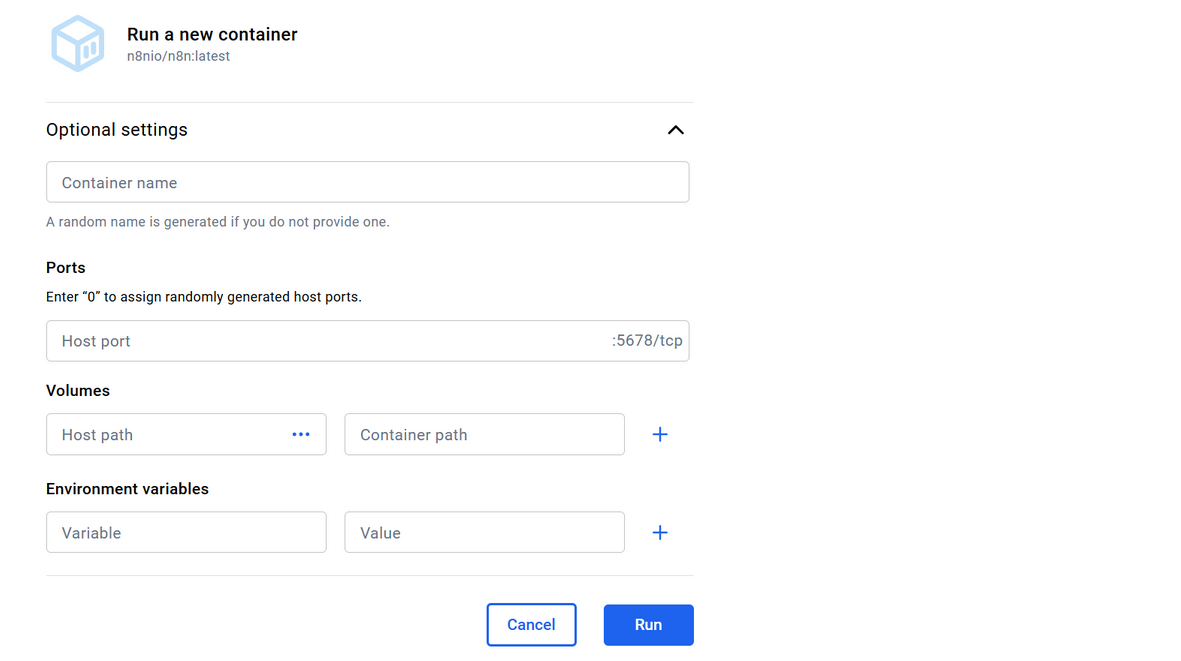
<img width="759" height="470" alt="Image" src="https://github.com/user-attachments/assets/d7360c7e-ff0a-4ec1-b038-b1d17df11598" />

then click [image], run the n8n:



<img width="743" height="264" alt="Image" src="https://github.com/user-attachments/assets/fd1d0d90-0f22-4526-91b6-47ad33bf7127" />

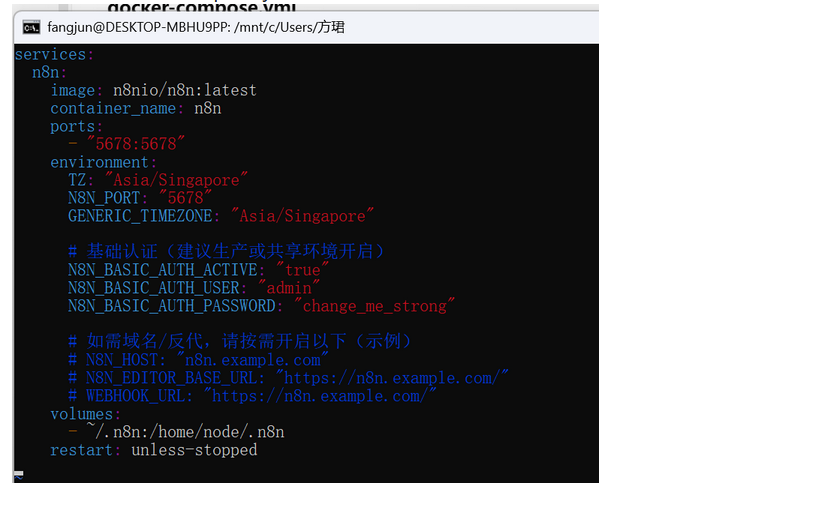
do the optional settings:



<img width="471" height="466" alt="Image" src="https://github.com/user-attachments/assets/cdf23a43-1d96-4758-867c-7652ba4fcfba" />

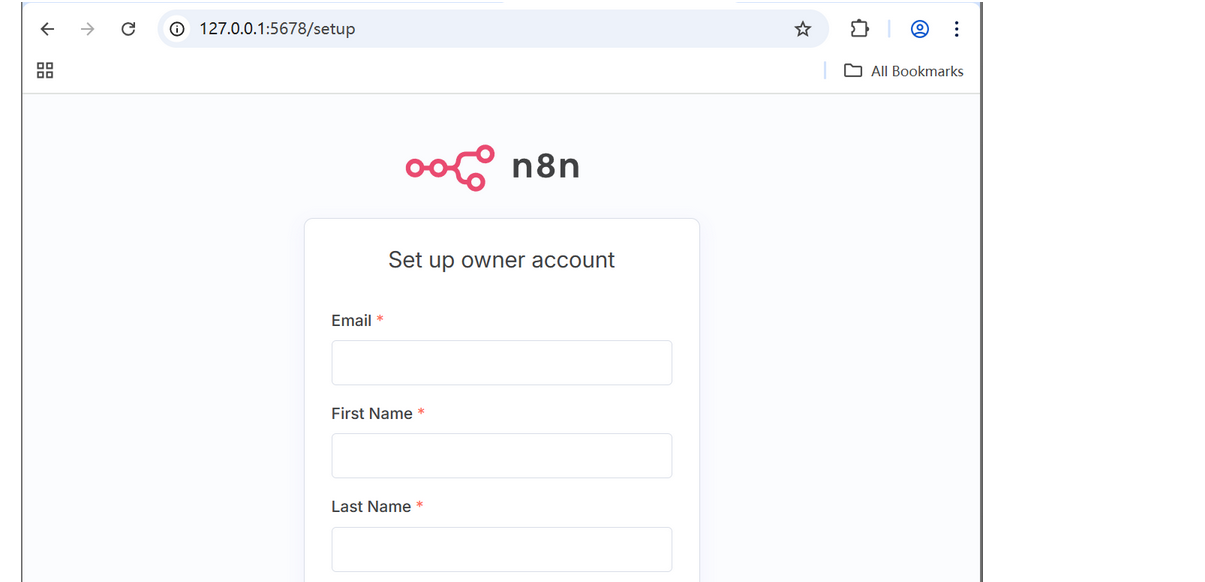
and then deploy n8n (n8n.io) with Docker sucessfully.

creat the Docker-compose.yml file



<img width="391" height="320" alt="Image" src="https://github.com/user-attachments/assets/3d956d8c-f93f-4763-b971-4063cc234858" />

then we can visit n8n from the port 127.0.0.1:5678



<img width="641" height="642" alt="Image" src="https://github.com/user-attachments/assets/56ab250e-c925-4270-97c5-76c342e02c47" />