Run with Docker (Development Mode)

Prerequisites

These prerequisites are already installed if you are using innowing's computer.

If you are using your own server, please follow the steps below to install the prerequisites:

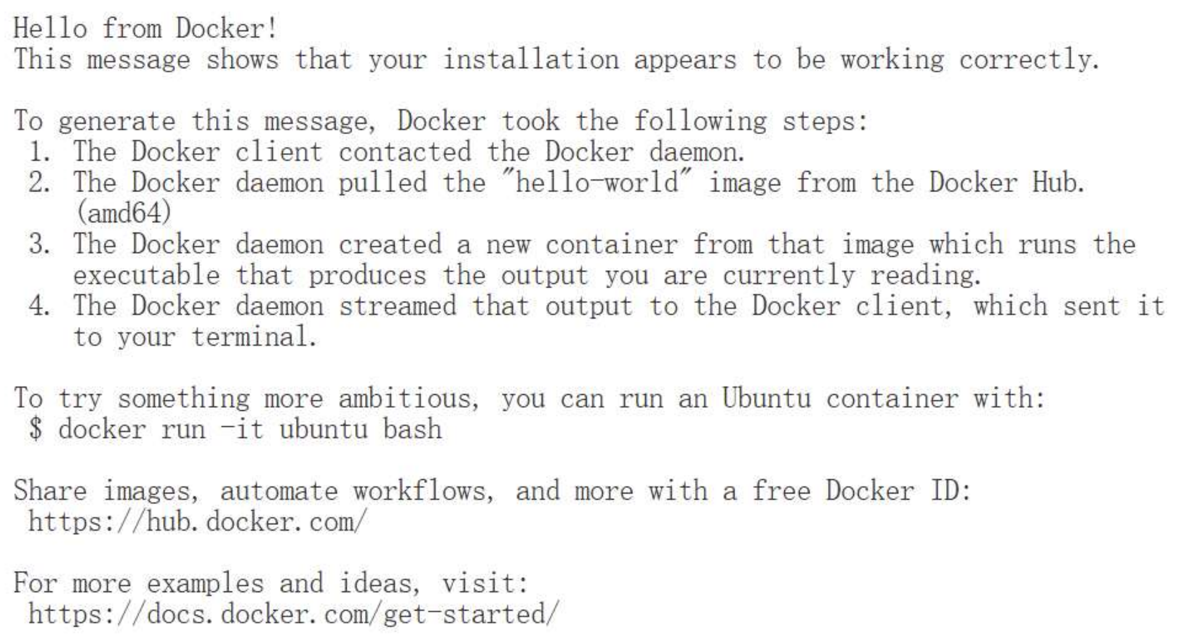
Docker and Docker Compose installed

- Install on linux (https://docs.docker.com/engine/install/)
- Install on Windows (https://docs.docker.com/desktop/install/windows-install/)
- Install on Mac (https://docs.docker.com/desktop/install/mac-install/)

To check your docker installation, run:

```
docker run hello-world
```

If you see the following message, then your installation is working correctly:



```
Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

NVIDIA GPU Support Required (if you want to use cpu only, skip this):

- NVIDIA GPU with CUDA support
- For Windows (https://docs.nvidia.com/cuda/cuda-installation-guide-microsoft-windows/)
- For Linux (https://docs.nvidia.com/cuda/cuda-installation-guide-linux/)
- NVIDIA Container Toolkit installed
- For Windows (https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/install-guide.html#docker) you need to install the NVIDIA Container Toolkit in wsl.
- For Linux (https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/install-guide.html#docker)

Quick Start

Make sure the docker daemon is running

In innowing's environment, you can double-click the Docker Desktop icon to start the daemon.

If you are using your own Linux server, run:

```
sudo systemctl start docker
```

Prepare the application code

```
# Unzip the application files
unzip mtr-chatbot.zip

# Navigate to the application directory
cd mtr-chatbot
```

Run with gpu

We use a smaller model compared to the one used on production, which is more suitable for single GPU but the performance will be lower than the larger model.

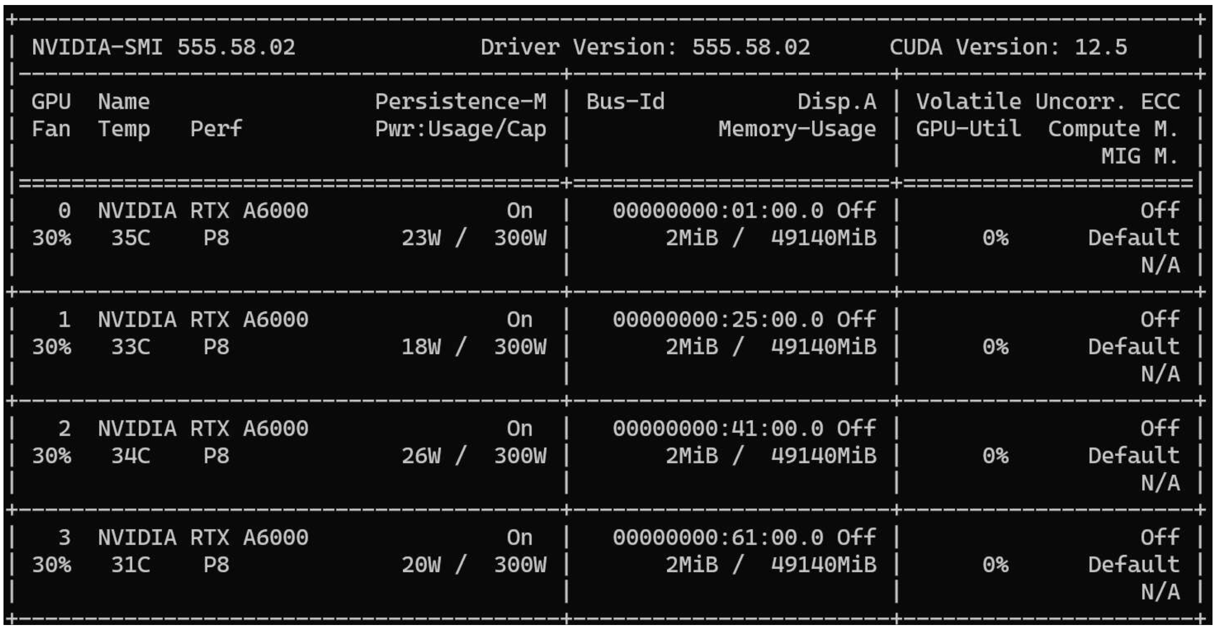
```
# Run in detached mode
docker compose up -d --build

# Pull necessary Ollama models (example)
docker exec -it mtr-ollama ollama pull deepseek-r1:8b
```

Now you can access the application at http://localhost:8501

Verify GPU Usage in Ollama

```
# Check if Ollama is using nvidia GPU
docker exec mtr-ollama nvidia-smi
```

If you see the GPU usage information, it means Ollama is successfully utilizing the GPU.



```
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 555.58.02              Driver Version: 555.58.02      CUDA Version: 12.5     |
|-----------------------------------------+------------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |               MIG M. |
|=========================================+========================+======================|
|   0  NVIDIA RTX A6000               On  |   00000000:01:00.0 Off |                  Off |
| 30%   35C    P8             23W /  300W |       2MiB /  49140MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
|   1  NVIDIA RTX A6000               On  |   00000000:25:00.0 Off |                  Off |
| 30%   33C    P8             18W /  300W |       2MiB /  49140MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
|   2  NVIDIA RTX A6000               On  |   00000000:41:00.0 Off |                  Off |
| 30%   34C    P8             26W /  300W |       2MiB /  49140MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
|   3  NVIDIA RTX A6000               On  |   00000000:61:00.0 Off |                  Off |
| 30%   31C    P8             20W /  300W |       2MiB /  49140MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
```

Or run with cpu only (not recommended)

For testing purposes only, we will use a much smaller model, which is more suitable for CPU usage but the performance will be significantly lower.

```
# Run in detached mode
docker compose -f docker-compose.cpu.yml up -d --build

# Pull necessary Ollama models (example)
docker exec -it mtr-ollama ollama pull deepseek-r1:1.5b
```

Now you can access the application at http://localhost:8501

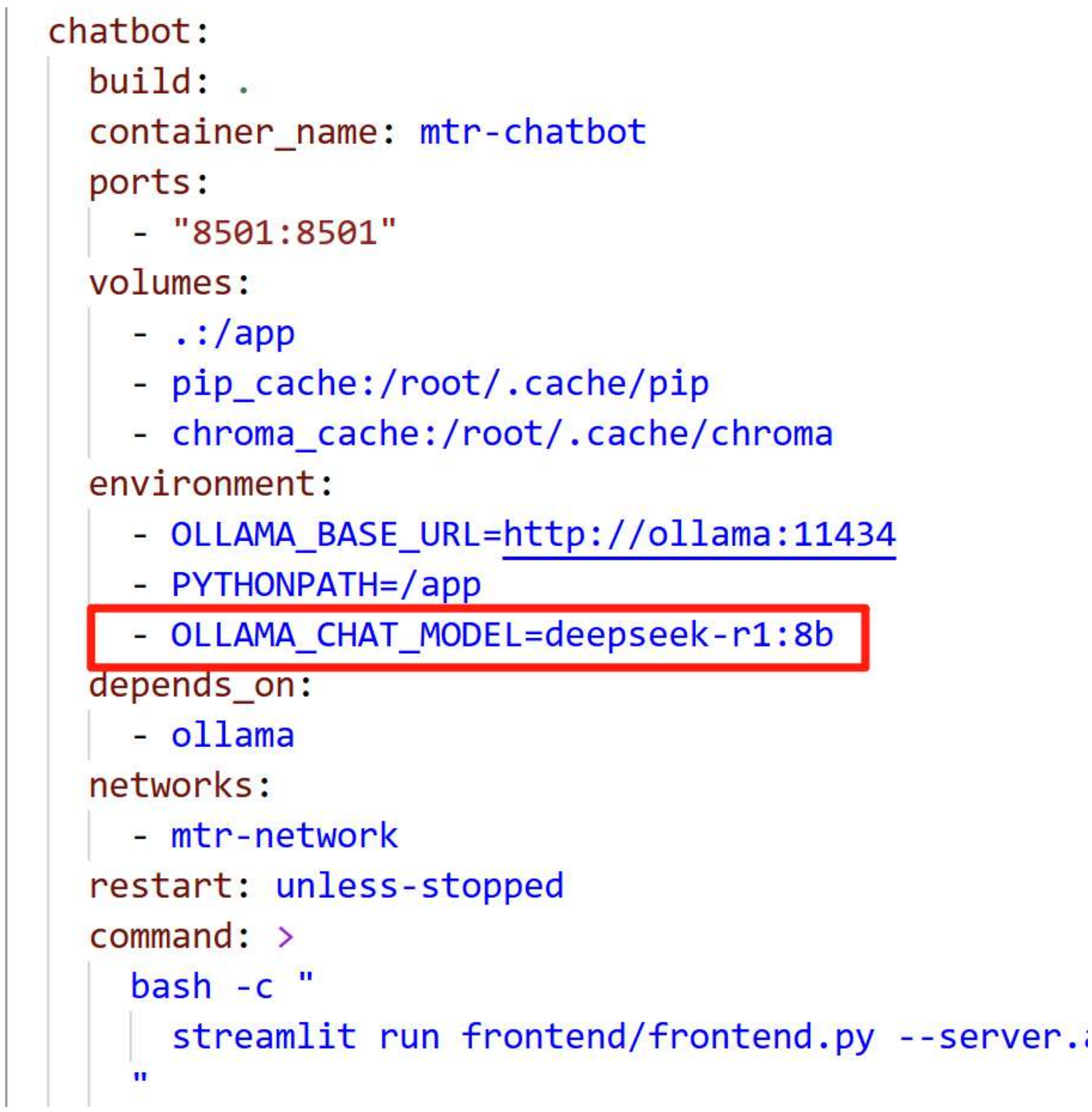
View the logs

```
# view the logs
docker compose logs -f
# view the chatbot logs only
docker compose logs -f chatbot
```

Change Models

You can change the models used by modifying the `docker-compose.yml` or `docker-compose.cpu.yml` file.

For example, to change the model for Ollama, update the `OLLAMA_CHAT_MODEL` environment variable in the `ollama` service section:



```
chatbot:
  build: .
  container_name: mtr-chatbot
  ports:
    - "8501:8501"
  volumes:
    - .:/app
    - pip_cache:/root/.cache/pip
    - chroma_cache:/root/.cache/chroma
  environment:
    - OLLAMA_BASE_URL=http://ollama:11434
    - PYTHONPATH=/app
    - OLLAMA_CHAT_MODEL=deepseek-r1:8b
  depends_on:
    - ollama
  networks:
    - mtr-network
  restart: unless-stopped
  command: >
    bash -c "
      streamlit run frontend/frontend.py --server.
    "
```

After making changes to the `docker-compose.yml` or `docker-compose.cpu.yml` file, restart the services:

```
docker compose restart ollama chatbot
```

After restarting the services, pull required models:

```
docker exec mtr-ollama ollama pull deepseek-r1:8b
docker exec mtr-ollama ollama pull qwen3:8b
docker exec mtr-ollama ollama pull llama3.1:8b
# or any other model you need
```