ChatGPT

BadAi – Ambiente Docker Compose per ComfyUI

Il progetto **BadAi** è una configurazione Docker Compose che permette di eseguire **ComfyUI** (un'interfaccia grafica a nodi per Stable Diffusion) con supporto GPU e plugin IP-Adapter Plus, soddisfacendo tutti i requisiti richiesti. Di seguito descriviamo la struttura del progetto e i file principali inclusi.

Struttura del Progetto

```
BadAi/
├── docker-compose.yml
├── Dockerfile
├── entrypoint.sh
├── models/        # Volume persistente per i modelli
└── output/        # Volume persistente per le immagini generate
```

- **Volumi persistenti**: La cartella `models/` sul host è montata come `/models` nel container ed è collegata alla directory dei modelli di ComfyUI (tramite link simbolico) in modo da conservare i modelli scaricati. Allo stesso modo `output/` è montata come `/output` nel container per salvare fuori dal container le immagini generate.
- **Immagine base GPU**: Il `Dockerfile` utilizza Ubuntu 22.04 con **CUDA 12.1** e include **PyTorch** già installato (build compatibile CUDA 12.1) per abilitare l'accelerazione GPU.
- **Installazione automatica**: All'avvio, uno script `entrypoint.sh` esegue l'installazione delle dipendenze Python di ComfyUI, clona il plugin **ComfyUI_IPAdapter_plus** se non presente e scarica al primo avvio i modelli richiesti (IP-Adapter SDXL e CLIP ViT-H-14), posizionandoli nelle relative sottodirectory sotto `/models` come raccomandato [1] [2]. Infine, lo script avvia ComfyUI.

Di seguito presentiamo in dettaglio il contenuto di ciascun file.

File `docker-compose.yml`

Questo file definisce il servizio Docker Compose denominato **badai**. Esso costruisce l'immagine dal Dockerfile corrente, abilita l'uso della GPU (NVIDIA), espone la porta web di ComfyUI e monta i volumi persistenti per modelli e output.

```yaml
version: "3.8"
services:
  badai:
    build: .
    image: badai:latest
    container_name: badai
    ports:
      - "8188:8188"
```

```
    volumes:
      - ./models:/models
      - ./output:/output
    # Abilita il supporto GPU NVIDIA
    deploy:
      resources:
        reservations:
          devices:
            - driver: nvidia
              count: all
              capabilities: ["gpu"]
    environment:
      - NVIDIA_VISIBLE_DEVICES=all
      - NVIDIA_DRIVER_CAPABILITIES=compute,utility
```

Note: La porta predefinita di ComfyUI è la **8188**, quindi viene pubblicata sullo stesso numero (8188) per l'accesso dall'host. La sezione `deploy` con `devices: gpu` e le variabili d'ambiente NVIDIA assicurano che il container utilizzi il runtime NVIDIA (CUDA) su Windows + WSL2 (è necessario aver installato il *NVIDIA Container Toolkit* in WSL2).

File `Dockerfile`

Il Dockerfile configura l'immagine Docker con tutti i componenti necessari:

```
# Base image: Ubuntu 22.04 con CUDA 12.1 e cuDNN 8
FROM nvidia/cuda:12.1.1-cudnn8-runtime-ubuntu22.04

ENV DEBIAN_FRONTEND=noninteractive

# Installazione di Python 3, pip, git e wget nel container
RUN apt-get update && apt-get install -y --no-install-recommends \
    python3 python3-pip git wget && \
    apt-get clean && rm -rf /var/lib/apt/lists/*

# Installazione di PyTorch con supporto CUDA 12.1
RUN pip3 install --no-cache-dir torch torchvision torchaudio \
    --index-url https://download.pytorch.org/whl/cu121

# Clona il repository ComfyUI nell'immagine
RUN git clone https://github.com/comfyanonymous/ComfyUI.git /opt/ComfyUI

# Copia dello script di entrypoint nel container
COPY entrypoint.sh /entrypoint.sh
RUN chmod +x /entrypoint.sh

# Imposta la working directory
WORKDIR /opt/ComfyUI

# Collega le cartelle dei modelli e output ai volumi persistenti
```

```
RUN mkdir -p /models /output && \
    rm -rf /opt/ComfyUI/models /opt/ComfyUI/output && \
    ln -s /models /opt/ComfyUI/models && \
    ln -s /output /opt/ComfyUI/output

# Espone la porta di ComfyUI
EXPOSE 8188

# Configura l'entrypoint di default
ENTRYPOINT ["/entrypoint.sh"]
```

Spiegazione dei passi principali:

- Viene utilizzata l'immagine NVIDIA CUDA 12.1 (runtime) su Ubuntu 22.04 come base. Ciò garantisce la presenza delle librerie CUDA necessarie.
- Si installano pacchetti di sistema essenziali e si utilizza pip per installare **PyTorch** (torch, torchvision, torchaudio) con le specifiche *wheels* per CUDA 12.1.
- Si clona il repository di ComfyUI direttamente nell'immagine, posizionandolo in `/opt/ComfyUI`. (In questo modo il codice di ComfyUI è pronto all'uso all'interno dell'immagine).
- Si copia lo script `entrypoint.sh` e lo si rende eseguibile.
- **Persistenza modelli/output:** la directory `/opt/ComfyUI/models` e `/opt/ComfyUI/output` (quelle predefinite di ComfyUI per modelli e immagini) vengono rimosse e rimpiazzate con link simbolici che puntano a `/models` e `/output` (le cartelle montate come volumi). In questo modo, quando ComfyUI salva o legge file da `ComfyUI/models` e `ComfyUI/output`, in realtà utilizza i volumi persistenti sul disco host.
- Si espone la porta 8188 per consentire l'accesso all'interfaccia web.
- L'entrypoint del container è lo script `entrypoint.sh` (descritto sotto), che configurerà l'ambiente all'avvio.

File `entrypoint.sh`

Questo script viene eseguito ogni volta che il container parte. Esso si occupa di:

1. Installare le dipendenze Python di ComfyUI (dal `requirements.txt` del progetto).
2. Clonare il plugin **ComfyUI_IPAdapter_plus** (se non già presente) nella cartella dei custom nodes di ComfyUI.
3. Creare le cartelle per i modelli IP-Adapter e CLIP all'interno di `/models` e scaricare, al primo avvio, i file dei modelli richiesti (se non già presenti nei volumi):
4. `ip-adapter_sdxl_vit-h.safetensors` (modello IP-Adapter SDXL) nella cartella `/models/ipadapter`.
5. `CLIP-ViT-H-14-laion2B-s32B-b79K.bin` (modello CLIP ViT-H-14) nella cartella `/models/clip_vision`.
6. Avviare ComfyUI con supporto GPU.

Ecco il contenuto dello script:

```
#!/bin/bash
set -e
```

```
# 1. Installa le dipendenze Python di ComfyUI
pip3 install --no-cache-dir -r /opt/ComfyUI/requirements.txt

# 2. Installa il plugin ComfyUI_IPAdapter_plus se non presente
if [ ! -d "/opt/ComfyUI/custom_nodes/ComfyUI_IPAdapter_plus" ]; then
    echo "Installing ComfyUI_IPAdapter_plus plugin..."
    git clone https://github.com/cubiq/ComfyUI_IPAdapter_plus.git \
        /opt/ComfyUI/custom_nodes/ComfyUI_IPAdapter_plus
fi

# Crea le cartelle modelli se non esistono già
mkdir -p /models/clip_vision /models/ipadapter

# 3. Scarica i modelli necessari al primo avvio
# IP-Adapter SDXL model
if [ ! -f "/models/ipadapter/ip-adapter_sdxl_vit-h.safetensors" ]; then
    echo "Downloading ip-adapter_sdxl_vit-h.safetensors..."
    wget -q --show-progress -O /models/ipadapter/ip-adapter_sdxl_vit-
h.safetensors \
        https://huggingface.co/h94/IP-Adapter/resolve/main/sdxl_models/ip-
adapter_sdxl_vit-h.safetensors
fi

# CLIP ViT-H-14 model
if [ ! -f "/models/clip_vision/CLIP-ViT-H-14-laion2B-s32B-b79K.bin" ]; then
    echo "Downloading CLIP-ViT-H-14-laion2B-s32B-b79K.bin..."
    wget -q --show-progress -O /models/clip_vision/CLIP-ViT-H-14-laion2B-
s32B-b79K.bin \
        https://huggingface.co/laion/CLIP-ViT-H-14-laion2B-s32B-b79K/resolve/
main/CLIP-ViT-H-14-laion2B-s32B-b79K.bin
fi

# 4. Avvia ComfyUI (server web sulla porta 8188)
cd /opt/ComfyUI
exec python3 main.py ${CLI_ARGS}
```

Dettagli importanti:

- Usiamo `set -e` perché in caso di errore in uno dei passi vogliamo interrompere l'esecuzione (evitando di far partire ComfyUI senza dipendenze corrette).
- La clonazione di **ComfyUI_IPAdapter_plus** inserisce il plugin nella cartella `ComfyUI/custom_nodes/` del repository ComfyUI come richiesto nelle istruzioni del plugin [3].
- I modelli vengono scaricati dal repository **HuggingFace** appropriato: il modello IP-Adapter SDXL viene preso dall'utente *h94* (repo *IP-Adapter*) e il modello CLIP ViT-H-14 da *laion*. I percorsi di destinazione (`/models/ipadapter/...` e `/models/clip_vision/...`) corrispondono a quelli attesi dal plugin IP-Adapter Plus [1]. Notare che abbiamo mantenuto l'estensione `.bin` per il modello CLIP come indicato (in HuggingFace viene fornito come `.bin` e il plugin lo supporta) [4].

- Infine, viene eseguito `python3 main.py` nella directory di ComfyUI per avviare l'interfaccia. (È possibile passare variabili d'ambiente `CLI_ARGS` per abilitare opzioni extra di ComfyUI; per esempio `--use-pytorch-cross-attention` o `--listen` se necessario).

Utilizzo del Progetto

Una volta che i file sono pronti, è possibile avviare il sistema eseguendo il comando:

```
docker compose up --build
```

Questo comando costruirà l'immagine Docker (integrando ComfyUI e il plugin) e lancerà il container **badai**. Al primo avvio lo script di entrypoint scaricherà automaticamente i modelli IP-Adapter SDXL e CLIP richiesti (circa qualche centinaio di MB, potrebbe richiedere tempo). Dopodiché ComfyUI sarà disponibile all'indirizzo http://localhost:8188 sul browser host.

Le immagini generate saranno salvate nella cartella `output/` sul host, e i modelli scaricati rimarranno nella cartella `models/` sul host per utilizzi successivi (evitando di riscaricarli).

Nota: In caso di aggiornamenti di ComfyUI o del plugin, sarà sufficiente ricostruire l'immagine o eliminare la cartella del plugin nel volume per forzarne la reinstallazione all'avvio successivo. Assicurarsi di avere i driver NVIDIA correttamente installati su Windows e il supporto CUDA attivo in WSL2.

Di seguito è allegato un archivio ZIP **BadAi.zip** contenente tutti i file descritti (comprensivo di `docker-compose.yml`, `Dockerfile`, `entrypoint.sh` e le cartelle vuote `models/` e `output/`). È possibile scaricarlo e utilizzarlo direttamente.

【51†download】 **BadAi.zip** (tutti i file del progetto)

---

1 2 3 4 GitHub - cubiq/ComfyUI_IPAdapter_plus
https://github.com/cubiq/ComfyUI_IPAdapter_plus