Install Ollama Docker Image

Step 0: Pull the image

```
$ sudo docker pull ollama/ollama
```

Step 0: Deploy admin scripts [OPTIONAL]

```
$ mkdir -p $HOME/bin/ollama
$ cp start-ollama.sh stop-ollama.sh $HOME/bin/ollama
```

Update .bashrc:

```
export PATH=$PATH:$HOME/bin/ollama
```

Source .bashrc:

```
$ source ~/.bashrc
```

Step 1: Download the LLMs

Download the target LLM(s) locally:

- llama-2-7b-chat.Q4_K_S.gguf (https://huggingface.co/TheBloke/Llama-2-7B-Chat-GGUF/resolve/main/llama-2-7b-chat.Q2_K.gguf)
- llama-3.2-1b-instruct-q8_0.gguf (https://huggingface.co/hugging-quants/Llama-3.2-1B-Instruct-Q8_0-GGUF)

Place them into a suitable directory on your PC. For example, in the following steps the following directory is used: **/home/ml/pretrained-models/llms/llama/**. Adapt to your environment.

Step 2: Create the Modelfiles

Create the ollama's configuration files related to these two models: **modelfiles**.
You create one Modelfile per local LLM:

Somewhere in your project directory: **/MY_PROJECT_TOPDIR/**

```
$ mkdir -p ollama-modelfiles
$ cd ollama-modelfiles
$ vi Modelfile.llama-2-7b-chat.Q4_K_S.docker.v1
```

```
FROM /root/llms/llama/llama-2-7b-chat.Q4_K_S.gguf

$ vi Modelfile.llama-3.2-1b-instruct-q8_0.docker.v1

FROM /root/llms/llama/llama-3.2-1b-instruct-q8_0.gguf
```

Step 3: Start the Ollama container (on CPUs)

```
$ mkdir -p /home/ml/ollama
```

```
$ docker stop ollama ; docker rm ollama
$ sudo docker run -d \
  -v /home/ml/ollama:/root/.ollama \
  -v /home/ml/pretrained-models/llms:/root/llms \
  -v
/home/ml/pretrained-models/llms/ollama-modelfiles:/root/ollama-modelfiles \
  -p 11434:11434 \
  --name ollama \
  --restart always \
  --add-host=host.docker.internal:host-gateway \
  ollama/ollama
```

Explanations:

WARNING

Adapt the folders hierarchy to your own environment. For instance, /home/ml/ollama, /home/ml/pretrained-models/llms, etc, should be changed to reflect your own directory structure.
It is a good ideas to externalize the ollama-modelfiles directory in order to be accessible for all projects ; one can copy it somewhere near the llms:

```
/home/ml/pretrained-models/llms
├── llama
│   ├── llama-2-7b-chat.Q4_K_S.gguf
│   └── llama-3.2-1b-instruct-q8_0.gguf
└── ollama-modelfiles
    ├── Modelfile.llama-2-7b-chat.Q4_K_S.docker.v1
    └── Modelfile.llama-3.2-1b-instruct-q8_0.docker.v1
```

- The directory `/home/ml/ollama` is where Ollama will put its personal stuff
- Three volumes are created using -v command line option

INFO

Simply run **$ start-ollama.sh** if you have deployed the provided script. See the script documentation for deploying it.

Step 3 bis: Start the Ollama container (on GPUs)

TODO

Step 4: Register the LLMs in Ollama

List registered models:

```
$ docker exec -it ollama ollama list
```

IT should be empty!

Register the LLMs:

```
$ docker exec -it ollama ollama create llama-2-7b-chat -f
/root/ollama-modelfiles/Modelfile.llama-2-7b-chat.Q4_K_S.docker.v1
...
success
```

```
$ docker exec -it ollama ollama create llama-3.2-1b-instruct -f
/root/ollama-modelfiles/Modelfile.llama-3.2-1b-instruct-q8_0.docker.v1
...
success
```

List registered models:

```
$ docker exec -it ollama ollama list
NAME                          ID              SIZE      MODIFIED
llama-3.2-1b-instruct:latest  2aede794639c    1.3 GB    8 seconds ago
llama-2-7b-chat:latest        b9cacec83cce    3.9 GB    12 days ago
```

Step 5: Prompt the LLMs

Make a simple test by prompting one of the models:

```
$ docker exec -it ollama ollama run llama-3.2-1b-instruct
>>> Send a message (/? for help)

>>> what is the capital of Côte d'Ivoire?
```

```
The capital of Côte d'Ivoire (Ivory Coast) is Yamoussoukro.
```

```
>>> can you speak french?
Non, je ne parle pas français. I can try to communicate in French,
though...
```

Ok your Ollama Docker installation if functional. Type **/?** for help or **/bye** to exit the prompt.

Step 5: Getting your Ollama server urls

Normally the server is already running and listening on the port **11434** (specified early when starting the container).

Its urls should be (for Q&A) `http://localhost:11434/api/generate` and (for Chat) `http://localhost:11434/api/chat`.
Example:

```
$ curl http://localhost:11434/api/generate -d '{
  "model": "llama-3.2-1b-instruct",
  "prompt":"What is the capital of Ivory Coast?"
}'
```

Response:

```
...
{"model":"llama-3.2-1b-instruct",...,"response":" is","done":false}
{"model":"llama-3.2-1b-instruct",...,"response":" Yam","done":false}
{"model":"llama-3.2-1b-instruct",...,"response":"ou","done":false}
{"model":"llama-3.2-1b-instruct",...,"response":"ss","done":false}
{"model":"llama-3.2-1b-instruct",...,"response":"ou","done":false}
{"model":"llama-3.2-1b-instruct",...,"response":"k","done":false}
{"model":"llama-3.2-1b-instruct",...,"response":"ro","done":false}
...
```

As you see, the response is provided ; but not very humanly readable… We'll use appropriate tools to get more comfort for prompting our models.

Step 6: Stop Ollama

```
$ docker stop ollama ; docker rm ollama
```

or simply **$ stop-ollama.sh** if the provided script is deployed.

Troubleshooting

Contact the trainer.

END OF DOCUMENT.