IPEX vLLM Serving Demo with openwebui and chatbox

Backend with vLLM Serving

1. Start docker using `backend-ipex-docker.sh`
   - change `-v <model-path>:/llm/models` to your model path
   - change `-v <script-path>:/llm/workspace` to the script file path

```
bash backend-ipex-docker.sh
```

2. Start IPEX vLLM Serving using `vllm-serving` in docker container

> Only using tp4pp2 need change the openwebui docker backend code

go into docker container

```
docker exec -it ipex-llm-b6 bash
```

and start vllm serving

```
bash vllm-deepseek32b-serving.sh
```

Frontend with openwebui or chatbox(web)

openwebui

1. start docker using `frontend-openwebui-docker.sh` change `OPENAI_API_BASE_URL`'s ip to

```
bash frontend-openwebui-docker.sh
```

2. visit https://host-ip:3000 sign up or sign in
   - username: `bigdl@intel.com`
   - password: `intel123`

chatbox(web)

1. map vllm serving machine port to localhost(chatbox webui can only access to local network only, not LAN) using following command(need change the `<username>@<server-ip>`) and type your password to auth it

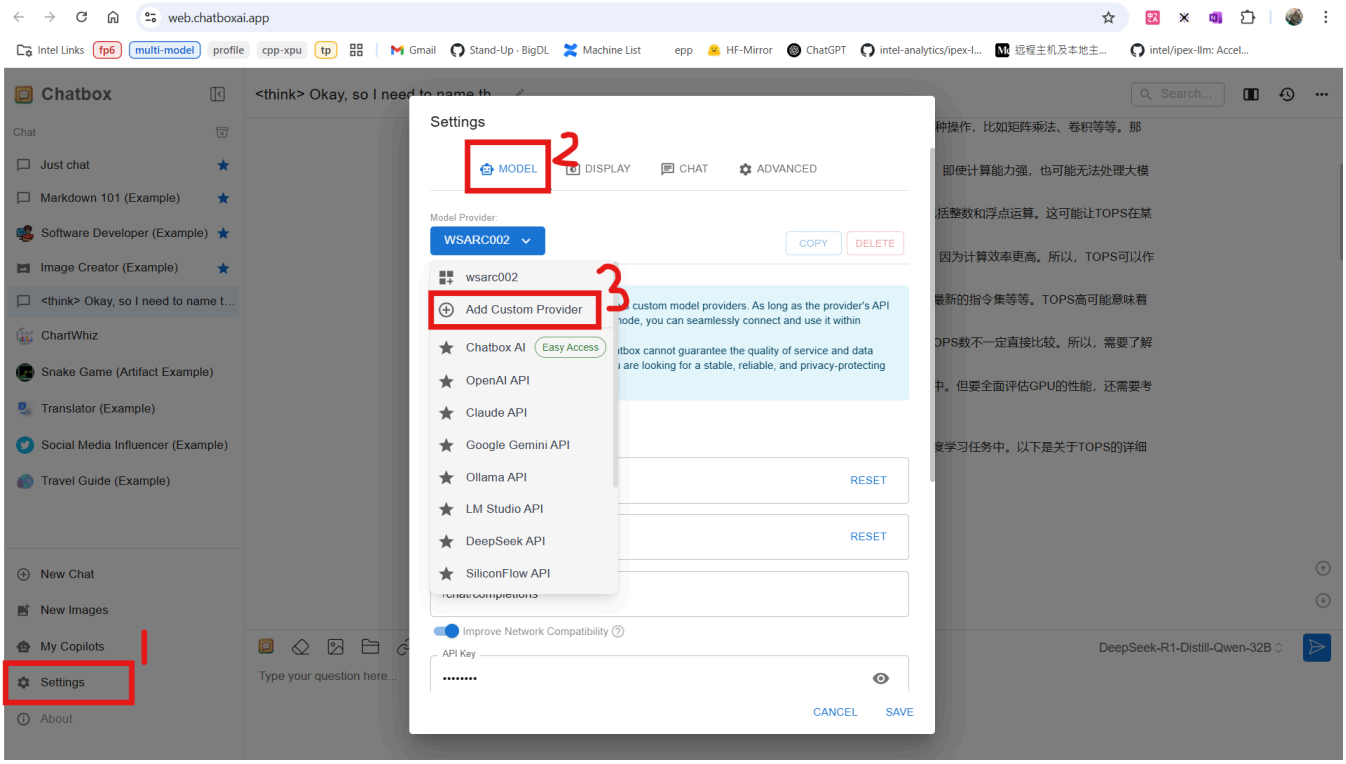
```
ssh -L 8001:localhost:8001 intel@10.238.154.133
```

you can use following command(need change the `"Authorization: Bearer <api-key>"` to the in your vllm serving start script) in your machine to check if port mapping is successful

```
curl http://localhost:8001/v1/models -H "Content-Type: application/json" -H "Authorization:
```

expect output like this, which contain the vllm serving model name

2. visit https://web.chatboxai.app/ website, and click `Settings -> MODEL -> Add Custom Provider` button like this:



3. configure and save it

   - API Mode: default `OpenAI API Compatible`
   - Name: any name you like
   - API Host: `http://localhost:8001/v1` (need map remote vllm serving to local host like step 1, and **only support http not https**)
   - API Path: default `/chat/completions`
   - `Improve Network Compatibility`: enable and disable not affect
   - API Key: `<api-key>` in your vllm-serving script

here is an example: