(nginxloadbalancer)=

Using Nginx

This document shows how to launch multiple vLLM serving containers and use Nginx to act as a load balancer between the servers.

Table of contents:

(nginxloadbalancer-nginx-build)=

Build Nginx Container

This guide assumes that you have just cloned the vLLM project and you're currently in the vllm root directory.

```console
export vllm_root=`pwd`
```

Create a file named `Dockerfile.nginx`:

```console
FROM nginx:latest
RUN rm /etc/nginx/conf.d/default.conf
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

Build the container:

```console
docker build . -f Dockerfile.nginx --tag nginx-lb
```

(nginxloadbalancer-nginx-conf)=

Create Simple Nginx Config file

Create a file named `nginx_conf/nginx.conf`. Note that you can add as many servers as you'd like. In the below example we'll start with two. To add more, add another `server vllmN:8000 max_fails=3 fail_timeout=10000s;` entry to `upstream backend`.

```console
upstream backend {
    least_conn;
    server vllm0:8000 max_fails=3 fail_timeout=10000s;
    server vllm1:8000 max_fails=3 fail_timeout=10000s;
}
server {
    listen 80;
    location / {
        proxy_pass http://backend;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

(nginxloadbalancer-nginx-vllm-container)=

Build vLLM Container

```console
cd $vllm_root
docker build -f Dockerfile . --tag vllm
```

If you are behind proxy, you can pass the proxy settings to the docker build command as shown below:

```console
cd $vllm_root
docker build -f Dockerfile . --tag vllm --build-arg http_proxy=$http_proxy --build-arg https_proxy=$https_proxy
```

(nginxloadbalancer-nginx-docker-network)=

Create Docker Network

```console
docker network create vllm_nginx
```

(nginxloadbalancer-nginx-launch-container)=

Launch vLLM Containers

Notes:

- If you have your HuggingFace models cached somewhere else, update `hf_cache_dir` below.
- If you don't have an existing HuggingFace cache you will want to start `vllm0` and wait for the model to complete downloading and the server to be ready. This will ensure that `vllm1` can leverage the model you just downloaded and it won't have to be downloaded again.
- The below example assumes GPU backend used. If you are using CPU backend, remove `--gpus all`, add `VLLM_CPU_KVCACHE_SPACE` and `VLLM_CPU_OMP_THREADS_BIND` environment variables to the docker run command.

- Adjust the model name that you want to use in your vLLM servers if you don't want to use `Llama-2-7b-chat-hf`.

```console
mkdir -p ~/.cache/huggingface/hub/
hf_cache_dir=~/.cache/huggingface/
docker run -itd --ipc host --privileged --network vllm_nginx --gpus all --shm-size=10.24gb -v $hf_cache_dir:/root/.cache/huggingface/ -p 8081:8000 --name vllm0 vllm --model meta-llama/Llama-2-7b-chat-hf
docker run -itd --ipc host --privileged --network vllm_nginx --gpus all --shm-size=10.24gb -v $hf_cache_dir:/root/.cache/huggingface/ -p 8082:8000 --name vllm1 vllm --model meta-llama/Llama-2-7b-chat-hf
```

:::{note}
If you are behind proxy, you can pass the proxy settings to the docker run command via `-e http_proxy=$http_proxy -e https_proxy=$https_proxy`.
:::

(nginxloadbalancer-nginx-launch-nginx)=

Launch Nginx

```console
docker run -itd -p 8000:80 --network vllm_nginx -v ./nginx_conf/:/etc/nginx/conf.d/ --name nginx-lb nginx-lb:latest
```

(nginxloadbalancer-nginx-verify-nginx)=

Verify That vLLM Servers Are Ready

```console
docker logs vllm0 | grep Uvicorn
docker logs vllm1 | grep Uvicorn
```

Both outputs should look like this:

```console
INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
```