AI-04533 OpenWebTTS Docker Deployment

Show me a recommended Docker Compose configuration for running OpenWebTTS with CPU-optimized models on Linux. Use dockerlab-network because I am using NPM (not cloudflare DNS), the directory is /home/rarcayan/dockerlab/apps/openwebtts.

Here is a ready-to-deploy `docker-compose.yml` tailored for your setup.

Since you are running Nginx Proxy Manager (NPM) on the existing `dockerlab-network` and working out of `/home/rarcayan/dockerlab/apps/openwebtts`, this configuration connects directly to your network proxy without requiring exposed host ports unless you want local IP access for troubleshooting.

Directory & File Setup

First, navigate to your app directory and create the folder structure:

Bash

```
cd /home/rarcayan/dockerlab/apps/openwebtts
mkdir -p data/cache
```

`docker-compose.yml`

Create `docker-compose.yml` inside `/home/rarcayan/dockerlab/apps/openwebtts/`:

YAML

```
version: "3.8"

services:
  openwebtts:
    image: synesthesiam/opentts:en
    container_name: openwebtts
    restart: unless-stopped
    command:
      - "--cache"
      - "/cache"
      - "--no-espeak"  # Disables legacy robotic eSpeak voices to save memory/CPU
    volumes:
      - ./data/cache:/cache
    environment:
      - TZ=Asia/Manila  # Set to your preferred local timezone
    networks:
      - dockerlab-network
    # Optional: Uncomment if you want to test directly via http://<SERVER_IP>:5500
    # ports:
```

```
    #   - "5500:5500"

networks:
  dockerlab-network:
    external: true
```

Key Configuration Highlights

1. **CPU Efficiency (** `synesthesiam/opentts:en` **):**
   Using the `:en` image tag loads English voice models (including lightweight Piper, Larynx, and Flite models) without the heavy memory overhead of downloading all global language packages.
2. **Persistent Audio Caching (** `--cache /cache` **):**
   By passing `--cache /cache` and mounting `./data/cache`, synthesized audio segments are stored locally. If you replay a section or jump back in a chapter, it loads instantly from disk rather than re-synthesizing on CPU.
3. **NPM Integration:**
   By attaching to `dockerlab-network`, NPM can route traffic directly to `http://openwebtts:5500` internally—no need to expose port 5500 to the public host interface.

Deployment Commands

From `/home/rarcayan/dockerlab/apps/openwebtts`, run:

Bash

```
# Pull image and start container in detached mode
docker compose up -d

# Check startup logs
docker compose logs -f
```

Nginx Proxy Manager (NPM) Setup

1. Open your NPM Web UI.
2. Add a new **Proxy Host**:
   - **Domain Names:** `openwebtts.yourdomain.com` (or your chosen local/subdomain)
   - **Scheme:** `http`
   - **Forward Hostname / IP:** `openwebtts` *(container name on* `dockerlab-network` *)*
   - **Forward Port:** `5500`
   - **Websockets Support:** Enabled (recommended for streaming audio)

Once NPM routes the subdomain, open it in your browser or point your HTTP client/mobile reader directly to the Web UI endpoint.