Setting up the Streaming Server

Introduction

This guide provides step-by-step instructions to set up the Streaming Server from scratch. This self-hosted solution allows you to download and stream video/audio files via a Telegram bot and a web interface. The system enables users to interact with the bot to request media, which is then processed and served through a web player.

By following this guide, you will replicate the server's setup using Docker, ensuring scalability and easy deployment.

Prerequisites

Before starting, make sure you have the following tools installed on your machine:

- Docker: The server is Dockerized, meaning all components are run inside containers.
- Docker Compose: A tool to define and run multi-container Docker applications.
- Git: Used to clone the repository and manage version control.

Cloning the Repository

To begin, you need to clone the Streaming Server repository from GitHub to your local machine:

```
git clone https://github.com/PumukyDev/stream-server.git
cd stream-server
```

Once you've cloned the repository and navigated to the project directory you can configure the server with your own information.

Configuration

Before running the server, a few configurations must be set up, including the Telegram bot and environment variables. Follow these steps carefully.

Telegram Bot Configuration

1. To interact with the Telegram bot, you need to create your own bot. This is done using the Telegram BotFather.
    - Open Telegram and search for **BotFather**.
    - Type `/newbot` and follow the instructions to create a new bot.

    - Once your bot is created, you will receive a unique **API token**. Keep this token safe as you will need it to configure the bot.
2. You will use this token to interact with the Telegram API.

Setting Up the Environment

Now, you need to configure environment variables for your server. Create a `.env` file in the telegram-bot directory of the project. This file will store sensitive information, such as the Telegram bot API token.

Example `.env` file:

```
TELEGRAM_API_TOKEN=your-telegram-bot-api-token-here
```

Make sure to replace `your-telegram-bot-api-token-here` with the actual token you received from BotFather.

Nginx Web Server Configuration (optional)

If you want to watch/listen your streaming in a web browser you can customize the Nginx web server settings by modifying the `nginx/nginx.conf` file. This file defines how the server will handle incoming requests, including stream routing and access control.

The Nginx configuration file is already set up for most use cases, but you can adjust it if you need to change port numbers, add SSL certificates, or modify security settings.

Starting the Server

Once the configuration is complete, you can start the server using Docker Compose. Follow these steps:

Build and start the Docker containers:

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

The `--build` flag ensures that Docker will rebuild the containers with the latest changes, while the `-d` flag runs the containers in detached mode, meaning they will run in the background.

Verify that everything is working:

After starting the containers, open your web browser and navigate to `http://localhost`. If everything is configured correctly, you should see the web interface for the streaming service.

Check the logs (optional):

If something goes wrong, you can check the logs of your containers to debug any issues:

```
docker-compose logs
```

This command will show logs from all running containers, which can help you identify errors related to the web server, bot, or other services.

Accessing the Telegram Bot

Once the server is running, you can access the Telegram bot:

1. **Search for your bot in Telegram:**
   - Open the Telegram app and search for your bot.
   - Alternatively, you can scan the QR code generated by your bot if you have save it
2. **Interact with the bot:**
   - Send a YouTube link using the `/dv` (video download) or `/da` (audio download) command. The bot will download the media and provide you with a streaming link.

     ```
     Example: /dv https://www.youtube.com/watch?v=dQw4w9WgXcQ
     ```
3. **Receive the streaming link:**
   - Once the bot processes your request, it will send you a link to access the media in a web player.

Accessing the Web Player

Once you have the streaming link, you can open it in your web browser or in vlc. The web player allows you to stream audio or video content in real-time.

1. **Open the streaming link:**
   - The bot will provide a link in the format: <code><a href="http://<your-server-ip>:<port>/video/<media-file&gt" class="bare">http://<your-server-ip>:<port>/video/<media-file&gt;</code>.
   - Open this link in your web browser.
2. **Choose between video or audio playback:**
   - You can choose the type of content you wish to stream, either audio or video, based on your preferences.
3. **Start streaming:**
   - The media will start streaming directly in your browser without needing to install any

additional software.

Stopping the Server

If you want to stop the server and remove the containers, use the following Docker Compose command:

```
docker-compose down
```

This will stop and remove the running containers, but all data (e.g., downloaded media) will remain unless you explicitly remove the volumes.

If you want to stop the containers without removing them, you can use:

```
docker-compose stop
```

This will stop the containers but leave them on your system for later use.

Troubleshooting

Here are some common issues you might encounter and their solutions:

Docker Containers Fail to Start

- Ensure Docker and Docker Compose are properly installed.
- Check the `.env` file to make sure the Telegram bot token is correctly set.
- Run `docker-compose build` to rebuild the containers if you encounter issues.
- Check logs/ directory is beeing generated correcly if icecast2 container is stopping down.
- Icescast2 container will shut down if there is no songs to reproduce, make sure it has some songs before running it.

Bot Not Responding

- Make sure the API token in the `.env` file is correct.
- Ensure the server has internet access to communicate with Telegram's servers.
- Check the bot's status by visiting the BotFather on Telegram and using the `/status` command.

Nginx Errors (e.g., 502 Bad Gateway)

- Make sure Nginx is correctly configured and pointing to the right services.
- Check the logs in the `logs/` directory for any detailed error messages.

- Ensure all services (e.g., Telegram bot and web server) are running.
- Some times it is solved simply by restarting the server.