Ship Proxy System

This project demonstrates a **Python client-server setup using Docker**, where the client acts as an HTTP proxy and the server handles backend logic.

The client forwards requests from users to external websites and returns the responses.

Project Structure

```
ship-proxy-system/
 ├── client/
 │  ├── client.py
 │  └── Dockerfile
 ├── server/
 │  ├── server.py
 │  └── Dockerfile
 └── docker-compose.yml
```

- **client/**: Client proxy code, exposes port `8080`.
- **server/**: Backend server code, exposes port `9999`.
- **docker**-compose.yml: Starts both client and server containers together.

Prerequisites

- Docker Desktop installed (Windows / Mac / Linux)
- Docker Hub account (`swagatika957`)

Docker Hub Images

- Client: `swagatika957/client:latest`
- Server: `swagatika957/server:latest`

Running the Project

Option 1: Using Docker Compose (recommended)

~~~
```powershell
docker-compose up
~~~

- Starts both **server** and **client** containers.
- Client accessible at: `http://localhost:8080`
- Server accessible at: `http://localhost:9999`

Stop containers with:

```
docker-compose down
```

Option 2: Running Containers Individually

```
docker run -p 9999:9999 swagatika957/server:latest
docker run -p 8080:8080 swagatika957/client:latest
```

> Ensure the **server container** is running before starting the client.

Testing the Proxy

Example `curl.exe` commands (Windows)

- **GET request**:

```
curl.exe -x http://localhost:8080 http://httpforever.com/
curl.exe -x http://localhost:8080 https://example.com/
```

- **POST request**:

```
curl.exe -x http://localhost:8080 -X POST -d "hello=world" https://httpbin.org/post
```

- **PUT request**:

```
curl.exe -x http://localhost:8080 -X PUT -d "update=test" https://httpbin.org/put
```

- **DELETE request**:

```
curl.exe -x http://localhost:8080 -X DELETE https://httpbin.org/delete
```

- **Delayed response test**:

```
curl.exe -x http://localhost:8080 https://httpbin.org/delay/3
```

These tests confirm that the client proxy supports **all HTTP methods** and handles multiple requests consistently.
