Yes, absolutely! Docker is an excellent tool for testing this kind of distributed system setup on a single Linux machine. You can simulate the three nodes, each running Nginx and a Flask application, using Docker containers and Docker Compose.

Here's how you can set it up:

### 1. Project Structure

Create a directory for your project, and inside it, create the following files:

my-distributed-app/  
├── app/  
│ └── app.py  
├── nginx/  
│ └── nginx.conf  
├── Dockerfile.app  
├── Dockerfile.nginx  
└── docker-compose.yml

### 2. app/app.py (Flask Application)

This is the same Flask application code as before.

app/app.py (Flask Application)

Jul 25, 2:35 PM

Open

### 3. Dockerfile.app (for Flask Application)

Dockerfile.app (for Flask Application)

Jul 25, 2:35 PM

Open

### 4. nginx/nginx.conf (Nginx Configuration)

**Important:** In Docker Compose, container service names (e.g., app1, app2, app3) act as hostnames within the Docker network. So, the upstream block in Nginx will refer to these service names.

nginx/nginx.conf (Nginx Configuration)

Jul 25, 2:35 PM

Open

### 5. Dockerfile.nginx (for Nginx Proxy)

Dockerfile.nginx (for Nginx Proxy)

Jul 25, 2:35 PM

Open

### 6. docker-compose.yml (Orchestration)

This file defines all the services (containers) and how they interact. We'll create three Flask app containers and three Nginx proxy containers.

docker-compose.yml (Docker Compose Configuration)

Jul 25, 2:35 PM

Open

### 7. How to Test on Your Linux Machine

1. **Install Docker and Docker Compose:** If you don't have them, follow the official Docker installation guide for your Linux distribution.
   * [Install Docker Engine](https://docs.docker.com/engine/install/)
   * [Install Docker Compose](https://docs.docker.com/compose/install/)
2. **Navigate to Project Directory:**  
   Bash  
   cd my-distributed-app
3. Build and Run the Containers:  
   This command will build the Docker images (if not already built) and start all the services defined in docker-compose.yml. The -d flag runs them in detached mode (in the background).  
   Bash  
   docker-compose up --build -d  
     
   You should see output indicating that containers are being created and started.
4. **Verify Containers are Running:**  
   Bash  
   docker-compose ps  
     
   You should see app1, app2, app3, nginx1, nginx2, nginx3 all in the Up state.
5. Test the Load Balancing and Stickiness:  
   Now, you can use curl to send requests. We'll simulate the ALB by hitting different Nginx instances (exposed on different ports on your host machine: 80, 8081, 8082).  
   **Test 1: Consistent Hashing for a Specific User ID (e.g., "Alice")**  
   Send multiple requests for X-User-ID: Alice to different Nginx instances. You should observe that the processed\_by\_node in the response is **always the same** for "Alice", regardless of which Nginx proxy (80, 8081, or 8082) you hit.  
   Bash  
   echo "--- Testing User Alice ---"  
   curl -H "X-User-ID: Alice" http://localhost:80/  
   curl -H "X-User-ID: Alice" http://localhost:8081/  
   curl -H "X-User-ID: Alice" http://localhost:8082/  
   curl -H "X-User-ID: Alice" http://localhost:80/  
     
   *Expected Output:* For all these Alice requests, the processed\_by\_node should consistently show node1, node2, or node3 (whichever node Nginx's consistent hash maps "Alice" to).  
   **Test 2: Consistent Hashing for Another User ID (e.g., "Bob")**  
   Now try with a different user ID. "Bob" should consistently map to a *different* (or potentially the same, but consistently so) node than "Alice".  
   Bash  
   echo "--- Testing User Bob ---"  
   curl -H "X-User-ID: Bob" http://localhost:80/  
   curl -H "X-User-ID: Bob" http://localhost:8081/  
   curl -H "X-User-ID: Bob" http://localhost:8082/  
   curl -H "X-User-ID: Bob" http://localhost:80/  
     
   *Expected Output:* For all these Bob requests, the processed\_by\_node should consistently show one specific node (e.g., if Alice mapped to node1, Bob might map to node2 or node3).  
   **Test 3: Missing Header**  
   Bash  
   echo "--- Testing Missing Header ---"  
   curl http://localhost:80/  
     
   *Expected Output:* You should get a 400 Bad Request error, as configured in Nginx.
6. View Logs (Optional):  
   To see the logs from all services (including the Flask app logs showing which node received the request):  
   Bash  
   docker-compose logs -f  
     
   Press Ctrl+C to exit the log stream.
7. Stop and Remove Containers:  
   When you're done testing, stop and remove all the containers and networks created by Docker Compose:  
   Bash  
   docker-compose down

This Docker setup provides a powerful and convenient way to test the distributed routing logic locally before deploying it to actual cloud instances.