Test Automation Engineer Coding Challenge

Welcome to the coding challenge for the Test Automation Engineer position.

This challenge consists of two main tests/suites designed to assess your ability to automate testing processes for Waku nodes, utilizing Docker containers and a mini test automation framework of your choice.

Requirements

- **Languages:** For creating your mini test automation framework use Python and Pytest please.
- **Output:** Ideally a Github PR or similar.

Test Suite 1: Basic Node Operation (easy)

Objective

Verify that a single Waku node can start, publish messages, and respond to API requests.

Steps

1. **Start a Waku Node:** Use Docker to run a Waku node with the provided command (use any ports/ip):

```
docker run -i -t \
  -p 21161:21161 \
  -p 21162:21162 \
  -p 21163:21163 \
  -p 21164:21164 \
  -p 21165:21165 \
  wakuorg/nwaku:v0.24.0 \
  --listen-address=0.0.0.0 \
  --rest=true \
```

```
    --rest-admin=true \
    --websocket-support=true \
    --log-level=TRACE \
    --rest-relay-cache-capacity=100 \
    --websocket-port=21163 \
    --rest-port=21161 \
    --tcp-port=21162 \
    --discv5-udp-port=21164 \
    --rest-address=0.0.0.0 \
    --nat=extip:172.18.111.226 \
    --peer-exchange=true \
    --discv5-discovery=true \
    --relay=true
```

2. **Verify Node Information:** Confirm that the node's debug information is accessible. Make a GET request to `/debug/v1/info`, and store the `enrUri` in a variable.
3. **Subscribe to a Topic:** Send a POST request to `/relay/v1/auto/subscriptions` in order to subscribe to a topic. Ex:

```
curl -X POST "http://127.0.0.1:21161/relay/v1/auto/subscriptions" -H "accep
t: text/plain" -H "content-type: application/json" -d '["/my-app/2/chatroom
-1/proto"]'
```

4. **Publish a Message:** Send a POST request to `/relay/v1/auto/messages` in order to publish a message. Ex:

```
curl -X POST "http://127.0.0.1:21161/relay/v1/auto/messages" -H "content-t
ype: application/json" -d '{"payload":"UmVsYXkgd29ya3MhIQ==","conten
tTopic":"/my-app/2/chatroom-1/proto","timestamp":0}'
```

5. **Confirm Message Publication:** Ensure the message was successfully published by making a GET request to `/relay/v1/auto/messages/%2Fmy-app%2F2%2Fchatroom-1%2Fproto` and validating the response.

Test Suite 2: Inter-Node Communication (advanced)

Objective

Establish communication between two Waku nodes and verify message transmission from one node to the other.

Steps

1. **Repeat Test Suite 1:** Follow the steps in Test Suite 1 to start and set up the first node `node1`.
2. **Start a Second Node `node2`:** Repeat the process for node1 with modified port mappings, external IP (`extip`) and the additional flag `--discv5-bootstrap-node` set to the `enrUri` of `node1`.
3. **Create a Docker Network:** Create a Docker network named `waku`. Ex:

   ```
   docker network create --driver bridge --subnet 172.18.0.0/16 --gateway 17
   2.18.0.1 waku
   ```

4. **Connect Nodes to the Network:** Attach both node containers to the `waku` network, assigning them specific IPs that match their external IPs (`extip`).
5. **Verify Autoconnection:** Confirm that the nodes have successfully connected by making a GET request to `/admin/v1/peers` and checking that the `node2` address can be seen. This takes some time, make a wait mechanism.
6. Subscribe `node2` to the topic just like you did with `node1`
7. **Message Transmission:** Publish a message from `node1` and verify that it is received by `node2`.

Notes:

- The shell commands provided (e.g., Docker, curl) are intended as examples/guides, and candidates should use whatever tools or methods they consider best to accomplish the task.

Additional Instructions

- **README:** Provide a README file explaining how to set up the project and run the tests locally. This should include instructions for installing any dependencies, configuring the environment, and executing the test suites.
- **Framework Usage:** It's recommended to utilize a test automation framework for this challenge. Your framework choice should facilitate structured and maintainable tests. Adhering to best practices is also important for a successful submission.
- **Structure:** Organize your tests in a manner that you find logical and efficient. Include assertions for all responses to ensure the expected outcomes.
- **Waku Documentation:** For more detailed information on how Waku works, including node operations and API specifications, please refer to the official documentation available at Waku Documentation. This resource provides valuable insights into the functionalities and capabilities of Waku nodes, which may assist in the development and troubleshooting of your test cases.
- **Completion**: We encourage you to submit your challenge even if it is not fully completed. It's important for us to observe your programming, testing, and problem-solving skills, which can be demonstrated through a partially completed challenge. Your approach to tackling the tasks and any documentation you provide about your process and decisions will be valuable for our assessment.