Weather Application

Originally from: https://github.com/eficode/weatherapp

This simple weather application shows forecast of the current weather at user's location.

API Key from Open Weather Map is needed for the application to work.

Application Usage with Docker

Requirements: Compatible Docker Host, please see Docker documentation.

Run following command in terminal:

1. Install Docker
2. Install docker-compose
3. Execute: `cd /path/to/this/repository`
4. For production: `APPID=API_KEY_HERE NODE_ENV=production docker-compose up --build backend frontend nginx`
5. For development: `APPID=API_KEY_HERE docker-compose up frontend backend`
6. There are environment variables available:
   - These can be set by adding them in front the compose command and seperating them by space.
   - `MAP_ENDPOINT` sets the base URL for Weather API endpoint. Optional.
   - `APPID` sets the Open Weather Map API Key. Mandatory.
7. Wait a few minutes and please access the service with browser at:
   - Production: http://localhost
   - Development: http://localhost:8000

Application Usage with Ansible

Requirements: Ansible, remote instance to run Ansible

Ansible is recommended when no ready host with docker and docker-compose exists. Currently it has only been tested on Ubuntu 22.04 LTS. It is only meant for production usage. Please run following commands in terminal:

1. Get Ansible first
2. Acquire eg. Amazon AWS EC2 Instance. Ubuntu LTS is a good choice.
   i. It should have TCP ports 80 and 9000 enabled.
   ii. In AWS this requires creating a new security group.
   iii. Recommended way is to generate SSH Key Pair in PEM format and load it with eg with Bash:
   - `eval $(ssh-agent) && ssh-add ssh-key-name-here.pem`
3. Create host configuration file eg. `weather.hosts`
   `[weather] host1 ansible_host=AWS_INSTANCE_IP_HERE ansible_ssh_user=ubuntu`
4. Check host is reachable: `ansible -i weather.hosts host1 -m ping`
5. Run setup script:
   `ansible-playbook -i weather.hosts --extra-vars "APPID=API_KEY_HERE" --tags setup,start playbook.yml`
6. Wait a moment and please access the service with browser at: `http://AWS_INSTANCE_IP_HERE`
7. Run shutdown script: `ansible-playbook -i /usr/local/etc/ansible/hosts --tags stop playbook.yml`

Application Usage via local execution (development)

Requirements: npm

This is not recommended way but the application can be started locally via npm. Run following commands in terminal.

1. Make sure you have npm
2. `cd /path/to/this/repository`
3. Run: `APPID=API_KEY_HERE bash scripts/local.sh`
4. Please access the service with browser at: http://localhost:8000

Application Demo at Amazon AWS

The application demo is available at: http://13.51.48.14/

Application Testing Instructions

Requirements: Equal to section "Application Usage with Docker"

The application can be tested by following commands in terminal:

1. `cd /path/to/this/repository`
2. `docker-compose up --build testing`
   i. The command will run backend, frontend and integration tests.

Services

The docker-compose file specifies several services:

1. frontend: This handles development server and building for production.
2. backend: This handles backend requests from frontend to Weather API servers.
3. mockapi: This simulates Open Weather MAP API for testing.
4. nginx: This provides production web server, and serves frontend assets.
5. testing: This contains testing scripts to run unit and integration tests.

Remarks to the reviewer

1. The unit tests for Weather component don't test all cases which would happen for a real-world version.
2. Integration tests to check that the container scripts, env. vars etc. do what they are supposed.
3. The integration test files sometimes due to firefox crashing. In real-world app it would be investigated.
4. There's not much verification of data from the backend opening door for XSS attacks.
5. For Backend tests, fetch() was supposed to be mocked but apparently the common libs are not compatible.
6. There is no access control for any of the backend services
7. There hasn't been any performance testing or profiling for frontend or backend services.
8. Not all browsers are tested.
9. The application could be fuzzed by sending anomaly data via HTTP requests. This could reveal issues.
10. The error message are not presented to the user when backend or API goes down.
11. The repository could have CI features from eg. CircleCI which executes some automated tests or builds.
12. There are some ESLint false positives, these have been marked.
13. I would not use bash for anything more complex, and not copy-paste the script.
14. There are no coverage maps or reports.
15. Most browsers won't permi location sharing without HTTPS.
16. The Mock API server has no tests other than the integration test.
17. The Dockerfiles and entry files have a workaround for issue where npm doesn't install global packages correctly.
18. Ports are static for now
19. XZ or brotli compression for logs

20. There could be NGINX reverse-proxy for more robust web server.
    i. Also HTTPS and certificates.
    ii. Serve backend or frontend via regexp, or even vhost.
21. Using memcached or redis for data storage would decrease request load.
22. There would be room for improvement on layout/aesthetic side
    i. Graphs using some sort of JS library
    ii. Colors for weather icons, and wind direction icons
    iii. Bootstrap themes
    iv. Animated backgrounds based on weather
23. W3C HTML validator does not like the tags React produces even though it shouldn't use custom tags.
24. It would be possible to do things like XHTML schema validation.
25. Outdated, deprecated and unused packages could be automatically checked.
26. Ideally the application would be available from docker registry or artifactory.
27. There are possibility of updating-breaking changes because not all packages are downloaded with version freeze.
28. Ideally there would be testing for different phones and eg. screenreaders.
29. Ideally it would also be tested for accessibility for users with disabilities.
30. Ideally ther woudl be compression and minimizatio
31. fetch-mock uses deprecated library querystring

Contact

Ari Timonen