End report

Implemented optional features:

- None

Instructions for examiner to test the system:

- Clone repository with 'git clone https://compse140.devops-gitlab.rd.tuni.fi/gdviir/devops.git'
- Navigate to the repository with 'cd devops'
- Run command 'docker compose up --build -d'
- Open 'localhost:8198' in browser
- Login with the credentials in 'login.txt' (username: vbox | password: 123)
- When the user enters credentials correctly, the web page is served, and the state is changed to 'RUNNING'. Now the user can interact with the buttons on the web page and the system functions as described in the API except the system does not ask for credentials again unless the browser is closed completely.
- Alternatively, the API can be tested by using curl commands
- To clean up, run command 'docker compose down --volumes'

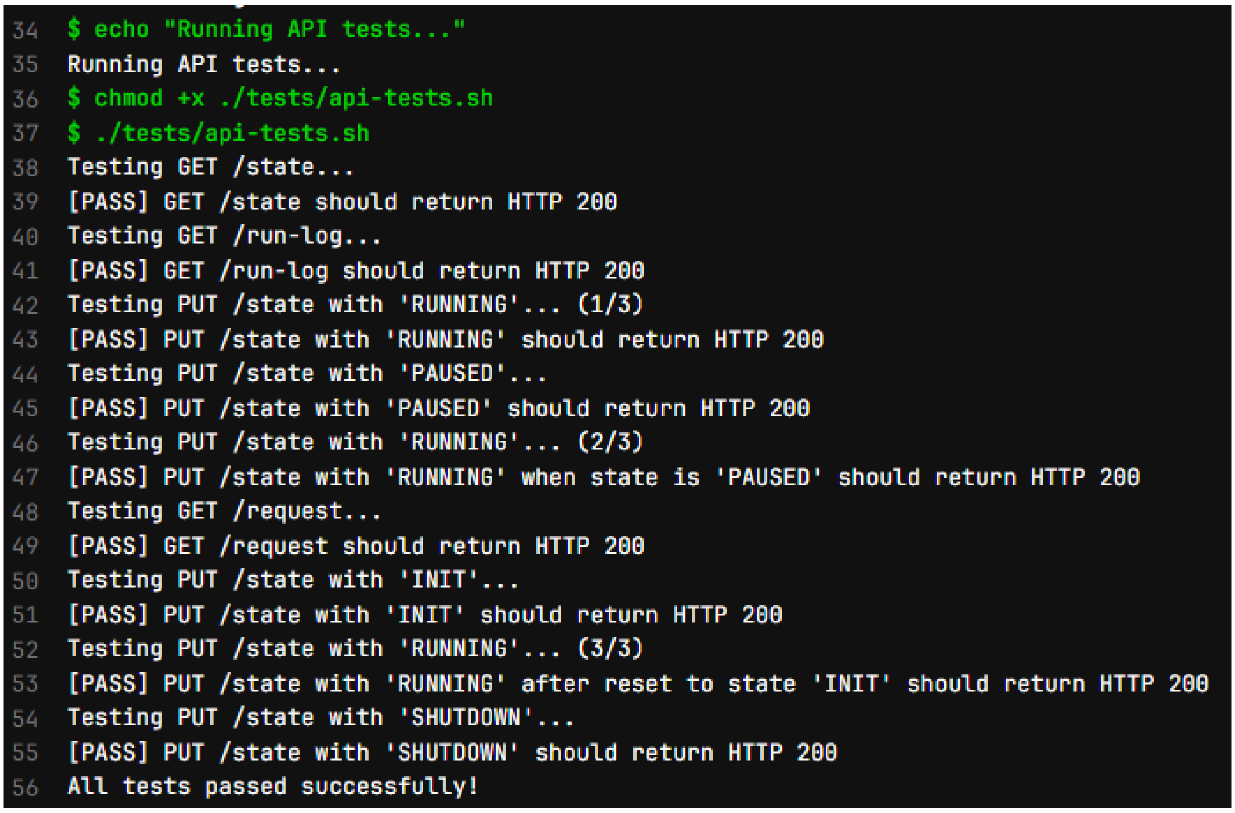
In developing the application, I used VirtualBox and had an Ubuntu virtual machine with default configuration. My host machine has an x86-64 architecture CPU. Docker version 27.3.1 and Docker Compose version v2.29.7.

Description of the CI/CD pipeline

- I always pushed my changes straight to the project branch. It is my only branch in GitLab.
- I used docker compose to build the project.
- I used a shell executor as my pipeline runner, and I have a shell script that has my test cases. The script is run in the test stage of the pipeline.
- I used docker compose to deploy the application i.e. the simple way.

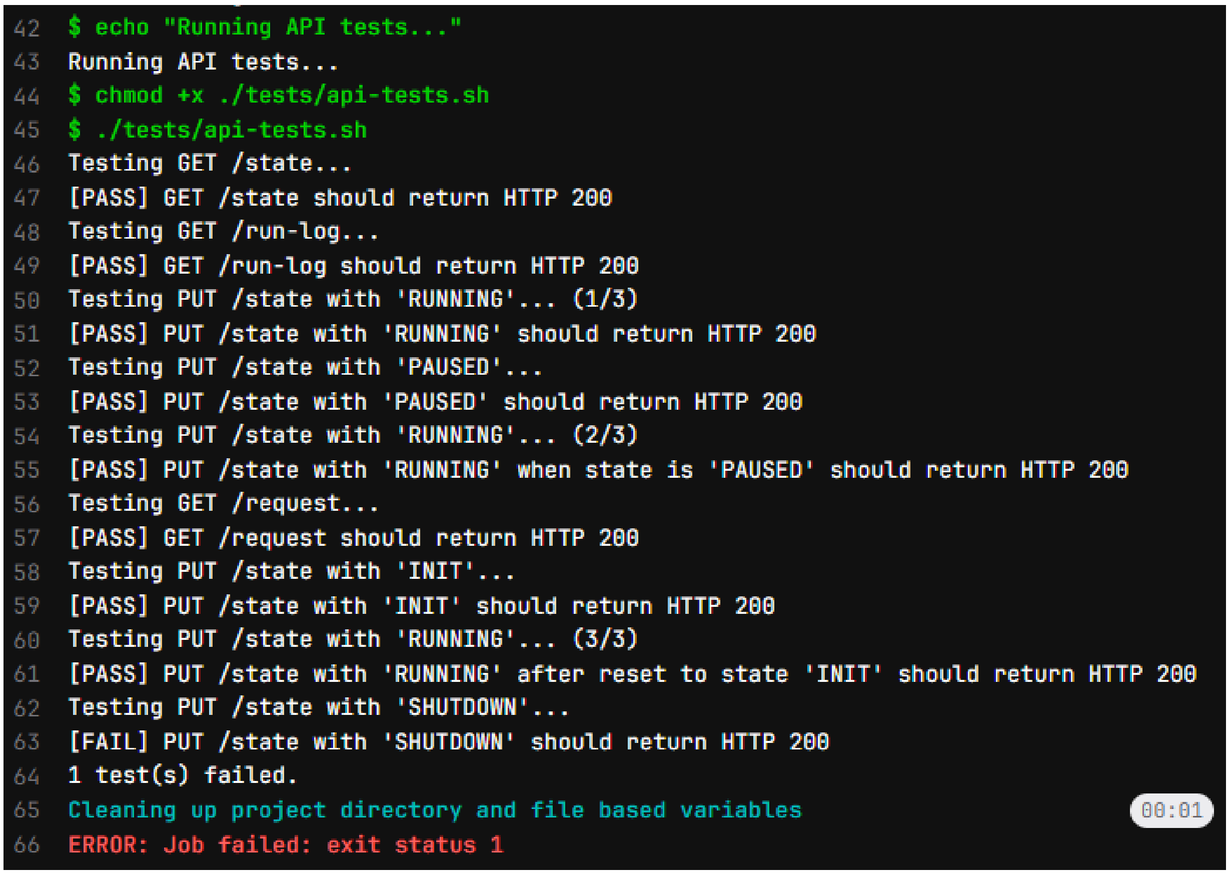
Example runs of the pipeline

All tests passing:



```
$ echo "Running API tests..."
Running API tests...
$ chmod +x ./tests/api-tests.sh
$ ./tests/api-tests.sh
Testing GET /state...
[PASS] GET /state should return HTTP 200
Testing GET /run-log...
[PASS] GET /run-log should return HTTP 200
Testing PUT /state with 'RUNNING'... (1/3)
[PASS] PUT /state with 'RUNNING' should return HTTP 200
Testing PUT /state with 'PAUSED'...
[PASS] PUT /state with 'PAUSED' should return HTTP 200
Testing PUT /state with 'RUNNING'... (2/3)
[PASS] PUT /state with 'RUNNING' when state is 'PAUSED' should return HTTP 200
Testing GET /request...
[PASS] GET /request should return HTTP 200
Testing PUT /state with 'INIT'...
[PASS] PUT /state with 'INIT' should return HTTP 200
Testing PUT /state with 'RUNNING'... (3/3)
[PASS] PUT /state with 'RUNNING' after reset to state 'INIT' should return HTTP 200
Testing PUT /state with 'SHUTDOWN'...
[PASS] PUT /state with 'SHUTDOWN' should return HTTP 200
All tests passed successfully!
```

One test failing:



```
$ echo "Running API tests..."
Running API tests...
$ chmod +x ./tests/api-tests.sh
$ ./tests/api-tests.sh
Testing GET /state...
[PASS] GET /state should return HTTP 200
Testing GET /run-log...
[PASS] GET /run-log should return HTTP 200
Testing PUT /state with 'RUNNING'... (1/3)
[PASS] PUT /state with 'RUNNING' should return HTTP 200
Testing PUT /state with 'PAUSED'...
[PASS] PUT /state with 'PAUSED' should return HTTP 200
Testing PUT /state with 'RUNNING'... (2/3)
[PASS] PUT /state with 'RUNNING' when state is 'PAUSED' should return HTTP 200
Testing GET /request...
[PASS] GET /request should return HTTP 200
Testing PUT /state with 'INIT'...
[PASS] PUT /state with 'INIT' should return HTTP 200
Testing PUT /state with 'RUNNING'... (3/3)
[PASS] PUT /state with 'RUNNING' after reset to state 'INIT' should return HTTP 200
Testing PUT /state with 'SHUTDOWN'...
[FAIL] PUT /state with 'SHUTDOWN' should return HTTP 200
1 test(s) failed.
Cleaning up project directory and file based variables                         00:01
ERROR: Job failed: exit status 1
```

Reflections

Main learnings:

- How to build a CI/CD pipeline from scratch
- See the value in creating such a pipeline
- Introduction to nginx
- Familiarize more with docker
- Plenty of new things in Linux, such as using grep, registering the pipeline runner etc.

Worst difficulties:

- Trying to get the user to log out using nginx basic authentication seems impossible and their documentation says that there is no logout feature. This is the only problem I am aware of in my system regarding the API specifications. I tried, for example, restarting the nginx web page container when the state was changed to 'INIT' and checking browser cookies, local storage, session storage, but found nothing there.
- Getting the response back from a successful PUT /state 'SHUTDOWN'. I had to thread the call to my shutdown function in service1, otherwise I did not get the response back. I'm not happy with the way it is implemented currently.
- Implementing a single source of truth for the state and run-log for all the service1 replicas. I used a volume in docker compose, and I think the implementation is very clean. The files for the state and run-log are removed when state is changed to 'SHUTDOWN'. The run-log or state files in the volume will not be cleared if the system is stopped some other way.
- At first, I did not register the gitlab-runner with sudo so it did not pick up any jobs in the pipeline. After lots of searching I found the issue.

Hours used

Around 20 hours.