Instructions for TA

I was unable to finish the project in a way that I would be happy with but leaving for exchange studies and other responsibilities in life made me too busy to do so.

Instructions to run the project on your machine:

Git clone the project

cd to the correct folder

run: “**docker-compose -f "amqp/docker-compose.yml" up -d --build**“

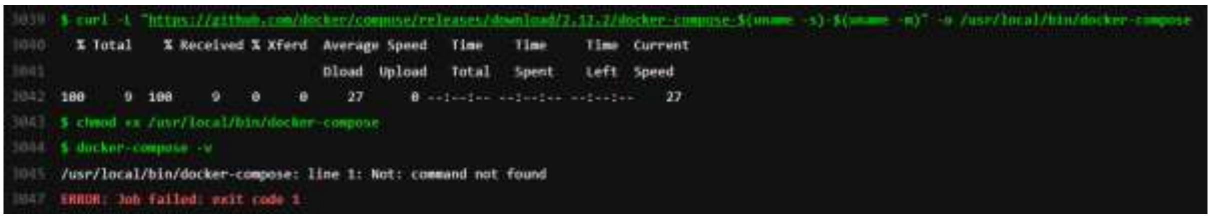
You can run API tests from the API container by running “**docker exec amqp-api-1 npm run test**”

Implemented optional features:
-Sonarqube static analysis

Description of the CI/CD pipeline

I never got this fully working with the final version of the amqp project. The issue was at first that I was unable to find a working way to install the correct version of docker-compose in the ci/cd pipeline, and when I got it installed, I couldn’t connect to the docker daemon. Tried many ways, but I was only able to make version 1.xx work, which doesn’t support health checks for example, that I use in the rest of script. Having the runners registered in privileged mode might had fixed the issue, but even though I attempted many times I couldn’t accomplish this.

For example, I tried the following method but couldn’t make docker-compose runnable:



```
$ curl -L "https://github.com/docker/compose/releases/download/2.12.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100     9  100     9    0     0     27      0 --:--:-- --:--:-- --:--:--    27
$ chmod +x /usr/local/bin/docker-compose
$ docker-compose -v
/usr/local/bin/docker-compose: line 1: Not: command not found
ERROR: Job failed: exit code 1
```

In version management each feature should be developed in their own branch. The branch could be named for example “feature/api-tests”. Then when development is done, it should be merged with main branch. That merge request could start the ci/cd pipeline that ensures that the code works, thus ensuring that the main branch is always a working version of the code.

For testing I used Jest and Supertest.

“Jest is a JavaScript testing framework designed to ensure correctness of any JavaScript codebase. It allows you to write tests with an approachable, familiar, and feature-rich API that gives you results quickly.” - https://jestjs.io/

“The motivation with this module is to provide a high-level abstraction for testing HTTP, while still allowing you to drop down to the lower-level API provided by superagent.” - https://www.npmjs.com/package/supertest

Jest and Supertest can be used together to make nice tests for APIs. Jest uses Supertest to make http requests to the API and then Jest can be used to evaluate the results and so on.

Cases:
Cases are described in detail in the test files. Jest and supertest are nice libraries, as the test code basically documents itself.

get /state, /messages and /run-log

These simple endpoints are tested the same. They should receive response with status 200 and content type text/html.

put /state

Put state tests each state change and then invalid input. Each happy case should return 200 status and sad case should return status 400.

Deployment and monitoring:

Locally the system is deployed using docker-compose up, exact command in the first chapter. While operating the system each service has their own logging done with console.log function. These logs can be accessed through docker with docker logs command. Rabbitmq container dashboard for monitoring can be accessed trough http://localhost:15672/.

Example runs of the pipeline

Failing example of the ci/cd pipeline:

```
[0KRunning with gitlab-runner 15.6.1 (133d7e76)[0;m
[0K  on uus e2b3WUtU[0;m
section_start:1672131521:prepare_executor
[0K[0K[36;1mPreparing the "docker" executor[0;m[0;m
[0KUsing Docker executor with image docker:stable ...[0;m
[0KStarting service docker:dind ...[0;m
[0KPulling docker image docker:dind ...[0;m
[0KUsing docker image
sha256:d736edfbfb0cb958492a1209140d3d1706596305a5d8a22b63c9fca65a51e578 for
docker:dind with digest
docker@sha256:0af3fdff5eb68de84dd4c7f6b0abe68f52683f49fa2bbc9eb027aefdac28e
842 ...[0;m
[0KWaiting for services to be up and running (timeout 30 seconds)...[0;m

[0;33m*** WARNING:[0;m Service runner-e2b3wutu-project-34-concurrent-0-
7e62cc7eec4baeba-docker-0 probably didn't start properly.

Health check error:
```

```
service "runner-e2b3wutu-project-34-concurrent-0-7e62cc7eec4baeba-docker-0-
wait-for-service" timeout

Health check container logs:


Service container logs:
2022-12-27T08:59:00.415894672Z Certificate request self-signature ok
2022-12-27T08:59:00.415922012Z subject=CN = docker:dind server
2022-12-27T08:59:00.425836654Z /certs/server/cert.pem: OK
2022-12-27T08:59:01.933300205Z Certificate request self-signature ok
2022-12-27T08:59:01.933329095Z subject=CN = docker:dind client
2022-12-27T08:59:01.943627537Z /certs/client/cert.pem: OK
2022-12-27T08:59:01.967660695Z ip: can't find device 'ip_tables'
2022-12-27T08:59:01.968193146Z modprobe: can't change directory to
'/lib/modules': No such file or directory
2022-12-27T08:59:01.969796668Z mount: permission denied (are you root?)

[0;33m*********[0;m

[0KPulling docker image docker:stable ...[0;m
[0KUsing docker image
sha256:b0757c55a1fdbb59c378fd34dde3e12bd25f68094dd69546cf5ca00ddbaa7a33 for
docker:stable with digest
docker@sha256:fd4d028713fd05a1fb896412805daed82c4a0cc84331d8dad00cb596d7ce3
e3a ...[0;m
section_end:1672131569:prepare_executor
[0Ksection_start:1672131569:prepare_script
[0K[0K[36;1mPreparing environment[0;m[0;m
Running on runner-e2b3wutu-project-34-concurrent-0 via 0bfcd6297bcd...
section_end:1672131571:prepare_script
[0Ksection_start:1672131571:get_sources
[0K[0K[36;1mGetting source from Git repository[0;m[0;m
[32;1mFetching changes with git depth set to 20...[0;m
Reinitialized existing Git repository in /builds/gitlab-instance-
8cadb184/asd/.git/
[32;1mChecking out 9d48af02 as project...[0;m
Removing api/

[32;1mSkipping Git submodules setup[0;m
section_end:1672131573:get_sources
[0Ksection_start:1672131573:step_script
[0K[0K[36;1mExecuting "step_script" stage of the job script[0;m[0;m
[0KUsing docker image
sha256:b0757c55a1fdbb59c378fd34dde3e12bd25f68094dd69546cf5ca00ddbaa7a33 for
docker:stable with digest
docker@sha256:fd4d028713fd05a1fb896412805daed82c4a0cc84331d8dad00cb596d7ce3
e3a ...[0;m
[32;1m$ apk update[0;m
fetch http://dl-
cdn.alpinelinux.org/alpine/v3.12/main/x86_64/APKINDEX.tar.gz
fetch http://dl-
cdn.alpinelinux.org/alpine/v3.12/community/x86_64/APKINDEX.tar.gz
v3.12.12-45-g8116a127ec [http://dl-cdn.alpinelinux.org/alpine/v3.12/main]
v3.12.12-46-g1df69604c4 [http://dl-
cdn.alpinelinux.org/alpine/v3.12/community]
OK: 12778 distinct packages available
[32;1m$ apk add nodejs npm[0;m
(1/7) Installing brotli-libs (1.0.9-r1)
(2/7) Installing c-ares (1.17.2-r0)
(3/7) Installing libgcc (9.3.0-r2)
```

```
(4/7) Installing nghttp2-libs (1.41.0-r0)
(5/7) Installing libstdc++ (9.3.0-r2)
(6/7) Installing nodejs (12.22.12-r0)
(7/7) Installing npm (12.22.12-r0)
Executing busybox-1.31.1-r19.trigger
OK: 69 MiB in 27 packages
[32;1m$ apk add docker docker-compose[0;m
(1/57) Installing libseccomp (2.4.4-r0)
(2/57) Installing runc (1.0.0_rc95-r0)
(3/57) Installing containerd (1.4.4-r0)
(4/57) Installing libmnl (1.0.4-r0)
(5/57) Installing libnftnl-libs (1.1.6-r0)
(6/57) Installing iptables (1.8.4-r2)
(7/57) Installing tini-static (0.19.0-r0)
(8/57) Installing device-mapper-libs (2.02.186-r1)
(9/57) Installing docker-engine (20.10.3-r0)
(10/57) Installing docker-cli (20.10.3-r0)
(11/57) Installing docker (20.10.3-r0)
Executing docker-20.10.3-r0.pre-install
(12/57) Installing libbz2 (1.0.8-r1)
(13/57) Installing expat (2.2.10-r4)
(14/57) Installing libffi (3.3-r2)
(15/57) Installing gdbm (1.13-r1)
(16/57) Installing xz-libs (5.2.5-r1)
(17/57) Installing readline (8.0.4-r0)
(18/57) Installing sqlite-libs (3.32.1-r1)
(19/57) Installing python3 (3.8.10-r0)
(20/57) Installing py3-ordered-set (4.0.1-r0)
(21/57) Installing py3-appdirs (1.4.4-r1)
(22/57) Installing py3-parsing (2.4.7-r0)
(23/57) Installing py3-six (1.15.0-r0)
(24/57) Installing py3-packaging (20.4-r0)
(25/57) Installing py3-setuptools (47.0.0-r0)
(26/57) Installing dockerpy-creds (0.4.0-r2)
(27/57) Installing py3-cparser (2.20-r0)
(28/57) Installing py3-cffi (1.14.0-r2)
(29/57) Installing py3-idna (2.9-r0)
(30/57) Installing py3-asn1crypto (1.3.0-r0)
(31/57) Installing py3-cryptography (2.9.2-r0)
(32/57) Installing py3-ipaddress (1.0.23-r1)
(33/57) Installing py3-chardet (3.0.4-r4)
(34/57) Installing py3-certifi (2020.4.5.1-r0)
(35/57) Installing py3-urllib3 (1.25.9-r0)
(36/57) Installing py3-requests (2.23.0-r0)
(37/57) Installing py3-websocket-client (0.57.0-r0)
(38/57) Installing docker-py (4.2.0-r0)
(39/57) Installing py3-cached-property (1.5.1-r1)
(40/57) Installing py3-distro (1.5.0-r1)
(41/57) Installing py3-dockerpty (0.4.1-r2)
(42/57) Installing py3-docopt (0.6.2-r5)
(43/57) Installing py3-more-itertools (8.3.0-r0)
(44/57) Installing py3-zipp (1.0.0-r0)
(45/57) Installing py3-importlib-metadata (1.6.0-r0)
(46/57) Installing py3-pyrsistent (0.16.0-r0)
(47/57) Installing py3-attrs (19.3.0-r1)
(48/57) Installing py3-jsonschema (3.2.0-r0)
(49/57) Installing py3-asn1 (0.4.7-r2)
(50/57) Installing py3-bcrypt (3.1.7-r2)
(51/57) Installing py3-pynacl (1.4.0-r0)
(52/57) Installing py3-paramiko (2.7.1-r0)
(53/57) Installing py3-pysocks (1.7.1-r1)
```

```
(54/57) Installing py3-texttable (1.6.2-r1)
(55/57) Installing yaml (0.2.4-r1)
(56/57) Installing py3-yaml (5.3.1-r1)
(57/57) Installing docker-compose (1.25.4-r3)
Executing busybox-1.31.1-r19.trigger
OK: 432 MiB in 84 packages
[32;1m$ cd api[0;m
/bin/sh: cd: line 124: can't cd to api: No such file or directory
section_end:1672131587:step_script
[0K[31;1mERROR: Job failed: exit code 2
[0;m
```

Successful run of the tests ran locally: (couldn't make these run in the pipeline as stated before)

```
❯ docker exec amqp-api-1 npm run test


> api@1.0.0 test

> jest --verbose --runInBand ./tests


  console.log

    Incoming state chage:  INIT  ->  PAUSED


      at log (server.js:48:13)


  console.log

    Begin putState


      at log (endpoints/state.js:178:13)


  console.log

    Pausing orig container


      at log (endpoints/state.js:187:17)


  console.log

    Written:  2022-12-27T09:13:36.888Z: INIT
    at log (server.js:28:17)

  console.log
    Orig paused

      at log (endpoints/state.js:190:50)

::ffff:127.0.0.1 - - [27/Dec/2022:09:13:36 +0000] "PUT /state HTTP/1.1" 200 6 "-" "-"
  console.log
    Written:  2022-12-27T09:13:36.990Z: PAUSED

      at log (server.js:28:17)

  console.log
    Incoming state chage:  PAUSED  ->  RUNNING

      at log (server.js:48:13)

  console.log
    Begin putState

      at log (endpoints/state.js:178:13)

  console.log
    Restaring orig

      at log (endpoints/state.js:200:17)

  console.log
    Container 6df7d09142d0f8b98fac1a07b48c36f58b68290fa18e84607c7b568381331ff0 restarted
successfully
```

```
      at ClientRequest.log (endpoints/state.js:136:25)

  console.log
    undefined

      at log (endpoints/state.js:203:26)

::ffff:127.0.0.1 - - [27/Dec/2022:09:13:47 +0000] "PUT /state HTTP/1.1" 200 7 "-" "-"
  console.log
    Written:  2022-12-27T09:13:47.784Z: RUNNING

      at log (server.js:28:17)

  console.log
    Incoming state chage:  RUNNING  ->  INIT

      at log (server.js:48:13)

  console.log
    Begin putState

      at log (endpoints/state.js:178:13)

  console.log
    Restarting all containers

      at log (endpoints/state.js:180:17)

  console.log
    Get containers done
    at IncomingMessage.log (endpoints/state.js:58:25)

  console.log
    Restart done

      at log (endpoints/state.js:184:23)

::ffff:127.0.0.1 - - [27/Dec/2022:09:13:47 +0000] "PUT /state HTTP/1.1" 200 7 "-" "-"
  console.log
    Written:  2022-12-27T09:13:47.805Z: INIT

      at log (server.js:28:17)

  console.log
    Written:  2022-12-27T09:13:47.805Z: RUNNING

      at log (server.js:28:17)

  console.log
    Incoming state chage:  RUNNING  ->  INVALID

      at log (server.js:48:13)

::ffff:127.0.0.1 - - [27/Dec/2022:09:13:47 +0000] "PUT /state HTTP/1.1" 400 51 "-" "-"
  console.log
    Incoming state chage:  RUNNING  ->  SHUTDOWN

      at log (server.js:48:13)

  console.log
  Begin putState

    at log (endpoints/state.js:178:13)

 console.log
  Shutting down all containers

    at log (endpoints/state.js:193:17)

 console.log
  Get containers done

    at IncomingMessage.log (endpoints/state.js:58:25)

 console.log
  All containers stopped

    at log (endpoints/state.js:197:23)

::ffff:127.0.0.1 - - [27/Dec/2022:09:13:47 +0000] "PUT /state HTTP/1.1" 200 8 "-" "-"
FAIL tests/putState.test.js (11.265 s)
 PUT /state
  ✓ Changes state to paused (69 ms)
  ✕ Changes state to running (10791 ms)
  ✓ Changes state to init (20 ms)
  ✓ Returns error if new state is invalid (7 ms)
  ✓ Changes state to shutdown (18 ms)

 ● PUT /state › Changes state to running
```

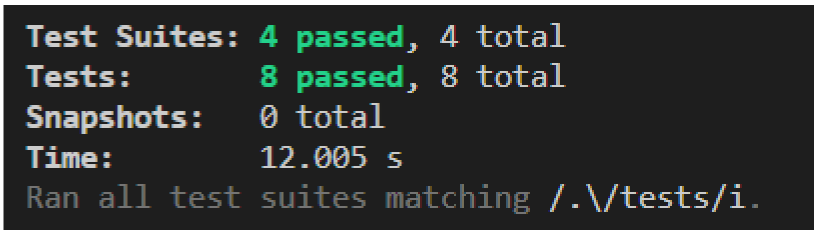
```
    Test functions cannot both take a 'done' callback and return something. Either use a 'done'
  callback, or return a promise.

    Returned value: Promise {}


      19 |     })
      20 |     jest.setTimeout(15000)
    > 21 |     it('Changes state to running', async (done) => {
         |     ^
      22 |         const newState = 'RUNNING'
      23 |         supertest(server)
      24 |             .put('/state')

      at it (tests/putState.test.js:21:5)
      at Object.describe (tests/putState.test.js:9:1)


  ● Cannot log after tests are done. Did you forget to wait for something async in your test?
    Attempted to log "Written:  2022-12-27T09:13:47.831Z: SHUTDOWN
    ".

      26 |             console.error(err)
      27 |         }
    > 28 |         console.log('Written: ', message)
         |                 ^
      29 |     })
      30 |     return
      31 | }

      at console.log (node_modules/@jest/console/build/CustomConsole.js:141:10)
      at log (server.js:28:17)
      at node_modules/graceful-fs/graceful-fs.js:61:14
  console.log
    Written:  2022-12-27T09:13:48.002Z: INIT

      at log (server.js:28:17)

::ffff:127.0.0.1 - - [27/Dec/2022:09:13:48 +0000] "GET /messages HTTP/1.1" 200 359 "-" "-"
PASS tests/runlog.test.js
  GET /run-log endpoint tests
    ✓ recieves status 200 and response is text (87 ms)

  console.log
    Written:  2022-12-27T09:13:48.244Z: INIT

      at log (server.js:28:17)

::ffff:127.0.0.1 - - [27/Dec/2022:09:13:48 +0000] "GET /messages HTTP/1.1" 200 359 "-" "-"
PASS tests/messages.test.js
  GET /messages endpoint tests
    ✓ recieves status 200 and response is text (21 ms)

::ffff:127.0.0.1 - - [27/Dec/2022:09:13:48 +0000] "GET /state HTTP/1.1" 200 4 "-" "-"
  console.log
    Written:  2022-12-27T09:13:48.423Z: INIT

      at log (server.js:28:17)

PASS tests/getState.test.js
  GET /state endpoint tests
    ✓ recieves status 200 and response is text (10 ms)
```

```
Test Suites: 1 failed, 3 passed, 4 total
Tests:       1 failed, 7 passed, 8 total
Snapshots:   0 total
Time:        11.896 s
Ran all test suites matching /.\/tests/i.

  ●  Cannot log after tests are done. Did you forget to wait for something async in your test?
    Attempted to log "Container bfa4de6e328feda95e407749d07e13c3e771510c261a6230e158fa11cd503209 restarted successfully".

      134 |       const callback = (res) => {
      135 |         if (res.statusCode === 204) {
    > 136 |           console.log(`Container ${containerId} restarted successfully`)
          |                   ^
      137 |           resolve([true, undefined])
      138 |         } else {
      139 |           console.error(

      at console.log (node_modules/@jest/console/build/CustomConsole.js:141:10)
      at ClientRequest.log (endpoints/state.js:136:25)

Jest did not exit one second after the test run has completed.

This usually means that there are asynchronous operations that weren't stopped in your tests. Consider running Jest with `--detectOpenHandles` to troubleshoot this issue.

  ●  Cannot log after tests are done. Did you forget to wait for something async in your test?
    Attempted to log "Container 6df7d09142d0f8b98fac1a07b48c36f58b68290fa18e84607c7b568381331ff0 restarted successfully".

      134 |       const callback = (res) => {
```

```
      135 |           if (res.statusCode === 204) {
    > 136 |               console.log(`Container ${containerId} restarted successfully`)
          |                   ^
      137 |               resolve([true, undefined])
      138 |           } else {
      139 |               console.error(

      at console.log (node_modules/@jest/console/build/CustomConsole.js:141:10)
      at ClientRequest.log (endpoints/state.js:136:25)



  ● Cannot log after tests are done. Did you forget to wait for something async in your test?
    Attempted to log "Container
b6aadc356659a81280f844965ce5670721a23488a06cc8d48cad67d19562bf2e restarted
successfully".

      134 |       const callback = (res) => {
      135 |           if (res.statusCode === 204) {
    > 136 |               console.log(`Container ${containerId} restarted successfully`)
          |                   ^
      137 |               resolve([true, undefined])
      138 |           } else {
      139 |               console.error(

      at console.log (node_modules/@jest/console/build/CustomConsole.js:141:10)
      at ClientRequest.log (endpoints/state.js:136:25)



  ● Cannot log after tests are done. Did you forget to wait for something async in your test?
    Attempted to log "Container
d94b5a6cdfa148ddd2c03364847aee1dd25c2ac1286dd0d4a14f56b6b0535864 restarted
successfully".
```

```
  134 |       const callback = (res) => {
  135 |         if (res.statusCode === 204) {
> 136 |           console.log(`Container ${containerId} restarted successfully`)
      |                   ^
  137 |           resolve([true, undefined])
  138 |         } else {
  139 |           console.error(

  at console.log (node_modules/@jest/console/build/CustomConsole.js:141:10)
  at ClientRequest.log (endpoints/state.js:136:25)
```

There are some minor issues with console.logs.



```
Test Suites: 4 passed, 4 total
Tests:       8 passed, 8 total
Snapshots:   0 total
Time:        12.005 s
Ran all test suites matching /.\/tests/i.
```

Reflections

Main learnings where with hosting own gitlab and gitlab runners as well as more docker-compose and unix sockets. It was interesting to learn the basics of running a platform for software development. Using unix sockets to control the containers was also new to me.

Some things that should had been done better:

-More use of feature branches when developing. I was not motivated to do this from the start as the single developer on the project, but this would make for a much cleaner version history.

-Maybe another way of running the ci/cd tests could had been implemented to completely go around using docker-compose inside the ci/cd pipeline. For example, there could be another test container that runs the tests outside the docker-compose cluster of containers.

-Running the gitlab-runner in privileged mode

-Other base images could be used to optimize performance. Ubuntu is quite large image and has lots of features not really needed in the pipeline. The upside and reason I used ubuntu is that at least it works

In the end I just didn't have enough time to really learn all the things I needed to make the whole pipeline run smoothly.

Overall, this project has taught me a lot about what practical work is in DevOps and has made me more interested in the field. I am disappointed at how hard the project was and how I was unable to finish it in a way that would make me satisfied.

I used around 60-80 hours on the project, so hopefully I could at least pass the course after all this work ☹