

# KangasmakiProject End report

Patric Kangasmäki – K428678 – [Patric.kangasmaki@tuni.fi](mailto:Patric.kangasmaki@tuni.fi)

Niclas Kangasmäki – 425572 – [Niclas.kangasmaki@tuni.fi](mailto:Niclas.kangasmaki@tuni.fi)

## Description of the CI/CD pipeline.

Our CI/CD pipeline is created with GitHub actions. The pipeline will automatically run the docker-file and tests created for the project. Our final pipelines are in the repository:

<https://github.com/NiclasKa/KangasmakiProject>

## GitHub actions (Extra work since we used GitHub instead of GitLab)

GitHub actions is a CI/CD pipeline, where you can create workflows that automatically execute and give feed back on the wanted pieces. It is often used to first build, then test and finally deploy applications. In our application, we have . Docker-workflow is used to build and compose our docker-files. We also had a work-flow for automated nodeJS testing with Jest but we got an issue where the docker wouldn't run in the background and our application couldn't retrieve data from the API in the workflow since it wasn't possible to connect to containers and connecting to localhost resulted in an error. In our application, the containers connect to each other so it's a must to have them running. For that reason, the tests would always fail and we had to turn to manual testing locally in the localhost.

The usage is really simple. You can create your custom workflows from actions tab in GitHub web, and you want, there are also ready-to-use foundations for different workflows for many different languages. Using GitHub actions can help you with the repetitive tasks, where you would normally have to do many different steps to accomplish your goals. Workflows also gives you accurate feedback and can be used to validate whether your pull-request works or not. Actions can also be combined with each other.

Our docker-workflow always runs when you either:

push: branches: [main] or

pull\_request: branches [main]

(Although we always used different branch, which we then merged to main branch, so we could have left out the main-push action.)

## Main learnings and worst difficulties

*Learnings:* Deeper understanding of the Docker and RabbitMQ. CI/CL pipelines were new for both of us, so we got our hands on that as well. This project taught us how to use GitHub in a more advanced way (versions, actions)

*Difficulties:* Using Docker and Pipelines was unknown to both of us before this course, so we had some difficulties with the new technologies. Our biggest difficulty was making http request between the containers in our docker-compose file. We got a lot of 'connection-refused'-error message and since it is not the clearest one, we had problems troubleshooting it. Our first idea was to enable CORS, but it did not help us. After some time and refactoring our files, we found out we were trying to connect to localhost, instead of the correct URI.

Another difficulty we had was with troubleshooting and testing. Since the server starts slowly and waiting for containers and RabbitMQ to start takes a lot of time, even small changes were time consuming.

We got our tests working during the pipeline, but we could not manage to make the test calls between different containers work. We decide to remove the automated tests from the project, and now they should be run manually.

## Our roles in the project

We built most of this project together. We had separated small tasks for each, but the biggest changes and features were mainly done together. Since the COVID situation does not ban meetings of 2, we did a lot of the coding in the same location helping each other. This way we could really block out any soloing and mysterious things from happening and we stayed updated all the way through the project. The rest structure of the api was assigned to Patric and RabbitMQ-messaging to Niclas, although we still helped each other on both tasks. Troubleshooting was mostly made by Niclas and the report was mostly written by Patric.

## Amount hours used in the project

We used approximately 45 hours together (90 hours in total) on this project. A lot of the time went into researching, planning, and troubleshooting.

## Analysis

### Description of the project

#### Docker-container

The application is deployed in Docker containers. Every service has its own docker file, and they are composed together in Docker Compose. The docker should be set up running using docker-compose up – build –d. D command makes it run on background, so the tests can be still run.

Some of the containers use volumes, so that the files can be accessed by other containers.

## RabbitMQ message queue

RabbitMQ is used as the message broker. It is used to handle the messages and to pass them forward. In our application, the broker sends messages every 3 seconds and they get passed to the reader which writes them for our API to use.

## Testing

Testing is done with the testing framework Jest. Supertest is used to test the API. Testing is done manually at the API folder by running 'npm run test'. Test tests cover GET and PUT requests that were used in the application.

## Troubleshoot (optional feature)

The troubleshoot is running at localhost:8082. There you can find some relevant information about the system, such as server start time, status of different services and the requests user has made to the servers. The requests user makes to the troubleshooting service has been excluded from the list.

## What could have been done differently

We used way too much trial and error. We should have gathered deeper understanding of the technologies used before trying too hard and spending too much time googling cryptic problems that we could not understand.

Also, we should have worked on a one feature at a time in a clear order. One thing that caused a lot of problems was that we worked on features that depended on other features (which were broken) and that gave us hard time troubleshooting, since we tried fixing the new one instead of going deeper in the application.

## Run logs of successful and failing tests.

Successful test run:

```
npm run test
```

```
> api@1.0.0 test /mnt/c/school/git/DevOps/KangasmakiProject/api
```

```
> jest
```

```
PASS tests/api_test.test.js
```

```
✓ get messages (79 ms)
```

✓ get state (25 ms)

✓ put state (50 ms)

✓ get runlog (8 ms)

GET /state 200 7.286 ms - 19

PUT /state 200 39.779 ms - 18

GET /state 200 0.386 ms - 18

GET /run-log 200 0.588 ms - 156

Test Suites: 1 passed, 1 total

Tests: 4 passed, 4 total

Snapshots: 0 total

Time: 6.574 s

Ran all test suites.

Failing test run:

Run npm ci && npm run test

added 576 packages in 7.174s

> api@1.0.0 test /home/runner/work/KangasmakiProject/KangasmakiProject/API

> jest

FAIL tests/api\_test.test.js

✗ get messages (79 ms)

✓ get state (11 ms)

console.error

✓ put state (29 ms)

Error: Error: getaddrinfo EAI\_AGAIN httpserv

✓ get runlog (4 ms)

● get messages

Network Error

Test Suites: 1 failed, 1 total

GET /state 200 0.255 ms - 18

Tests: 1 failed, 3 passed, 4 total

GET /run-log 200 0.373 ms - 148

Snapshots: 0 total

Time: 1.729 s

Ran all test suites.

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.

Error: Process completed with exit code 1.

## Version history:

*Note: this is not an accurate representation of the workload from each member since most of the coding was done together with one computer with both of us dropping ideas.*

Add files via upload	pkangasmaki	22 minutes ago
Delete End-report.docx	pkangasmaki	22 minutes ago
End report	Niclas.Kangasmaki	4 hours ago
Merge pull request #10 from NiclasKa/troubleshoot_ready	NiclasKa	5 hours ago
troubleshoot_ready origin/troubleshoot_ready finished troubleshoot	Niclas.Kangasmaki	5 hours ago
Add files via upload	pkangasmaki	8 hours ago
Update docker-image.yml	NiclasKa	8 hours ago
Update README.md	NiclasKa	8 hours ago
Merge pull request #9 from NiclasKa/fixingTest	NiclasKa	8 hours ago
Update docker-image.yml	NiclasKa	8 hours ago
fixingTest origin/fixingTest Use localhost in tests	Niclas.Kangasmaki	8 hours ago
Update docker-image.yml	NiclasKa	8 hours ago
Merge pull request #8 from NiclasKa/testFix	NiclasKa	8 hours ago
testFix origin/testFix fixed async	Niclas.Kangasmaki	8 hours ago
fixing	Niclas.Kangasmaki	9 hours ago
Merge branch 'main' into testFix	Niclas.Kangasmaki	9 hours ago
fixed httpserv and api test	Niclas.Kangasmaki	9 hours ago
Update docker-image.yml	NiclasKa	9 hours ago
Update docker-image.yml	NiclasKa	9 hours ago
Merge pull request #7 from NiclasKa/troubleshoot	NiclasKa	9 hours ago
troubleshoot origin/troubleshoot troubleshoot	Niclas.Kangasmaki	9 hours ago
Create node.js.yml	pkangasmaki	9 hours ago
Merge pull request #6 from NiclasKa/httpsevDisconnect	NiclasKa	11 hours ago
httpsevDisconnect origin/httpsevDisconnect Httpserv checks if the state is shutdown	Niclas.Kangasmaki	11 hours ago
Merge pull request #5 from NiclasKa/fixHttpRequest	pkangasmaki	11 hours ago
origin/fixHttpRequest err connection refused fixed	Patric Kangasmaki	11 hours ago
Merge pull request #4 from NiclasKa/httpservReforaged	NiclasKa	12 hours ago
httpservReforaged origin/httpservReforaged Reforaged obse, imed and orig	Niclas.Kangasmaki	12 hours ago
Reforaged httpserv	Niclas.Kangasmaki	14 hours ago
Merge pull request #3 from NiclasKa/testsAndGetMessages	NiclasKa	1 day ago
testsAndGetMessages origin/testsAndGetMessages More tests	Niclas.Kangasmaki	1 day ago
Merge pull request #1 from NiclasKa/patric	NiclasKa	1 day ago
Merge pull request #2 from NiclasKa/testsAndGetMessages	pkangasmaki	1 day ago

	Merge pull request #2 from NiclasKa/testsAndGetMessages	pkangasmaki	1 day ago
	tests and get messages	Niclas.Kangasmaki	1 day ago
	origin/patric branch test	Patric Kangasmäki	13 days ago
	Hotfix to merging problems	Patric Kangasmäki	13 days ago
	quickfix	Patric Kangasmäki	13 days ago
	read me test	Patric Kangasmäki	13 days ago
	Commit test	Patric Kangasmäki	13 days ago
	Delete docker-image.yml	NiclasKa	14 days ago
	remote test new	Patric Kangasmäki	17 days ago
	setup test4	Patric Kangasmäki	17 days ago
	Merge branch 'main' of https://github.com/NiclasKa/KangasmakiProject into main	Patric Kangasmäki	17 days ago
	setup test3	Patric Kangasmäki	17 days ago
	readme	Niclas.Kangasmaki	17 days ago
	setup test2	Patric Kangasmäki	17 days ago
	Test 7	Niclas.Kangasmaki	17 days ago
	test 5 and 6	Niclas.Kangasmaki	17 days ago
	ads	Niclas.Kangasmaki	17 days ago
	Merge branch 'main' of https://github.com/NiclasKa/KangasmakiProject into main	Niclas.Kangasmaki	17 days ago
	test 4	Niclas.Kangasmaki	17 days ago
	Merge branch 'main' of https://github.com/NiclasKa/KangasmakiProject into main	Patric Kangasmäki	17 days ago
	test commit	Patric Kangasmäki	17 days ago
	Test 3	Niclas.Kangasmaki	17 days ago
	Test commit 2	Niclas.Kangasmaki	17 days ago
	Merge branch 'main' of https://github.com/NiclasKa/KangasmakiProject into main	Niclas.Kangasmaki	17 days ago
	test commit	Niclas.Kangasmaki	17 days ago
	Create workflow	NiclasKa	17 days ago
	Initial commit	Niclas.Kangasmaki	17 days ago
	Initial commit	NiclasKa	17 days ago
	Update README.md	pkangasmaki	21 days ago
	Create docker-image.yml	pkangasmaki	21 days ago
	RabbitMQ files for project	Patric Kangasmäki	21 days ago
	Initial commit	pkangasmaki	21 days ago

*Version history showing our commits from 21 days ago when we started the project.*