

## 1. Instructions for the teaching assistant

Implemented optional features.

- ✓ Implemented a static analysis step in the pipeline by using SonarQube.
- ✓ Implemented *GET /mqstatistic* endpoint.

Instructions for examiner to test the system.

### 1. To run the system's basic requirements,

- Clone the project using the following command.

```
git clone -b project https://course-gitlab.tuni.fi/compse140-fall2023/fnshja.git
```

- Change directory to the project.

```
cd fnshja
```

- Build the system using the following command.

```
docker-compose build --no-cache
```

- Run the system using the following command.

```
docker-compose up -d
```

### 2. Test the system's API endpoints.

Wait for a bit until all the services become ready (Approximately 20-25 seconds).

*Note: - It is assumed that **PUT /state** endpoint is not called to initialize the service. The service will automatically start from **INIT** state and automatically switch to **RUINNING** state without the need of the **PUT /state** API call with **"INIT"***

- Use curl/Postman to test the system

- `curl localhost:8083/state -X PUT -d "PAUSED" -H "Content-Type: text/plain" -H "Accept: text/plain"`

- `curl localhost:8083/state -X PUT -d "RUNNING" -H "Content-Type: text/plain" -H "Accept: text/plain"`
- `curl localhost:8083/state -X PUT -d "INIT" -H "Content-Type: text/plain" -H "Accept: text/plain"`
- `curl localhost:8083/state -X PUT -d "SHUTDOWN" -H "Content-Type: text/plain" -H "Accept: text/plain"`
- `curl localhost:8083/state -X GET -H "Content-Type: text/plain" -H "Accept: text/plain"`
- `curl localhost:8083/messages -X GET -H "Content-Type: text/plain" -H "Accept: text/plain"`
- `curl localhost:8083/run-log -X GET -H "Content-Type: text/plain" -H "Accept: text/plain"`
- `curl localhost:8083/mqstatistic -X GET -H "Content-Type: application/json" -H "Accept: application/json"`

### 3. Test SonarQube Integration.

- Run SonarQube docker container using the below command.

```
docker run -d --name sonarqube -p 9000:9000 -p 9092:9092
sonarqube
```

- Login to the SonarQube by using default admin/admin credentials and generate a new user token by navigating to User > My Account > Security (<http://localhost/account/security>).

The screenshot shows the SonarQube web interface at `localhost:9000/account/security`. The user is logged in as 'Administrator'. The 'Security' tab is active. Under the 'Tokens' section, there is a 'Generate Tokens' form with fields for 'Name', 'Type', and 'Expires In'. Below this is a table of existing tokens. The table has columns: Name, Type, Project, Last use, Created, Expiration, and Actions. One token is listed: 'token-1', 'User', '1 hour ago', 'January 30, 2024', 'February 29, 2024', and a 'Revoke' button. Below the table is a section to 'Enter a new password' with fields for 'Old Password', 'New Password', and 'Confirm Password', and an 'Update' button.

Name	Type	Project	Last use	Created	Expiration	Actions
token-1	User		1 hour ago	January 30, 2024	February 29, 2024	Revoke

- Update the `-Dsonar.login= <TOKEN>` value with the previously generated token in `.gitlab-ci.yml`

The screenshot shows a GitLab CI/CD pipeline configuration file named `.gitlab-ci.yml`. The file is 561 B in size. The configuration includes a job named `test` with a script that runs `cd tests && pwd && go test -v ./... && cd ..`. The `sonar` section of the configuration is highlighted in yellow, showing the `-Dsonar.login` value updated with the token `squ_52c349d75d62086aeac4e1ae0e1b3a2a3b27f376`.

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19 : cd tests && pwd && go test -v ./... && cd ..
20
21
22
23
24 -Dsonar.projectKey=api-gateway -Dsonar.sources=. -Dsonar.host.url=http://localhost:9000 -Dsonar.login=squ_52c349d75d62086aeac4e1ae0e1b3a2a3b27f376
25
26
27
28
29
30
31
```

- Install Sonar Scanner in your Linux machine using following commands and create a symbolic link

```
wget https://binaries.sonarsource.com/Distribution/sonar-scanner-cli/sonar-scanner-cli-4.8.0.2856-linux.zip
```

```

unzip sonar-scanner-cli-4.8.0.2856-linux.zip

sudo mv sonar-scanner-4.8.0.2856-linux /opt/sonar-scanner

ln -s /opt/sonar-scanner/bin/sonar-scanner /usr/local/bin/sonar-scanner

```

## 2. Description of the CI/CD pipeline

- **Version Control and Branching:**

Git was used as Version Control System (VCS) and Gitlab as the centralized VCS platform. One repository was used to build, test, and deploy the system more efficiently with Gitlab CI and Other remote git repository was used to keep the final code. Created “project” branch in both repositories from the “exercise2” branch and used “project” branch to do changes during the implementations.

```

sharada-107454@107454-001LB: /media/sharada-107454/ADL/Personal/DevOps/Project/fnshja$ git remote -v
origin  https://course-gitlab.tuni.fi/compse140-fall2023/fnshja.git (fetch)
origin  https://course-gitlab.tuni.fi/compse140-fall2023/fnshja.git (push)
origin-ci https://sharada_jayaweera:glpat-3EXtm3BF1dZs9La5xa-i@compse140.devops-gitlab.rd.tuni.fi/sharada_jayaweera/sharada_jayaweera_private_project.git (fetch)
origin-ci https://sharada_jayaweera:glpat-3EXtm3BF1dZs9La5xa-i@compse140.devops-gitlab.rd.tuni.fi/sharada_jayaweera/sharada_jayaweera_private_project.git (push)
sharada-107454@107454-001LB: /media/sharada-107454/ADL/Personal/DevOps/Project/fnshja$

```

- **Building tools**

Used Go (Golang) and Java (Spring Boot) as the programming languages and frameworks in the project. For Golang “build” build tool was used and for Java Spring Boot, Maven was used.

- **Testing; tools and test cases**

Testing was mainly done on the api-gateway service.

**Test framework:** - testing (Golang)

## Test cases

- Test GET /messages endpoint.
  - Expected Response Code = 200
  - Expected Content-Type = "text/plain"
- Test PUT /state endpoint for valid state values ("INIT", "PAUSED", "RUNNING", "SHUTDOWN")
  - Expected Response Code = 200
  - Expected Content-Type = "text/plain"
  - Expected Response = "Successfully Updated State"
- Test PUT /state endpoint for invalid state values.
  - Expected Response Code = 400
  - Expected Content-Type = "text/plain"
  - Expected Response = "Invalid State Value"
- Test GET /state endpoint.
  - Expected Response Code = 200
  - Expected Content-Type = "text/plain"
- Test GET /run-log endpoint.
  - Expected Response Code = 200
  - Expected Content-Type = "text/plain"
- Test GET /mqstatistic endpoint.
  - Expected Response Code = 200
  - Expected Content-Type = "application/json"

## • Packing

Packaging done with docker.

## • Deployment

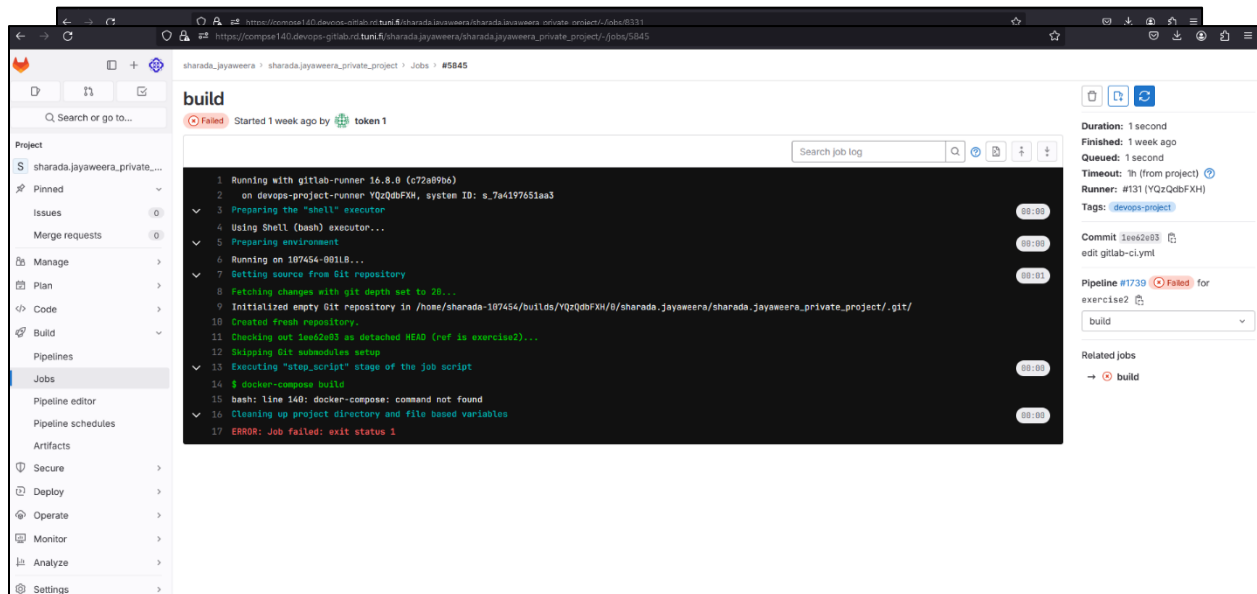
Local deployment done with docker-compose using ***docker-compose up -d*** command.

## • Operating; monitoring

Did not implement

### 3. Example runs of the pipeline

#### ➤ Successful Build Stage



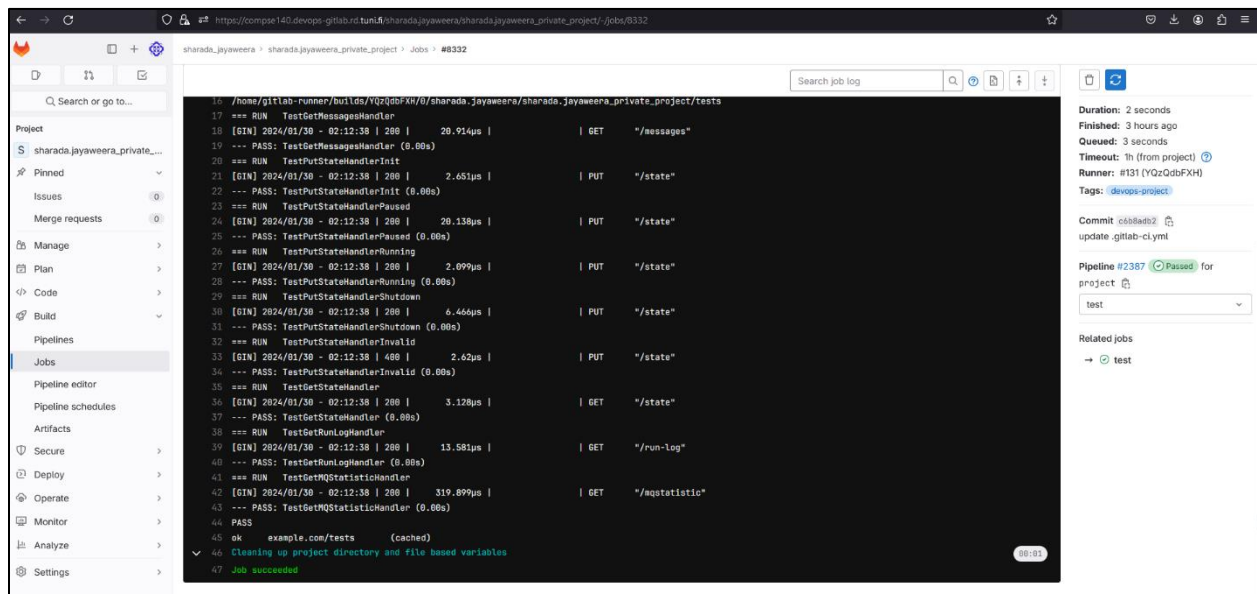
The screenshot shows a web interface for a CI/CD pipeline. The left sidebar contains a navigation menu with options like Project, Pinned, Issues, Merge requests, Manage, Plan, Code, Build, Pipelines, Jobs, Pipeline editor, Pipeline schedules, Artifacts, Secure, Deploy, Operate, Monitor, Analyze, and Settings. The main area displays the 'build' job, which is marked as 'Failed' with a red circle icon. The job log shows the following steps:

```
1 Running with gitlab-runner 16.8.0 (c72a89b6)
2 on devops-project-runner YQzQdbFXH, system ID: s_7a4197651aa3
3 #Preparing the "shell" executor...
4 Using Shell (bash) executor...
5 #Preparing environment
6 Running on 107454-001L8...
7 Getting source from Git repository
8 Fetching changes with git depth set to 30...
9 Initialized empty Git repository in /home/sharada-107454/builds/YQzQdbFXH/0/sharada.jayaweera/sharada.jayaweera_private_project/.git/
10 Creating fresh repository...
11 Checking out 1aa2d093 as detached HEAD (ref is exercise2)...
12 Skipping Git submodules setup
13 Executing "step_script" stage of the job script
14 $ docker-compose build
15 bash: line 140: docker-compose: command not found
16 Cleaning up project directory and file based variables
17 ERROR: Job failed: exit status 1
```

On the right side, there is a summary of the job: Duration: 1 second, Finished: 1 week ago, Queued: 1 second, Timeout: 1h (from project), Runner: #131 (YQzQdbFXH), Tags: devops-project, Commit: 1aa2d093, edit gitlab-ci.yml, Pipeline #1730 (Failed) for exercise2, build, and Related jobs: build.

#### ➤ Failed Build Stage

#### ➤ Passed Test Cases Scenario



The screenshot shows a web interface for a CI/CD pipeline. The left sidebar contains a navigation menu with options like Project, Pinned, Issues, Merge requests, Manage, Plan, Code, Build, Pipelines, Jobs, Pipeline editor, Pipeline schedules, Artifacts, Secure, Deploy, Operate, Monitor, Analyze, and Settings. The main area displays the 'test' job, which is marked as 'Passed' with a green circle icon. The job log shows the following steps:

```
15 /home/gitlab-runner/builds/YQzQdbFXH/0/sharada.jayaweera/sharada.jayaweera_private_project/tests
16 == RUN TestGetMessagesHandler
17 [GIN] 2024/01/30 - 02:12:38 | 200 | 20.914µs | GET | "/messages"
18 --- PASS: TestGetMessagesHandler (0.00s)
19 == RUN TestPutStateHandlerInit
20 [GIN] 2024/01/30 - 02:12:38 | 200 | 2.651µs | PUT | "/state"
21 --- PASS: TestPutStateHandlerInit (0.00s)
22 == RUN TestPutStateHandlerPaused
23 [GIN] 2024/01/30 - 02:12:38 | 200 | 20.138µs | PUT | "/state"
24 --- PASS: TestPutStateHandlerPaused (0.00s)
25 == RUN TestPutStateHandlerRunning
26 [GIN] 2024/01/30 - 02:12:38 | 200 | 2.099µs | PUT | "/state"
27 --- PASS: TestPutStateHandlerRunning (0.00s)
28 == RUN TestPutStateHandlerShutdown
29 [GIN] 2024/01/30 - 02:12:38 | 200 | 4.466µs | PUT | "/state"
30 --- PASS: TestPutStateHandlerShutdown (0.00s)
31 == RUN TestPutStateHandlerInvalid
32 [GIN] 2024/01/30 - 02:12:38 | 400 | 2.62µs | PUT | "/state"
33 --- PASS: TestPutStateHandlerInvalid (0.00s)
34 == RUN TestGetStateHandler
35 [GIN] 2024/01/30 - 02:12:38 | 200 | 3.128µs | GET | "/state"
36 --- PASS: TestGetStateHandler (0.00s)
37 == RUN TestGetRunLogHandler
38 [GIN] 2024/01/30 - 02:12:38 | 200 | 13.581µs | GET | "/run-log"
39 --- PASS: TestGetRunLogHandler (0.00s)
40 == RUN TestGetMQStatisticHandler
41 [GIN] 2024/01/30 - 02:12:38 | 200 | 319.899µs | GET | "/mqstatistic"
42 --- PASS: TestGetMQStatisticHandler (0.00s)
43 PASS
44 ok example.com/tests (cached)
45 Cleaning up project directory and file based variables
46 Job succeeded
```

On the right side, there is a summary of the job: Duration: 2 seconds, Finished: 3 hours ago, Queued: 3 seconds, Timeout: 1h (from project), Runner: #131 (YQzQdbFXH), Tags: devops-project, Commit: c0b8ad02, update gitlab-ci.yml, Pipeline #2387 (Passed) for project, test, and Related jobs: test.

## ➤ Failed Test Cases Scenario

The screenshot displays the GitHub Actions interface for a workflow named "sharada\_jayaweera\_private\_project". The left sidebar shows navigation options like Project, Pinned, Issues, Merge requests, Manage, Plan, Code, Build, Pipelines, Jobs (selected), Pipeline editor, Pipeline schedules, Artifacts, Secure, Deploy, Operate, Monitor, Analyze, and Settings.

The main area shows the workflow run details for job #0054. A search bar at the top allows filtering logs. The log output is as follows:

```

1 Running with gitlab-runner 16.8.0 (c72a89b6)
2   on devops-project-runner YQzQdbFXH, system ID: s_7a4197651aa3
3   ✓ Preparing the "shell" executor...
4   Using Shell (bash) executor...
5   Preparing environment...
6   Running on 187454-001ls...
7   Getting source from Git repository
8   Fetching changes with git depth set to 20...
9   Reinitialized existing Git repository in /home/gitlab-runner/builds/YQzQdbFXH/sharada_jayaweera/sharada_jayaweera_private_project/.git/
10  Checking out 7d815fed as detached HEAD (ref is project)...
11  Skipping Git submodule setup
12  Executing "setup_script" stage of the job script
13  $ echo "Running Tests ...." && pod && cd tests && pod && go test ./... && cd ..
14  Running Tests .....
15  /home/gitlab-runner/builds/YQzQdbFXH/sharada_jayaweera/sharada_jayaweera_private_project
16  /home/gitlab-runner/builds/YQzQdbFXH/sharada_jayaweera/sharada_jayaweera_private_project/tests
17  [ ]
18  [ ]
19  [ ]
20  [GIN-debug] [WARNING] Running in "debug" mode. Switch to "release" mode in production.
21    - using env: export GIN_MODE=release
22    - using code: gin.SetMode(gin.ReleaseMode)
23  [GIN-debug] GET    /messages                --> example.com/api-gateway/handlers.GetMessagesHandler (1 handlers)
24  --- FAIL: TestGetMessagesHandler (0.00s)
25      handlers_test.go:21: Expected status code 200, got 500
26      handlers_test.go:26: Expected Content-Type text/plain; charset=utf-8, got application/json; charset=utf-8
27  FAIL
28  FAIL    example.com/tests        0.018s
29  FAIL
30  Cleaning up project directory and file based variables
31  ERROR: Job failed: exit status 1

```

On the right side, summary statistics are provided:

- Duration:** 2 seconds
- Finished:** 1 week ago
- Queued:** 3 seconds
- Timeout:** 1h (from project)
- Runner:** #131 (YQzQdbFXH)
- Tags:** `devops-project`
- Comments:** 1 comment (update get messages feature)
- Pipeline:** #1795 (Failed for project)
- Related jobs:** → test

➤ SonarQube Check

The screenshot displays the SonarQube web application. On the left sidebar, the navigation menu includes options like Project, Pinned, Issues, Merge requests, Manage, Plan, Code, Build, Pipelines, Jobs, Pipeline editor, Pipeline schedules, Artifacts, Secure, Deploy, Operate, Monitor, Analyze, and Settings. The main area shows the details of a specific pipeline run.

**Pipeline Details:**

- Name:** sharada.jayaweera\_private\_project
- Status:** Completed (green checkmark)
- Duration:** 18 seconds
- Finished:** 3 hours ago
- Queued:** 0 seconds
- Timeout:** (from project)
- Runner:** YQZ-QdbFXH
- Tags:** devops-project
- Commit:** c6b8ad2
- Action:** update gitlab-clymt
- Pipeline ID:** #2387
- Status:** Passed
- Project:** sonarqube

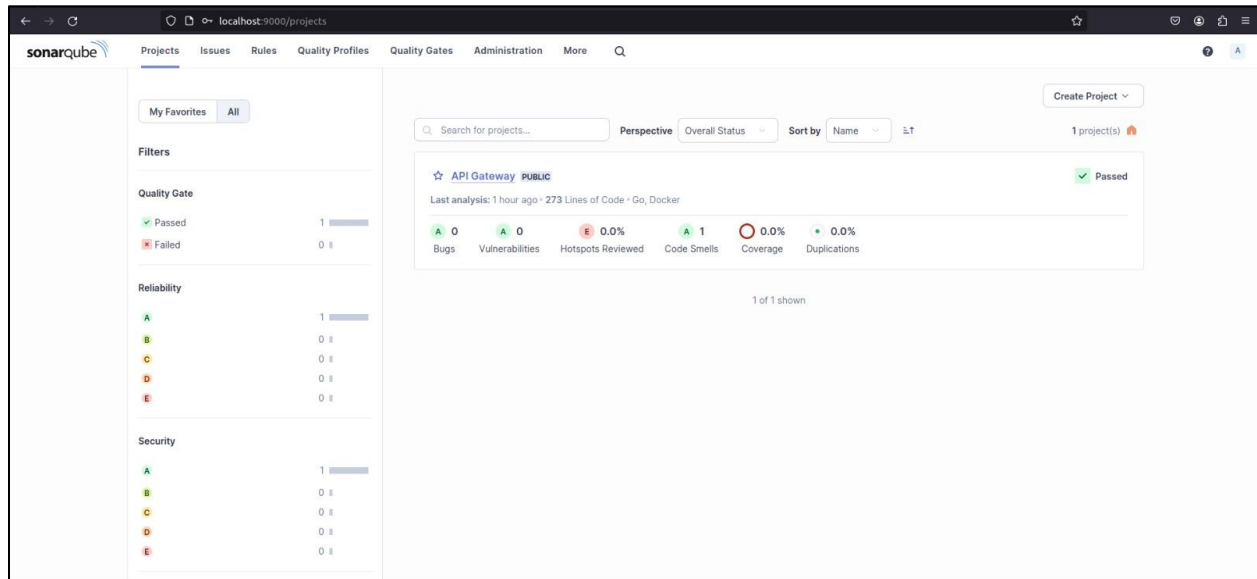
**Related jobs:**

- sonarqube

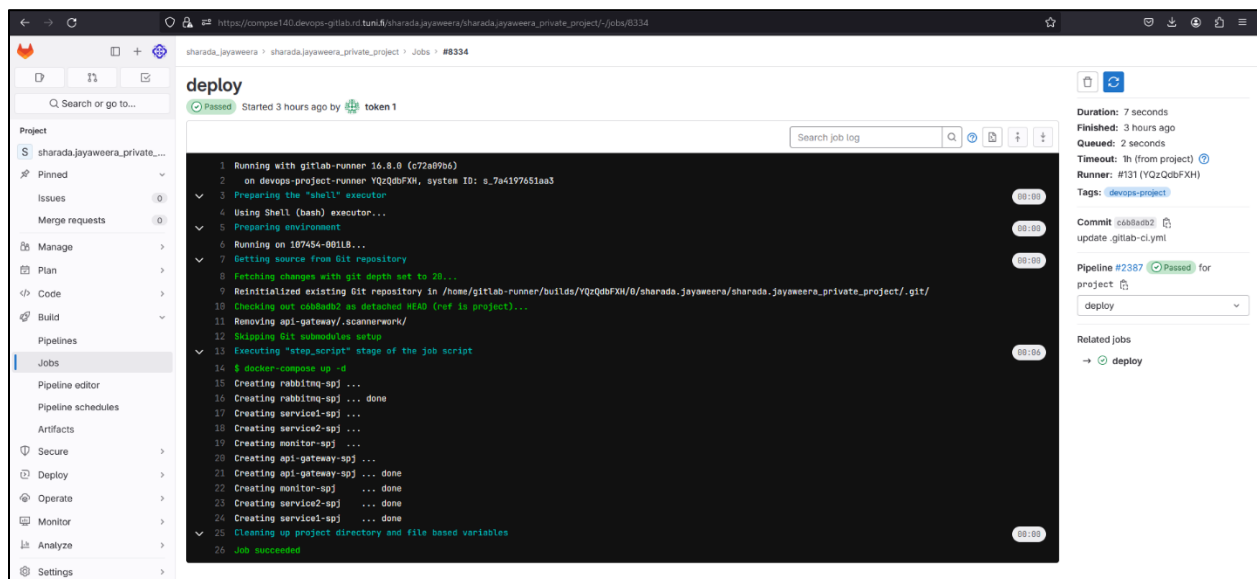
**Log Output:**

```

354 18:21:49.364 DEBUG: Detection of duplications for /home/gitlab-runner/builds/YQZ-QdbFXH/0/sharada.jayaweera/sharada.jayaweera_private_project/api-gateway/config.go
355 18:21:49.396 DEBUG: Detection of duplications for /home/gitlab-runner/builds/YQZ-QdbFXH/0/sharada.jayaweera/sharada.jayaweera_private_project/api-gateway/handlers.go
356 18:21:49.406 DEBUG: Detection of duplications for /home/gitlab-runner/builds/YQZ-QdbFXH/0/sharada.jayaweera/sharada.jayaweera_private_project/api-gateway/main.go
357 18:21:49.407 DEBUG: Detection of duplications for /home/gitlab-runner/builds/YQZ-QdbFXH/0/sharada.jayaweera/sharada.jayaweera_private_project/api-gateway/models/statResponseModel.go
358 18:21:49.408 DEBUG: Detection of duplications for /home/gitlab-runner/builds/YQZ-QdbFXH/0/sharada.jayaweera/sharada.jayaweera_private_project/api-gateway/models/statModel.go
359 18:21:49.409 INFO: CPU Executor CPU calculation finished (done) | time=25ms
360 18:21:49.422 DEBUG: SCM revision ID 'c6b8adb21bd9f9c0458b9a2c1e7cf9a7b999a65b'
361 18:21:49.575 INFO: Analysis report generated in 16ms, dir size=159.4 kB
362 18:21:49.619 INFO: Analysis report compressed in 43ms, zip size=26.9 kB
363 18:21:49.619 INFO: Analysis report generated in /home/gitlab-runner/builds/YQZ-QdbFXH/0/sharada.jayaweera/sharada.jayaweera_private_project/api-gateway/.scannerwork/scanner-report
364 18:21:49.620 DEBUG: Upload report
365 18:21:49.669 DEBUG: POST 200 http://localhost:9080/api/cv/submit?projectKey=api-gateway&projectName=API%2Bgateway | time=68ms
366 18:21:49.684 INFO: Analysis report uploaded in 73ms
367 18:21:49.692 DEBUG: Report metadata written to /home/gitlab-runner/builds/YQZ-QdbFXH/0/sharada.jayaweera/sharada.jayaweera_private_project/api-gateway/.scannerwork/report-task.txt
368 18:21:49.696 INFO: ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9080/dashboard?id=api-gateway
369 18:21:49.696 INFO: Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
370 18:21:49.696 INFO: More about the report processing at http://localhost:9080/api/cv/task?id=AvisblUwPdvxxXColFtzg
371 18:21:49.702 DEBUG: Post-jobs :
372 18:21:49.717 INFO: Analysis total time: 13.54 s
373 18:21:49.722 INFO: -----
374 18:21:49.722 INFO: EXECUTION SUCCESS
375 18:21:49.722 INFO: -----
376 18:21:49.722 INFO: Total time: 16.469s
377 18:21:49.821 INFO: Final Memory: 21M/74M
378 18:21:49.821 INFO: -----
379 Cleaning up project directory and file based variables
380 Job succeeded
  
```

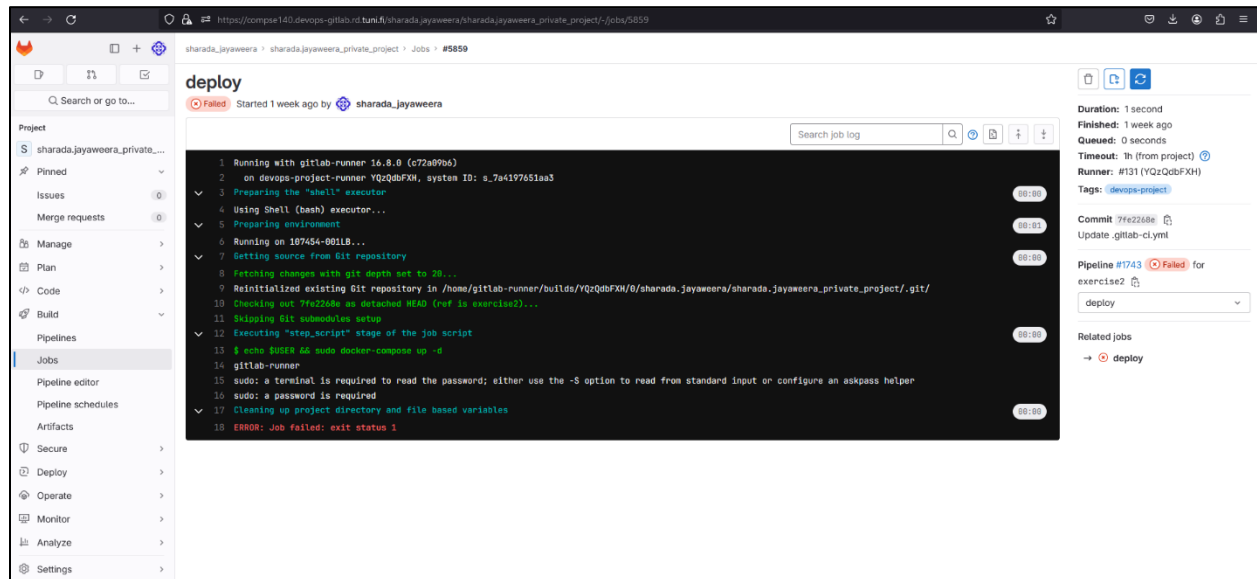


## ➤ Successful Deploy Stage





## ➤ Failed Deploy Stage



## 4. Reflections

### Main learnings and worst difficulties

#### ➤ Stopping running containers

I used Go client for the Docker Engine API in **service1-spj**. Using this client, the running docker containers in the host can be stopped. However, host **/var/run/docker.sock** should be mounted to the container to communicate with the host docker system. Hence, the relevant volume mount is added in the **docker-compose.yaml** file for **service1-spj**.

#### ➤ Test cases writing

Writing test cases inside a separate "tests" folder seems rather strange given that the programming language frameworks has a built-in way and procedure of writing and running test cases. It is difficult to get a code coverage value in this manner as the tests files are isolated from the source code. Hence, the code coverage is zero in SonarQube. It is possible to get a code coverage by including the test cases together with the source code as required by the language framework. In my opinion it is not a good practice to have test cases in a separate folder as it generates additional overhead and difficulties.

Amount effort (hours) used.

Around 50 hours.