

## 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.

*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 **RUNNING** 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"`

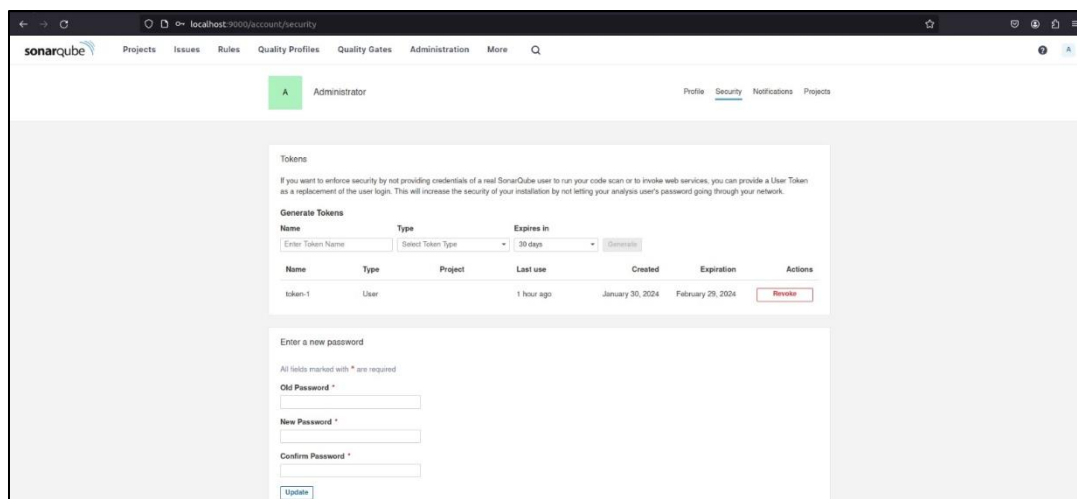
- `curl localhost:8083/state -X PUT -d "INIT" -H "Content-Type: text/plain" -H "Accept: text/plain"`
- `curl localhost:8083/state -X PUT -d "SHUTDOWN" -H "Content-Type: text/plain" -H "Accept: text/plain"`
- `curl localhost:8083/state -X GET -H "Content-Type: text/plain" -H "Accept: text/plain"`
- `curl 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.

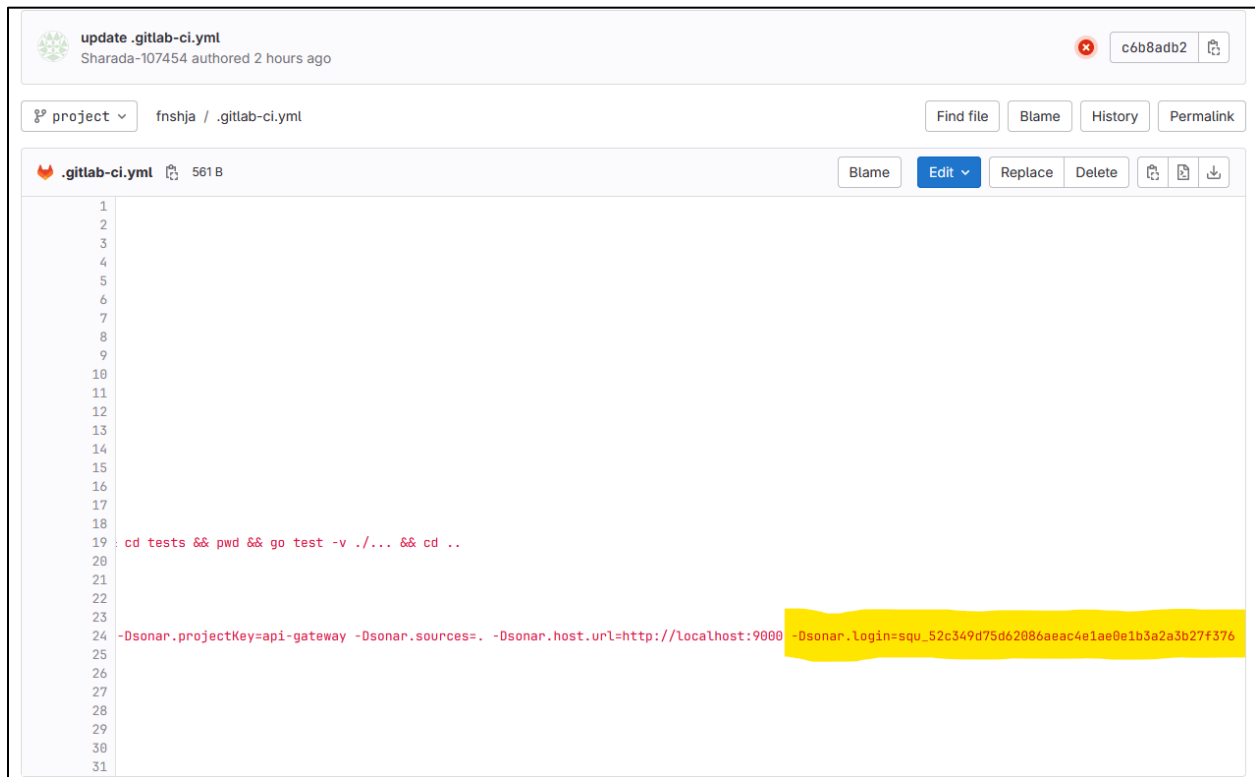
- Run SonarQube docker container using the below command.

```
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>).



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



- 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.tunl.fl/compse140-fall2023/fnshja.git (fetch)
origin  https://course-gitlab.tunl.fl/compse140-fall2023/fnshja.git (push)
origin-ci https://sharada.jayaweera:glpat-3EXtn3Bf1dZs9La5xa-1@compse140.devops-gitlab.rd.tunl.fl/sharada.jayaweera/sharada.jayaweera_private_project.git (fetch)
origin-ci https://sharada.jayaweera:glpat-3EXtn3Bf1dZs9La5xa-1@compse140.devops-gitlab.rd.tunl.fl/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

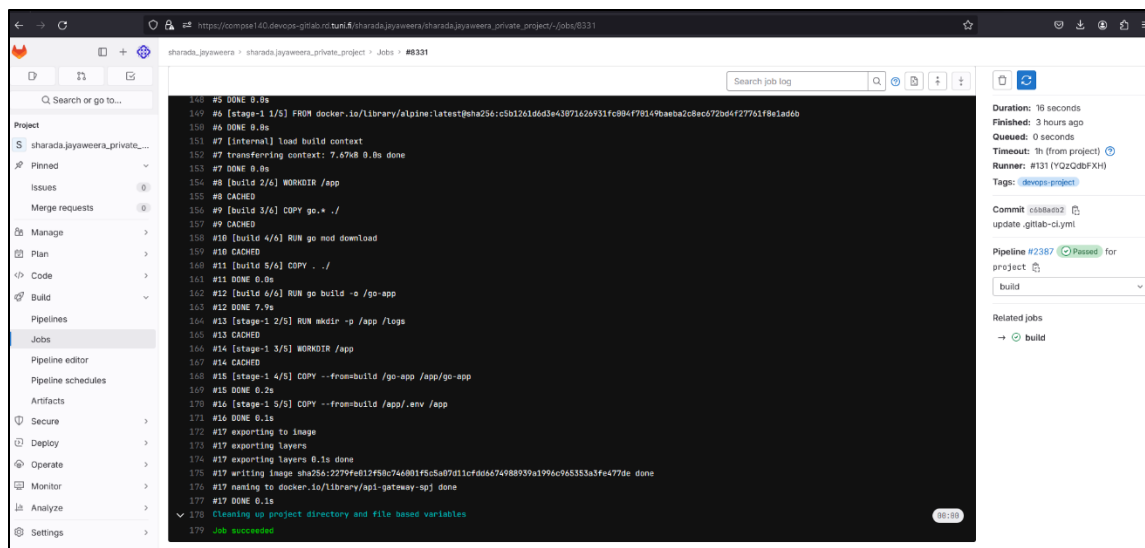
Local deployment done with docker-compose using ***docker-compose up -d*** command.

## • Operating; monitoring

Did not implement

## 3. Example runs of the pipeline

### ➤ Successful Build Stage



The screenshot displays a GitHub Actions workflow run for a project named 'sharada\_jayweera'. The interface shows the 'Jobs' tab selected in the left sidebar. The main area displays the job log for the 'build' job, which is marked as 'Succeeded'. The log shows the following steps:

- 100 #5 DONE 0.0s
- 101 #6 [stage-1 1/3] FROM docker.io/library/alpine:latest@sha256:c5b1261d6d3e43071626931fc094f79149bae2c8ec672bd4f27761f8e1ad60
- 102 #6 DONE 0.0s
- 103 #7 [internal] load build context
- 104 #7 transferring context: 7.67kB 0.0s done
- 105 #7 DONE 0.0s
- 106 #8 [build 2/6] WORKDIR /app
- 107 #8 CACHED
- 108 #9 [build 3/6] COPY go.\* ./
- 109 #9 CACHED
- 110 #10 [build 4/6] RUN go mod download
- 111 #10 CACHED
- 112 #11 [build 5/6] COPY . ./
- 113 #11 DONE 0.0s
- 114 #12 [build 6/6] RUN go build -o /go-app
- 115 #12 DONE 7.9s
- 116 #13 [stage-1 2/3] RUN mkdir -p /app /logs
- 117 #13 CACHED
- 118 #14 [stage-1 3/3] WORKDIR /app
- 119 #14 CACHED
- 120 #15 [stage-1 4/5] COPY --from=build /go-app /app/go-app
- 121 #15 DONE 0.2s
- 122 #16 [stage-1 5/5] COPY --from=build /app/.env /app
- 123 #16 DONE 0.1s
- 124 #17 exporting to image
- 125 #17 exporting layers
- 126 #17 writing image sha256:2279f6d12f9dc746801f5c8a07d11cf0d6674988939a199ac96553a3fe477de done
- 127 #17 naming to docker.io/library/api-gateway-spj done
- 128 #17 DONE 0.1s
- 129 Cleaning up project directory and file based variables
- 130 Job succeeded

On the right side, the job summary shows:

- Duration: 16 seconds
- Finished: 3 hours ago
- Queued: 0 seconds
- Timeout: 1h (from project)
- Runner: #131 (YQZG8FXXH)
- Tags: devops-project
- Commit: c5b1261d6d3e43071626931fc094f79149bae2c8ec672bd4f27761f8e1ad60
- update .github-ci.yml
- Pipeline #2387 Passed for project
- build
- Related jobs: build

## ➤ Failed Build Stage

The screenshot shows a GitLab CI/CD interface for a project named 'sharada.jayaweera\_private\_project'. The 'build' job is highlighted in the left sidebar. The job status is 'Failed', indicated by a red circle with a white 'X'. The job was started 1 week ago by user 'token1'. The main panel displays the job's log, which shows the following steps:

- Running with gitlab-runner 16.8.0 (c72a89b6)
- on devops-project-runner YQzQdbFXH, system ID: s\_7a4197a51aa3
- Preparing the "shell" executor
- Using Shell (bash) executor...
- Preparing environment
- Running on 107454-0018...
- Setting source from Git repository
- Fetching changes with git depth set to 20...
- Initialized empty Git repository in /home/sharada-107454/builds/YQzQdbFXH/0/sharada.jayaweera/sharada.jayaweera\_private\_project/.git/
- Created fresh repository.
- Checking out 1aa2a63 as detached HEAD (ref is exercise2)...
- Skipping Git submodules setup
- Executing "step\_script" stage of the job script
- \$ docker-compose build
- bash: line 140: docker-compose: command not found
- Cleaning up project directory and file based variables
- ERROR: Job failed: exit status 1

On the right side, the job's metadata is displayed:

- Duration: 1 second
- Finished: 1 week ago
- Queued: 1 second
- Timeout: 1h (from project)
- Runner: #131 (YQzQdbFXH)
- Tags: devops-project
- Commit: 1aa2a63
- edit gitlab-ci.yml
- Pipeline #1730: Failed for exercise2
- build
- Related jobs: build

## ➤ Passed Test Cases Scenario

The screenshot shows a GitLab CI/CD interface for the same project. The 'test' job is highlighted in the left sidebar. The job status is 'Passed', indicated by a green circle with a white checkmark. The job was finished 3 hours ago. The main panel displays the job's log, which shows the following steps:

- /home/gitlab-runner/builds/YQzQdbFXH/0/sharada.jayaweera/sharada.jayaweera\_private\_project/tests
- == RUN TestGetMessagesHandler
- [GIN] 2024/01/30 - 02:12:38 | 200 | 20.914µs | GET | "/"messages"
- PASS: TestGetMessagesHandler (0.00s)
- == RUN TestPutStateHandlerInit
- [GIN] 2024/01/30 - 02:12:38 | 200 | 2.651µs | PUT | "/"state"
- PASS: TestPutStateHandlerInit (0.00s)
- == RUN TestPutStateHandlerPaused
- [GIN] 2024/01/30 - 02:12:38 | 200 | 20.138µs | PUT | "/"state"
- PASS: TestPutStateHandlerPaused (0.00s)
- == RUN TestPutStateHandlerRunning
- [GIN] 2024/01/30 - 02:12:38 | 200 | 2.099µs | PUT | "/"state"
- PASS: TestPutStateHandlerRunning (0.00s)
- == RUN TestPutStateHandlerShutdown
- [GIN] 2024/01/30 - 02:12:38 | 200 | 6.466µs | PUT | "/"state"
- PASS: TestPutStateHandlerShutdown (0.00s)
- == RUN TestPutStateHandlerInvalid
- [GIN] 2024/01/30 - 02:12:38 | 400 | 2.62µs | PUT | "/"state"
- PASS: TestPutStateHandlerInvalid (0.00s)
- == RUN TestGetStateHandler
- [GIN] 2024/01/30 - 02:12:38 | 200 | 3.128µs | GET | "/"state"
- PASS: TestGetStateHandler (0.00s)
- == RUN TestGetRunLogHandler
- [GIN] 2024/01/30 - 02:12:38 | 200 | 13.581µs | GET | "/"run-log"
- PASS: TestGetRunLogHandler (0.00s)
- == RUN TestGetMQStatisticHandler
- [GIN] 2024/01/30 - 02:12:38 | 200 | 319.899µs | GET | "/"mqstatistic"
- PASS: TestGetMQStatisticHandler (0.00s)
- PASS
- ok example.com/tests (cached)
- Cleaning up project directory and file based variables
- Job succeeded

On the right side, the job's metadata is displayed:

- Duration: 2 seconds
- Finished: 3 hours ago
- Queued: 3 seconds
- Timeout: 1h (from project)
- Runner: #131 (YQzQdbFXH)
- Tags: devops-project
- Commit: c6b6a62
- update gitlab-ci.yml
- Pipeline #2387: Passed for project
- test
- Related jobs: test

## ➤ Failed Test Cases Scenario

The screenshot displays a GitLab CI/CD interface for a pipeline job #8054. The job is titled "sharada.jayaweera" and is part of a project named "sharada.jayaweera\_private\_project". The job is in a "Failed" state, indicated by a red circle with a white 'X' icon. The job log shows the following steps:

- 1 Running with gitlab-runner 16.8.0 (c72a996)
- 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 107454-0018...
- 7 Getting source from Git repository
- 8 Fetching changes with git depth set to 20...
- 9 Initializing existing Git repository in /home/gitlab-runner/builds/YQzQdbFXH/0/sharada.jayaweera/sharada.jayaweera\_private\_project/.git/
- 10 Checking out 70b15f6d as detached HEAD (ref is project)...
- 11 Skipping Git submodule setup
- 12 Executing "step\_script" stage of the job script
- 13 \$ echo "Running Tests ...." && pwd && cd tests && pwd && go test ./... && cd ..
- 14 Running Tests ....
- 15 /home/gitlab-runner/builds/YQzQdbFXH/0/sharada.jayaweera/sharada.jayaweera\_private\_project
- 16 /home/gitlab-runner/builds/YQzQdbFXH/0/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 example.com/tests 0.018s
- 29 FAIL
- 30 Cleaning up project directory and file based variables
- 31 ERROR: Job failed: exit status 1

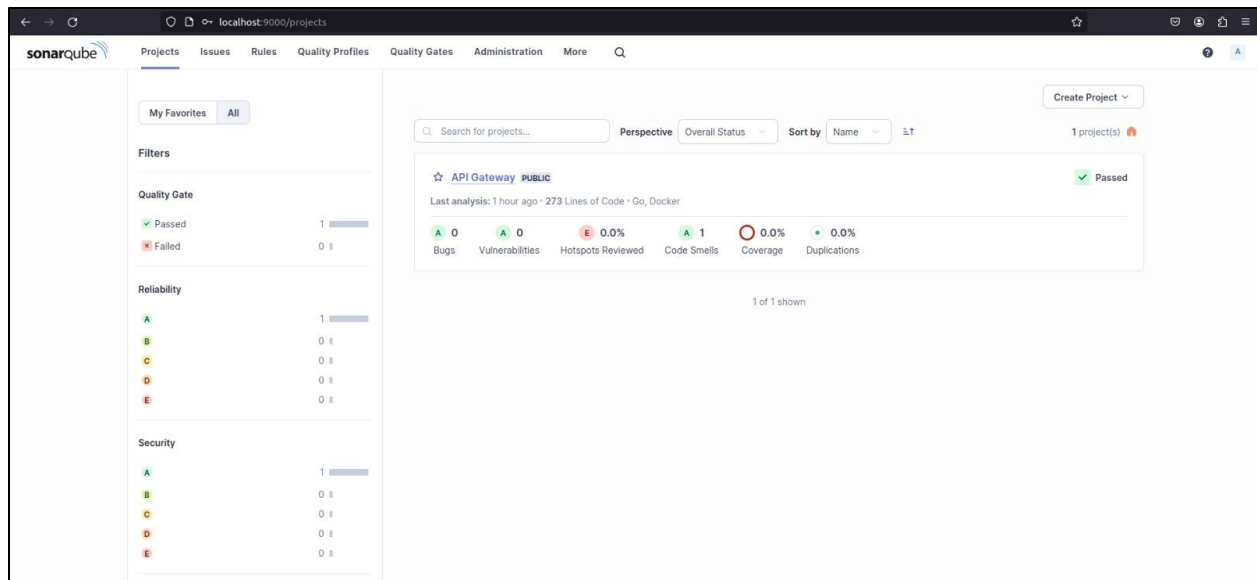
The job summary on the right indicates a duration of 2 seconds, finished 1 week ago, and a timeout of 1h (from project). The runner is #131 (YQzQdbFXH). The pipeline #1796 is also shown as failed for the project.

## ➤ SonarQube Check

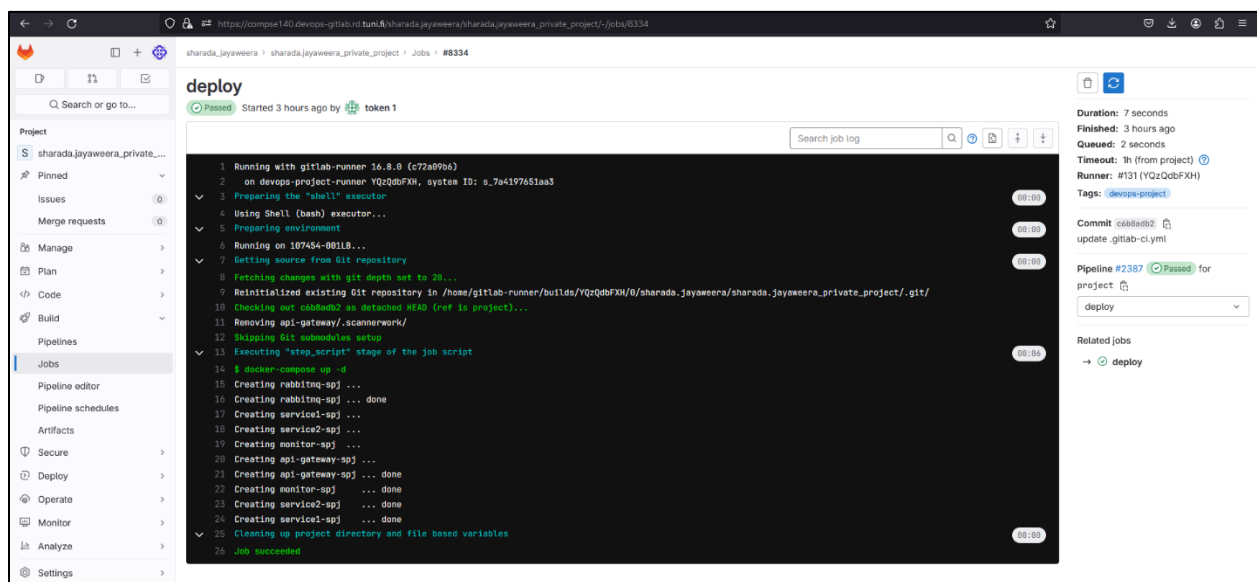
The screenshot displays a GitLab CI/CD interface for a pipeline job #8333. The job is titled "sharada.jayaweera" and is part of a project named "sharada.jayaweera\_private\_project". The job is in a "Succeeded" state, indicated by a green circle with a white checkmark icon. The job log shows the following steps:

- 345 18:21:49.394 DEBUG: Detection of duplications for /home/gitlab-runner/builds/YQzQdbFXH/0/sharada.jayaweera/sharada.jayaweera\_private\_project/api-gateway/config.go
- 346 18:21:49.396 DEBUG: Detection of duplications for /home/gitlab-runner/builds/YQzQdbFXH/0/sharada.jayaweera/sharada.jayaweera\_private\_project/api-gateway/handler
- 347 18:21:49.407 DEBUG: Detection of duplications for /home/gitlab-runner/builds/YQzQdbFXH/0/sharada.jayaweera/sharada.jayaweera\_private\_project/api-gateway/main.go
- 348 18:21:49.408 DEBUG: Detection of duplications for /home/gitlab-runner/builds/YQzQdbFXH/0/sharada.jayaweera/sharada.jayaweera\_private\_project/api-gateway/models/
- 349 18:21:49.409 INFO: CPD Executor CPD calculation finished (done) | time=2ms
- 350 18:21:49.422 DEBUG: SCM revision ID 'c6b8ad2212bdf59c450b9a2c1e7c9a7b9996e5b'
- 351 18:21:49.575 INFO: Analysis report generated in 14ms, dir size=159.4 kB
- 352 18:21:49.619 INFO: Analysis report compressed in 43ms, zip size=26.9 kB
- 353 18:21:49.619 INFO: Analysis report generated in /home/gitlab-runner/builds/YQzQdbFXH/0/sharada.jayaweera/sharada.jayaweera\_private\_project/api-gateway/.scannerwo
- 354 18:21:49.620 DEBUG: Upload report
- 355 18:21:49.689 DEBUG: POST 200 http://localhost:9000/api/cv/submit/project?projectKey=api-gateway&projectName=APIGateway | time=68ms
- 356 18:21:49.694 INFO: Analysis report uploaded in 73ms
- 357 18:21:49.695 DEBUG: Report metadata written to /home/gitlab-runner/builds/YQzQdbFXH/0/sharada.jayaweera/sharada.jayaweera\_private\_project/api-gateway/.scannerwo
- 358 18:21:49.696 INFO: ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=api-gateway
- 359 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
- 360 18:21:49.696 INFO: More about the report processing at http://localhost:9000/api/cv/task?id=AV1bLUPrDxxvKCaTfzg
- 361 18:21:49.702 DEBUG: Post-jobs :
- 362 18:21:49.717 INFO: Analysis total time: 13.546 s
- 363 18:21:49.722 INFO: -----
- 364 18:21:49.722 INFO: EXECUTION SUCCESS
- 365 18:21:49.722 INFO: -----
- 366 18:21:49.722 INFO: Total time: 16.469s
- 367 18:21:49.821 INFO: Final Memory: 21M/74M
- 368 18:21:49.821 INFO: -----
- 369 Cleaning up project directory and file based variables
- 370 Job succeeded

The job summary on the right indicates a duration of 18 seconds, finished 3 hours ago, and a timeout of 0 seconds. The runner is #131 (YQzQdbFXH). The pipeline #2387 is also shown as passed for the project.

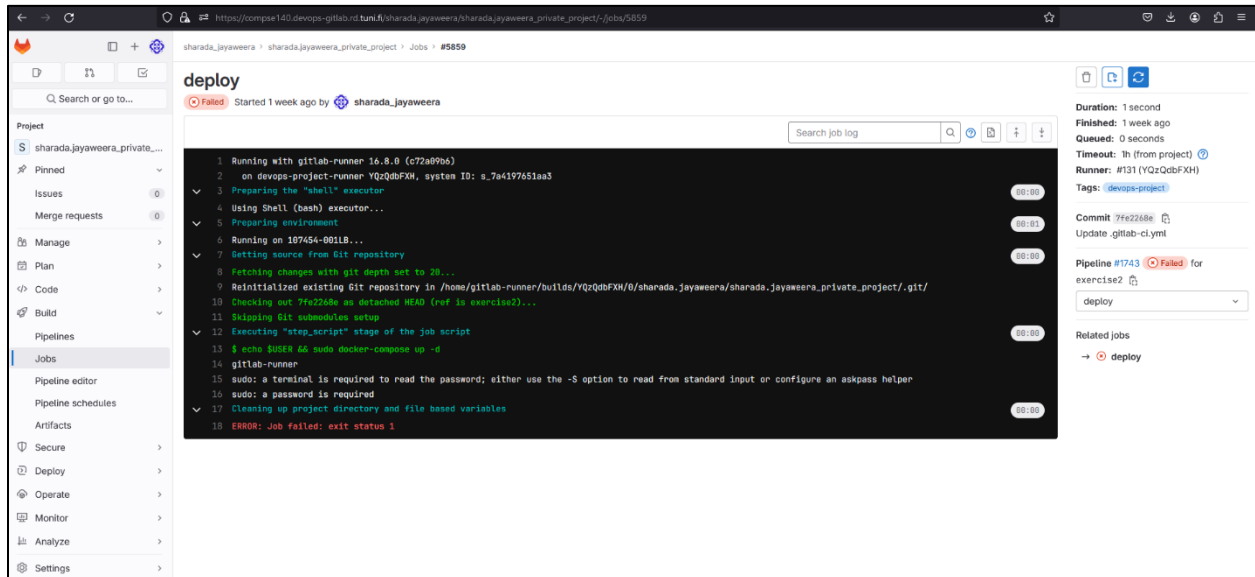


## ➤ Successful Deploy Stage





## ➤ 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.