DevOps Project: CI/CD Pipeline

COMP.SE.140

MARKUS PALOMÄKI

Contents

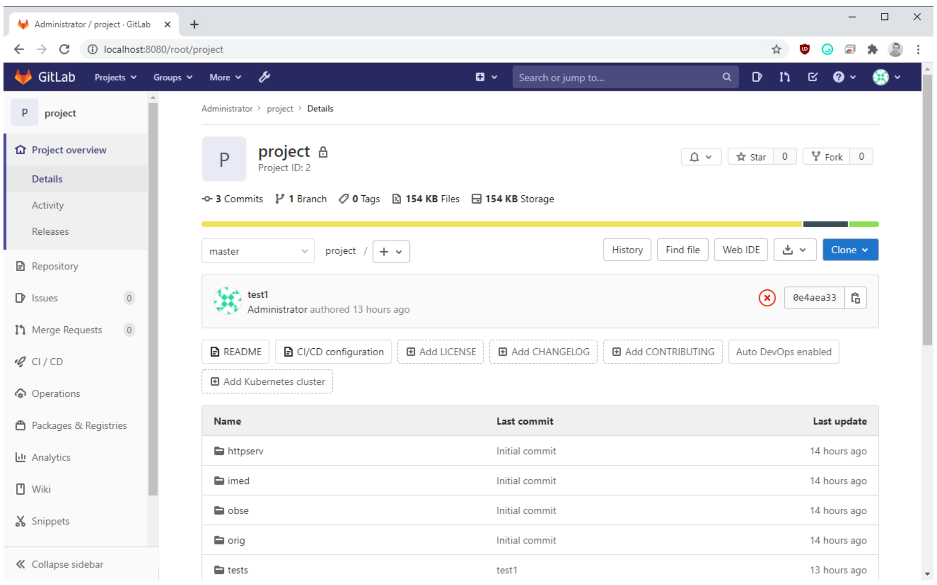
Introduction

The project was made for Tampere Universities course *Continuous Development and Deployment – DevOps* (COMP.SE.140). The idea of the course's final project was to experience with CI/CD pipeline, which was running on GitLab; to implement automated pipeline which started running when the system noticed a new GIT change in GitLab repository.

With the help of created CI/CD pipeline the included message queue software project was developed further with test-drive development way.

Implementation

The system is using Docker containers. Each part of system has an own container for different ideas. The main two containers include GitLab web service and GitLab runner, which is an application for GitLab CI/CD and which runs jobs in GitLab's pipeline.



Name	Last commit	Last update
httpserv	Initial commit	14 hours ago
imed	Initial commit	14 hours ago
obse	Initial commit	14 hours ago
orig	Initial commit	14 hours ago
tests	test1	13 hours ago

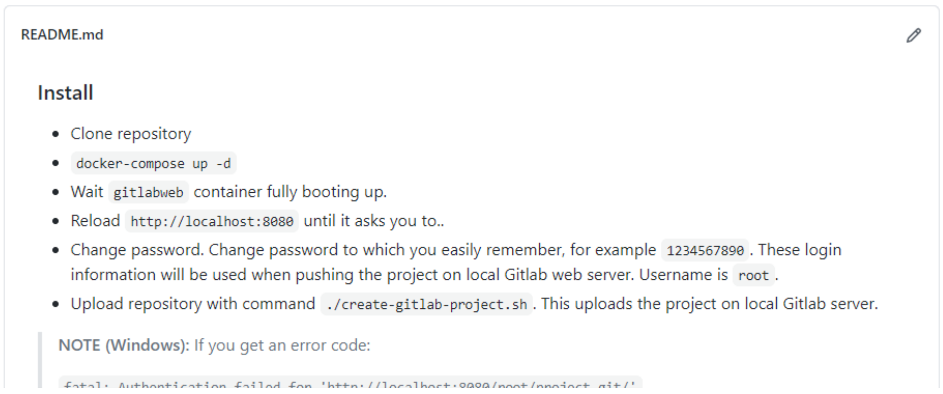
Figure 1 Screenshot from GitLab running in container

GitLab needs files for building and testing, so the project includes another project's files. With GitLab changes are saved with GIT and with pipelines code is built and tested.

The applications what the project is handling on GitLab is previous exercise which sends messages using RabbitMQ. With RabbitMQ applications could route messages from application to other using message-queues and topics.

Installation

The project includes README.md file which tells step-by-step how to install the GitLab environment after cloning the repository. It has couple notes about the issues which users could face while installing or setting environment up. More about these issues later in another topic.



README.md

Install

- Clone repository
- `docker-compose up -d`
- Wait `gitlabweb` container fully booting up.
- Reload `http://localhost:8080` until it asks you to..
- Change password. Change password to which you easily remember, for example `1234567890`. These login information will be used when pushing the project on local Gitlab web server. Username is `root`.
- Upload repository with command `./create-gitlab-project.sh`. This uploads the project on local Gitlab server.

> **NOTE (Windows):** If you get an error code:

```
fatal: Authentication failed for 'http://localhost:8080/root/project.git/'
```

Figure 2 Screenshot from GitHub of README file

Pipelines

When GitLab runner detects changes in GitLab's GIT repository, it launches jobs in pipelines. This setup is made with the file "*.gitlab-ci.yml*". The first pipeline creates a new Docker container with *docker:latest* image and install all necessary packages such as *NodeJS* and *npm.* NodeJS is a JavaScript run-time environment for testing files which are using Mocha test framework and Chai (HTTP integration testing library). NPM is package manager for NodeJS.

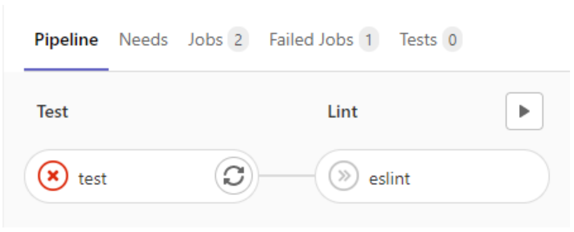
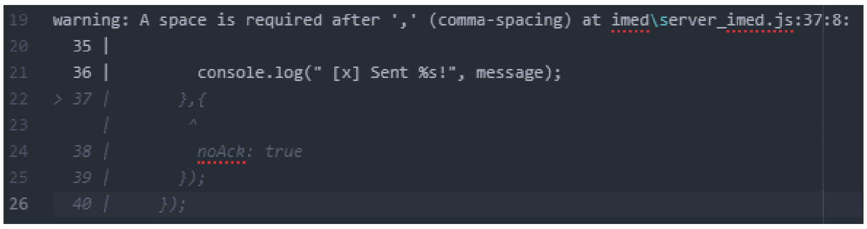


Figure 3 Screenshot from GitLab pipelines

After installing it launches four containers for each apps which are stored in GIT repository, waits with using a custom-made script that the software's messages are sent, and after this runs tests.

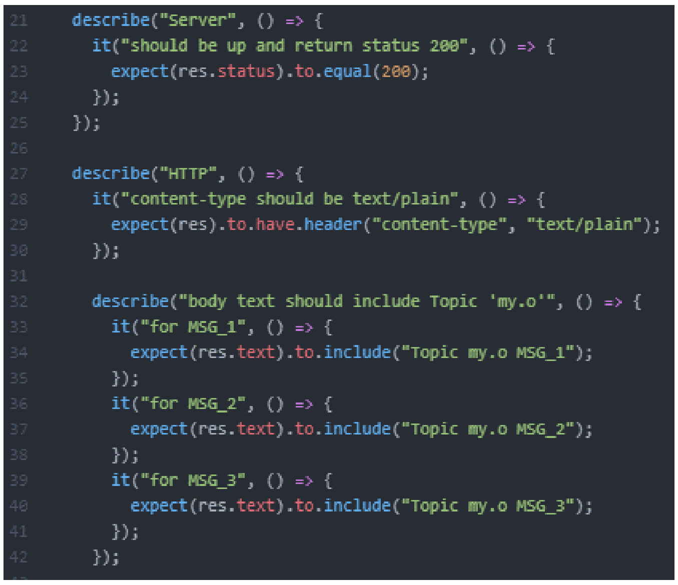
When the test stage is done, the next stage is executed. It will run a ESLint analyser which goes thru files and detects problems based to ESLint settings. It shows issues on logs such as missing semicolons. The lint-state also creates a report in "tests" folder with name "lint-report.md", but this feature needs more developing. After all stages are ran, the CI removes all Docker containers which it started.



```
warning: A space is required after ',' (comma-spacing) at imed\server_imed.js:37:8:
  35 |
  36 |         console.log(" [x] Sent %s!", message);
> 37 |       },{
     |        ^
  38 |         noAck: true
  39 |       });
  40 |     });
```

Figure 4 Screenshot from lint-report.md file showing a warning about missing space (comma-spacing rule)

Tests



```
describe("Server", () => {
  it("should be up and return status 200", () => {
    expect(res.status).to.equal(200);
  });
});

describe("HTTP", () => {
  it("content-type should be text/plain", () => {
    expect(res).to.have.header("content-type", "text/plain");
  });

  describe("body text should include Topic 'my.o'", () => {
    it("for MSG_1", () => {
      expect(res.text).to.include("Topic my.o MSG_1");
    });
    it("for MSG_2", () => {
      expect(res.text).to.include("Topic my.o MSG_2");
    });
    it("for MSG_3", () => {
      expect(res.text).to.include("Topic my.o MSG_3");
    });
  });
```

Figure 5 A part of the testing file

The project includes only one test file for the server software which shows all sent and received messages on RabbitMQ. The test file detects is the web service up and if it is, the other tests will be executed. It checks does the web page return header value 200 and checks does it show correct text.

Problems and issues

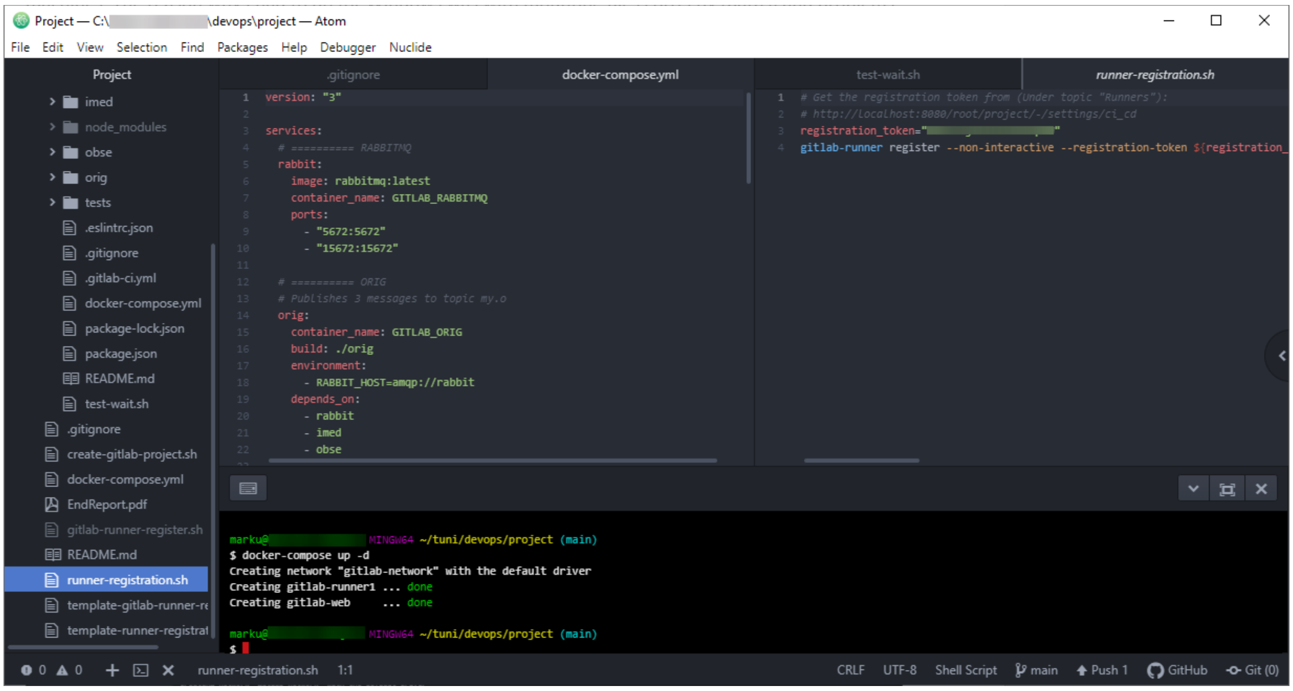
Developing the project on Linux OS was not possible because limited storage page on computer. Also, an option to run the environment on virtual machine was also not used because it may reduce the speed of containers running on virtual machine and other containers running in containers (DIND).

Most of the time I faced issues with GitLab and its pipelines. When multiple other projects with GIT were developed on the same computer, Windows' stored credentials made impossible to connect or push the newest version of software to local GitLab. The way how it was resolved was to remove specific GIT credentials from Windows settings with administrator rights.

Building the project and tested with pipelines also faced couple problems. Sometimes the environment worked, but on other day with the same files and settings something did not work. This happened a few times which made me stuck without knowing where the issue is. On Windows and Docker for Windows, it is not always clear is the newest changes really in use or not; a couple times I found out the newest changes was not in use in containers.

Error such as *"`ERROR: Failed to remove network for build`"* and *"ERROR: Job failed: invalid volume specification`"* was thrown which was fixed with changes, which did not work anymore in another day. I tried to run containers with special flags to build and force recreating, but a couple of times I found out nothing changed when flags were used.

For registering the GitLab runner application I had to create two registration scripts for Windows and Linux machines. The reason why I had to do for Windows was with mounting files correctly (paths) and problems with ending lines. If *"docker exec"* command with flag *"—docker-volumes"* was used on Windows and GIT Bash for Windows, it used Windows system paths even the string was *""/var/run/docker.sock:/var/run/docker.sock""*. On Linux it would work, but Windows changed the string, so it included C:\ disk's name and other folders in the file path. When the command was run, it tried to find "C:\" from Linux system even the string did not include that or other Windows folder names. To fix this I created another registration script, which does not include "docker exec" command for executing a command in a specific container. I created step-by-steps how the file can be run in specific container.



```
version: "3"

services:
  # ========== RABBITMQ
  rabbit:
    image: rabbitmq:latest
    container_name: GITLAB_RABBITMQ
    ports:
      - "5672:5672"
      - "15672:15672"

  # ========== ORIG
  # Publishes 3 messages to topic my.o
  orig:
    container_name: GITLAB_ORIG
    build: ./orig
    environment:
      - RABBIT_HOST=amqp://rabbit
    depends_on:
      - rabbit
      - imed
      - obse
```

```
# Get the registration token from (Under topic "Runners"):
# http://localhost:8080/root/project/-/settings/ci_cd
registration_token="[redacted]"
gitlab-runner register --non-interactive --registration-token ${registration_
```

```
marku@[redacted] MINGW64 ~/tuni/devops/project (main)
$ docker-compose up -d
Creating network "gitlab-network" with the default driver
Creating gitlab-runner1 ... done
Creating gitlab-web     ... done

marku@[redacted] MINGW64 ~/tuni/devops/project (main)
$ 
```

Figure 6 Working environment on Atom

Line endings were another issue. When the GitLab runner was registered, it added “\n” and “\r” endings because incorrect file format which Docker image’s Alpine Linux distribution incorrectly found. Even the script file was looking okay on main computer and in the GitLab runner container with command “cat”, it used the endings while commands were executed. To fix this I had to edit more the registration script so it would not have empty lines or any other slashes.

When containers were set up, Docker found containers which was stored on main computer. Even the docker-container file -- which is used to setup and run multiple containers with one command -- was run in another container, it still saw these containers which located on main computer. I had to rename container names so that overlapping did not happen. Maybe this behaviour is normal for Docker containers.

Conclusion

After facing and solving multiple issues the project did not complete. Faced issues and problems with running DIND and the whole environment took most of the time. The project staid at state where it could not download a specific docker image because network issues (a guess). This problem was familiar earlier and I could fix it, but with the fixes the port numbers were incorrect and Mocha tests faced error about timeout connection to the container which showed results. Because the time was used only battling with the Docker and GitLab issues, the application developing did not happen.

I also had difficulties with motivation and organizing enough time for university because distance learning. Also, it is been a busy Fall when I had to book an extra course from last year. But even I have lack of coding routing and skills which made developing slower and harder, I found out I have learned so many new ideas and technic things while working with the project. The first thing which I will do is to switch main operating system from Windows to Linux, or to create a virtual machine for code developing and accept the slower running speeds.

Even the project was sometimes too much for me and because of that I gave up once or a couple times, I wanted to finish this project. I am aware the project is not enough to pass the course, but I’m already prepared for the idea to re-take this DevOps course again in next time with more available time.