COMP.SE.140-2025-2026-1-TAU

Continuous Development and Deployment – DevOps Tuukka Salonen

Project report 7.12.2025

Authentication

The login credentials are:

- Username: adminuser
- Password: adminpassword

Optional features

The project includes JWT-token based authentication. The console and the API are protected and cannot be accessed without logging in. When user logs in, a cookie is generated for the browser, and the generated access token is displayed in the console. You can use the token to be able to make requests directly to the API having the token in authorization headers.

More advanced container metrics were also added to the console. They display metrics for each of the containers. The metrics include container name, image, current state, uptime, CPU usage percentage, current memory usage and memory usage percentage of the available space. Also, average, min and max response times are kept. It displays the duration of requests to the API.

Pipeline description

In the first stage, the pipeline builds each of the services with their native languages to generate executables. Next, in the testing stage, the pipeline does basic sanity tests. Service1 test checks that the entry point can be run, service2 test checks the module import, storage test checks that the module was created and lastly the console test checks that the module can be imported.

Then, in the package stage, each of the images are built and then pushed to Harbor. After that, smoke tests are performed to make sure the containers can be started and run. Lastly, the deployment stage deploys the code in the virtual machine with Ansible.

The pipeline is the same between versions 1.0 and 1.1 except for the image names. 1.0 builds and pushes images with “blue” postfix and 1.1 does the same with “green” postfix. Storage service image is the same between the versions as it does not contain any differences.

Reflection

The way I implemented the version switching wasn’t the most optimal way. If I had more time I would have tried to do it some other way. In this implementation, Nginx runs outside the containers, and the version switch is done by changing the link of the Nginx conf file. It works but due to it being called from the container, and the switch being done on the host, it became a problem. I solved it by running a script using SSH from the container. I did the same with the discard, which runs a script that shuts down the containers. This doesn’t seem like a good solution but at least it works. I also had some doubts regarding running the versions and my solution was to just run both at the same time, which is probably the correct way.

I could have made the project more generally usable, for example I used direct blue/green identifiers, which would not be used in a real environment. Version numbers would have been better. Also, I had quite a many hardcoded values related to the versioning. They could have been environment variables. I had also some uncertainty about the instructions. I was uncertain what uptime was meant to change from hours to minutes in the version change. I decided that it is done to new log entries.

Biggest learning was probably having everything as one. As the previous exercises had smaller parts, this project combined it all. That gave some perspective about what is needed as a whole.

Effort

Due to being busy with other courses and work, I couldn’t spend much time on this project. If I had more time, I could have improved it. However, I’d estimate that I spent around 25 hours on this project.