Prize Motivation for Docks

Team: TripleParity
Client: Compiax

Team Members
Francois Mentz
Connor Armand du Plooy
Raymond De Vos
Evert Geldenhuys
Anna-Mari Helberg
Paul Wood

Contents

1 Project Prizes Motivation

1.1 Kindle Technologies - Fit for Purpose

In the beginning, we did not quite understand what the purpose of Docks should be but after consulting the client again and playing around with Docker the real world problem became clear: provide a secure web user interface for using Docker .

Docker is a tool designed to make it easier to create, deploy and run applications using lightweight virtualization. It provides a command line interface (CLI) and RESTful API. Maintaining and deploying applications often involve multiple people. Providing multiple people access to the Docker CLI requires Secure Shell access (SSH) as root to the server running Docker . If the server is secure it will only provide SSH access using public/private keys, which reduces convenience and restricts access to devices that are SSH capable and holds a private key. The Docker API lacks functions which are provided by the CLI, so it cannot be used on its own.

The requirements of the problem domain were solved by Docks . That is because we did implement an easier way for using Docker than using the CLI. But, we also added the extra security that was lost when we moved from the CLI to a user interface with the implementation of the two factor authentication.

1.2 Quant - Most innovative use of open source

Docks is a completely open source project to enhance an existing open source tool and is licensed under the copyleft *GNU General Public License v3.0.* Our project extends the existing open source Docker platform by integrating with the Docker Swarm container orchestration project's API to provide advanced functionality.

Docks only makes use of fully open source libraries such as Angular and Node.js. All packages used are open source on NPM.

Apart from ensuring our project is free and open, we have also innovated in the open source space in the form of a new package published by the Docks team. We have identified a limitation in the Docker swarm stack model in that the translation from a swarm specification file to a service running on the swarm is a one-way process. We did not find any existing open source tools that can reverse the translation. This limitation meant that Docks or any similar project could not be easily integrated into an existing swarm.

To solve this, our team developed, tested and published an independent open source package that can take the machine-specification from Docker and convert it back into a human readable format that can easily be included by existing projects. The package is licensed under the permissive open source *ISC License*. The package can be freely downloaded from NPM (npmjs.com/package/docker-api-to-compose). The source code is published on GitHub and can be viewed under the TripleParity organisation.

1.3 Singular - Software Engineering Excellence

Software Security was always our focus when we were designing and building Docks . Our team structure was quite strange but it worked out quite well. We followed the Scrum framework where Evert was the scrum master. We then made a soft split between back- and frontend where Raymond was "sub-"scrum master for the backend and Francois "sub-"scrum master for the frontend. The rest of the team members (Anna-Marie, Armand and Paul) worked mainly on frontend elements and integration with the backend. However, Raymond has his time working on frontend elements and Anna-Marie and Armand also worked on the backend. Hence, we call it a soft split and that allowed for transparency between members.

From looking at our documentation, one can clearly understand what use purpose and use for Docks is. We clearly followed strict coding standards and review processes when developing the system. The whole group used the same IDE (Visual Studio Code) where ES Lint was active which means that linting errors were accounted for and easily solvable. Minimum two group members were required to review and approve pull requests into develop but we always aimed to gather at least 3 if not all of the team members's feedback.

We used Telegram and Slack as our main method of communicating with each other. We planned and managed every issue that needed to be taken care of with the help of Zenhub. Data integrity was ensured through the review process but also with the help of Travis which monitored every push and every pull request.

If the Travis build fails then the pull request cannot be approved. In this fashion, we always ensured that the code in master is up to scratch and correct.