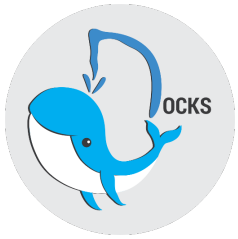

Prize Motivation for Docks

Team: TripleParity
Client: Compiax

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

Contents

1 Project Prizes Motivation

1.1 Kindle Technologies - Fit for Purpose

At the start, we did not quite understand where our project (Docks) would fit in or what purpose it would serve. The client made it clear that they had a valid use for Docks since they heavily rely on Docker for their applications and required specific features not provided by other projects. The initial scope for Docks was a lightweight management interface providing advanced functionality on top of the existing Docker swarm framework and with excellent security.

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 . Our interface serves as a much simpler way of using Docker than 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. Docks was developed in close collaboration with our client who plans on using it in their live, production Docker swarm environment.

Every feature added was added with real-life usage in mind due to our client's experience using Docker. There is a need for a lightweight, drop-in Docker swarm manager with two-factor authentication and a simple, easy to use user interface and we believe that Docks is fit for this purpose.

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 our main focus while we were designing and building Docks . Our team structure was peculiar but it was chosen for optimal performance. We adhered to the Scrum framework as much as possible and as such had a scrum master. Two key focus areas were identified from our team members' experiences: frontend and backend. We decided to roughly split our team into two scrum teams; one for each identified focus area. The teams did not work independently and all issues were raised across the entire TripleParity team for discussion. Members of the sub-teams were encouraged to work on other focus areas as well. We discovered that a smaller team working with high cohesion and focussing on one area outperformed a larger team that

struggled to collaborate.

We spent a lot of effort to ensure our documentation is clear and concise. Due to Docker being a complicated and convoluted project, we made sure users can quickly get a good grasp of Docker concepts from reading our documentation. Clear and strict coding standards were applied and thorough review processes were implemented while developing the system. Our development environment was consistent across our entire team; each member used the same editor (Visual Studio Code) with active syntax linting provided by ESLint. Strict linting resulted in many small bugs being caught and fixed early in the development cycle and consistent styling conventions allowed our code to be easy to read and maintain. A minimum of two team members was required to review and approve pull requests but in practice, every pull request got peer-reviewed by 3 or more developers. This ensured that only high-quality code ended up in our main branches.

We used Telegram and Slack as our main methods of communication and we planned and managed every issue with the help of Zenhub and Github. Data integrity was ensured through the review process but also with the help of automated continuous testing and integration tools such as Travis which monitored every push and pull request. A pull request could only be approved if all the tests passed on Travis. This ensured bugs were caught early and subsequently quickly removed before merging with develop / master.