May15-17

Project Plan

Version 1.0

Caleb Brose, Chris Fogerty, Nick Miller, Rob Sheehy, Zach Taylor
September 30, 2014

Contents

Problem Statement

Docker is a relatively new platform designed to simplify the process of software creation and distribution for distributed and scalable applications. It has become popular with many tech companies including Google, Microsoft, Netflix, and EBay. Groups have attempted to make management applications for Docker in the past, but there still is no single application that can be used to fully utilize all of Docker's capabilities. The goal of this project, then, is to build an open-source management system for Docker images and containers. Ideally, this system will allow users to deploy containers to various cloud platforms, build and upload images to a Docker registry, and view statistics on containers that are currently running.

Deliverables

Minimal Viable Product

Our plan is to quickly create a minimal viable product (MVP) that accomplishes the main goals of our application, then work with Workiva to further iterate on the base product. Ideally, this will be available by week 10 of the first semester.

Future Iterations

Please see the project schedule for expectations of work after MVP completion.

Specifications

Minimal Viable Product

The MVP will allow the user to manage their Docker containers on various Google Cloud Engine (GCE) instances. The user will be able to add, update, and remove the cloud instances stored in our database, as well as any Docker images stored on our registry. The user will be able to see what images have been deployed and what cloud instance they have been deployed to, as well as information about what containers are currently running. While our final goal is to be able to interface with multiple cloud providers, our goal for the MVP is to interface with just the GCE.

Concept Sketch/Mockup

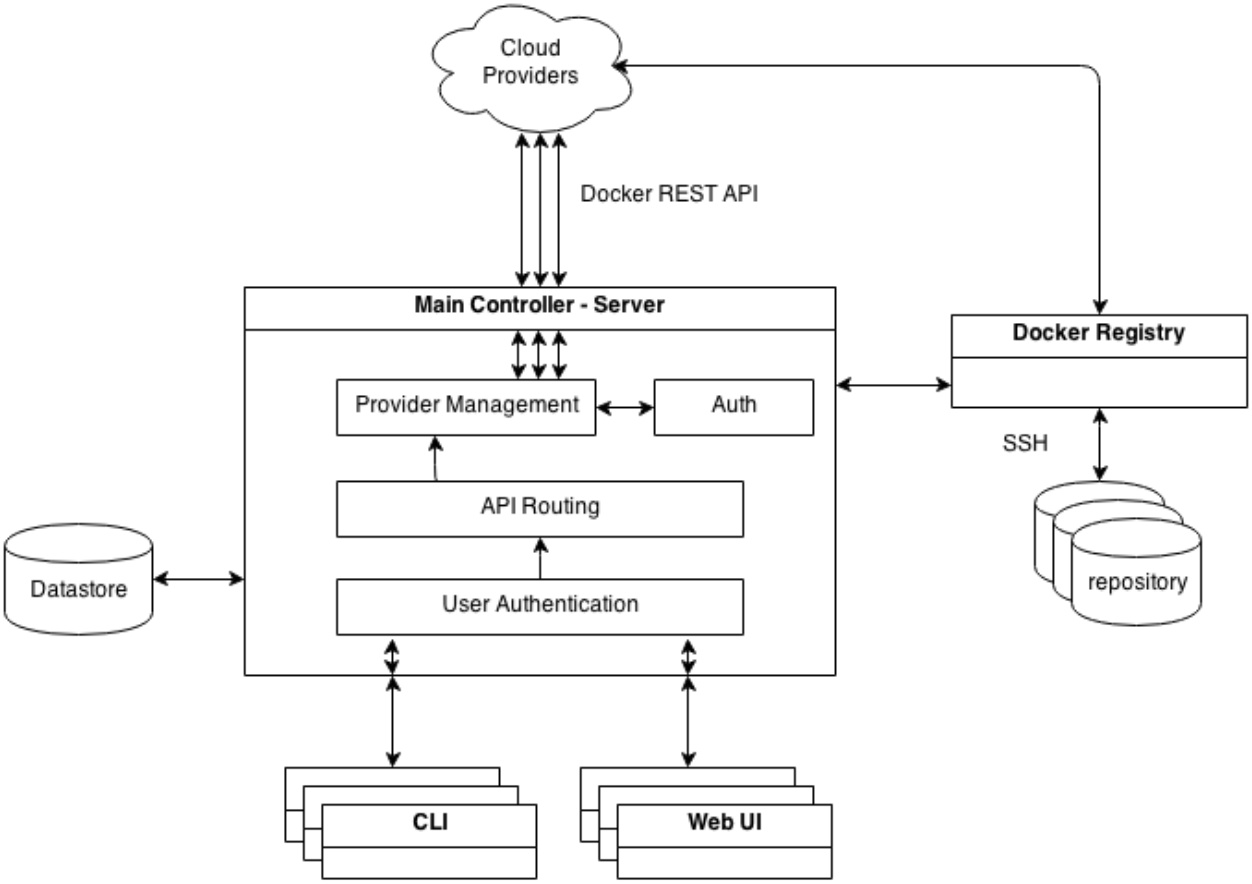

Figure 1: MVP Concept Sketch

User Interface Description

This is currently just a sample of what the user interface could look like. We will work closely with Workiva to come up with something workable that conveys the relevant information.

Figure 2

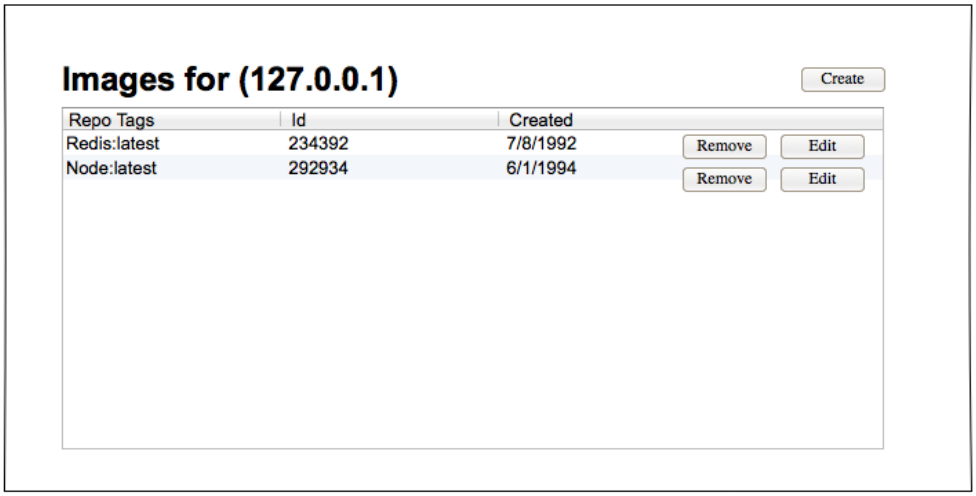

Figure 3

Figure 4

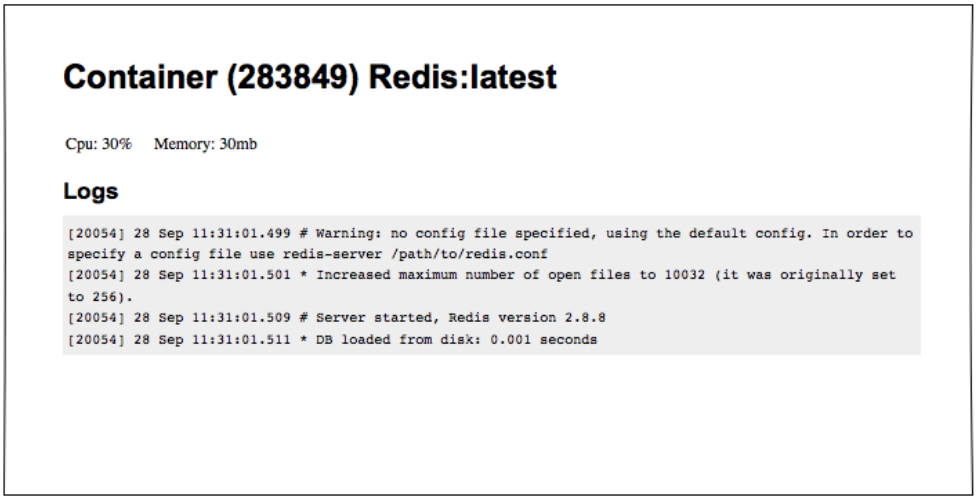

Figure 5

Functional Requirements

Administrators must be able to add, remove, edit addresses that point to a pre-constructed Docker server. These address can either be IPs or domain names. (Figure 2)

Developers/Administrators should be able to add, remove, and edit images stored on any Docker server listed in the app. These images should be pulled from a specified Docker registry whether it be Docker Hub or an in house registry. This involves asking for the image name and registry to target. (Figure 3)

Developers/Administrators should be able to start and stop containers running on any Docker server listed in the app. These containers are spawned from pre-existing images inside a target Docker server. Starting a container involves providing an exposed port, command, and environment variables to run. (Figure 4)

Developers/Administrators should be able to view running containers in real time. These views should consist of logs, CPU usage, and memory consumption for each container. (Figure 5)

Developers/Administrators should be able to login into the system with an email and password. Without both the Developers/Administrators shouldn't be allowed access to the app.

Authenticated Administrators should be able to add, remove, and edit existing developers/admins in the app.

Non-Functional Requirements

System should be secure and not allow unauthorized control. This means that our site should protect against XSS attacks, session hijacking, unauthenticated requests, and exposure of sensitive container/server data. HTTPS should be the only protocol used for web communication between a user and a server to mitigate various communication security exploits.

Code should be easy to maintain and understand. That is, commit comments should be clear and concise while following standard git conventions. Code written in go must follow the predefined style guides and effective tips from google. Similarly for Python and PEP-8. Each new contribution must also be code review by at least 2 other team members before master merge.

Work Breakdown Structure

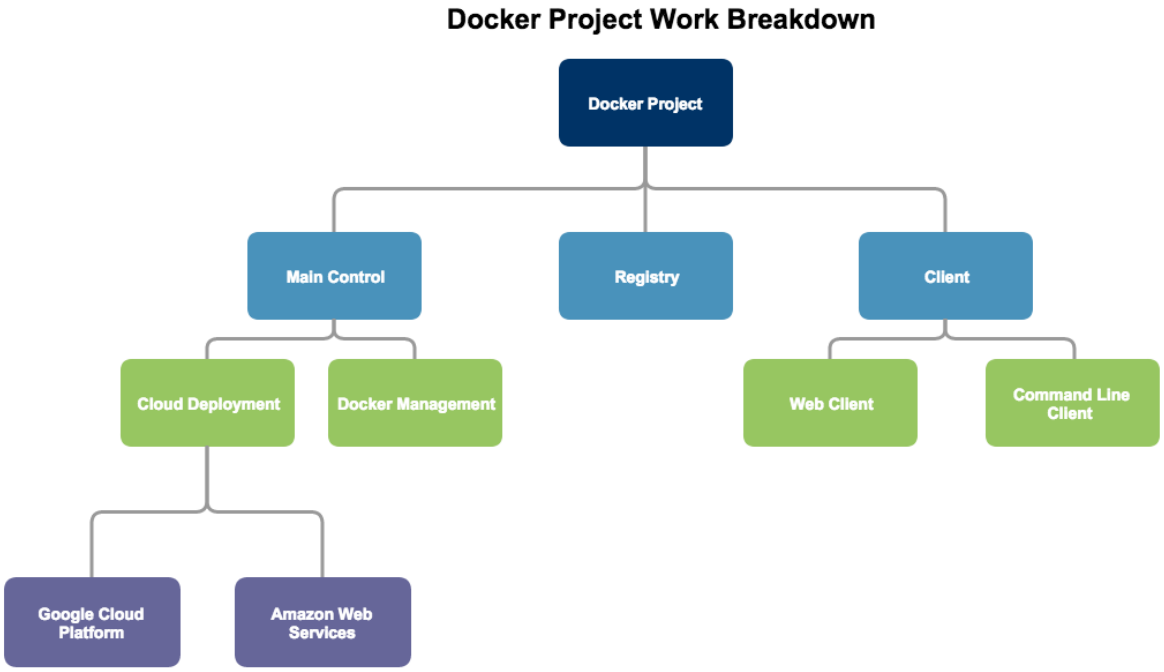

Figure 6: Project Work Breakdown

Resource Requirements

Resource	Purpose	Means of acquiring	Estimated cost
Shared Google Cloud Services developer account	Used to test and deploy our application	Provided by Workiva	Approx $20/month
Amazon Web Services developer account	Used in late development to test multi-provider functionality	Provided by Workiva	Free for low usage
Raspberry Pi	Used for automating our development process	Chris has purchased	$50

PROJECT SCHEDULE

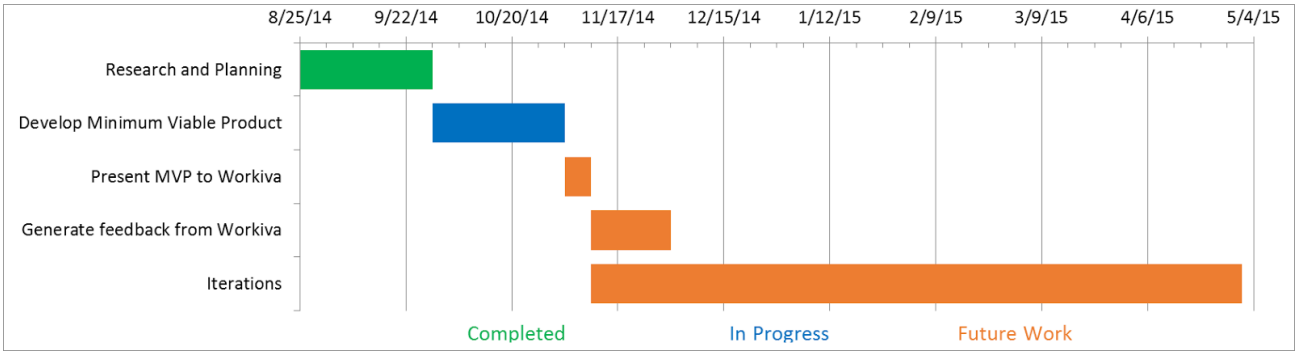

Figure 7: Project Schedule

Note: After the MVP is released to Workiva, we will be able to iterate upon this first release by taking feature requests and bug reports. For example, interfacing with additional cloud providers such as AWS will be a feature added after the initial release.

RISKS

Risks	**Impact**	**Likelihood**	**Rating**
Feature Creep - the unintended expansion of our product's features which causes loss in simplicity or scope	Major (3)	Likely (2)	Medium (6)
Competing Products - other similar products are released open or closed source and overlap in functionality of our product	Minor (1)	Very likely (3)	Low (3)

APPENDIX A: TABLE OF FIGURES