Using Containers to Isolate Remote Code Execution for an Online Development Environment

Joseph Fazzino
24026478

Supervisor: Dr. Hong Wei

29th April 2019

Contents

0.1 Abstract

250 - 300 words

outline aims, methods, implementation, achievements, and conclusions

0.2 Acknowledgements

I'd like to acknowledge Dr. Hong Wei for being my project supervisor. Dan Justin and Dr. Martine Magnan for their continued support through the early stages of my career. Suhail Parmar for contributing to my caffeine levels and providing help with Docker and UNIX. Dan Davis and Max Denning for helping me brainstorm the idea for the project and enduring me talking about frontend for the past year. And finally, my parents, Paul and Joanna Fazzino for being unwaveringly supportive and great role models.

0.3 Glossary of Terms and Abbreviations

API - Application Platform Interface

Chapter 1

Introduction

There is a theory which states that if ever anyone discovers exactly what the Universe is for and why it is here, it will instantly disappear and be replaced by something even more bizarre and inexplicable. There is another theory which states that this has already happened.

Figure 1.1: The Universe

Chapter 2

Problem Articulation & Technical Specification

2.1 Context

As computers have become more pervasive, coding has become a skill that has graduated past being something that only people who work in laboratories need to concern themselves with, to a skill that has become highly desirable commercially and is starting to be taught in the regular curriculum to children studying at a primary level education[1]. This new demand for beginner friendly coding tools lends itself nicely to the promise of an online based environment where people can get started with basic coding concepts without having to trawl through documentation and technical detail about how to get running with one of the popular languages/tools available. This has led to an explosion of popularity for web applications such as `codecademy.com` which offer pre-made, executable exercises for a number of languages. A similar platform `repl.it` offers a more open and free-form experience and attempts to recreate the environment a developer may have on their machine through the web browser along with online compilation.

2.2 Problem Statement

A common pattern with the current platforms that exist is that they provide a strict sandbox within the confines of a predetermined configuration that the user selects, for example, in `codecademy` and `repl.it` you're stuck in the environment you pick when you start desired tool. An argument can be made that this makes a new developers life easier as they don't have to consider the more nuanced parts of the file system or learn any sort of terminal commands. However, it seems as though there would be value in a system that can provide both the ease of use that current existing solutions offer and also the freedom to explore a full environment with an array of tools preconfigured that encourage exploration without compromising the security and integrity of the underlying system.

2.3 Technical Specification

The objectives of this project are:

1. Create a platform where users can write/execute code
2. Give every user their own personal environment
3. Eliminate the need for locally installed tooling
4. Provide a system that encourages exploration into the world of development

2.3.1 Writing and Executing Code

Functional Requirements

- Code will be able to be typed using the platform
- Code will be able to be saved
- Code will be able to be read from the platform
- Code will be able to be executed

Non-Functional Requirements

- A good variety of languages will be supported
- The basic features of a code editor will be available (i.e. syntax highlighting)
- Code that is executing will not stall the platform

2.3.2 Personal Environments

Functional Requirements

- A personal environment will be allocated to every user

Non-Functional Requirements

- The personal environments will be isolated from the rest of the system
- The personal environments will be isolated from each other
- The personal environments will perform well and be responsive to user input
- If a personal environment fails then it will be restarted

2.3.3 Local Tooling Replacement

Functional Requirements

- High quality tools will be available to the user
- Industry standard tools will be available to the user
- The system will eliminate the need for local tooling

Non-Functional Requirements

- Popular tools will be researched and considered before being added to the system
- Tools will be standardised across the system
- Tools will be customisable to the users needs
- Tools will behave in a responsive manner

2.3.4 Encourage Exploration into Development

Functional Requirements

- Implement exercises for users to do
- Allow creation of exercises by users

Non-Functional Requirements

- Allow any exercise to be shared
- Assign difficulty level to exercises
- Provide an open area for the user to explore their personal environment

Chapter 3

Literature Review

Chapter 4

The Solution Approach

Chapter 5

Implementation

Chapter 6

Testing: Verification and Validation

Chapter 7

Discussion: Contribution and Reflection

Chapter 8

Social, Legal, Health and Safety and Ethical Issues

Chapter 9

Conclusion and Future Improvements

Bibliography

[1] S. Chalmers, "Why schools in england are teaching 5 year olds how to code," Oct. 2014. [Online]. Available: https://www.bloomberg.com/news/2014-10-15/why-schools-in-england-are-teaching-5-year-olds-how-to-code.html (visited on 04/01/2019).