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

The following is a list of abbreviations that are commonly used in this document:

API - Application Platform Interface

RTC - Real-Time Communication

REPL - Read-Evaluate-Print-Loop

HCI - Human Computer Interaction

UX - User Experience

DX - Developer Experience

UI - User Interface

JS - JavaScript

OS - Operating System

PaaS - Platform as a Service

VM - Virtual Machine

VPN - Virtual Private Network

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

Based on the problem statement the potential scope for the project is very broad, there are companies and teams of developers that have the sole goal of making sure their online environments are providing users with as smooth an experience as they would expect if they had installed the tools locally.

This project will focus on the essential functionality required to behave as an online development environment while supporting a good variety of languages and offering a space which encourages exploration into different coding concepts.

With the above in mind the enumerated 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

As an essential requirement for the development experience, the ability to edit and execute code is crucial to satisfy the overarching objective of creating an online environment. The execution of code presents a significant technical challenge however as the only code execution that can be done remotely is on a web browser which must be able to execute HTML, CSS and JavaScript. Mobile applications developed for iOS and Android are not capable of 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

The need for the space that the user occupies to feel personal is a vital element to local development environment and therefore must be well implemented for an online equivalent.

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

Tooling has been through some big changes both in web browsers and locally. Web browsers have got to the point where they are so powerful that some of the most popular desktop software is being powered by them[2]. It is important to provide tools that will help those new to development, while also offering experience in tools that are of a high quality.

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

Lowering the barrier to entry through the requirements stated above will inherently make it easier to explore development but more steps can be taken in order to engage users with the system such as allowing them to create short coding exercises that can be shared with friends or on social media.

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

2.4 Stakeholders

This project has a number of relevant stakeholders with various degrees of interest in the outcomes. All of them will be considered during the construction of the system.

The Developer - Joseph Fazzino

The developer of the system is responsible for making 100% of the the technical decisions and is responsible for delivering a fully functioning system adhering to the technical specification found in Section 2.3 of this report.

Project Supervisor - Dr. Hong Wei

The supervisor of this project is overseeing the development and design process that is being undertaken.

They provide guidance when it comes to essential functionality and ways that technical requirements can be implemented.

User - Beginner Level Developer

Those new to development will not have experience with the terminology and syntax that exists in programming and wider computer science. They may have an understanding of basic coding concepts taught to them during formal education.

The beginner user should be able to use the system in order to become more familiar with generic programming concepts. The exercises available through the system will likely be the area they spend the most time.

User - Intermediate Level Developer

A user more familiar with the general work flow of a developer will be able to understand certain levels of nuance of how a system might be implemented and consider how they may solve certain problems.

This kind of user would benefit more from the ability to have a playground to explore the system in so they can understand the functionality that it provides and maybe try to explore the extent to which it works.

User - Experienced Level Developer

This user will have successfully developed systems with a high level of complexity and will most likely have specialised knowledge in a certain domain/environment.

This type of developer will be difficult to convince the benefits of an online working environment when they undoubtedly have a solution that works well for them locally.

2.5 Constraints

Some constraints on the development of the project exist.

- Permanent deployment - as the system is likely to be complex, deploying it will be costly and time consuming. Test deployment will be done to experiment with configuration settings in the system but a permanent live deployment will not be.
- Computer resource availability - the system will be constrained performance wise by the resources available during development meaning that any stress tests are not representative of a deployed system
- Significant testing base - as the system will not be deployed it will be difficult to adequately test the system in the manner which it would be used by end user. A different method of testing will have to be explored.

2.6 Assumptions

A number of assumptions must be made to reasonably meet the technical requirements.

- The users will have a reliable internet connection
- The users will have the necessary software/hardware configuration in order to access the system (e.g. a modern web browser)

Chapter 3

Literature Review

This chapter focuses on an analysis of existing systems in the online development space and the literature that provides context to the decisions that were made during development. It looks at the advancement of the web browser and discusses the functionality provided which can lead to this type of project to be developed. It also looks at the current state of the art of virtualization as it relates to personalised environments and how the advent of Containers has fundamentally changed the way users interact with virtual environments, as illustrated by supporting literature.

3.1 The Web Browser

The web browser has been through a myriad of changes since it's birth in the early 1990s. An excellent article titled "Mosaic and the World-Wide-Web"[3] illustrates the problems that were prominent in the early stages of the industry such as the lack of a search engine leading to difficulty in finding resources. It also illustrates what were considered leaps in progress at the time such as Mosaic being the first browser to support in-line multimedia and to have a 'back' and 'forward' button.

Such concepts that were developed during the time are still very relevant, the general TCP/IP stack had been determined and the HTTP protocol was in use. An issue with HTTP is the protocol by design has latency embedded as it is designed for sending structured messages. For a project attempting to create an environment that mirrors a local set up online latency is a big hurdle and the need for it to be real-time is key.

3.2 Real-Time Communication

An experiment was done in 2012 discussing the performance of different RTC methods by Professors at the University of New Brunswick[4].

HTTP polling is an attempt to solve the real-time issue however it is still built on top of a system not designed for real-time, full duplex communication. **HTTP long-polling** is another solution that sticks to the HTTP protocol but reduces the number of wasteful requests by having the server intelligently not respond to the request if there is no information available and hang until a timeout or information becomes available.

A modern solution to this is the **WebSocket** protocol proposed in RFC 6455 [5] which aims to reduce latency by a factor of 3 compared to HTTP in the real-time communication aspect. It is a fully duplexed, bidirectional communication channel that uses physical sockets to connect machines.

3.3 Online Developer Environments

A number of existing solutions providing online development environments exist and have been analysed for the purpose of this review.

3.3.1 Repl.it

Repl.it is very similar to the idea proposed in the Problem Statement (Section 2.2) and a lot of the requirements lined out in Section 2.3. It offers a huge array of Repl templates available for users to get started with many languages/frameworks very quickly. It also uses the Monaco Editor provided by Microsoft in order to provide a first class text editor experience.

Repl.it takes advantage of containerisation in order to gives users the full developer experience when visiting the system [6]. The system also uses it's own container orchestration software in order to scale the instances available to users up and down depending on demand.

Every code result that is available to be viewed/run is viewable through a special .repl.run subdomain. This includes long running processes like web servers which are able to be hosted from these subdomains and be always accessible. This means you could create several repls which all connect to each other like a full system.

Technically the system is very impressive, something that the system doesn't recreate quite as smoothly as a local environment would is a small amount of latency between a key being pressed and the corresponding value appearing in the REPL itself.

The system also seems to remove all previously typed entries of the REPL on every press of the *Run* button. This suggests that it is giving you a new REPL instance on every execution which isn't how a local environment works.

From a HCI point of view the website feels very smooth to use and is not frustrating to use other than the latency noted when typing directly into the running container via the REPL.

Repl.it is clearly very focused on the objective of replacing local development environments and does a good job of fulfilling that need.

3.3.2 Codecademy

Codecademy is a educational focused online environment designed to teach users how to code. Ranging in topics from beginning web development to a course to the IBM Watson API. It is a more directed experience than Repl.it as users are performing tasks for exercises but they are typing code into a similar environment, the code is executed and the result is displayed to the user.

Codecademy does allow access directly to the REPL but if code is entered into the editor which allows for user input such as the `input()` function in Python. Then it interprets the input correctly.

The Codecademy web application is clearly a very complicated system and it shows by how unresponsive it feels when navigating from page to page. The page does a full refresh even though there are elements which do not change on the screen page to page. This leads to a frustrating wait looking a blank screen between page loads.

It is clear that Codecademy is a focused environment to encourage new developers to get into development by offering an easy to start environment and heavily directed experience. It is not concerned with the idea of replacing local development environments so much as making sure that it's not something beginners should need to think of when wanted to get to know a new tool.

3.3.3 Glitch

Glitch is a web application that is focused on trying to cultivate a social coding community that encourages developers to help each other out and build mini applications with JavaScript and Node.js. It provides an online coding environment that uses containers to isolate the users runtime.

Glitch is clearly focused heavily on the social aspect as on the homepage they have a section dedicated to users asking for help so more experienced coders can help them achieve their goals with the applications they want to build. It also showcases user made projects on the homepage which can be *Remixed* which is similar to forking a repository on GitHub for other users to modify.

In terms of design, the website has a very colourful friendly interface. A feature which is particularly notable is in each project editor there is an option to view *Container Stats* where the CPU usage in %, Memory usage in bytes and additional relevant information can be found. There is also guidelines on the technical restrictions to projects that are run in Glitch.

3.4 Virtual Machines and Containers

3.4.1 Virtual Machines

Virtualisation is a technique in computing that, most commonly, is seen by users in the **Virtual Machine** *(VM)* software. Virtual Machines are heavily utilised to provide virtual desktop environments on top of a users already existing desktop. The advantages of which are a sandbox environment for potentially harmful operations, such as when penetration testers are trying to fingerprint a virus. The option of trying a different OS without needing to dedicate a partition of disk space to it or deal with a dual booting set up is another user facing benefit of virtual machines.

In the enterprise world, Virtual Machines are being used to host their customers appli-

cations in a full PaaS solution so customers no longer have to worry about hosting their own web servers or other online services.

The general way of interacting with fully virtualised environments is through a hypervisor which is a tool that is responsible for provisioning and monitoring Virtual Machines [7]. The hypervisor allocates resources such as memory and CPU cores from the host machine that the VM is allowed to consume. When the VM is shut down these resources are freed and can be used by the host system once again. The hypervisor also allows the VM to use a different base operating system than the one that is on the host machine as it provides a whole *Guest OS*.

3.4.2 Containers

Containers are a much lighter virtualisation method than Virtual Machines despite the functionality being similar. They achieve this as they are much closer to the systems 'bare metal' as any commands that are executed through a container are running on the host's hardware. This means that there is no need for a hypervisor as containers have direct access to the resources. Usage limits can be set in the configuration of container *images* which will be discussed further during this subsection.

As Containers traditionally don't utilise a hypervisor the biggest difference between them is that the engine that powers the container provisioning software such as the *Docker Engine* isn't able to virtualise an environment based on a different OS. This is more by design however as it is what gives containers their 'light weight' quality as they aren't having to simulate a whole kernel.

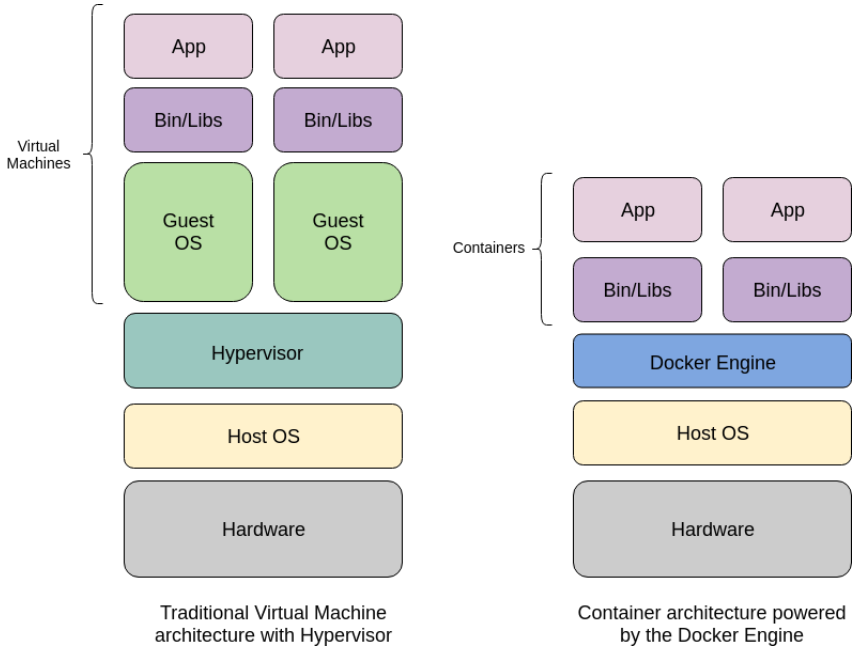

Figure 3.1: Architecture of Virtual Machines vs. Containers

https://ieeexplore.ieee.org/abstract/document/6903537

‘Talking about containers vs VMs for PaaS software

https://ieeexplore.ieee.org/abstract/document/7158965

‘More focused on Docker

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).

[2] T. Claburn, "Carlo has a head for apps and a body (tag) for rendering: Google takes on electron with js desktop app toolset," Nov. 2018. [Online]. Available: https://www.theregister.co.uk/2018/11/02/carlo_chromium/ (visited on 04/02/2019).

[3] R. J. Vetter, C. Spell, and J. Ward, "Mosaic and the world-wide-web," Oct. 1994. [Online]. Available: http://vision.unipv.it/wdt-cim/articoli/00318591.pdf (visited on 04/03/2019).

[4] V. Pimentel and B. G. Nickerson, "Communication and displaying real-time data with websocket," May 2012. [Online]. Available: https://ieeexplore.ieee.org/document/6197172 (visited on 04/03/2019).

[5] I. Fette and A. Melnikov, "The websocket protocol," Dec. 2011. [Online]. Available: https://tools.ietf.org/html/rfc6455 (visited on 04/03/2019).

[6] F. Lardinois, "Repl.it lets you program in your browser," 2018. [Online]. Available: https://techcrunch.com/2018/03/15/repl-it-lets-you-program-in-your-browser/ (visited on 04/03/2019).

[7] M. G. Sumastre, "Virtualization 101: What is a hypervisor?," Feb. 2013. [Online]. Available: https://www.pluralsight.com/blog/it-ops/what-is-hypervisor (visited on 04/04/2019).