Waterford Institute *of* Technology

Project Report

Bachelor of Science (Hons) Applied Computing

RubiX - Functions as a Container

Ciaran Roche - 20037160

Supervisor: Dr. Rosanne Birney

April 16, 2019

Abstract

Modern cloud centric computing has seen a number of technologies and approaches emerge as candidates for auto-inclusion in development stacks. In particular, Kubernetes and developing for a serverless world are fast becoming the de facto way of working as their benefits are clear. Simple, yet clever configurations can bring unrivalled scalability, coupled with huge cost saving over the lifecycle of a service. These are among the primary drivers for attracting developers towards this paradigm. However, the barriers to entry for serverless development are high. There exists a steep learning curve as well as mundane scaffolding which obscures the valuable configurations developers should be focusing on. RubiX is an Open Source Framework aimed at reducing the barrier of entry to serverless development on Knative, the Kubernetes-based platform to build, deploy, and manage modern serverless workloads. Using RubiX, a developer can focus on their functions, with the freedom to develop in a fluid manner, using a number of SDK's which support multiple languages. This allows developers to focus on the real problems at hand and take advantage of being able to deploy in a serverless world.

Plagiarism Declaration

Unless otherwise stated all the work contained in this report is my own. I also state that I have not submitted this work for any other course of study leading to an academic award.

Acknowledgements

A very special gratitude goes to my supervisor Dr. Rosanne Birney for her advice and guidance through out this semester. Without it my project would not be where it is today. I also wish to extend my gratitude to the entire computing staff, who gave me the drive and motivation needed to push myself beyond my abilities.

My sincere thanks to Dr. Leigh Griffin, Aiden Keating, Laura Fitzgerald and the entire team at Red Hat Waterford. Their guidance and support throughout my internship and extending to my project has been incredible. Giving me not only the knowledge and tools needed for my final year of college but to progress my career in this field.

To all my family, without their encouragement and support I would not be writing this today. For that I will be forever grateful.

To Amy who has been an inspiration to me throughout this journey. For listening to me waffle on for hours, and sharing our time together with my laptop, I will be eternally grateful. Now to get our weekends back.

And finally, last but by no means least my fellow students. Their collaboration and friendship has made my time at WIT a memorable one. I wish you all the best in your FYP's and your future careers.

Contents

List of Figures

List of Tables

Glossary

API Application Programming Interfaces, a set of functions and procedures that allow the creation of applications which access the features or data of an operating system, application, or other service. ix, 16, 32

AWS Lambda Amazon Web Services Lambda, a service that lets you run code without provisioning or managing servers. 1

BaaS Backend as a Service, a cloud computing service model that serves as the middleware that provides developers with ways to connect their web and mobile applications to cloud services via API's and SDK's. 1

BSD-style license Berkeley Software Distribution style license, Is a family of permissive free software licenses, imposing minimal restrictions on the use and distribution of covered software. 15

Buildah A tool that facilitates building Open Container Initiative container images. 16

CLI Command Line Interface, is a means of interacting with a computer program where the user issues commands to the program in the form of successive lines of text. 3, 14–16, 32

Cloud Foundry Foundation A nonprofit that oversees an open source platform and is a collaborative project of the Linux Foundation. 1

Container An isolated piece of software bundled with everything it needs to run on the host. 3, 6, 16

CRI Container Runtime Interface, at the lowest layers of a Kubernetes node is the software that starts and stops containers. ix

Cri-O Is an implementation of the Kubernetes CRI to enable using OCI compatible runtimes. It is a lightweight alternative to using Docker as the runtime for kubernetes. 16

CVE Common Vulnerabilities and Exposures is a dictionary-type list of standardized names for vulnerabilities and other information related to security exposures. CVE aims to standardize the names for all publicly known vulnerabilities and security exposures.. 24

Docker A piece of software which enables orchestration and management of containers on a host. 3, 15, 16, 27, 32

FaaC Functions as a Container, a term coined for this project as best describes the end result. Being a function that is a container allowing a user utilize Frameworks such as Knative. 2

FaaS Functions as a Service, a service is a category of cloud computing services that provides a platform allowing customers to develop, run, and manage application functionalities. Without the complexity of building and maintaining the infrastructure typically associated with developing and launching an app. 1

GitHub Is a web-based hosting service for version control using Git. 17

GoLang Is an open source programming language designed by Google. Go is a statically typed, compiled language in the tradition of C, with the added benefits of memory safety, garbage collection, structural typing, and CSP-style concurrency. 14–18

Kata Is a new open source project building extremely lightweight virtual machines that seamlessly plugs into the containers ecosystem. 16

Knative A set of middleware components that are essential to build modern, source-centric, and container-based applications that can run anywhere – on premises, in the cloud, or even in a third-party data center. ix, 1, 2, 16

Kubernetes An open-source system for automating deployment, scaling, and management of containerised applications. 2, 3, 15, 27

OCI Open Container Initiative, Is a lightweight, open governance structure (project), formed under the auspices of the Linux Foundation, for the express purpose of creating open industry standards around container formats and runtime. ix, 16

Open-Source Is a type of computer software whose source code is released under a license in which the copyright holder grants users the rights to study, change, and distribute the software to anyone and for any purpose. 2, 14, 15

Quay.io is a cloud hosted service for building, storing and distributing Docker containers. 6

SDK Software Development Kit, Is a set of programs used by a computer programmer to write application programs.. ix, 3, 16, 17, 31

Serverless Applications that significantly depend on third-party services (knows as Backend as a Service or "BaaS") or on custom code that's run in ephemeral containers (Function as a Service or "FaaS"). 1–3, 5, 17

1 Introduction

1.1 Motivation

As seen in Figure 1, serverless computing is the number one growing cloud service (*RightScale 2018 State of the Cloud Report*, 2018). As with many trends in software, there is no one clear view of what serverless is. It was first used to describe applications that fully incorporate third-party, cloud-hosted applications and services. Another term often used to describe these type of architectures is Backend as a Service or BaaS. Today serverless is the term used to describe applications where the server-side logic is written by the applications developer but unlike traditional architectures it is event-triggered, meaning that a server is not constantly running but instead events trigger the compute containers (Fowler, 2018). This is often referred to as Functions as a Service or FaaS. A popular implementation of this is AWS Lambda. For the purpose of this report. I will be using the (Fowler, 2018) definition of serverless.

Place	Service	Growth Rate	2017 Use	2018 Use
#1	Serverless	**75%**	12%	21%
#2	Container-as-a-service	**36%**	14%	19%
#3	DBaaS SQL	**26%**	35%	44%
#4	DBaaS NoSQL	**22%**	23%	28%
#5	DRaaS	**21%**	14%	17%

Figure 1: *Top Growing Cloud Services* – RightScale 2018 State of the Cloud Report *(2018)*

One of the main benefits and motivation for the project is that it reduces operational cost, as one only pays for the compute power they need as opposed to a constant running server instance. Another aspect of the operational costs stem from the development as developers can focus on business logic of their application over building the infrastructure around their application (Fowler, 2018).

The Cloud Foundry Foundation[1] conducted a global survey of their users and found that 22% are already using serverless technology (*Where PaaS, Containers and Serverless Stand in a Multi-Platform World*, 2018). And with almost half their users evaluating the technology. This kind of trend is supported by research conducted by Cloudability (*State of Cloud 2018 Report*, 2018). This suggests that the area of serverless computing is growing rapidly.

Further backing the growth of serverless, Google announced the launch of their Knative[2] project

[1] https://www.cloudfoundry.org – a multi-cloud application platform.
[2] https://cloud.google.com/knative/ – middleware components for container-based applications.

which is based on the Kubernetes[3] engine and developed by Google, Pivotal, IBM, Red Hat and SAP. Knative provides a suite of developer-focused middleware tools for building, deploying and managing serverless applications on the Kubernetes engine (Bryant, 2018).

The adoption rate of serverless technologies, mixed with a personal interest in the migration of conventional monolithic applications to microservice serverless applications, led me to explore problems around serverless that could be solved as a topic for my Final Year Project.

1.2 Problem Statement

There are a number of identified problems with serverless technologies. A report by DigitalOcean highlighted that the biggest challenge faced by serverless was the ability to monitor and debug (*Currents: A quarterly report on developer trends in the cloud*, 2018). This was shown to be down to the layers of abstraction by particular vendors and their serverless services.
While Knative looks after the orchestration of a serverless application within Kubernetes seamlessly, thus reducing the number of layers of abstraction, it is up to the developer to build, configure and package their functions in a container. This pattern goes against one of the main benefits of serverless in that a developer can focus their time on the business logic over the configuration of the infrastructure thus reducing operational costs.
This project will look to solve both problems outlined above and will be discussed in detail in Section 1.3.

1.3 Aims and Objectives

With the aim to solve the problems outlined in Section 1.2 the overall goal is to provide an open-source tool to allow software engineers to easily create FaaC to be utilised by Knative.

AO1 - Provide support for building FaaC in multiple languages. This will reduce the cost of entry, as developers can use a language that best fits their use case or expertise.

AO2 - The ability to easily debug functions. Solving the main identified problem with serverless, providing the means for easy debugging will help increase adoption rate.

AO3 - Confidence in a Function as a Container being as secure as possible. By removing layers of abstraction the developers can have confidence in their containers being up to date and as secure as possible.

AO4 - The project to be transparent and developed in the open. This will allow for the community to mould and adapt the project to best fit their needs, overall increasing the quality and the uptake of the project.

[3]https://kubernetes.io – container-orchestration system for automating deployment.

AO5 - The means to control the life cycle of a Function as a Container. This will increase the speed of local development by providing the developer with the tools needed to fit their development needs, thus reducing the need to switch between tools.

The project will utilise common patterns that would be native to a software engineer already working in the space of Containers and Kubernetes. It will consist of the following:

- **CLI:** An easily extensible command line interface following the same work flow such as tools like Docker CLI and Kubernetes Kubectl.
- **SDK's:** A suite of SDK's to support the creation of Functions as a Container in multiple programming paradigms.
- **Showcase Application:** To demonstrate a modern serverless application running in the Kubernetes engine.

1.4 Contributions

The end result of this project will be a Framework which provides a suite of SDK's and a CLI that will reduce the barrier of entry to serverless development. The project will be developed in the open with the goal to build a community that will allow the development be guided by the needs of the serverless community as a whole. On the back of the project a number of applications to talk at conferences will be submitted, with a local tech talk already confirmed for April 2019.

1.5 Outline

This report is broken into seven sections. This section, the first, is an overall introduction covering motives and the problem to be solved along with contributions that I intend to make. In section two, the work carried out in Semester One is summarised. The third section looks at the methodologies this project follows. Section four examines the technological choices made. Section five outlines the tools used to aid development of the project. The system architecture, requirements and user stories make up the design of the project, which is captured in Section six. Section seven discusses the implementation of the project, consisting of Scrum Ceremonies along with personal reflections on each sprint. Finally section eight reviews the project direction, reviewing the work completed. The learning outcomes and a final personal reflection are also included in this section.

This project is sponsored by Red Hat, acting as stakeholders in the project. As stakeholders Red Hat fulfill the role of product owner ensuring the development of this product meets the requirements set to solve the problem stated in Section 1.2. This is not a paid or monetary sponsorship.

This project is supervised by Dr. Rosanne Birney.

2 Semester One Summary

Throughout the first Semester I set out to identify a problem within the serverless space. Upon identifying the problem a proposed solution was sought. This led to the identification of technologies which could solve the problem. A number of tools and methodologies were then chosen to help in the development of this solution. A number of design patterns were explored and looked at, resulting in the architecture chosen for development this Semester.

To ensure full technical feasibility, two Prototypes were developed. The first was used as a benchmark for further iterations, ensuring that work carried out within this project not only improves the user experience, but also improves on the existing solution technically. The second prototype was developed following the design laid out in Section 6.

2.1 Prototype 1

As outlined in Section 1.3, essentially a Function as a Container is a function wrapped in a server and packaged as an image. Through the development of this prototype a server was built in Typescript. Included in this server were two endpoints that satisfy a HTTP Get Request and a HTTP Post Request. This was built with the aid of a common Node Module *Express*[4] and followed recommended patterns of Middlewares / pipelines. This is a simple abstraction in that the output of one unit/function is the input for the next
This prototype serves as a benchmark for further implementations as the project looks to provide a better solution to the development of a Function as a Container. It provides a baseline of a typical Container for us to compare. The image for this prototype can be found at Appendix F.

2.2 Prototype 2

The resulting code from this Prototype of the SDK can be found at Appendix G, along with a showcase application built to demo it, which can be found at Appendix H. Completed in this Prototype is the ground work for the SDK, written in Typescript allowing a user to create a Function as a Container. This provided the feasibility for the project and the design outlined throughout this paper.

Prototype	Size	Vulnerabilities-	Latency
One	75.3MB	90	15ms
Two	23.4MB	0	7ms

Table 1: Prototype Comparison Table

2.3 Prototype Summary

A number of arbitrary statistics were taken from the two prototypes based on freely available statistics from Quay.io[5]. These results can be seen in Table 1. As we can see our prototype following the design structure outlined in this report has made improvements in both size and latency, while reducing the number of vulnerabilities within our Container. This in my opinion is due to stripping away third party modules and dependencies, as it allows us to focus on what we need with out accumulating a large dependency tree which we do not control. This prototype will now be focused on for the remainder of this project.

[4]https://expressjs.com – Minimalist web framework for Node.js
[5]https://quay.io – Container Repository

3 Methodology

Agile is the chosen methodology that this project followed, and in particular the Scrum variation of Agile. The main reason behind chosing Scrum was that all decisions made within the Scrum methodology are based on what is known, this will be discussed in Section 3.1. Scrum is also the most popular variant of Agile (*State of Agile Report*, 2018) and is well understood by the community.

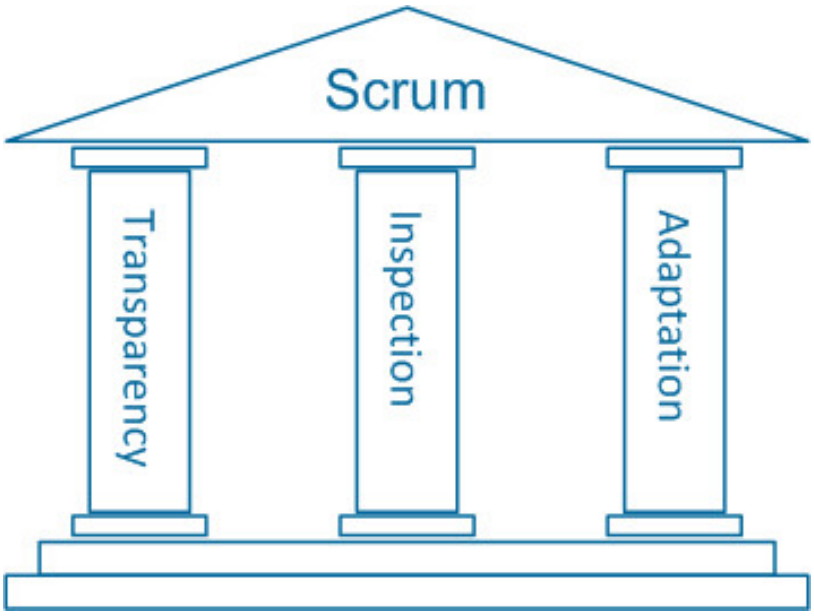

Figure 2: *3 Pillars of Scrum* The Scrum Guide™ *(2018)*

3.1 Agile

The Agile methodology comes in many flavors and frameworks (*Manifesto for Agile Software Development*, n.d.), and those who follow an Agile methodology do so by adapting it to best suit their needs. The same approach is taken for this project in that it follows closely to the Scrum framework. The Scrum Guide defines Scrum as a "framework for developing, delivering and sustaining complex products" (Schwaber and Sutherland, 2017). The Scrum framework suited this project due to it being based on empiricism, which asserts that decisions are based on what is known. It employs this decision flow through an incremental approach to optimise predictability and to control risk. The Scrum framework can be broken into three pillars: transparency, inspection and adaptation.

3.1.1 Transparency

All aspects of a process must be visible to all who are involved and responsible for the outcome. Transparency requires these aspects to be defined so that any observers can share an understanding. I will achieve this by my Trello Cards and these reports.

3.1.2 Inspection

Scrum employs frequent inspections of Scrum artifacts, which are outlined in Section 3.3, and the overall progress toward a Sprint Goal to detect and identify and undesirable variances. These inspections should not be so frequent that they take away from the work but that they increase the standard and quality of the deliverables from each Sprint. The Scrum Artifacts, Section 3.3, will allow me achieve this.

3.1.3 Adaptation

During these inspections if it is determined that one or more aspects have deviated from the acceptable limits then the process must be adjusted. All adjustments must be made as soon as possible to minimize any further deviation. This will be achieved through the Scrum Artifacts.

3.1.4 Acceptance Criteria

For every story in the project, Acceptance Criteria will be set. It is this criteria that will ensure requirements are met per story and will be used by myself as a Developer and my ScrumMaster to ensure work is complete.

3.1.5 Definition of Done

A critical aspect to Scrum is the Definition of Done. This will be a list of activities that will be used to verify that the criteria is met and the features are truly complete.
As a lone developer in a time sensitive project, being able to definitively know when I am done with a task and can move on is essential to ensure the project meets its deadline and a product is delivered. The Definition of Done for this project is defined and discussed in detail in Section 3.4.

3.2 Scrum Roles

Throughout this project a number of people took on a number of different roles to fit with the Scrum Methodology. Some roles have been adapted to suit this project and are outlined below.

- **Scrum Master** - Dr. Rosanne Birney. Project supervisor Rosanne will also take the role as Scrum Master. As Scrum Master to this project Rosanne will facilitate weekly stand ups to ensure work is progressing with minimal impediments. This ensures that the project concludes with a high value product.
- **Product Owners** - Dr. Leigh Griffin (Red Hat) and Laura Fitzgerald (Red Hat). Taking an adapted approach to Product Owners it is their job to guide the overall scrum team through the scrum process. In comparison to the traditional Product Owner taking an authoritative role, this authority falls to myself as Developer to this project where I retain general say.
- **Developer** - Ciaran Roche. Following the traditional approach to a developer within a Scrum team. It is my job to do the work of delivering a potentially releasable product at the end of each Sprint. To that end all traditional team roles, such as front end, backend, UX will fall to me.
- **Stakeholder** - Aiden Keating (Red Hat). It is the job of the stakeholder to advise decisions based on their knowledge of the technologies used. As Aiden possesses several years working within the Kubernetes ecosystem along with many open source projects, his insight and advise will be critical during the early development days prior to community involvement.

3.3 Scrum Artifacts

Figure 3 shows artifacts as outlined in the Scrum Guide (Schwaber and Sutherland, 2017) which allow for inspection and adaptation of aspects of this project and how they are incorporated around a Sprint.

- **Sprint Planning** - There was planning sessions to evaluate the project scope and refine the backlog to best suit the needs of the project. These meetings involve the Product Owners. This meeting defined a goal for the next Sprint and looked at taking large development tasks and break them into smaller tasks and fit them to meet the overall goal for the coming Sprint.
- **Sprint** - The sprint is a time boxed unit where development work is carried out on the tasks assigned during the Sprint Planning artifact. These sprints were relatively short to suit with the college semester and to allow for highly achievable goals and giving time to allow for any change. The sprints consisted of two weeks to allow enough time for a shippable increment.
- **Sprint Review** - Once a sprint was over a review meeting took place and the work toward the

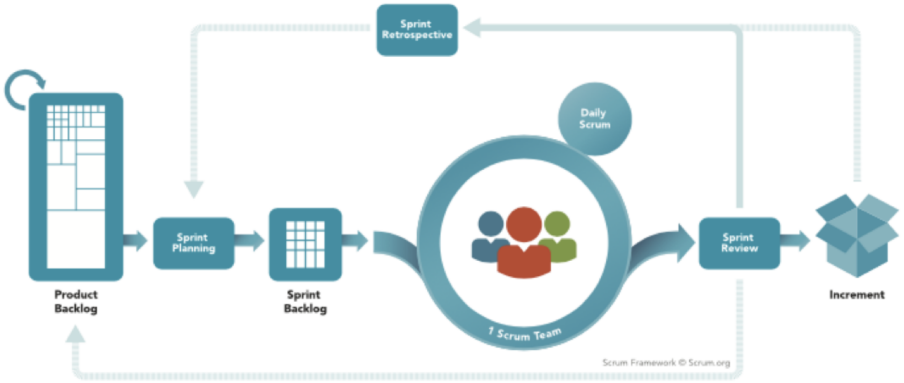

Figure 3: *Scrum Process Outline* The Scrum Guide™ *(2018)*

sprint goal was evaluated. All work completed was shown in a demo to the product owners. This helped gauge progress of the project and helped with defining the next sprint during the following planning meeting. It also allowed us review the priority of the backlog.

- **Sprint Retrospective** - This meeting took place following the review meeting, where it is looked at for areas that worked well and more so areas that did not work during the previous sprint. This allowed scope for improvements in sprint performance, the outcome was taken into consideration during the following planning meeting. This rapid feedback and course correction ensured the success of the project.

3.4 Completion and Handover

The development for this project will have the following stipulations:
The Definition of Done for each sprint will be as follows:

- Each User Story has been tracked within JIRA.
- Each User Story is implemented.
- Each JIRA ticket contains a link to completed work.
- A Sprint Review and Sprint Retrospective meeting has been held.

The project will adhere to the following Definition of Done:

- User Stories have been categorised as MVP and Stretch Goals during initial backlog refinement.
- All User Stories belong to a Theme and contain an overview story.
- All code is committed to GitHub repositories belonging to Rubix Functions Github Org.
- All code is accompanied by documentation
- Latest image is available on Quay.io
- Latest package(s) are available at chosen repository.

This project required the following deliverables upon completion:

- Presentation of the project.
- Project Report.
- Descriptive poster of project.
- Demonstrative video of project usage.

3.5 Continuous Integration

Continuous Integration is the process where work is integrated frequently, usually on a daily basis by multiple people which leads to multiple integrations per day (Fowler, 2006). While this project was developed by a single person it had a continuous build cycle, this ensured a high quality overall, thus reducing the number of bugs. To achieve this, integration pipelines were built for each repository associated with this project. The tools to be used will be outlined in more detail in the following sections.

Figure 4 shows the outline of these pipelines. When work was committed to GitHub through tool integrations with GitHub a number of checks were triggered. Circle CI, which is discussed in

Section 5.2.1 was responsible for building the repository and carrying out all unit tests in an isolated environment. Coveralls, which is discussed in Section 5.2.2, checked the code base for test coverage thus reducing technical debt.

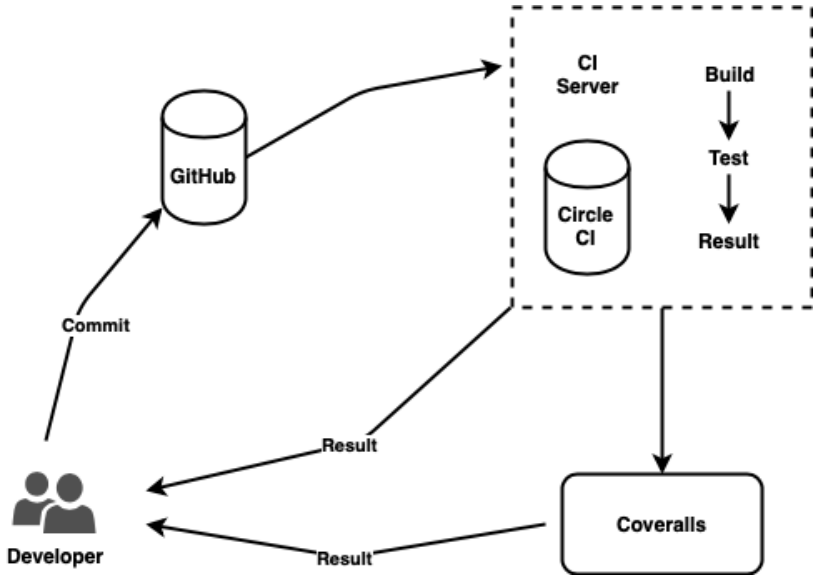

Figure 4: *Continuous Integration Workflow*

3.6 Continuous Integration/Continuous Delivery

Due to short development sprints, producing a shippable increment every two weeks Continuous Delivery was built into the Continuous Integration allowing for the automation of releases and bug fixes.

3.7 Testing Approach

Behavior Driven Development (BDD) was chosen for the approach to be followed for this project as Behavior Driven Development is the combination of Test Driven Development, domain-driven design and object-oriented analysis.
BDD is a set of practices that aim to reduce common wasteful activities in software development, such as rework needed due to misunderstood or vague requirements, or technical debt due to reluctance to refactor code. It achieves this by using a natural language to express desired behaviors and expected outcomes, based on these constructs test scripts are written and implemented into a CI/CD workflow (Solis and Wang, 2011). The result is a closer relationship to the acceptance criteria set at the user story level, thus BDD aligns closely with the chosen Agile Methodology which, was discussed in

Section 3.1.
Upon completion of the POC discussed in Section 2, constructs will be written outlining all expected and desired behaviors. These constructs will be used throughout the project to drive the development and tests to be written.

3.8 Open Source

To ensure the project followed the Open Standards Requirement for Software set by the Open Source Initiative. In order to comply with the requirement and ensure that the project is legally Open Source, a number of criteria where met (*Open Standards Requirement for Software*, 2018).

- **No Intentional Secrets** - The project must not withhold any detail that is necessary for interoperable implementation. No versions of the project can be released if in violation to the OSR
- **Availability** - The project must be freely and publicly available under royalty-free terms.
- **Patents** - All patents associated with the project must be licensed under royalty-free terms for unrestricted use or be covered by a promise of non-assertion when practiced by open source software.
- **No Agreements** - There must not be any requirement for execution of a license agreement.
- **No OSR-Incompatible Dependencies** - Implementation of the standard must not require any other technology that fails to meet the criteria of this requirement.

A number of User Stories were defined around complying with the standard and creating the foundation for an Open Source community. These can be seen in Section I.6.

4 Technologies

In order to solve the problems outlined in Section 1.2, the correct technologies needed to be chosen. To ensure a correct fit a number of investigations were undertaken. This fits in sync with the Agile methodology, Section 3.1, in that through early and often prototyping and investigation, the correct technology is chosen. The choices made resulted in a prototype application which is discussed in detail in Section 2. This section will be broken into the following sub sections, each outlining different aspects of the project and the technologies chosen which are to be implemented early on in the Project:

- Command Line Interface
- Container Build Tools
- Software Development Kit

4.1 Command Line Interface

4.1.1 Investigation

Acceptance criteria, Section 3.1.4, was set in order to ensure the correct technology was chosen for the CLI.

- Must be easily extensible.
- Must be modular in design.
- Must appeal to developers.
- Must follow patterns users are familiar with.
- Must have minimal footprint.
- Must meet open-source criteria.

Based on the criteria, it was decided to go with GoLang and the reasoning will be discussed in detail in Section 4.1.2. Other technologies looked at include, NodeJs and Python but were dismissed because, while satisfying some of the acceptance criteria I felt GoLang would increase overall uptake of the project within the Open Source Community.

4.1.2 GoLang

GoLang[6] is an open source project developed by Google and distributed under a BSD-style license. GoLang was designed to be a modern computer programming paradigms to enable higher productivity while taking aspects from a number of different languages. GoLang is a specifically engineered language unlike others which have evolved in that the designers based characteristics of the language on what they considered to be the best elements of other modern languages. For example it is statically typed and efficient in the style of C++ and Java. It is productive and intuitive similar to that of Python or JavaScript.

Some of the main benefits include garbage collection and native concurrency while its novel type system allows for flexible and modular program construction (*GoLang*, 2018). The Stack Overflow Survey 2018 (*Stack Overflow Developer Survey 2018*, 2018) continues to show signs in GoLang's rise in popularity between developers which can be seen in Figure 5. This rise could be attributed to, tools like Docker and Kubernetes being built in GoLang. As these tools are open-source, willing contributions to them are made in GoLang. Utilizing the same language and libraries used to build these tools in the development of the project satisfies a number of the criteria set out in Section 4.1.1. GoLang appeals to developers, follows patterns users are familiar with and has a minimal footprint while meeting the open-source criteria.

Figure 5: *Stack Overflow Most Loved Languages* – Stack Overflow Developer Survey 2018 *(2018)*

GoLang features a plugin system. This allows developers build loosely coupled modular programs using packages that are compiled as shared object libraries. These libraries are then loaded and bound together dynamically at runtime (Vivien, 2017). By using GoLang's plugin system it will allow for greater extensibility in that each supported language by the tool will have its own library within the CLI maintaining a more modular structure to the overall design.

[6]https://golang.org – Google engineered programming language

4.2 Container Build Tools

4.2.1 Investigation

As of June 2015 due to the rise in popularity in container technologies, with emphasis on the rise of Docker, the Open Container Initiative (OCI) was born. It is the goal of the initiative to form an open standard in the Runtime Specification and the Image Specification of containers. Meaning containers could be built and run by a combination of tools and technologies (*OCI*, 2018). Just three years from the launch of OCI a number of interoperable tools are available. To find the best fit OCI tool for this project acceptance criteria was set for this investigation. The criteria is:

- Must be interoperable with technology chosen for CLI
- Must be OCI compliant
- Must support Runtime Specifications for local testing
- Must support Image Specifications for interoperablity with Knative

Based on the criteria outlined, Docker was chosen for the project and will be discussed in detail in Section 4.2.2. Other technologies looked at during this investigation have been Kata[7], Buildah[8] and Cri-O[9]. However, they were not deemed suitable because of the unavailability and maturity of their APIs.

4.2.2 Docker

Docker is a Container platform that controls the entire life cycle of a Container. A Container is a unit of software that packages up an applications code and all its dependencies and enables it to run on any computing environment with a suitable runtime. A Docker image is an executable package which includes everything needed to run. It is not until runtime that an image becomes a Container. This runtime can be any OCI compliant engine (*Docker*, 2018).

For the purpose of the project the focus will be on both the creation of an image and the execution of the image on the Docker engine. This can be satisfied through the use of the Docker GoLang SDK. An example can be seen in Figure 6, which shows the GoLang equivalent to using the Docker client to list all running containers within the Docker engine.

This shows that Docker satisfies the criteria set out in Section 4.2.1, as it is interoperable with the chosen technology for the CLI in Section 4.1. Docker is also OCI compliant and supporting both Runtime Specification and Imagine Specifications.

[7]https://katacontainers.io – Open Source Container building tool
[8]https://github.com/containers/buildah – Open Source Container building tool
[9]http://cri-o.io – Open Source Container runtime

```go
package main

import (
  "fmt"

  "github.com/docker/docker/api/types"
  "github.com/docker/docker/client"
  "golang.org/x/net/context"
)

// new environment
func main() {
  cli, err := client.NewEnvClient()
  if err != nil {
    panic(err)
  }
    // list all container options
  containers, err := cli.ContainerList(context.Background(), types.
      ContainerListOptions{})
  if err != nil {
    panic(err)
  }
    // foll all containers, print
  for _, container := range containers {
    fmt.Println(container.ID)
  }
}
```

Figure 6: *Docker SDK Example*

4.3 Software Development Kit

4.3.1 Investigation

It is the SDK's job to abstract all of the configuration away from the user and allow them to focus on the logic behind their functions. The SDK will add the most value to the project by reducing the cost of admission to the development of serverless functions. Due to this fact, a great deal of consideration had to be given to choose the initial supported paradigm. As this would be the flagship SDK to this project and will be used for the Community Launch of the project as well as demo's at a local tech talk.

The approach taking in this project involved looking at the most common and popular programming languages. Thus giving a greater probability in the tool being of value to users and reducing the knowledge needed to use the tool. As can be seen in Figure 7, from the Stack Overflow Survey (*Stack Overflow Developer Survey 2018*, 2018), JavaScript is top of the poll. This is further backed from the GitHub Insights Statistics 2018 *The State of the Octoverse* (2018), viewable in Figure 8, which shows a total of 2.5 million Pull Requests. Pull Requests are a method for contributing to projects on GitHub, thus being a good indicator for usage of GoLang through contributions.

Figure 7: *Stack Overflow Most Popular Technologies* – Stack Overflow Developer Survey 2018 *(2018)*

While based on the results of supporting JavaScript and supplying an SDK for JavaScript a decision was made to develop the SDK in TypeScript. The reasoning behind this decision will be discussed in Section 4.3.2. A number of other supported languages where also chosen, these were put into the product backlog and completed by the end of the project. Among the other languages chosen where Python, GoLang and Java.

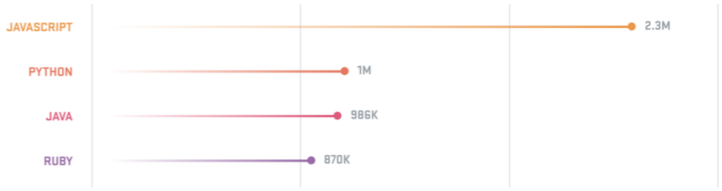

Figure 8: *GitHub Number of Pull Requests* – The State of the Octoverse *(2018)*

4.3.2 Typescript

TypeScript could be considered as modern JavaScript. This consideration is made as in comparison to JavaScript, TypeScript is strongly typed in that you must declare types and variables. Types can be summed up as the simplest units of data such as numbers, strings, structures and boolean values. Where as JavaScript, being dynamically typed, does not know what type a variable is until it is instantiated at run-time. This can lead to problems by false assumptions on variables, whereas with TypeScript these false assumptions can be caught during development. With added features like strict null checks, interfaces and schemas, TypeScript will overall improve the quality and usability of this project (*TypeScript*, 2018).

Further backing the decision to use TypeScript is its interoperablity to JavaScript this means that JavaScript modules can be used and understood by the TypeScript compiler. It also means that TypeScript transcompiles down to JavaScript meaning that any TypeScript code can be used in a JavaScript ecosystem. This will allow for the leveraging of the community and ecosystem behind JavaScript and being able to use available modules will overall decrease development time and allow for more complex features with less overhead.

5 Tools

5.1 Project Management Tools

5.1.1 JIRA

JIRA is a proprietary issue tracking product developed by Atlassian. It allows for the planning, tracking and management of Agile projects with stories and tasks broken down into a series of JIRA tickets. These tickets are stored in the backlog within JIRA allowing for the planning of sprints and managing the overall goal of each sprint.
As discussed in Section 6.6, to meet the void between JIRA, Agile and Scrum in what defines an epic, story and task a number of rules were set. Each story that was defined had a number of tasks associated with. These tasks were small achievable increments that would lead to satisfying the acceptance criteria set for each story. The story would be seen in JIRA as an Epic and the corresponding tasks would be outlined as tickets within JIRA.

5.1.2 Trello

Mentioned in Section 5.1.1, there is a disconnect between between JIRA, Agile and Scrum definitions. Due to this it was decided to incorporate Trello within the planning phases of the project. Trello is an online tool for tracking and project management. Due to its ease of use and a high standard of visualisation it made it easier to plan the User Stories for this project. The board used for planning can be found at Appendix C. This also allows for a higher level of abstraction, with Trello being more suited to story level discussions and JIRA more granular, technical level.

5.1.3 Trira

As Trello has a rich API a number of tools and plugins have been made to increase the usability of Trello. One tool which is used throughout this project is the Open-Source tool Trira. It can be found at Appendix D. It allows for the synchronisation between JIRA and Trello through the use of a command line tool. A separate Trello Board was set up for porting User Story Tasks to JIRA tickets. This can be found at Appendix E

5.1.4 GitHub

The architecture of this project as described in Section 6 will incorporate a number of different code bases. To allow for the ease of version control on multiple code bases GitHub is used. All code will

be housed within multiple repositories under an organisation on GitHub. The project willl adhere to semver standards for versioning.

5.2 Technical Tools

5.2.1 CircleCI

CircleCI is the tool chosen to handle all continuous integration of the platform. It will be incorporated into all GitHub repositories and will allow for the automation of deployments throughout the projects lifetime.

5.2.2 Coveralls

In order to eliminate technical debt Coveralls is used to highlight untested areas of the codebase. It will work with the continuous integration to expose any gaps.

6 Design

This section has notable differences to the Design section in the Status Report, this is due to changes and requirements to the project as more information became known. Following the Agile methodology allowed for an easy change to new requirements. The design laid out in this section is of the completed work to date.

6.1 System Architecture Overview

This project is essentially broke into multiple elements, each element consisting of its own architecture. The elements are as follows:

- Command Line Interface
- Software Development Kits
- Showcase Application

6.1.1 Command Line Interface

The CLI will be of modular design, consisting of a main application which will have common logic shared between all plugin modules. These plugin modules will consist of language specific logic associated with each of the supported languages of the product. The main application will utilise the Docker API to connect to the Docker Daemon. This can be seen illustrated in Figure 9

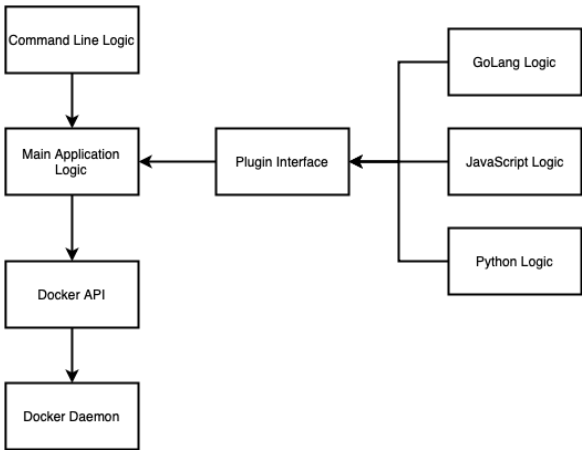

Figure 9: *Command Line Interface Architecture Diagram*

6.1.2 Software Development Kits

There will be a number of SDK's developed as part of this project, each SDK will correspond to a single supported language to the project. Each individual SDK will follow the same architecture. This architecture can be seen in the form of a Class Diagram in Section 6.15

6.1.3 Showcase Application

The purpose of the Showcase Application is to demonstrate the power of deploying applications on Kubernetes with Knative. The showcase architecture can be seen illustrated in Figure 10. It will be deployed on Google Cloud Platform and will consist of a React Single Page Application Frontend, with a REST API Backend which will leverage Googles Speech API to convert text to speech and store the speech output in Google Cloud Storage. The REST API will be built using the RubiX Framework and will show how the framework fits into a real world scenario.

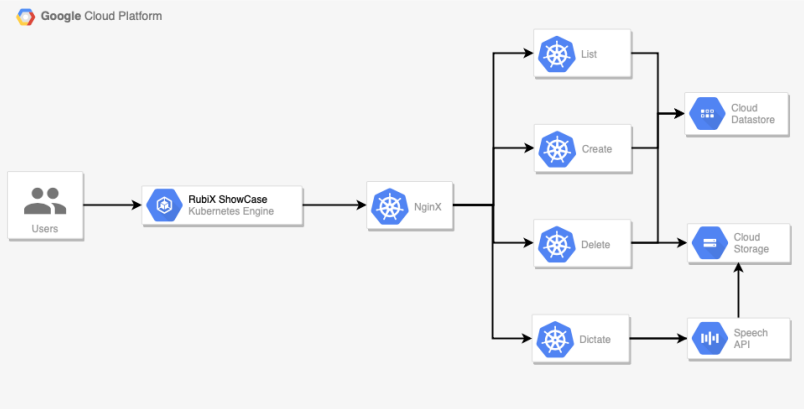

Figure 10: *Highlevel Showcase Architecture Diagram*

6.2 Requirements

Based on meetings with engineers currently working within the Knative project along with engineers working within RedHat, the problem which was discussed in Section 1.2 was identified. While identifying the problem a number of requirements were highlighted. These requirements will be split into two categories, Functional and Non-Functional.

It should be noted that these requirements were identified during the early stages of project planning and could be liable to change depending on the outcome of the Proof of Concept, discussed in Section 2.

6.3 Functional Requirements

The functional requirements listed below will be illustrated in Section 6.16 as Sequence Diagrams. Each requirement includes a rationale to help with understanding the purpose of it.

R1 - Bootstrapping of a Function as a Container
To allow a user easily create Functions as a Container, without abstracting anything from the user, giving freedom to test, debug and customise as needed.

R2 - Create a Function as a Container Image
To improve the work flow by saving a developer from switching between tools throughout the creation of a Function as a Container lifecycle.

R3 - Build a Function as a Container
To allow for local deployment to an OCI compliant runtime independent of Knative and Kubernetes.

R4 - Push a Function as a Container to a Registry
To allow for a means of secure storage for a developers Functions as a Container without introducing adding complexity to a developers workflow by utilising systems already used by a developer.

6.4 Non-Functional Requirements

The following is a number of non-functional requirements that were identified in the early stages of planning. These requirements fall under a number of broad categories.

R5 - Maintainability: The product as a whole must remain maintainable by implementing procedures to allow for the software elements to be modified to correct faults, improve performance and overall re-factoring of the code structure.

R6 - Quality: Processes need to be in place to guarantee a high quality in the overall code base across all elements within the project. These processes should be built into CI/CD tools like those discussed in Section 3

R7 - Security: Systems to be in place to ensure unauthorised, accidental or unintended usage of a users Functions as a Container does not happen. Processes to ensure that the latest versions of dependencies are used where feasible CVE

R8 - Documentation: Any addition to the project must be accompanied by concise and practical documentation. This documentation must be easily accessible to users.

R9 - Performance: All software must provide a minimal footprint on a user' s host, along with minimal response times. Thus reducing the time needed on CPU usage along with space taken on disk.

6.5 Core Requirements and Stretch Goals

All requirements were broken into a number of user stories. These user stories can be seen in Section 6.6. On the creation of the user stories and the core requirements a minimum viable product (MVP) was defined and outlined. An MVP is a product with all core features given minimal functionality in order to provide feedback to further improve on in later sprits. Each story was tagged with an id which is reflected in the tables in Section 6.6 with tags beginning with *mvp*. Stretch goals can be identified by tags beginning with *s*.

6.6 User Stories

Following the Agile Methodology, which is discussed in Section 3.1, the planning of this project is broken into Epics and User Stories. As this project uses JIRA to track and plan the development progress throughout sprints, which is discussed in Section 5.1.1, there is a disconnect between what is a JIRA epic and story in comparison to the definition of an Agile epic and story. In order to minimise this disconnect Trello was used during the planning, discussed in Section 5.1.2, and the following rules were applied.

User Stories would be broken into themes; each theme has an overview story. Each overview story will have a number of User Stories associated with it. These user stories would be treated as epics within JIRA and each epic will have a number of tasks associated with it.

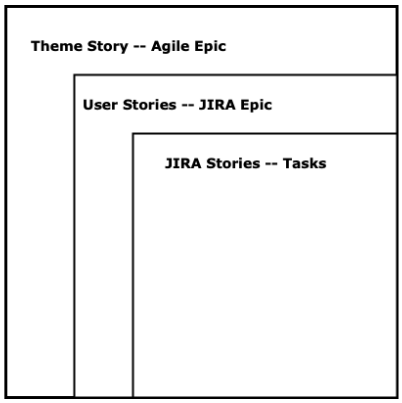

Figure 11: *Bridging the Agile – JIRA gap*

It must be noted that each User Story contains a *What, Why, How* along with *Acceptance Criteria* and *Tasks*. For brevity these will be left out of the document but can be found on the associated Trello Board found at Appendix C

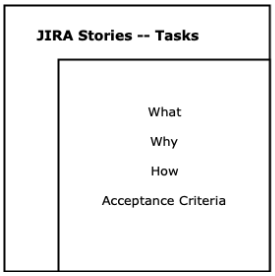

Figure 12: *JIRA Story – Task*

6.7 Themes

User Stories where thematically grouped under headings, to help with prioritisation and organisation. User story themes are as follows:

- Containers
- CLI
- Security
- Developer
- Community
- ShowCase

6.8 User Definitions

A number users have been defined in order to add clarity to the user stories:
1 – General User Definition: A user of the product who would ideally be a developer who is used to working with tools like Docker and Kubernetes, minimal admin privileges are granted
2 – Community User Definition: A user of the product who is also a contributor.
3 – Org Owner Definition: A developer to the product, with full admin and configuration privileges
NOTE: A Community User incorporates the user stories of a General user, while the Org Owner incorporates all users.

Note : The following sections contain all stories completed to date over the course of this project. The stories are sorted under their respected themes. For a full list of stories created for this project refer to Appendix I.

6.9 Containers

Overview Story

As a General User, I want a container function so that I can have full control over my FaaS

#	As a	I want to be able to	Such that
mvp1	General User	create a JavaScript container	I can leverage my own FaaS
mvp2	General User	handle HTTP requests in JavaScript	I can call my functions in a container
mvp3	Org Owner	ability to read and write metadata from the request req context	give more control to my functions
mvp4	General User	create GoLang functions as a container	I can leverage my own FaaS
mvp5	General User	create Python functions as a container	I can leverage my own FaaS
mvp6	General User	create Java functions as a container	I can leverage my own FaaS
mvp7	General User	create Haskell Functions as a container	I can leverage my own FaaS

6.10 Security

Overview Story

As a General User I want to have confidence in my container functions such that the container functions are as secure as possible

#	As a	I want to be able to	Such that
mvp17	General User	make my images as minimal as possible	I can ensure malicious code can not be injected or introduced to my containers during deployment
mvp18	Org Owner	have my systems protected from known CVEs	To ensure there is no known vulnerabilities in the latest release of the framework

6.11 CLI

Overview Story

As a General User, I want to be able to control the lifecycle of my container functions so that I can increase productivity while maintaining full control and ownership of my container functions

#	As a	I want to be able to	Such that
mvp8	General User	bootstrap the creating of container functions	productivity of using container functions is increased abstracting the mundane tasks away so I can focus on the functions.
mvp9	General User	push my container function to a registry	I can easily access and store my functions
mvp10	General User	easily create a container function	I maintain full control and ownership of my container function
mvp11	General User	bootstrap the creating of golang container functions	productivity using container functions is increased abstracting the mundane tasks away so i can focus on the functions
mvp12	General User	bootstrap the creating of python container functions	productivity using container functions is increased abstracting the mundane tasks away so i can focus on the functions
mvp13	General User	bootstrap the creating of haskell container functions	productivity using container functions is increased abstracting the mundane tasks away so i can focus on the functions

6.12 Developer

Overview Story

As an Admin User I want to ensure consistency throughout the project so that the project code base maintains a high standard

#	As a	I want to be able to	Such that
mvp14	Org Owner	build each commit to an org repo	I can track the progress of the project and highlight any issues as early as possible into the development.
mvp15	Org Owner	application to be available via a container registry	I can easily access, install and run the applications
mvp16	Org Owner	make SDKs available for installation through package management tools	It makes it easier for users to utilise the Framework

6.13 Community

Overview Story

As a General User I want this project to be developed in the open so that I can benefit from the transparency while being able to contribute to and give back to the project

#	**As a**	**I want to be able to**	**Such that**
mvp19	Community User	all repo's open and under a single organisation	I can easily find and browse the available repo's
mvp20	Community User	a clearly defined definition of done	I can easily contribute to the project and get involved
mvp21	Community User	have all documentation consolodated in one location	I can easily find and browse the documentation

6.14 Showcase

Overview Story

As a General User I want a showcase application to provide a real world example of the efficiency and usage of this framework as well as showing how Knative handles serverless workloads.

#	**As a**	**I want to be able to**	**Such that**
mvp22	Org Owner	arch diagrams for my showcase application	to help speed development of my showcase application
mvp23	Org Owner	REST Serverless functions	to show a common pattern developed in serverless architecture
mvp24	Org Owner	show a function that interacts with Google Speech API	the showcase application can convert text to speech
mvp25	Org Owner	I want a front end application	it will allow a user to access the serverless functions
mvp26	Org Owner	have a Google Project configured	it allows for a quick and easy environment to develop and test the applications in

6.15 SDK Class Diagram

Outlined in Section 6.1.2, there will be a number of SDK's developed throughout this project. Each SDK will follow the class structure outlined in Figure 13

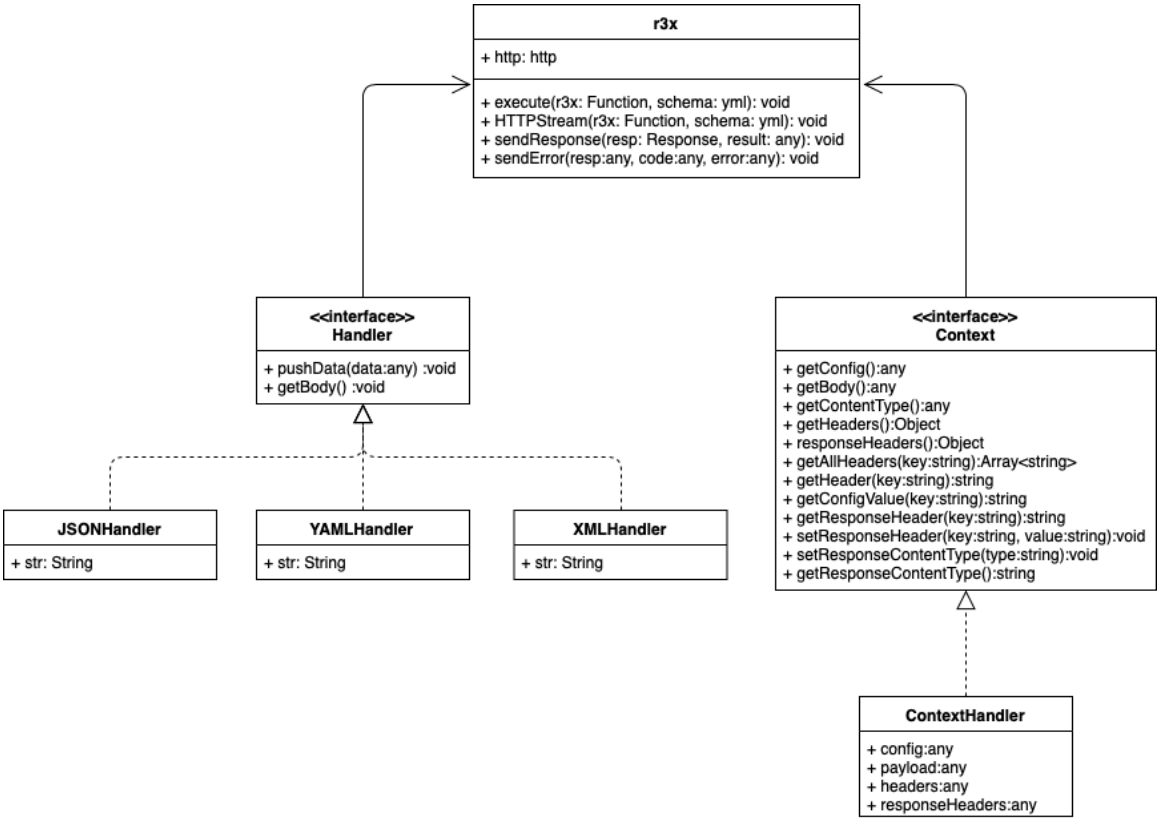

Figure 13: *SDK Class Diagram*

6.16 CLI Sequence Diagram

A number of sequence diagrams have been created to show object interactions between the CLI, Docker API and Docker Daemon.

6.16.1 Initializing a Project

Initializing a project via the command line, to improve the work flow and provide the scaffolding needed for a user to develop a function as a container.

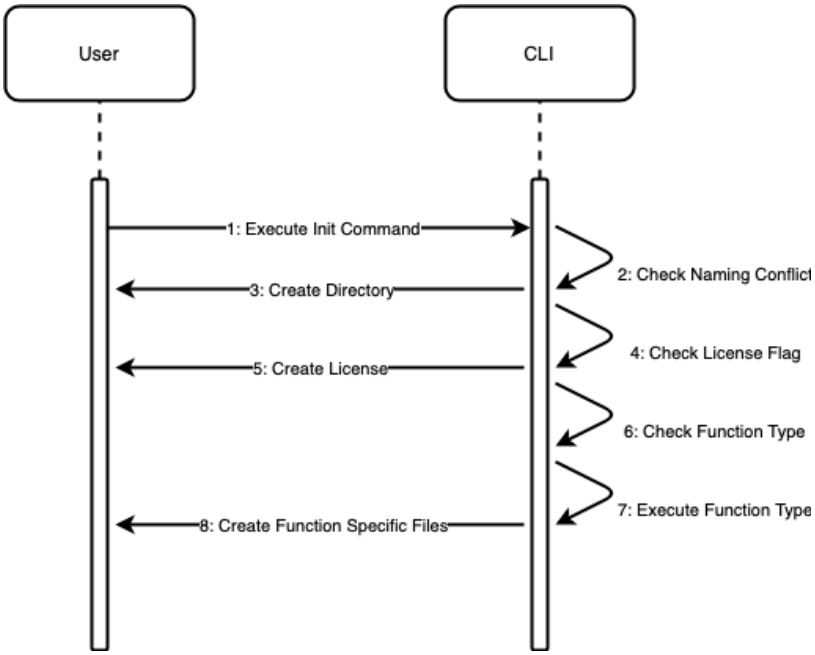

Figure 14: *CLI Sequence Diagram for Init a Project*

Sequence Steps

1. User executes the init command, passing flags to declare type of function and function type
2. CLI checks for naming conflicts before creating a new directory
3. CLI checks values for license type and creates a license
4. CLI checks the function type and executes function type logic to create function specific files

6.16.2 Building and Pushing Container

Building a Container locally and Pushing to a registry. It was decided to combine this functionality into a single command to add extra value to the CLI over using existing tools such as Docker CLI

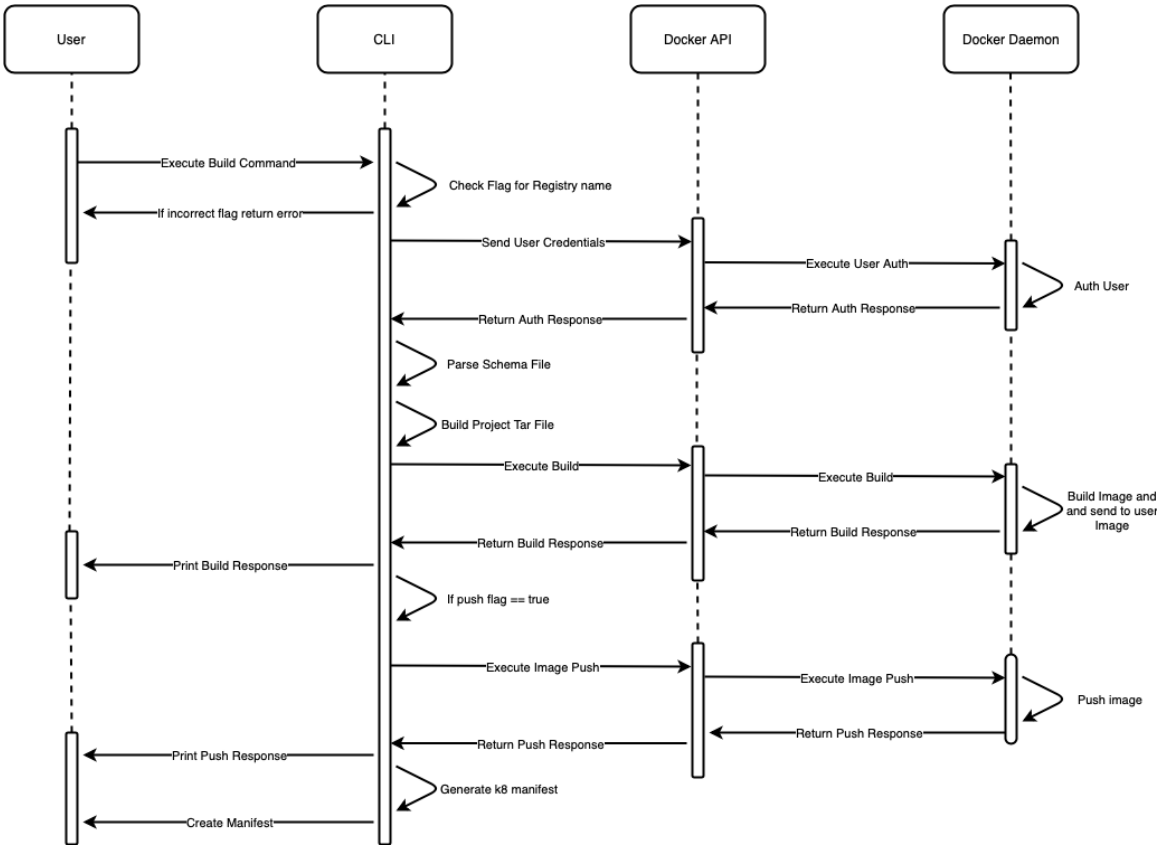

Figure 15: *CLI Sequence Diagram for Building and Pushing a Container*

Sequence Steps

1. User executes build command and the CLI checks passed flags
2. User is then authenticated through Docker
3. A TAR file is created of the project and a build is triggered
4. If the user has passed the push flag, then a push of the image is triggered on a successful build.
5. On a successful push, the Knative manifest is created.

7 Implementation

7.1 Sprint 0

7.1.1 Sprint Planning

A number of stories where completed last semester as part of building a prototype for the project. Following the Scrum methodology, a backlog grooming took place in order to reevaluate the current backlog, eliminating stories which where complete and reordering stories based on new insight to the project, discovered during the development work on the prototype.
With initial completion of the JavaScript SDK, it was decided to begin work on the CLI. Laying the ground work on the CLI during this early sprint would add value to the project by paving the way for further development later on.

7.1.2 Sprint Review

The following goals where achieved:

- CLI Repo was built and CI/CD configured.
- Cobra scaffolding was built for CLI
- Init Function was completed
- CLI Documentation was completed
- Restructure of JS SDK was completed

During the Sprint Review a Stake Holder highlighted that a user might want to change the name of their project after it has been initialized. This topic was discussed and a story created and added to the backlog.

7.1.3 Sprint Retrospective

What did we do well?

- Upskilling in Golang

What could have been done better?

- Better use of story points, as Sprint output was under estimated.
- Backlog refinement could of been improved

- User definitions could be clearer.

Actions

- Ciaran Roche should revise current backlog stories
- Story Points needed at the Story level over the use of Story Points at the task level
- Proper usage of labels to help organise sprints and tasks via trello as well as JIRA

7.1.4 Sprint Burndown

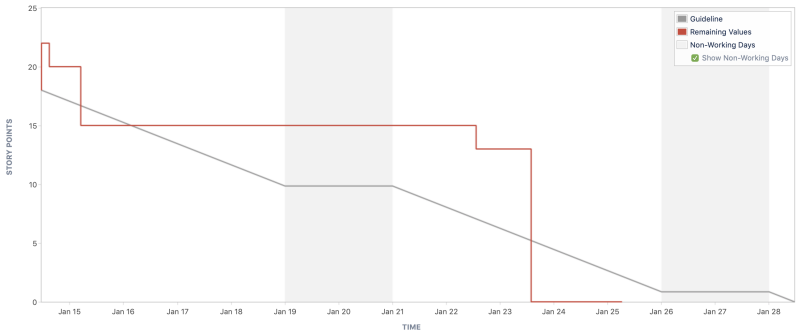

Figure 16: *Sprint 0 Burndown Chart*

7.1.5 Personal Reflection

This sprint highlighted areas which I overlooked, mainly those mentioned in the *Actions* from this Sprint Retrospective. The main take away from this sprint was the importance of Story Points, as it is a key factor in discovering sprint output and planning more efficient sprints in the future. I feel this is reflected nicely in Figure 16, which shows all tasks being completed ahead of sprint end. Overall I was happy with the outcome of this sprint as it was the first one there was bound to be teething problems.

7.2 Sprint 1

7.2.1 Sprint Planning

This sprint focused on two topics, the first being the CLI. It was decided to focus on adding capabilities to build and push an image to a repository. This feature would add the most value early on in development of the project. The second topic was to begin work on the showcase application. Taking this sprint to spike on a suitable environment to host the showcase along with begin work on

appropriate documentation for the entire project. As the project is spread across multiple repositories a central location for documentation will bring value and usability to those looking to use the tool.

7.2.2 Sprint Review

The following goals where achieved:

- Create functionality was added to CLI.
- Push functionality was added to CLI.
- Parsing of the RubiX schema was completed within the CLI.
- A Documentation Repo was set up.
- An environment was decided on, using Google Kubernetes Engine (GKE).
- Configuring and running a function within GKE was documented.

7.2.3 Sprint Retrospective

What did we do well?

- Finishing the Sprint ahead of time

What could have been done better?

- Better plan the Sprint to avoid early completion of Sprint
- Time management improvements as more work is consumed on the Weekends

Actions

- Refine story points, as what was considered a 5 might not be a 5 as we begin the third sprint, stories which where initially pointed might not be as complex as previously thought, leading to sprint work finishing early.
- Look to improve time management to ensure a more consistent completion of work over large chunks at the weekend.

7.2.4 Sprint Burndown

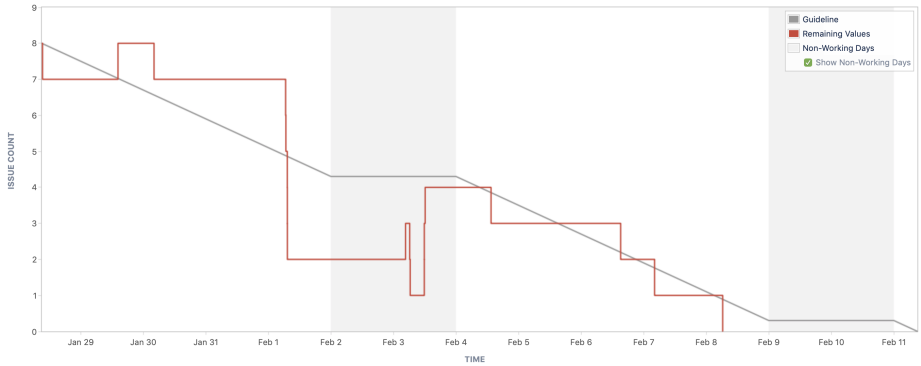

Figure 17: *Sprint 1 Burndown Chart*

7.2.5 Personal Reflection

Following the incremental improvements instilled in Scrum development, figure 17 shows a slight improvement on the previous sprint. While being an improvement I feel that there is quite a lot more room to further better my sprints. The burndown chart shows work was added to sprint due to over all sprint velocity being underestimated. The action points from this sprint reflect this with a full revision of story points needed to reevaluate the complexity of stories.

All that said, while it is still early days within the development process, I am beginning to think, was scrum the right approach for a final year project. My argument being, in college workloads change week on week, with focus on other assignments and modules. Would a methodology which is more reactive allowing for the flexibility to add and remove work at will from sprints, have being a better approach?

This is a question I hope the answer by the end of this semester, as it is still early and it takes time to figure out a teams skill level and velocity. In my case being a one person team the factors that affect velocity are different again. Overall I am quite happy with my progress as I feel in a stronger position being able to complete sprints early as opposed to not completing sprints. It adds to my excitement for the project to watch these changes sprint on sprint and see my project and my processes improve.

7.3 Sprint 2

7.3.1 Sprint Planning

A lot of time was given to refining the backlog and giving priority to stories that will add the greatest amount of value at this midway sprint.
With that, this sprint took the opportunity to return to the JavaScript SDK and finish it to a production quality. As the work completed to date on the SDK was work carried out during the proof of concept phase. Included in this work was ensuring the SDK fails safely with appropriate error handling. Along with a context parser to allow for metadata to be written to request headers.
Another story brought into this sprint looked to develop an SDK to support GoLang. This SDK would need to support all the features of the JavaScript SDK.
Two further stories where brought into the sprint to design the Showcase Application.

7.3.2 Sprint Review

The following goals where achieved:

- Complete work on JavaScript SDK
- Complete work on GoLang SDK
- Develop GoLang example app

7.3.3 Sprint Retrospective

Note : At the beginning of this sprint I was bed bound after coming down with flu. Unable to begin work sprint work till the second week of the sprint. **What did we do well?**

- Complete a vast amount of the allocated work in the final days of the sprint
- The reorder of the backlog during the planning

What could have been done better? Due to the sickness, not enough working time was consumed over the two week period to give us enough information on areas that could be improved.
Actions

- Take a RAID approach to planning.

The acronym RAID stands for Risks, Assumptions, Issues and Dependencies. This covers events that will have an adverse impact on the project if they occur. In this case working on a one person team my inability to work on the project during a sprint had an impact on output, along with giving no valuable information that could be used to further improve work output sprint on sprint.

7.3.4 Personal Reflection

Personally this was a disappointing sprint, in that I was unable to complete work due to being sick. When you begin a sprint you are committing to completing all work allocated to the sprint, not being able to complete this goal felt like a set back, as the previous sprints saw a higher output then what was originally planned.
That said it was encouraging to see the amount of work I was able to complete in a short space of time. I originally thought it would be an easy task to port the SDK from one language to another. As I began this work on the GoLang SDK I immediately saw this was not as straight forward as I predicted. Despite the original SDK being wrote in TypeScript which is strictly typed, it still allowed me some flexibility in my code. Writing the SDK in GoLang did not offer me any freedom, and with this, decisions needed to be made to insure the GoLang SDK performed as expected. This was a challenge as time was against me, but was also an enjoyable experience while I improved my efficiency with working with GoLang.
I feel the need to return to the question that came up in the previous decision, is Scrum the correct methodology a final year project should follow? While this question could be the topic of an entire research project, I feel after completing this sprint with an unforeseen event affecting the output of the work. I do think a more reactive approach to the methodology would allow a lot more freedom while alleviating some pressure. That said, the commitment to complete work outlined in the sprint, was a driving force behind the development that was completed. Having freedom here, would the work have being finished or would it have been pushed back? I cannot make up my mind, with two more sprints to go before this project is over will the benefits of working in a one person Scrum team become clear? Time will tell.

7.3.5 Burndown

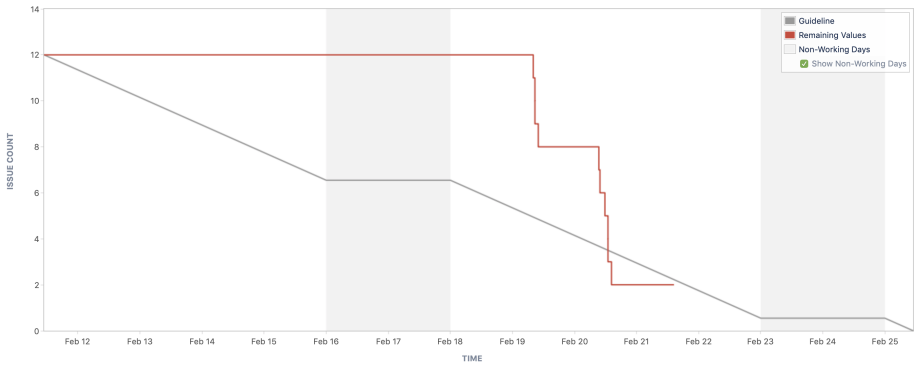

Figure 18: *Sprint 2 Burndown Chart*

7.4 Sprint 3

7.4.1 Sprint Planning

As with all sprint planning, choosing the stories to complete in the coming sprint is based on priority along with what will add the greatest value to the MVP at the end of this project. Looking back at the work completed so far, with the CLI in good shape, and progress being made toward adding SDK's to the frame work it was decided to further the frame works portfolio of SDK's. Therefore, stories for this sprint included the completion of both a Java and a Python. Along with stories which would allow for the bootstrapping and building of the new SDK's through the CLI.

7.4.2 Sprint Review

The following goals where achieved:

- Complete work on Python SDK
- Add Python functions to CLI
- Complete work on Java SDK
- Restructure GitHub Org and Documentation
- Complete planning work of Showcase Application

The following goals were not achieved:

- Add Java functions to CLI

7.4.3 Sprint Retrospective

What did we do well?

- Context switching between various languages.
- Sprint Planning is getting stronger and more accurate.

What could have been done better?

- Reach out for assistance quicker.

Actions

- Maintain channel of communication between developer and stakeholders throughout sprint.

7.4.4 Personal Reflection

Before delving into my personal reflection it is worth pointing out and adding context to a number of the above points made in the Retrospective. First, despite this being the first sprint not to achieve all the goals, the sprint planning is becoming more accurate as a flow for the sprint output is becoming more apparent and the value of story points becomes more refined. The reasoning behind not finishing all tasks is due to the nature of dealing with a strongly typed language such as Java, the planning for the SDK needed to be adapted to suit. While I waited to long to reach out for advice on how to overcome the problem at hand. When a solution was found I was left unable to deploy the SDK to Maven before the sprint end. Without a release of the SDK the work for bootstrapping and the build functions could not be complete.
With that said while the work on the Java SDK was disappointing on my behalf, but the overall sprint was enjoyable with lots of lessons learned. The most important lesson is choosing the right time to ask for help. I feel I left this too late in the sprint to be able to complete the sprint goal. Working on an FYP can be quite an isolated experience due to the work being carried out, is graded on the premise that you completed it yourself, along with the fact you are essentially a one person team. While in the 'real world' this type of environment would be extremely rare due to the emphasis on team work. I have to remind myself that while this is an FYP I have a team around me, to guide me and advice me. This team consists of my supervisor, the stakeholders and my lecturers.
Another important take away from this sprint revolves around context switching between paradigms. The initial SDK work was completed in TypeScript, while it requires types to be declared it still offers some of the freedom of dynamic languages. GoLang was the next SDK to be completed, which this is strongly typed it feels quite dynamic. It allows a certain amount of freedom, and this freedom lead to an easy transition from TypeScript to Golang. Python falls into this category, being an easy transition. I guess I got too comfortable working with these languages, which lead to me overlooking the complexity of creating a Java SDK. This is something I will be taking forward with me, to not make assumptions and where possible to ask questions and get verification.
With lessons learned and work progressing this was a challenging sprint. Leaving me eager to continue work and watch my own progression with juggling FYP, general college work and personal life. With less then half the semester left, the excitement and the nerves of completing this project are at an all time high.

7.5 Sprint 4

7.5.1 Sprint Planning

Work on developing the showcase application took precedence this Sprint. With two weeks given to create the showcase outlined in Section 6.1.3. This would allow the opportunity to see how the

RubiX framework handles working on a real world project. Along with building more familiarity with Knative and Kubernetes.

7.5.2 Sprint Review

The following goals where achieved:

- Complete CRUD functions with RubiX
- Develop a function that incorporates multiple API's with RubiX
- Develop frontend Application to consume RubiX functions
- Deploy Application and Functions to GKE

7.5.3 Sprint Retrospective

What did we do well?

- Collaboration with the Knative Community
- Gaining familiarity with the Google Cloud Platform

What could have been done better?

- More investigation into the tools I am using

Actions

- Maintain collaboration with the Knative Community

7.5.4 Personal Reflection

Looking at the burndown chart in Figure 19 it shows a steady progress of completion of tasks. While the progress was steady and consistent it was not an easy road. It required reaching out to the Knative community, several slack chats with Knative engineers, a GitHub issue and multiple google group threads later the app was deployed.
It was a good learning experience and showed the power of open source, having the resources to be able to reach out to those working and developing the software. As Knative is so new, without a full release and with consistent change some of what I wanted to achieve was yet to be documented and in one case has not been developed yet. One instance was Knative does not support CORS configuration at the Knative Ingress Gateway, resulting in access to services in a Knative cluster to be impossible. It was a Knative engineer who suggested using xip.io to create a new namespace for my cluster, and through this I was able to bypass the need for CORS config at the Ingress Gateway.

A simple yet clever workaround for an undeveloped feature.
Away from that I was personally happy with the progression of this sprint, showing my planning had improved and my sprint consumption had remained consistent. This was the first sprint in which I brought the framework beyond basic use cases, and I was quite happy with how the framework worked and handled. Which left me with a reassurance of the value to which this framework brings.

7.5.5 Burndown

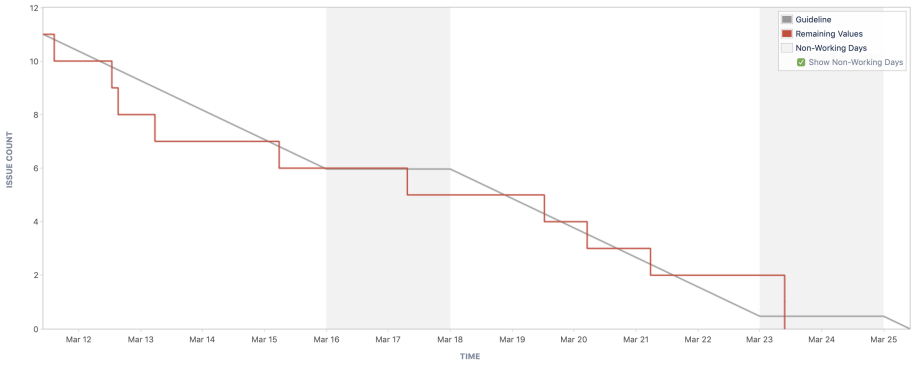

Figure 19: *Sprint 4 Burndown Chart*

7.6 Sprint 5

7.6.1 Sprint Planning

It was decided to make this the final sprint of the Project. So the tasks included creating the first MVP release of the Project. This included tasks of increasing test coverage in critical repos, documentation verification, and manual testing of all repos. A story was included in this sprint to develop a new SDK to support a functional programming paradigm (Haskell).

7.6.2 Sprint Review

The following goals where achieved:

- Increase test coverage to 90% in the CLI and JS SDK Repos
- Verified all documentation
- Completed v1 release of Framework
- Developed Haskell SDK

7.6.3 Sprint Retrospective

What did we do well?

- Complete all MVP tasks and release v1
- Haskell SDK

7.6.4 Personal Reflection

This was quite an emotional sprint due to being the final sprint for this current project. It gave me two weeks to step back and look back at the work completed over all sprints as I performed manual testing and verification.
The only down side to this sprint was in that, JIRA updated their sign in which left me locked out of JIRA, so that I could not get a burndown chart for my final sprint, while disappointing not to have the chart, it was not a burden or a set back as all planning work was completed in Trello it was quite an easy transition to track my tasks there.
Taking on the work of supporting Haskell in the framework, was one of the hardest challenges of the whole project. The differences between functional programming in comparison to dynamic languages or static, meant I could not leverage the existing patterns of other SDK's, I had to approach the Haskell SDK from a different angle. While challenging it was quite a learning experience, and I feel it lead to the development of a strong product.
A number of weeks back at the Waterford Tech Meet Up, Richard Rodgers spoke about the development of his latest startup, when he got to the topic on testing, he showed the variation of test coverage across his different services, ranging from zero percent to plus ninety percent in some cases. He argued that at the early stages of development of a new product, when there are goals to be met, in creating an MVP having full test coverage does not always work, and priority has to be given to core services and development of features. I took this mentality with my project as I strived for breath over depth, trying to gain as much experience and exposure through a single project as I could. This sprint gave me a sense that I completed this goal, as I look across the RubiX org on GitHub and see the variations of different paradigms, and the individual lessons and challenges each one possessed.

8 Summary

8.1 Project Direction

Currently the project has succeeded in delivering an MVP to the Product Owners. With the completion of the MVP I feel the project has successfully solved the problem outlined in Section 1.2, in providing a means to quickly and efficiently develop applications within Knative. The current state of the Project gives users an open door to explore what the Knative project has to offer, with minimal upskilling needed, giving more focus to Knative and its features.

8.1.1 Future Development

I feel the project currently solves the initial outlined problem, while there is work to be completed, which is documented and logged on RubiX backlog in Trello. With exposure and experience to the Kubernetes and Knative ecosystem I have discovered many new demands and needs for developers, this is further backed by the release of Quarkus[10]. A high level view of Quarkus, is a container-centric Java Stack with the main goal to spin up a Java container in sub second times.
The current state of RubiX provides the ground work to develop upon. Work that continues will solely focus on a single SDK, with the core goal of container first, developing near instant scale up and high density memory utilization in Kubernetes. As current trends in development within the Knative project are skewed toward scaling from 0 in sub second times. The container itself is an individual cog in this process, and development in this area is extremely valuable to the Knative community. RubiX is the perfect candidate to build upon, and this is work which I will begin on after the college term.

8.2 Review

8.2.1 Core Goals

The project delivered on all its core goals set out by the stakeholders.

AO1 - Completed SDK's in the following paradigms – TypeScript, GoLang, Python, Java and Haskell

AO2 - Each SDK handling and catching errors, with the ability for logging.

AO3 - Dockerfiles following the best practices outlined by Docker and the OCI.

AO4 - The project was developed in the open under the GitHub org RubixFunctions.

[10]https://quarkus.io – A Kubernetes Native Java stack.

AO5 - A CLI tool developed to abstract common usages and workflows through the development of Functions as a Container.

The project deliverables are as follows:

- JavaScript SDK
- GoLang SDK
- Python SDK
- Java SDK
- Haskell SDK
- Development CLI
- Knative Terraform
- RubiX/Knative Showcase Application
- RubiX Documentation
- RubiX HelloWorld Samples

Contributions

As Knative is an Open Source project, it gave the opportunity to contribute back to it over the course of the project. It is often a misconception that a contribution to an Open Source project has to be in the terms of code. But a contribution comes in many forms, with the goal of improving the project and developing something special and community driven. That said over the course of my Project it resulted in the following:

- One Pull Request to a Knative Repository.
- One Github Issue in a Knative Repository.
- Two authored Knative Google Group Threads.
- Responded to multiple Knative Google Group Threads.
- Partook in Slack Discussions.

8.3 Learning Outcomes

This project produced multiple Learning Outcomes, ranging from Technical and Non-Technical.

8.3.1 Technical Learning Outcomes

- Gained valuable insight into the containerization and orchestration of applications
- Developed an appreciation for quick prototyping and iterating within the development process, as is recommended by the Agile framework.
- Learned about CI/CD processes to build and deploy images, a common software development practice in the industry when developing as part of a large team
- Gain experience in context switching between multiple paradigms, a common trait of software engineers working within development teams on large projects.
- Realised common differences and patterns across different languages and being able to leverage these differences to solve particular problems.
- Reinforced the importance of documentation, and separating one's experience when writing documentation to provide a better user experience.

8.3.2 Non-Technical Learning Outcomes

- Experience was gained in dealing with Product Owners and Stakeholders, from listening to requirements and recommendations along with the confidence to push back with my own ideas.
- Learned the benefits of tools such as JIRA and Trello in planning work. Included here was learning how to properly track work progress and utilise this information to improve work output while reducing technical debt.
- Communication skills improved, from general day to day contact that is common in a development ecosystem, across mediums such as IRC, Email, Slack etc. Right up to gaining the confidence to present and project my ideas and thoughts to a larger audience.
- Writing experience and skill improved dramatically over the course of this project. This includes academic writing, technical documents, pull requests and code reviews.
- The ability to retrospectively self evaluate oneself, and learn from mistakes. This is a key skill not only in software engineering but in any career.

8.4 Personal Reflection

This has been an amazing journey from start to finish, I feel it has helped me progress as a not only a developer but also a person, learning many new skills and discovering new abilities, mixed with making new connections and gaining exposure to new and exciting platforms. This project will help me proceeding forward with a career Software Engineering.

The highlight to my project was the involvement of Red Hat and the Knative community, who assisted my problem-solving in several areas of this project. Overall, this demonstrates the importance and benefits a community can bring to a project. Open Source stretches far beyond a public repository on GitHub, and my FYP gave me an insight into what it takes to maintain such a project and how to successfully contribute to one. Having a contribution to the Knative project is a personal success for me, and something which I did not foresee when beginning my FYP.

Over the course of my time as an Applied Computing student, dealing with many projects and deadlines, along with my time as an Intern at Red Hat and StitcherAds, I have constantly been taught methods and processes on how to take an idea and see it to fruition. This project gave me to opportunity to take the lessons learned and put them into practice. It showed me the importance of initially understanding a problem. While the process of splitting a problem into small tasks can be quite time consuming and tedious, doing this at a granularity at which I never had to sole responsibility to do before proved to pay dividends as the project progressed. This exposed and opened new and previously unthought of avenues throughout the project. Overall this increased the project's value and when it came time to developing these tasks there was zero confusion allowing the completion of these tasks happen in a fluid and consistent manner.

The project followed the Scrum methodology; while not being targeted for a solo project I adopted the methodology to suit. Numerous times throughout this project I questioned was this the correct methodology to suit an FYP. Now retrospectively looking back I feel it was the correct choice, while my arguments against scrum revolved around it not being dynamic enough, as college circumstance change so quickly, it was often hard to adapt sprints to suit. While this is a valid argument, I feel the commitment of scrum is needed for an FYP, in that as college circumstances change you are obligated to finish all tasks in a current sprint. This ensures work gets completed regardless, and progress is consistently made. I feel being more dynamic would lead to less completed work with added pressure on the final deadline. That said this is a topic that could do with its own body of research. As it is such a broad area and the benefits of correct planning and management are apparent.

This project allowed me to explore multiple paradigms; this was one of my own personal goals, as I wanted to build my confidence as a software engineer. Unbeknownst to myself it opened my eyes to the differences across these paradigms and the value some can bring to particular problems and not others. While some languages possessed similarities, which I could leverage to solve the problem, others forced me to think of the problem in a different light and approach it differently. This plays

back into the importance of planning, and giving the time to break problems and tasks into small manageable chunks, ensuring all aspects are clear and the task is feasible. While increasing the overall challenge to the project, this taught me valuable lessons, and the importance of having the right tool for the job.

Overall I feel my project was a success, and has motivated me to pursue a career in Open Source and the Kubernetes eco-system along with continuing work on RubiX.

Appendices

A GitHub Organization

https://github.com/rubixFunctions

B Circle CI Repository

https://circleci.com/gh/rubixFunctions

C Trello User Stories

https://trello.com/invite/b/1tnWUrMa/9be869b4b608c1a9c8fb40ff4723f18b/r3x-user-stories

D Trira GitHub

https://github.com/aerogear/trira

E Trello Trira Board

https://trello.com/invite/b/VgT5oxJ3/d6a4fa66d09c13e37f3b6c0a7969d3a7/rubix-task-planning-board

F Prototype One

https://quay.io/repository/ciaranroche/r3x-poc-1

G Prototype Two

https://github.com/rubixFunctions/r3x-js-sdk

H SDK Showcase

https://github.com/rubixFunctions/r3x-js-showcase

I Semester 1 Stories

I.1 Containers

Overview Story

As a General User, I want a container function so that I can have full control over my FaaS

#	As a	I want to be able to	such that
mvp1	General User	I want a way to handle HTTP requests in JavaScript	I can call my functions in a container
mvp2	General User	I want to be able to create a JavaScript container	I can leverage my own FaaS
s1	General User	I want to be able to create GoLang functions as a container	I am not limited to one programming language
s2	General User	I want to be able to create Python functions as a container	I am not limited to one programming language

Table 2: Container User Stories

I.2 Testing

Overview Story

As a General User I want to easily test my container functions so that I can highlight any errors quickly and efficiently.

#	As a	I want to be able to	such that
s9	General User	I want to easily be able to stress test my container functions	I can ensure reliability in the functions under load
mvp9	General User	I want to easily be able to test my functions outputs	I can easily test for errors without the need to write explicit tests
mvp10	Admin User	I want my production pushes to only occur when all tests are green	no unseen errors make it through my CI

Table 3: Testing User Stories

I.3 CLI

Overview Story

As a General User, I want to be able to control the lifecycle of my container functions so that I can increase productivity while maintaining full control and ownership of my container functions

#	As a	I want to be able to	such that
mvp3	General User	I want to bootstrap the creating of container functions	productivity of using container functions is increased abstracting the startup tasks
mvp4	General User	I want to easily create a container function	I maintain full control and ownership of my container function.
mvp5	General User	I want to be able to push my container function to a registry	I can easily access and store my functions
mvp6	General User	I want to be able to easily list what container functions I have created	they do not get lost between my other regular containers
mvp7	General User	I want to easily deploy a container function	I can easily test my container function.
s3	General User	I want to easily deploy container functions to Kubernetes	I do not need to switch between tools throughout the container function development process.
mvp8	General User	I want to be able to delete container functions	I can remove unused container functions without the need of using native tooling.
s4	General User	I want the ability to create multiple container functions at once	I can increase productivity by controlling multiple function container functions.
s5	General User	I want the ability to bootstrap multiple container functions at once	I can increase productivity by controlling multiple function container functions.
s6	General User	I want the ability to deploy multiple container functions locally at once	I can increase productivity by controlling multiple function container functions.
s7	General User	I want to be able to update container functions	I can easily reuse existing functions
s8	General User	I want to be able to roll back container function updates	I can roll back use cases in the event of a problem I can recover

Table 4: CLI User Stories

I.4 Security

Overview Story

As a General User I want to have confidence in my container functions such that the container functions are as secure as possible

#	As a	I want to be able to	such that
s10	General User	I want the ability to run my container functions in read only mode	I can ensure malicious code can not be injected or introduced to my containers during deployment
mvp11	Admin User	I want to have my systems protected from known CVEs	I can have assurance throughout my development
s11	General User	I want to grant read and write permissions to my function containers	I can control who owns the function

Table 5: Security User Stories

I.5 Developer

Overview Story

As an Admin User I want to ensure consistency throughout the project so that the project code base maintains a high standard

#	As a	I want to be able to	such that
mvp12	Admin User	I want nightly builds of the project	I can track the progress of the project and highlight any issues as early as possible into the development.
mvp13	Admin User	I want builds on every Pull Request	I can ensure no potential breaking changes are merged to the master branch
mvp14	Admin User	I want security vulnerability scans to be conducted as part of the CI	any potentially harmful vulnerabilities are highlighted early in the development process
mvp15	Admin User	I want the application to be available via a container registry	I can easily access, install and run the applications
mvp16	Admin User	I want to make my SDKs available for installation through package management tools	any user can easily access and use the SDK's independent to using the tool itself

Table 6: Developer User Stories

I.6 Community

Overview Story

As a General User I want this project to be developed in the open so that I can benefit from the transparency while being able to contribute to and give back to the project

#	**As a**	**I want to be able to**	**such that**
mvp17	Community User	I want a medium to be able to get involved in discussions around the project	I can participate and help shape the project
mvp18	Community User	I want a medium to be able to receive updates about the project	I can stay up to date with the project
mvp19	Community User	I want a medium to quickly talk to project developers	I can easily have any questions answered and share knowledge
mvp20	Community User	I want all repos open and under a single organisation	I can easily find and browse the available repos
mvp21	Community User	I want to have all documentation consolidated in one location	I can easily find and browse the documentation
mvp22	Community User	I want a clearly defined definition of done	I can easily contribute to the project and get involved

Table 7: Community User Stories

Bibliography

Bryant, D. (2018), 'Google releases knative: A kubernetes framework to build, deploy, and manage serverless workloads'.
URL: *https://www.infoq.com/news/2018/07/knative-kubernetes-serverless*

Currents: A quarterly report on developer trends in the cloud (2018).
URL: *https://www.digitalocean.com/currents/june-2018/*

Docker (2018).
URL: *https://www.docker.com/*

Fowler, M. (2006), 'Continuous integration'.
URL: *https://martinfowler.com/articles/continuousIntegration.html*

Fowler, M. (2018), 'Serverless Architectures'. [online] Accessed: 6/10/2018.
URL: *https://martinfowler.com/articles/serverless.html*

GoLang (2018).
URL: *https://golang.org/doc/*

Manifesto for Agile Software Development (n.d.).
URL: *https://agilemanifesto.org/*

OCI (2018).
URL: *https://www.opencontainers.org/about*

Open Standards Requirement for Software (2018).
URL: *https://opensource.org/osr*

RightScale 2018 State of the Cloud Report (2018).
URL: *https://www.rightscale.com/lp/state-of-the-cloud*

Schwaber, K. and Sutherland, J. (2017), 'The Scrum Guide'. [online] Accessed: 29/10/2018.
URL: *https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf*

Solis, C. and Wang, X. (2011).
URL: *https://ieeexplore.ieee.org/document/6068372?arnumber=6068372*

Stack Overflow Developer Survey 2018 (2018).
URL: *https://insights.stackoverflow.com/survey/2018/*

State of Agile Report (2018).
URL: *https://explore.versionone.com/state-of-agile/versionone-12th-annual-state-of-agile-report*

State of Cloud 2018 Report (2018).
URL: *http://get.cloudability.com/ebook-state-of-cloud-2018.html*

The Scrum Guide^{TM} (2018).
URL: *https://www.scrumguides.org/scrum-guide.html*

The State of the Octoverse (2018).
URL: *https://octoverse.github.com/*

TypeScript (2018).
URL: *https://www.typescriptlang.org/*

Vivien, V. (2017), 'Writing modular go programs with plugins – learning the go programming language – medium'.
URL: *https://medium.com/learning-the-go-programming-language/writing-modular-go-programs-with-plugins-ec46381ee1a9*

Where PaaS, Containers and Serverless Stand in a Multi-Platform World (2018).
URL: *https://www.cloudfoundry.org/multi-platform-trend-report-2018/*