# K8s Checkmate

DESIGN DOCUMENT

Team 58
Workiva
Julie Rursch
Team Members/Roles
sdmay20-58@iastate.edu
https://sdmay20-58.sd.ece.iastate.edu/

Revised: v1

# Executive Summary

## Development Standards & Practices Used

No hardware, purely software

Agile

Modular code

Clean code

Well documented code

## Summary of Requirements

Runs in CLI

Parses Helm charts

Creates policies based on rules provided in templates

checks configs of helm charts and alerts if they fail

## Applicable Courses from Iowa State University Curriculum

**Docker**

According to the [Docker documentation](#), Docker is a platform for developers and sysadmins to develop, deploy, and run applications in containers.

Containerization is not a new concept, but Docker's improvements made it feasible for widespread practical use. At the heart of it, containerization is just another method of virtualization. Traditionally Virtual Machines (VMs) are used for this. VMs virtualize the entire system, from the hardware level to the application level. As a result, they are easier to use on different machines than containers that only virtualize from the operating system (OS) level up.

Docker containers are implemented differently. First, a Docker process is run on the host OS. Containers natively run on Linux through the Docker process; only

virtualizing the apps, binaries, and libraries it needs. All Docker containers are separate from each other, but share the OS kernel with the host OS. As a result, Docker containers are portable, lightweight, stackable, and scalable. A more in-depth explanation is detailed [here](#).

While still relatively young, Docker is a well-established technology. It is open-source, first released in 2013, and has widespread use by companies and developers around the world.

Most 300 level classes in the SE Department provide VMs for students to use, so we do gain some experience with virtualization. Unfortunately, this experience does not translate well to Docker's implementation.

Our project specifically targets Helm charts and Kubernetes. Kubernetes is a technology developed by Google for creating clusters of Docker containers. Kubernetes clusters can be difficult to set up, thus helm charts are a framework used to simplify this process. As Docker is not taught in any of our classes, by extension Kubernetes and Helm are not either.

Kubernetes, Helm, and Docker are all written in the Go programming language. Therefore we will be using Go for our project.


**Go**

Golang, also known as Go, is a language created by Google to solve a lot of the language level programming issues they were facing at the time. The first stable release of Go was in 2011, and the first major release was in 2012. Go is a fast-growing language that is becoming increasingly popular due to its size and flexibility. Unfortunately, Go is neither taught nor allowed to be used in any of the classes we have taken. However, much like learning any other programming language, the prior programming experience we have learned in the SE curriculum helps reduce the time it takes to learn.

While not explicitly taught any of the technologies we will be using for our project, we will be relying on our skills to learn them. Our skills were honed through the classes we have taken and as a result, we will be better able to teach ourselves what we need to know for our project.

## New Skills/Knowledge acquired that was not taught in courses

Configuration with Kubernetes

Parsing Files with Security Checks

Experience with Go Programming Language

Communicating to an Advisor of our group's progress

Communicating to an Advisor with team management

Manipulation of Helm Charts

Knowledge of Python in General

Helm configuration

Docker usage

# Table of Contents

## List of figures/tables/symbols/definitions (This should be similar to the project plan)

# 1 Introduction

## 1.1 ACKNOWLEDGMENT

**Julie Rursch** - Group Advisor

She meets with us at least once a month in person and communicates with us when needed. She is guiding our group by letting us know what is expected and she lets us know what is expected.

**Eric Anders** - Workiva

Eric Anders works with Workiva and is our group's Industry Advisor

## 1.2 PROBLEM AND PROJECT STATEMENT

There does not exist a tool that checks the configuration of helm charts nor the configurations produced by them. Linters and syntax checkers exist, however, they only check that the helm charts are formatted correctly. They do not check that clusters comply with predefined rule sets.

By creating an extensible framework we hope to provide a well documented, highly extensible, useful tool that prevents a lot of security issues.

There is nothing out there that exists on what we are trying to accomplish. Our drive is not only to give the open-source community this tool but also be the first people to craft a tool like this.

## 1.3 OPERATIONAL ENVIRONMENT

The end product will run in a Command Line Interface, and will not be exposed to unusually hazardous conditions. This is solely software-based, so the will be no expectations physically for the product. However, we do expect the end product to be able to run in Linux and MacOS.

## 1.4 REQUIREMENTS

Functional requirements

- System should parse and check Kubernetes configuration files
- System should parse and check helm charts
- Command-line interface should allow for easy interaction with the system
- System should alert the user of potential security vulnerabilities
- System should suggest how to fix potential security vulnerabilities

Economic requirements

Since the project is almost entirely software, there are very few economic requirements. There is no hardware that needs to be purchased or licenses that need to be paid for since the project will be entirely open-source.

Environmental requirements

Again, since the project is almost entirely software-bound, the physical environment has no effect on it. There is no hardware that could be exposed to the elements or poor weather. In terms of the computer architecture environment, the project should be able to run on macOS and *NIX systems.

UI requirements

To keep the program lightweight and portable, the UI will not consist of a GUI, but rather a command-line interface. This is plenty sufficient for usability and fulfilling the intended use cases.

## 1.5 INTENDED USERS AND USES

The intended user is the person conducting a security review for a Kubernetes project.

The intended use is to streamline and reduce user error in the process of checking security configurations against a defined ruleset.

## 1.6 ASSUMPTIONS AND LIMITATIONS

Assumptions:

We assume that the user will have a basic knowledge of Kubernetes security configurations.

We assume that the user will have a basic knowledge of security with files in general

These assumptions are made because the intended user is for someone who wants to perform security checks with Kubernetes. Someone who doesn't have this previous knowledge probably wouldn't be using this then.

Expectations:

The end product will be lightweight and run in a CLI.

We expect the end product to be able to run Linux and macOS.

We expect the end product to be able to effectively perform a security review.

The end product and deliverables for our senior design project are as follows.

- A lightweight and portable CLI program that can check, verify, and alert users about potentially insecure and vulnerable Kubernetes configurations and helm charts. Lightweight means that it must be a small program that can be downloaded quickly from any internet connection. Portable means that it is not system-dependent and can run on a multitude of Operating Systems.
- Extensive documentation on the installation and use of the program so that anyone will be able to understand and use this. This documentation will include readme markdown documentation for outside users intending to use the program. Additionally, this will include a well-commented code for the open-source community intending to clone and contribute to the repository.
- Open-source code for continual improvement by the open-source community. It is proven that open-source code is more cost-effective, quicker to develop, much more secure/transparent, and more extensible in the future by anyone. We are making our program open-sourced for the aforementioned reasons

The delivery dates for these deliverables is T.B.D due to the nature of the senior design program. We can estimate that the above will be ready sometime around May of 2020.

# 2. Specifications and Analysis

## 2.1 PROPOSED DESIGN

The application will parse and analyze Helm and Kubernetes security configuration files, and compare the results against a defined set of rules.

The application will run in CLI.

The application will accept templates of rule sets to compare to.

## 2.2 DESIGN ANALYSIS

Our group has been communicating with both our advisor and each other about team roles and planning for our code development. We have mostly been communicating through online messaging with our entire team. Although our team has also met in person as well. So far our meetings, whether that be face-to-face or online, have been very successful. Each one of us is able to understand what is expected and we are able to hold each other accountable for tasks that need to get done.

Our strengths are communication and expectations. Everyone in our team is okay with sharing their thoughts and ideas. Expectations are clearly understood and set as well. Our biggest weakness is availability. All of our members are extremely busy so finding times to meet in person is a challenge.

Observations and thoughts on our team style so far are mostly positive. We all are communicating effectively and getting tasks done on time.

Finally, our team members have been learning the GO language. This is the language that we will be developing our code in.

## 2.3 Development Process

We are using a Waterfall development process because our requirements are well-defined and will not change over the course of the project.

We are developing one part at a time, testing it with our tests, showing our advisors what we have how it works now, and testing it in the environment it will be used, and making any of the necessary changes.

## 2.4 Design Plan

We are developing our code one feature at a time. This ensures our code is modular. Each feature can run on its own and is imported into the main package. The high degree of modularity makes the code more extensible so that others can modify it for their purposes.

The GUIs are going to be written as local client/servers. This model reduces the overhead and allows them to be run from the terminal, regardless of the desktop environment. This also reduces the number of dependencies needed for the application.

As of now we do not know all of the dependencies and modules we will be using. We anticipate that we will only need modules and dependencies we will have will be from the standard library.

# 3. Statement of Work

## 3.1 Previous Work And Literature

Include relevant background/literature review for the project

– If similar products exist in the market, describe what has already been done

– If you are following previous work, cite that and discuss the **advantages/shortcomings**

– Note that while you are not expected to "compete" with other existing products / research groups, you should be able to differentiate your project from what is available

Detail any similar products or research done on this topic previously. Please cite your sources and include them in your references. All figures must be captioned and referenced in your text.

## 3.2 Technology Considerations

Highlight the strengths, weakness, and trade-offs made in technology available.

Discuss possible solutions and design alternatives

## 3.3 Task Decomposition

In order to solve the problem at hand, it helps to decompose it into multiple tasks and to understand interdependence among tasks.

## 3.4 Possible Risks And Risk Management

Include any concerns or details that may slow or hinder your plan as it is now. These may include anything to do with costs, materials, equipment, knowledge of area, accuracy issues, etc.

## 3.5 Project Proposed Milestones and Evaluation Criteria

What are some key milestones in your proposed project? Consider developing task-wise milestones. What tests will your group perform to confirm it works?

## 3.6 Project Tracking Procedures

What will your group use to track progress throughout the course of this and next semester?

## 3.7 Expected Results and Validation

What is the desired outcome?

How will you confirm that your solutions work at a **High level**?

# 4. Project Timeline, Estimated Resources, and Challenges

## 4.1 Project Timeline

• A realistic, well-planned schedule is an essential component of every well-planned project

• Most scheduling errors occur as the result of either not properly identifying all of the necessary activities (tasks and/or subtasks) or not properly estimating the amount of effort required to correctly complete the activity

• A detailed schedule is needed as a part of the plan:

– Start with a Gantt chart showing the tasks (that you developed in 3.3) and associated subtasks versus the proposed project calendar. The Gantt chart shall be referenced and summarized in the text.

– Annotate the Gantt chart with when each project deliverable will be delivered

• Completely compatible with an Agile development cycle if that's your thing

How would you plan for the project to be completed in two semesters? Represent with appropriate charts and tables or other means.

Make sure to include at least a couple paragraphs discussing the timeline and why it is being proposed. Include details that distinguish between design details for present project version and later stages of project.

## 4.2 Feasibility Assessment

Realistic projection of what the project will be. State foreseen challenges of the project.

## 4.3 Personnel Effort Requirements

Include a detailed estimate in the form of a table accompanied by a textual reference and explanation. This estimate shall be done on a task-by-task basis and should be based on the projected effort required to perform the task correctly and not just "X" hours per week for the number of weeks that the task is active

## 4.4 Other Resource Requirements

Identify the other resources aside from financial, such as parts and materials that are required to conduct the project.

## 4.5 Financial Requirements

If relevant, include the total financial resources required to conduct the project.

# 5. Testing and Implementation

Testing is an **extremely** important component of most projects, whether it involves a circuit, a process, or a software library

Although the tooling is usually significantly different, the testing process is typically quite similar regardless of CprE, EE, or SE themed project:

1. Define the needed types of tests (unit testing for modules, integrity testing for interfaces, user-study for functional and non-functional requirements)
    2. Define the individual items to be tested
    3. Define, design, and develop the actual test cases
    4. Determine the anticipated test results for each test case 5. Perform the actual tests
    6. Evaluate the actual test results
    7. Make the necessary changes to the product being tested 8. Perform any necessary retesting
    9. Document the entire testing process and its results

Include Functional and Non-Functional Testing, Modeling and Simulations, challenges you've determined.

## 5.1 Interface Specifications

– Discuss any hardware/software interfacing that you are working on for testing your project

## 5.2 Hardware and software

– Indicate any hardware and/or software used in the testing phase

– Provide brief, simple introductions for each to explain the usefulness of each

## 5.3 Functional Testing
Examples include unit, integration, system, acceptance testing

## 5.4 Non-Functional Testing

Testing for performance, security, usability, compatibility

## 5.5 Process

– Explain how each method indicated in Section 2 was tested

– Flow diagram of the process if applicable (should be for most projects)

## 5.6 Results

– List and explain any and all results obtained so far during the testing phase

- – Include failures and successes

- – Explain what you learned and how you are planning to change it as you progress with your project

- – If you are including figures, please include captions and cite it in the text

• This part will likely need to be refined in your 492 semester where the majority of the implementation and testing work will take place

-**Modeling and Simulation**: This could be logic analyzation, waveform outputs, block testing. 3D model renders, modeling graphs.

-List the **implementation Issues and Challenges**.

# 6. Closing Material

## 6.1 Conclusion

Summarize the work you have done so far. Briefly re-iterate your goals. Then, re-iterate the best plan of action (or solution) to achieving your goals and indicate why this surpasses all other possible solutions tested.

## 6.2 References

This will likely be different than in project plan, since these will be technical references versus related work / market survey references. Do professional citation style(ex. IEEE).

## 6.3 APPENDICES

Any additional information that would be helpful to the evaluation of your design document.

If you have any large graphs, tables, or similar that does not directly pertain to the problem but helps support it, include that here. This would also be a good area to include hardware/software manuals used. May include CAD files, circuit schematics, layout etc. PCB testing issues etc. Software bugs etc.