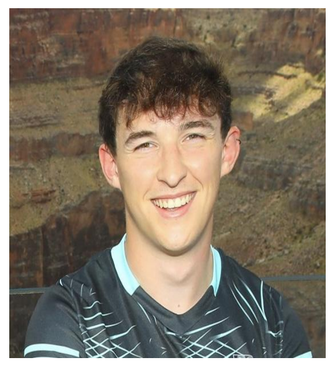
# AI Docker Linter & Optimizer:

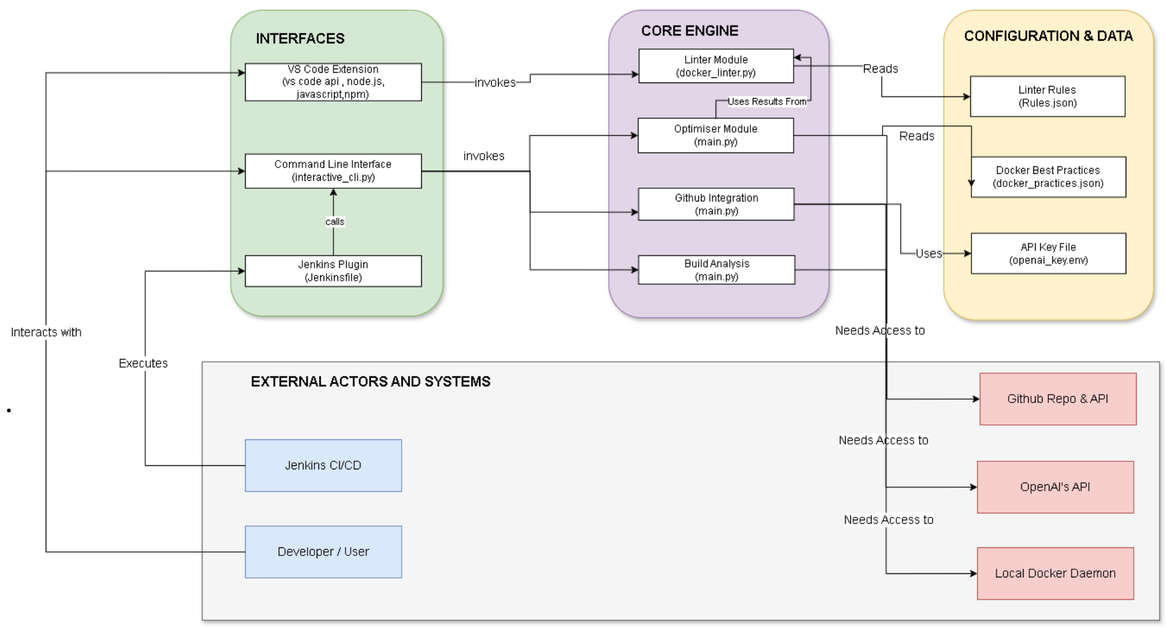
Peter Sheehan, BSc Honours in Computer Systems
Department of Computer Science,
MTU Cork, May 2025



## Introduction

In modern software development, Docker has become essential, yet crafting secure and efficient Dockerfiles remains a challenge. Inefficient images lead to slower builds, increased storage costs, and potential security vulnerabilities, while manually enforcing best practices across teams can be difficult. To address this, this project introduces an intelligent Docker linter and optimizer. This tool leverages AI to analyse Dockerfiles, identifying deviations from best practices and suggesting specific improvements for security and efficiency.

## Architecture




## Project Goal

The primary goal of this project is to simplify and automate the creation of secure, efficient, and maintainable Docker images. It aims to empower developers and DevOps teams reducing the complexity and manual effort required to adhere to Docker best practices, ultimately leading to faster builds, smaller images, and enhanced container security

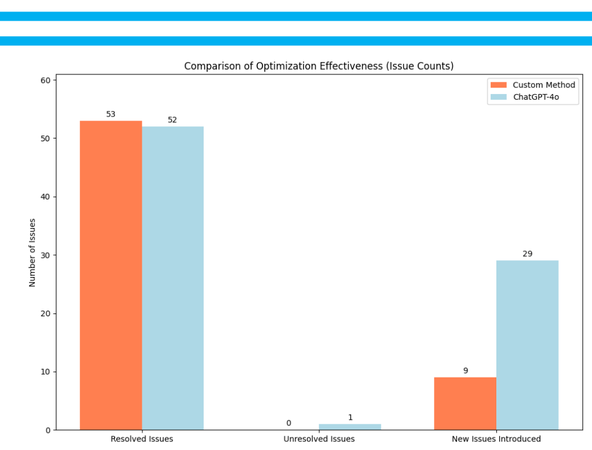
## Results and Conclusions

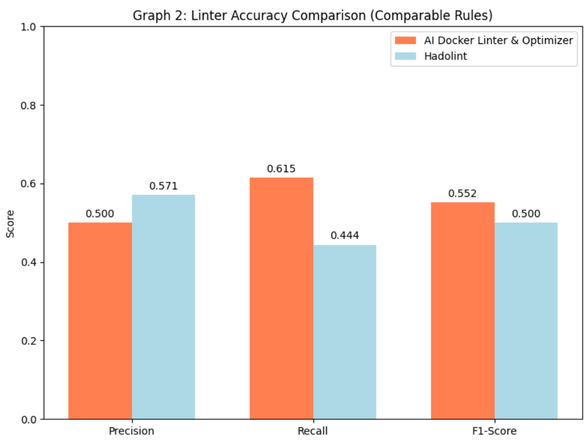
• Research and industry best practices emphasize the importance of optimizing Dockerfiles for smaller image sizes, improved security, and faster build times. Static linters like Hadolint are commonly used for identifying violations of these practices.

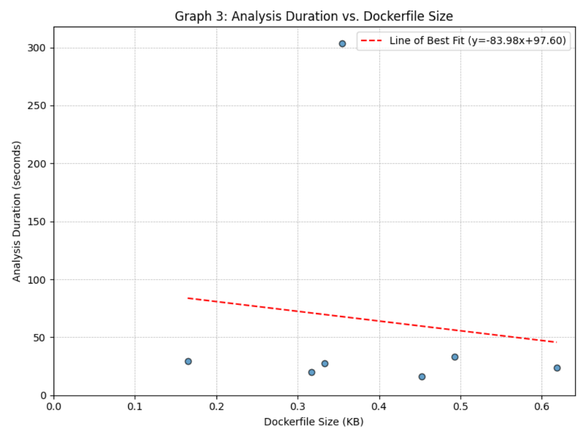
• Our data analysis confirms the effectiveness of automated optimization, demonstrating that the developed tool successfully built optimized images for 85.71% of the benchmark Dockerfiles, achieving an average image size reduction of 68.66% for those successful builds.

• Furthermore, the AI-driven linter component showed promising results compared to the established Hadolint tool on comparable rules, achieving superior Recall (0.480 vs 0.412) and a higher overall F1-Score (0.436 vs 0.412), indicating its potential to identify a broader range of issues, albeit with slightly lower Precision (0.400 vs 0.412).

• When comparing the end-to-end optimization effectiveness, the custom AI-driven approach significantly outperformed a generic baseline using ChatGPT-4o. The custom tool successfully resolved 100% of the annotated issues in the benchmark set (Recall: 1.0000), whereas the baseline ChatGPT-4o failed to resolve all issues (Recall: 0.9565). Furthermore, the custom method introduced fewer new potential issues (4 vs 6) during optimization, resulting in a superior overall F1-Score (0.9200 vs 0.8627). This highlights the value of incorporating specific linting results and targeted best practices context into the AI optimization prompt.




## Conclusions

This automation streamlines the development workflow, resulting in smaller images and faster builds with less manual intervention. Future development could introduce features like customizable rule sets or deeper integration with CI/CD pipelines

## References

1. Hadolint - https://github.com/hadolint/hadolint
2. Etc.
3. Etc.
4. Etc.

## Acknowledgments

The author would like to acknowledge Hristo Trifonov and Colin Manning for their ongoing support.