****

**National University of Computer & Emerging Sciences**

**Machine Learning Operations**

**Project Report: FYP Porting with MLOps tools**

**Group Members**

Momin Tariq 19I-0437

Safa Zaid Malik 19I-0555

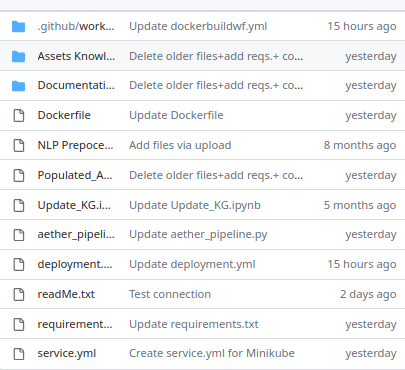
**Project Overview**

Our MLOps project is based on our Final Year Project, Aether, which provides an automation tool for voice to 3D scene generation in Unreal Engine 5. The objective of the MLOps project is to create a Continuous Integration and Continuous Deployment pipeline using selected tools that we learnt in this coursework.The report will highlight our implementation of a comprehensive CI/CD pipeline using GitHub Actions for Aether. Additionally, it discusses the utilization of Data Versioning Control (DVC) for managing large Megascans assets, the application containerization using Docker, and the setup of Kubernetes simulation via Minikube. These techniques and tools were instrumental in achieving a streamlined and efficient MLOps workflow for the development, deployment, and management of our project.

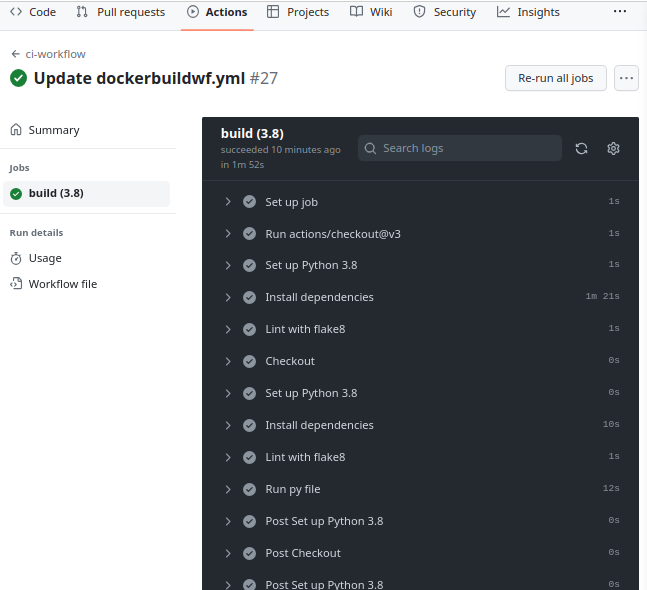
**Design & Implementation**

Building the continuous integration and continuous deployment (CI/CD) pipeline was essential to ensure smooth and reliable development of our project. To automate several steps of our pipeline, we used GitHub Actions, a potent workflow automation tool. The pipeline included the following significant steps:

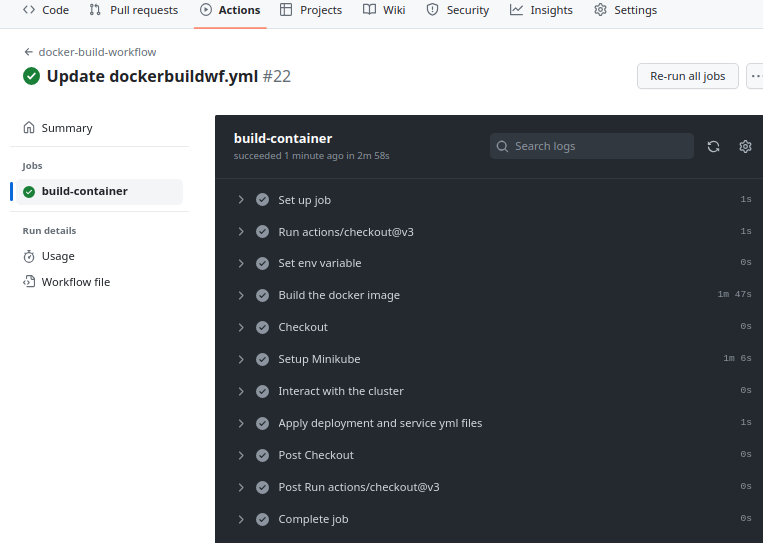
1. ***Version Control:*** To ensure a clear separation and managed access to the software, we kept separate branches for development and production. While separating team members' contributions between the primary production branch and the development branches, the version control system allowed for smooth cooperation among team members. Following is a snapshot of what our project scaffolding looks like:



1. ***Build Automation:*** By automating the build procedure, we made sure that any codebase modifications were swiftly integrated, built, and tested. Every push or pull request was accompanied by GitHub Actions, which started the build process and enabled early detection and feedback on dependency conflict and programming issues.



1. ***Containerization and Deployment:*** Our tool was containerized using Docker, making it simple to deploy across different platforms and ensured consistency in its behavior after getting rid of dependency problems The automatic production of Docker images was made possible using GitHub Actions, guaranteeing consistency and reproducibility.



1. ***Data Version Control (DVC):*** In machine learning projects, effectively managing huge datasets is essential. We used Data Version Control (DVC) to manage our massive Megascans assets efficiently. We were able to manage, share, and version these materials independently of the codebase thanks to DVC. We were able to maintain a manageable repository size while still having access to the required Megascans assets and the Knowledge Graph containing their metadata by decoupling the artifacts from the GitHub repository. Our Git workflow and DVC's seamless integration let us efficiently track changes to both code and data.
2. ***Minikube Setup for Kubernetes Simulation:*** We built up a Kubernetes cluster using Minikube to simulate and evaluate the scalability and performance of our application. We ran a single-node Kubernetes cluster locally using Minikube, simulating a production-like setting.