# Task 5 Deliverable

A problem that occurred when setting up the tensorboard is that after the nautilus servers went down the website still didn't work for one of the group members and they didn't know if it was still a server side issue or a client side issue as doing kubectl get pods on kubernetes showed the server as running without issues. Attempting to use access nautilus on different browsers yielded the same results. What eventually fixed this issue in the end was deleting the desktop deployment and restarting it. Another issue popped up when running the baseline RL algorithm. Our group initially had trouble implementing a stable baseline, as running the RL algorithm would give weird errors due to environment issues. To fix this issue, we had to create a new Python environment from the current one we were using. This time, we double-checked to make sure that the environment requirements matched the description of what environment stable baseline needed to be run on. However, this didn't completely fix the issue as we needed to reinstall certain packages like matplotlib, and numpy. After doing that, we were able to run the code with no missing module errors. However, we needed to make one last change as the simulator that the virtual car was supposed to run on was incorrect. Hence, we changed the version from CartPole-v1 to CarRacing-v2, which simulated a virtual environment depicting a town. In order to add moving cars to the environment, we added a separate parameter that stores the contents of the cars and iterated through the details of each outputted moving car.

We implemented a RL pipeline through the Stable Baselines3 library to train agents in a virtual environment, integrating advanced monitoring with TensorBoard and Weights & Biases (WandB). After training for 25,000 timesteps, the model is deployed to interact with the environment in real time, using a stochastic policy for exploration and rendering the car simulation. In addition, we implemented Proximal Policy Optimization (PPO) to train an agent on the CarRacing-v2 environment. It integrates with WandB to log hyperparameters, sync TensorBoard metrics, and optionally monitor performance and save models. The model is trained for 25,000 timesteps and logging is handled via a WandbCallback, which stores checkpoints and sends training data to the WandB dashboard for analysis and visualization.

Training the car racing agent seemed to go well. When we started training it, it started out very slow. It would slowly move down the track and steer back and forth a little while doing so. It was going slow enough that it didn't even make it to the first turn before resetting. Part way through training it was noticeably faster and was making it through a few turns before it would reset back to the start. For the most part it seemed to do a good job managing its speed and didn't get off the track that much. Once it had completed its training it was going noticeably faster through the track and was doing good staying on the track most of the time but ended up off the track more often than it had previously. When the car went off the track at high speeds it would end up spinning in circles until it eventually reset back to the start. Here is a link to a github repository with our code and pictures of our results.