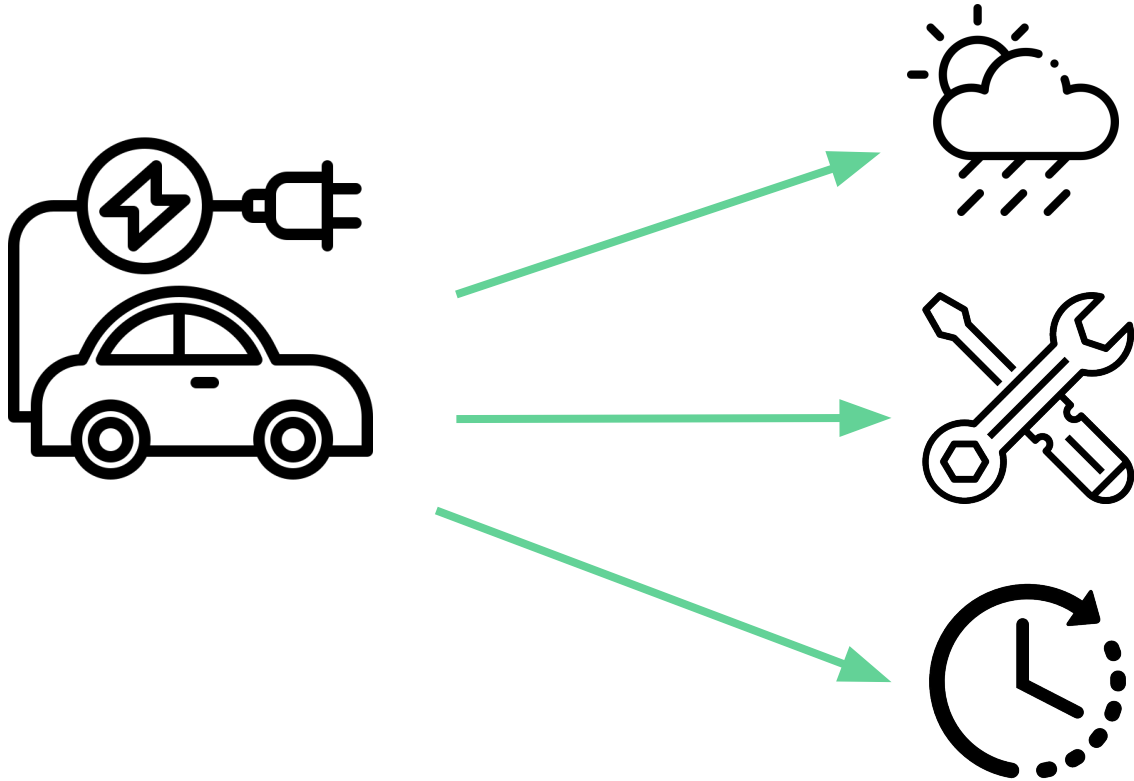


ME599 Project Presentation

Dockerization of CAV Codebase & CARLA

By: Urban Pistek

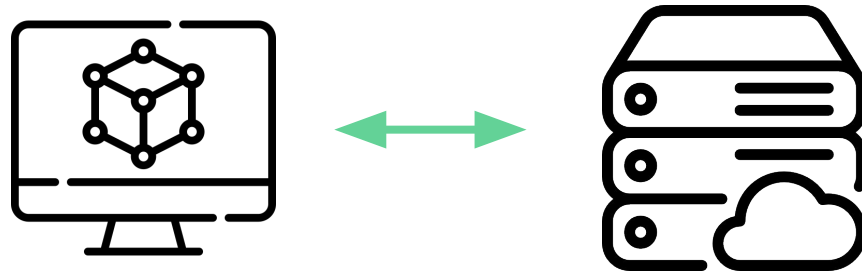
Background: Vehicle Software Development



Background: Vehicle Software Development

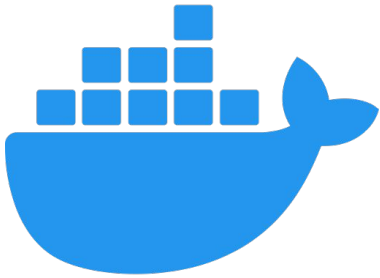
Need: Develop and test software without conflicting with rest of the team, in a controlled environment.

Solution: Leverage simulation and software tools to automate testing and parallelize development.



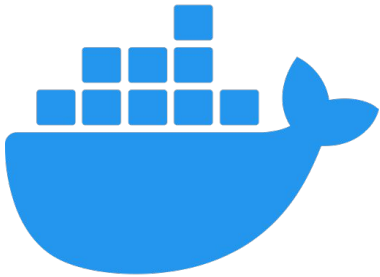
Background: Software & Simulation Tools

Solution: Leverage simulation and software tools to automate testing and parallelize development.



Engineering Methodology: Docker

Docker: A software platform that allows developers to easily create, deploy, and run applications in a virtual environment called a container.



Portability: Docker containers are platform-agnostic

Efficiency: Lightweight and use fewer resources than traditional virtual machines

Consistency: Built from a set of instructions called a Dockerfile, which ensures that the application is built and deployed in a consistent and repeatable way

Isolation: Docker containers provide a level of isolation between the application and the host system

Scalability: Docker makes it easy to scale applications horizontally by running multiple instances of the same container across multiple machines

Engineering Methodology: CARLA

CARLA: An open-source simulation platform designed for testing and developing autonomous driving systems.



Semi-Realistic Environment: Realistic and configurable simulation environment that includes realistic physics, weather conditions, and a range of urban and suburban scenarios

Safe & Cost-Effective Testing: Test autonomous driving algorithms and systems in a safe and controlled environment, without the need for real-world testing

Sensor Simulation: Supports lidar, radar, and cameras

Open-source & Extensible: Open-source platform, which means it can be customized and extended to meet the specific needs of individual developers and research teams

Engineering Methodology: ROS

ROS: ROS (Robot Operating System) is an open-source framework for building and programming robots

Modularity: Easy to reuse code and build on existing components

Flexibility: Supports a wide range of hardware and software platforms

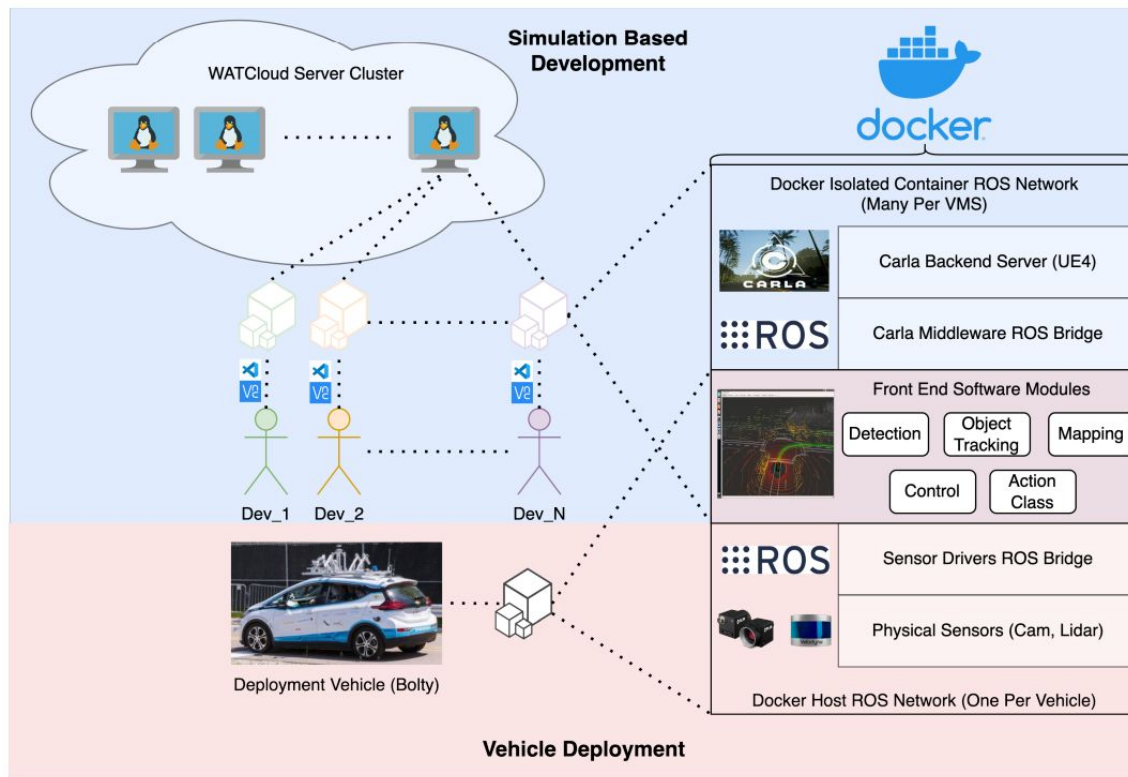
Large Community: Active community of developers, researchers, and users

Rich set of tools and libraries: Visualization tools, simulation environments, and software libraries

Open-source & Extensible: Open-source, which means it can be customized and extended to meet the specific needs of individual developers and research teams



Engineering Methodology: Architecture

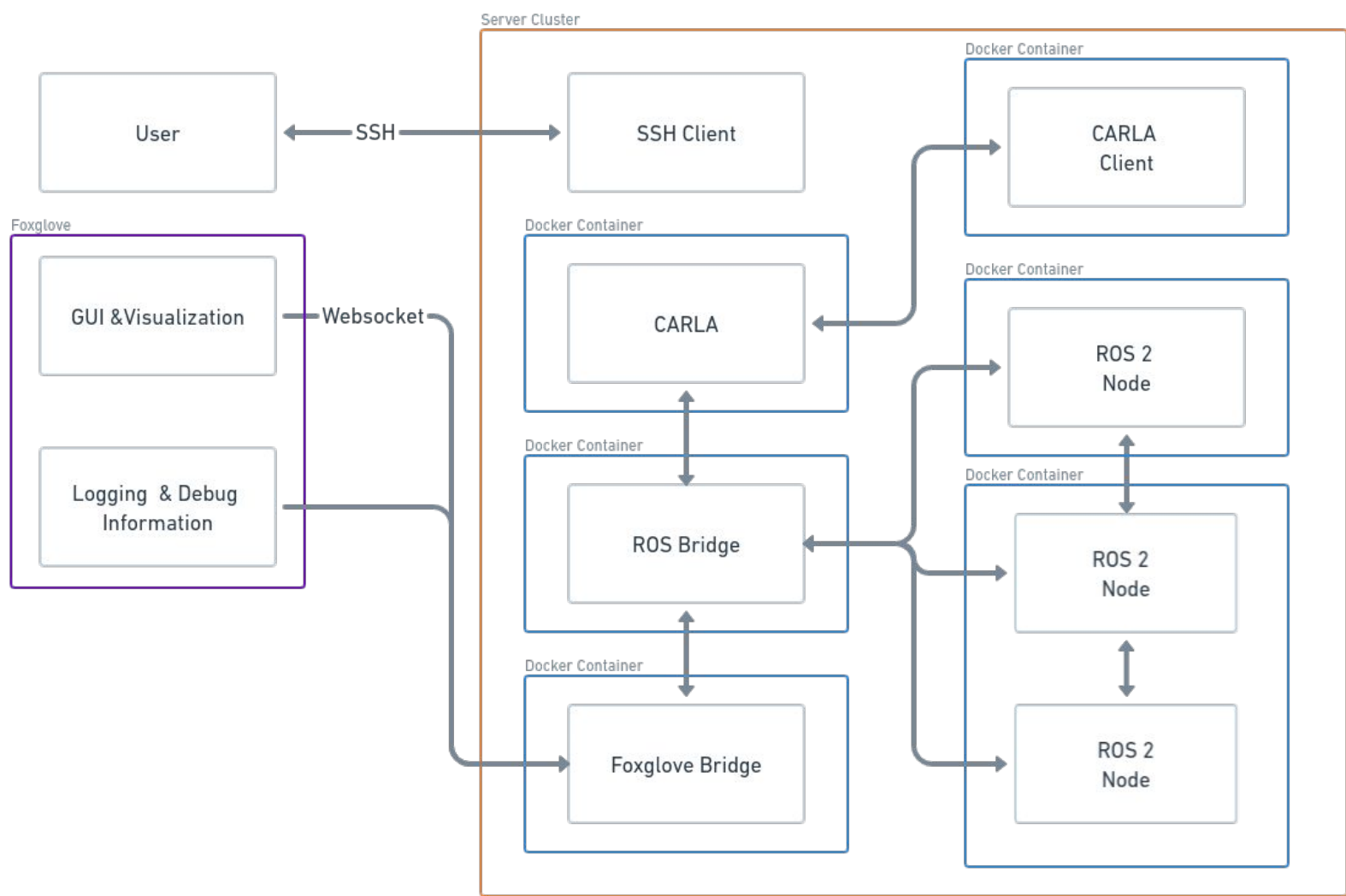


[1] WATonomous Software Architecture and Data Pipeline

Engineering Methodology: Architecture

Requirements:

1. Utilize ROS, CARLA and Docker and integrate them together
2. Able to run locally or in a server/cloud setting
3. Minimal environment setup / configuration - just deploy and run
4. Modular, able to run nodes separately
5. Able to scale by adding more nodes or tools
6. Can setting GUI / ROS data from server to local machine in real-time



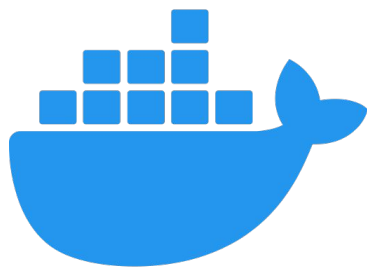
Challenges: Development

1. Building the ROS-Carla Bridge for ROS 2
2. Getting the ROS-Carla Bridge to connect to the Carla server
3. Compatibility between ROS / Carla / Python versions and releases
4. Getting a feed of the GUI remotely from the server instance
5. Using Foxglove for websocket data streaming

Challenges: Additional

1. Coordination with WATO for server access & other resources
2. Learning and getting familiar with CARLA
3. Less documentation for ROS 2 & newer Carla releases

Integration: Tools & Software Used

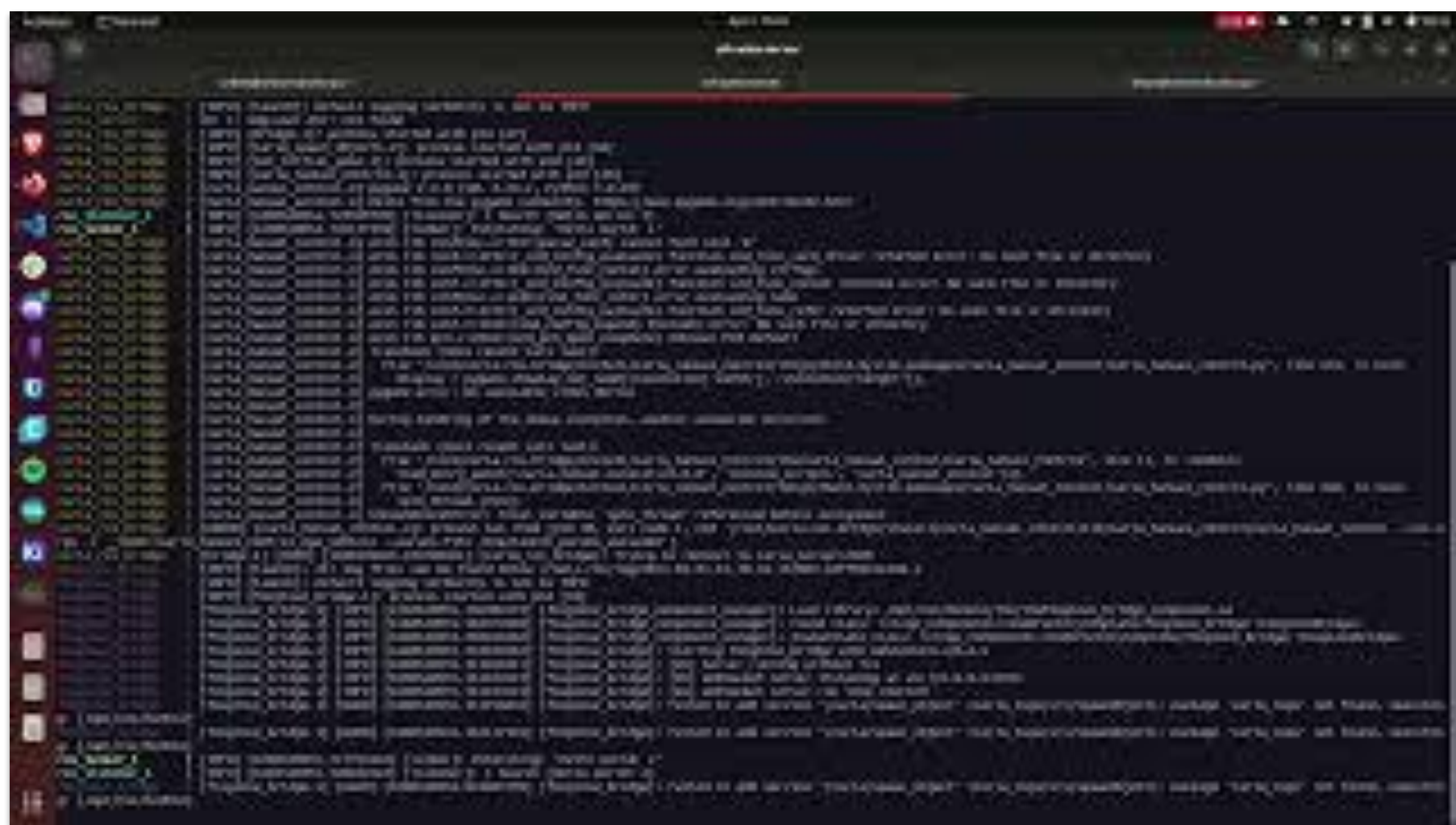


Integration: Demos

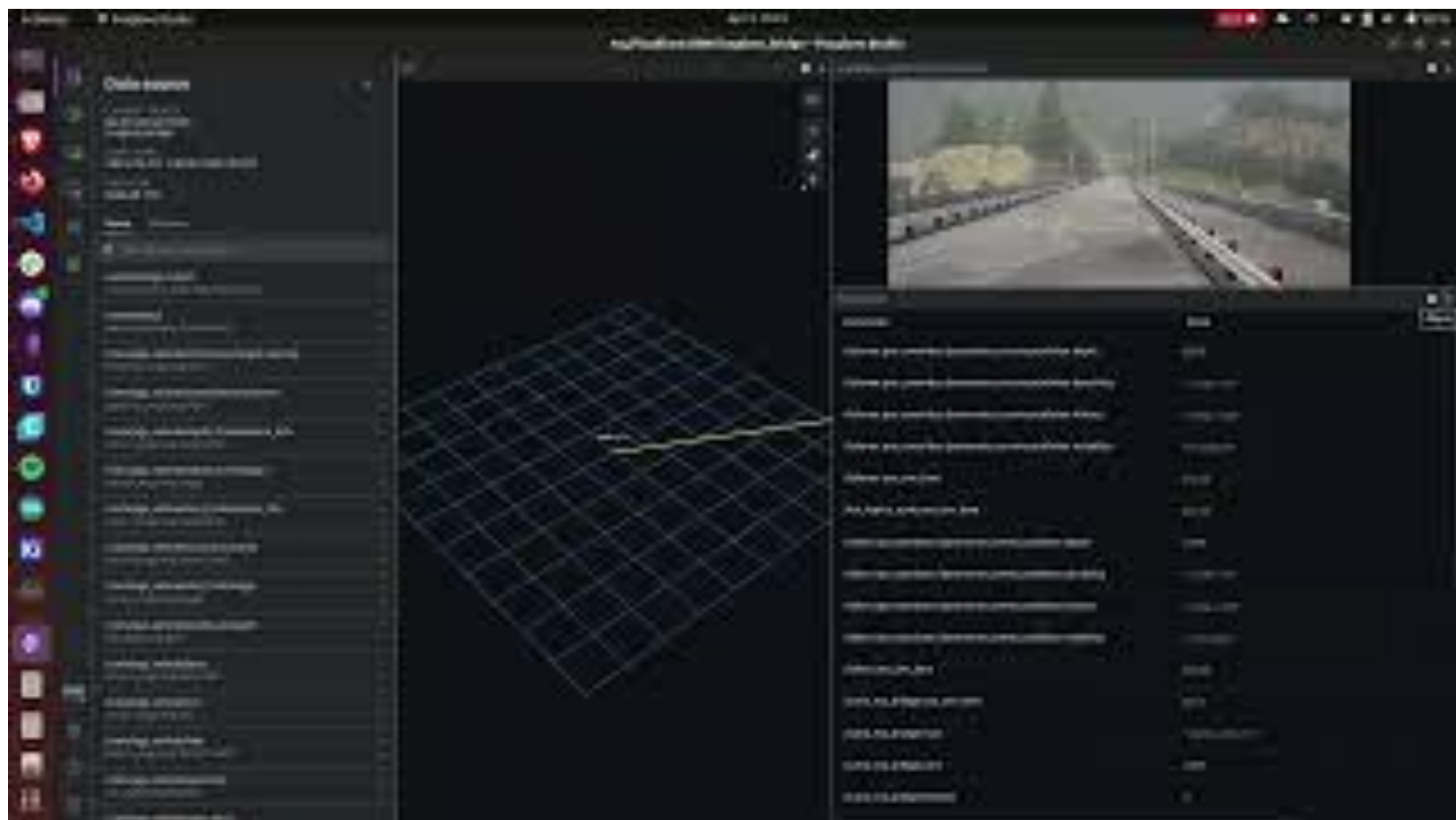
Demo 1

Demo 2

Demo 3



Application		Application		Application		Application		Application	
Application	Value	Application	Value	Application	Value	Application	Value	Application	Value
Application 1	100.0	Application 2	100.0	Application 3	100.0	Application 4	100.0	Application 5	100.0
Application 6	100.0	Application 7	100.0	Application 8	100.0	Application 9	100.0	Application 10	100.0
Application 11	100.0	Application 12	100.0	Application 13	100.0	Application 14	100.0	Application 15	100.0
Application 16	100.0	Application 17	100.0	Application 18	100.0	Application 19	100.0	Application 20	100.0
Application 21	100.0	Application 22	100.0	Application 23	100.0	Application 24	100.0	Application 25	100.0
Application 26	100.0	Application 27	100.0	Application 28	100.0	Application 29	100.0	Application 30	100.0
Application 31	100.0	Application 32	100.0	Application 33	100.0	Application 34	100.0	Application 35	100.0
Application 36	100.0	Application 37	100.0	Application 38	100.0	Application 39	100.0	Application 40	100.0
Application 41	100.0	Application 42	100.0	Application 43	100.0	Application 44	100.0	Application 45	100.0
Application 46	100.0	Application 47	100.0	Application 48	100.0	Application 49	100.0	Application 50	100.0
Application 51	100.0	Application 52	100.0	Application 53	100.0	Application 54	100.0	Application 55	100.0
Application 56	100.0	Application 57	100.0	Application 58	100.0	Application 59	100.0	Application 60	100.0
Application 61	100.0	Application 62	100.0	Application 63	100.0	Application 64	100.0	Application 65	100.0
Application 66	100.0	Application 67	100.0	Application 68	100.0	Application 69	100.0	Application 70	100.0
Application 71	100.0	Application 72	100.0	Application 73	100.0	Application 74	100.0	Application 75	100.0
Application 76	100.0	Application 77	100.0	Application 78	100.0	Application 79	100.0	Application 80	100.0
Application 81	100.0	Application 82	100.0	Application 83	100.0	Application 84	100.0	Application 85	100.0
Application 86	100.0	Application 87	100.0	Application 88	100.0	Application 89	100.0	Application 90	100.0
Application 91	100.0	Application 92	100.0	Application 93	100.0	Application 94	100.0	Application 95	100.0
Application 96	100.0	Application 97	100.0	Application 98	100.0	Application 99	100.0	Application 100	100.0



Recommendations: Improvements

1. Better configuration of Foxglove studio
2. Streamline docker images (optimize, make smaller)
3. Debug initial Carla-ROS-Bridge connection issue ([Issue#673](#))
4. Make wrapper to make deployment more configurable (Similar to [watod](#))
5. Work and collaborate with WATO more (Get server access, Join discord, join syncs with their simulation team members)

Lessons Learned: Knowledge

1. Better understanding of Carla, Docker & ROS
2. Better understanding of deploying docker clusters
3. Better understanding of dockerizing software & tools
4. Connect with experts and knowledgeable people early on
5. Collaborate everywhere possible

Summary: Links

[Find Example Project on Gitlab](#)

[Access to WATO Server](#)

[Running Carla in Docker](#)

[Foxglove Websocket Bridge](#)

Questions?
