Containerized Architecture Design for Portable Deployment of GENESIS Intelligent Agent Research Framework

Emily D. Carpenter
Marketing and Data Sciences
Anderson College of Business and Computing
Regis University, Denver, CO, USA
ecarpenter004@regis.edu

Abstract

This project continued data engineering contributions to the GENESIS (General Emergent Norms, Ethics, and Societies in Silico) research project at Regis University. The GENESIS project investigates whether intelligent computer agents are able to develop cooperative behaviors and moral reasoning through simulated interactions, specifically through the use of the Iterated Prisoner's Dilemma (IPD) testing environment [1].

The core simulation framework and the large language model (LLM) agent code have been provided by the principal investigators of the research project, and a persistent data storage solution was developed in the first practicum course (MSDS 692). This phase of practicum study containerized the research platform for portable deployment to allow the principal investigators to collaborate with academic partners in the field of Data Science to contribute to the ongoing Artificial Intelligence (AI) ethics research that explores the boundaries between optimization and moral behavior.

I. Introduction/Background

The purpose of this project was to contribute to the research on GENESIS (General Emergent Norms, Ethics, and Societies in Silico) proposed by Dr. Douglas Hart and Dr. Kellen Sorauf, Regis University, which investigates the behavioral patterns exhibited by intelligent agents and whether they follow Jonathan Haidt's six moral foundations, [2] starting with cooperation [1]. The research was conducted on the Framework for Observing and Researching Group Emergence (FORGE) platform via the department's compute cluster supporting LLM and multi-agent reinforcement learning (MARL).

This practicum contributed to the GENESIS research project by investigating and developing a solution to containerize the research code for distribution to other participating academic institutions and research teams in an effort to determine if intelligent agents participating in the Iterative Prisoner's Dilemma (IPD) testing environment to develop a sense of cooperation and morality. Additionally, the persistent storage solution provided in the first practicum (MSDS 692) was refined as needed based on feedback from current research participants.

II. Problem Statement

This practicum project continued ongoing research by Regis University professors that "investigates moral foundation emergence through institutional evolution," starting with "simple cooperation in fixed games" and culminating in "complex institutional ecosystems addressing cooperation, authority, loyalty, and potentially care and sanctity," behaviors analogous to Jonathan Haidt's moral foundations [1].

The programming code used in the research project was developed and deployed on a small compute cluster consisting of six Linux workstations of varying capabilities. By containerizing the project, the research project can be deployed in additional compute environments ranging from local clusters at universities to scalable cloud computing solutions that provide virtually unlimited resources (i.e. Amazon Web Services, Google Cloud, Microsoft Azure, Oracle Cloud, etc.).

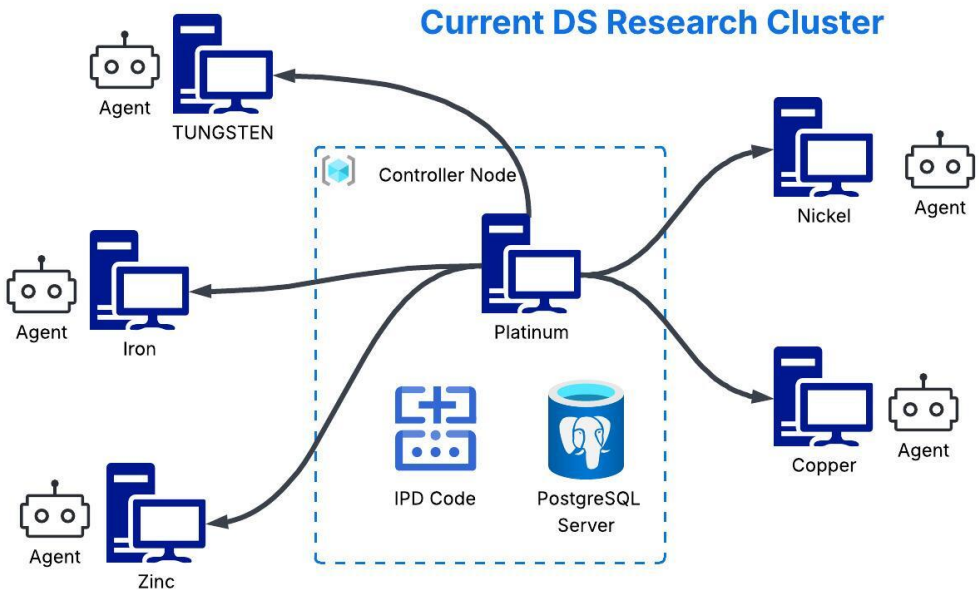

Fig. 1. Regis University Research Compute Cluster

Two possible architectures were considered for use in making the project code portable: Kubernetes and Docker Swarm. The first to be investigated was the lightweight Kubernetes distribution (K3s) given the popularity of Kubernetes in enterprise and commercial environments. Docker Swarm orchestration was also considered but was not pursued as discussed in further sections.

III. Related Work

The GENESIS research project draws on the body of work of Robert Axelrod [3], Robert Axelrod & William D. Hamilton [4], Jonathan Haidt [2], Steven Johnson [5], Elinor Ostrom [6] and Matt Ridley [7].

Software solutions considered for containerization orchestration include Kubernetes K3s [8] and Docker Swarm [9].

IV. Methodology/Approach

A. Existing Compute Cluster Research Platform

The GENESIS research platform operates on a six-machine compute cluster at Regis University (Figure 1). Each compute cluster machine (node) operates Ubuntu Linux with research software installed natively on each node (i.e. Ollama, PostgreSQL, Python, etc.) and one or more NVIDIA graphics processing units (GPUs) installed on each compute node. Research experiments are performed on the primary cluster node named ”Platinum” with Ollama agents distributed throughout the remaining nodes. Persistent storage was developed and installed on Platinum in early 2026.

Research Python code is designed to interface with the agent nodes in the cluster, and Ollama agents are managed on each cluster node through several shell scripts. However, management of each cluster node requires users to individually login through remote secure shell (SSH) and update software one cluster node at a time. Additionally, any issues with the Ollama services not handled through shell scripts require troubleshooting individually on each machine. This includes management of Ollama models used for research experiments.

While functional for the Regis University research team, this configuration is not readily made portable. Replicating the environment to a cloud compute cluster or another institution requires manually installing

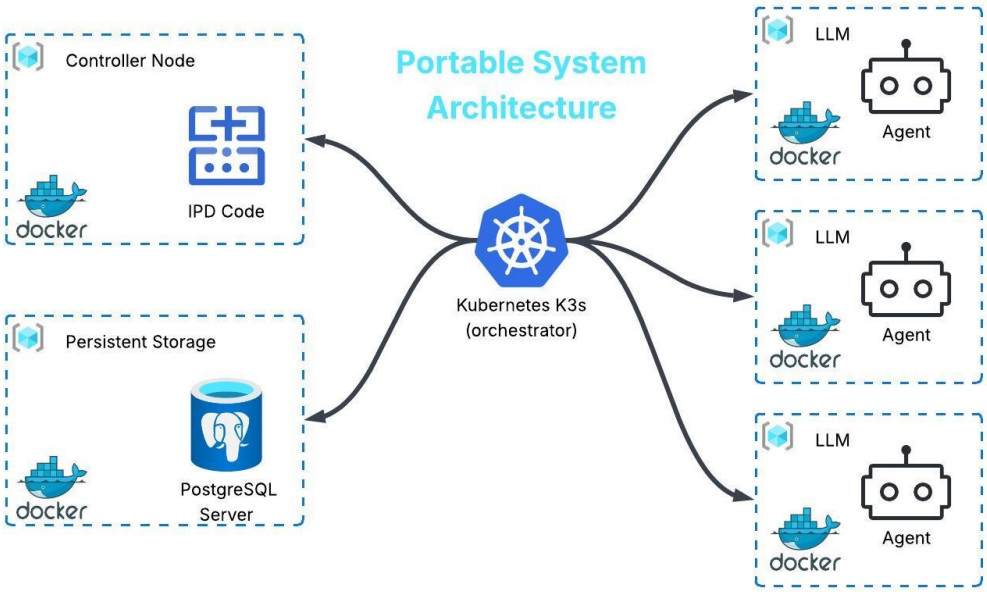

Fig. 2. Containerized Architecture Solution

and configuring each software component on every node, a process that is time-consuming, error-prone, and difficult to reproduce consistently.

B. Portability Solution

Two container orchestration platforms were considered for making the research platform portable: Lightweight Kubernetes (K3s) [8] and Docker Swarm [9]. Standard Kubernetes (K8s) is designed for enterprise-scale deployments and would be overkill for meeting the needs of the small number of agent nodes utilized in the GENESIS research [10] where K3s is specifically designed for small installations. Additionally, there are signs that the industry is moving away from Docker Swarm as a solution for managing clustered containers [11], [12]. For these reasons and to provide an industry-leading solution, Lightweight Kubernetes (K3s) was selected as the primary container orchestrator for this project.

K3s is a lightweight certified Kubernetes distribution originally developed by Rancher Labs as a reduced resource solution fully interoperable with Kubernetes APIs [10]. This application provides for easy installation via a simple command line shell script command that is compatible with a wide-range of backend data storage solutions (e.g. MySQL, PostgreSQL, sqlite3, etc.). Additionally, K3s doesn't require industry certification to understand and configure. This makes the application ideal for meeting small research cluster requirements [8].

Docker Swarm was a consideration at the start of this project, but given time constraints and issues experienced by users, this solution was not tested [12].

The existing bare-metal research cluster (Figure 1) was used as a model when designing the K3s Containerized Architecture. Several containers were configured to run on the controlling node Platinum: one to hold research code (forge-shell), one to host the persistent PostgreSQL database storage, and one for hosting an Ollama Agent. However, this agent was limited to the lightest of models due to limited hardware resources. Additional containers would be deployed to each of the other cluster nodes to host an Ollama Agent and configured to use the unique hardware resources each cluster node contained.

To meet the requirement of minimal management of cluster compute nodes, the software application Ansible was installed and configured for deploying the K3s Containerized Architecture solution. This application allows users to configure all requirements for a cluster on one controlling node and seamlessly manage installation and updates to all machines all without remotely logging in to each individual machine.

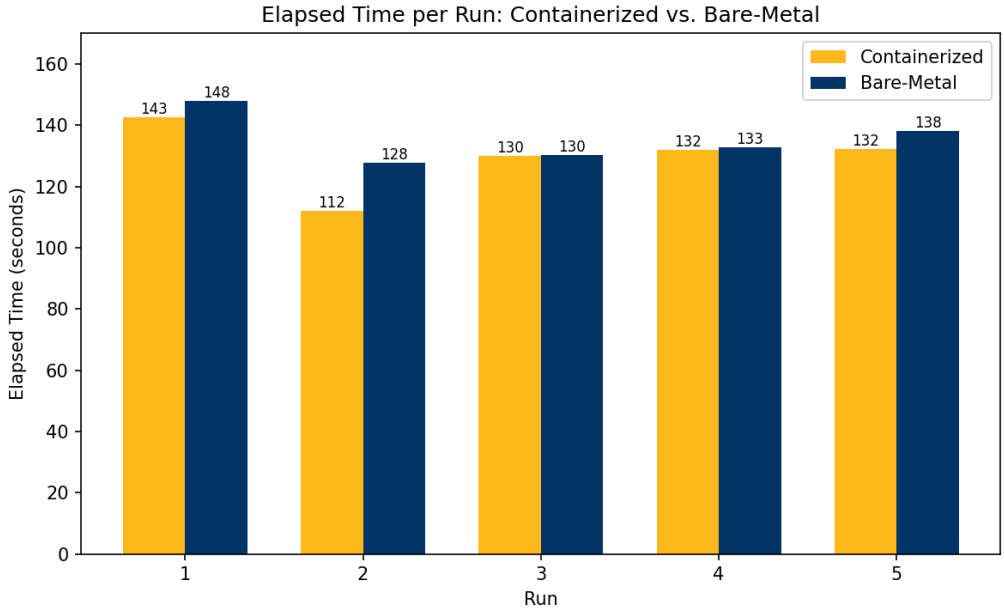

Fig. 3. Elapsed Time Per Experiment Run

Validation Runs Summary

Platform	Execution Runs	Total Rounds	Avg Elapsed Time (s)	Min Elapsed Time (s)	Max Elapsed Time (s)	Avg Cooperation Rate (Agent 0)	Avg Cooperation Rate (Agent1)
Bare-Metal	5	250	135.2689	127.6871	147.8621	0.776	0.764
Containerized	5	250	129.6929	111.937	142.5498	0.732	0.728

Fig. 4. IPD Research Code Execution Time Summary

Testing during the project saw dozens of installations and removal of software, all completed from the primary node (Platinum) without ever logging into a remote machine [13].

The resulting K3s Containerized Architecture is shown in Figure 2. Here you can see the controlling node (Platinum) hosting the research IPD Code and persistent storage solution container (PostgreSQL Server) as well as an Ollama Agent container running on each physical computing node of the Regis University Compute Cluster.

V. Results

To validate the Containerized Architecture, a series of experiments were conducted on both the containerized and bare-metal platforms using identical configurations. Five game sessions were executed on each platform, with each session consisting of 5 episodes of 10 rounds at a temperature of 0.2 (controls agent deterministic behavior, lower is more deterministic) using the llama3:8b-instruct-q5_K_M model. The bare-metal execution runs utilized the natively installed Ollama agent on the Tungsten compute node, and the containerized architecture execution runs interfaced with the K3s Ollama Agent configured on Tungsten.

Comparison of average elapsed time per session (250 IPD rounds) shows that the containerized Ollama Agent nodes completed execution in the same or better time when compared to the bare-metal execution runs (Figure 3). Overall, the average time to complete 250 rounds of IPD was approximately 135 seconds on bare-metal versus approximately 129 on the containerized architecture (Figure 4).

This shows that managing Ollama Agents inside Docker containers orchestrated by K3s had no observable impact to the execution of research experiments on the compute cluster. Initial impressions are

that the containerized architecture actually improved average execution time for research experiments. However, further extensive testing would be required to validate this statement.

VI. Conclusion

The GENESIS research platform was built and customized for use on the Regis University Compute Cluster. While the installation could be replicated on other platforms, this requires extensive work for a complete re-installation, with time extended for larger numbers of nodes.

This project researched, designed, and tested an alternative to the bare-metal architecture used at Regis University. The Containerized Architecture leverages industry standard applications such as Kubernetes (Lightweight version K3s), Ansible, and GitHub actions to package the research platform into a relatively portable deployment to cloud computing solutions or other hardware-based compute clusters with a relatively small investment to configure for deployment. This included incorporation of the persistent storage solution developed in the first practicum project by this author.

Additionally, containerization of Ollama Agents and Research code did not have an impact on code execution times. Instead, small-scale testing showed a slight improvement in execution times indicating that intensive experiment sessions may actually benefit from this architecture over bare-metal native installations.

The result is a research project that can be quickly cloned and deployed by other academic institutions to aid the research efforts of the principal investigators (PI). For example, given that agent instructions are presented in English language prompts, research teams will be able to test agent behaviors in other languages to monitor for emergent behavior by the participating agents.

Future enhancements could include automation of Ollama model management through the use of Ansible playbooks and agent deployments by leveraging Ansible Jinja2 templates. Additionally, larger scale tests should be completed on a cloud computing solution such as Amazon Web Services (AWS), Google Cloud Platform (GCP), or Microsoft Azure.

This work contributed to the GENESIS research project by providing a portable solution for principal investigators to more easily enable partner academic institutions to configure and deploy the project on compute clusters outside the Regis University Compute Cluster thus reducing barriers to collaborative efforts to further study of emergent behaviors and moral reasoning of intelligent agents.

Acknowledgments

The author wishes to thank Dr. Douglas Hart and Dr. Kellen Sorauf for the opportunity to participate in and contribute to the GENESIS research project.

Containerized Architecture and associated documentation developed with assistance from Claude (Anthropic; model: Opus 4.6) [14]. All code and content reviewed, edited, and approved by the author.

References

[1] D. Hart and K. Sorauf, "GENESIS project: From cooperation to conscience," 2025, Unpublished project documentation, Regis University.

[2] J. Haidt, *The Righteous Mind: Why Good People Are Divided by Politics and Religion*, 1st ed. Vintage, 2012.

[3] R. Axelrod, *The Evolution of Cooperation*. Basic Books, 1984. [Online]. Available: https://ee.stanford.edu/~hellman/Breakthrough/book/pdfs/axelrod.pdf

[4] R. Axelrod and W. D. Hamilton, "The evolution of cooperation," *Science*, vol. 211, no. 4489, pp. 1390–1396, 1981. [Online]. Available: http://www.jstor.org.dml.regis.edu/stable/1685895

[5] S. Johnson, *Emergence: The Connected Lives of Ants, Brains, Cities, and Software*. Scribner, 2004.

[6] E. Ostrom, *Governing the Commons*. Cambridge University Press, 2015.

[7] M. Ridley, *The Origins of Virtue: Human Instincts and the Evolution of Cooperation*. Penguin Publishing Group, 1998.

[8] K3s Project Authors, "K3s: Lightweight kubernetes," 2025, accessed: March 14, 2026. [Online]. Available: https://k3s.io

[9] Docker Inc., "Swarm mode," 2026, accessed: March 14, 2026. [Online]. Available: https://docs.docker.com/engine/swarm/

[10] E. Wahlquist, "K3s and K8s: Key differences and use cases explained," 2025, SUSE Communities. Accessed: April 2026. [Online]. Available: https://www.suse.com/c/k3s-and-k8s-key-differences-and-use-cases-explained/

[11] S. Muraru, “I almost ditched K3s for Docker Swarm (and why I didn’t),” 2025, accessed: April 2026. [Online]. Available: https://stefanmuraru.com/blog/k3s-vs-docker-swarm/
[12] Portainer, “5 best Docker Swarm alternatives & why you should migrate,” 2026, accessed: April 2026. [Online]. Available: https://www.portainer.io/blog/docker-swarm-alternatives
[13] Ansible Community, “Ansible documentation,” 2026, accessed: April 2026. [Online]. Available: https://docs.ansible.com/
[14] Anthropic, “Claude,” https://claude.ai/, 2025–2026, AI assistant used for database design, code development, and documentation.