Technical Report: Containerization of Abaqus FEA for Automated Cloud Orchestration

To: Dr. Qilin Li

From: Sasvidu Ranthul Abesinghe Mudiyanselage

Date: December 23, 2025

Subject: Sprint Update: Successful Engineering of the FEA Worker Layer & Automated Deployment Pipeline

1. Executive Summary

The most significant technical hurdle for this project was the deployment of the Abaqus Learning Edition (LE) environment. Traditional FEA software is notoriously difficult to automate due to GUI dependencies, Windows-centric licensing, and massive resource requirements.

This sprint successfully concluded with – to the best of our knowledge - the first-of-its-kind containerization of Abaqus LE 2024 within a lightweight Linux environment. This allows us to treat the simulator as a "stateless worker" that can be triggered by our AI Orchestrator, fulfilling the requirement for a scalable, cloud-native simulation pipeline.

2. The Technical Challenge: The "Legacy Wall"

During research and development, we identified three critical barriers to the planned architecture:

- **Platform Restriction:** Abaqus LE is officially supported only on Windows with a mandatory Graphical User Interface (GUI). This contradicts our goal of using Azure Linux-based serverless/container infrastructure.

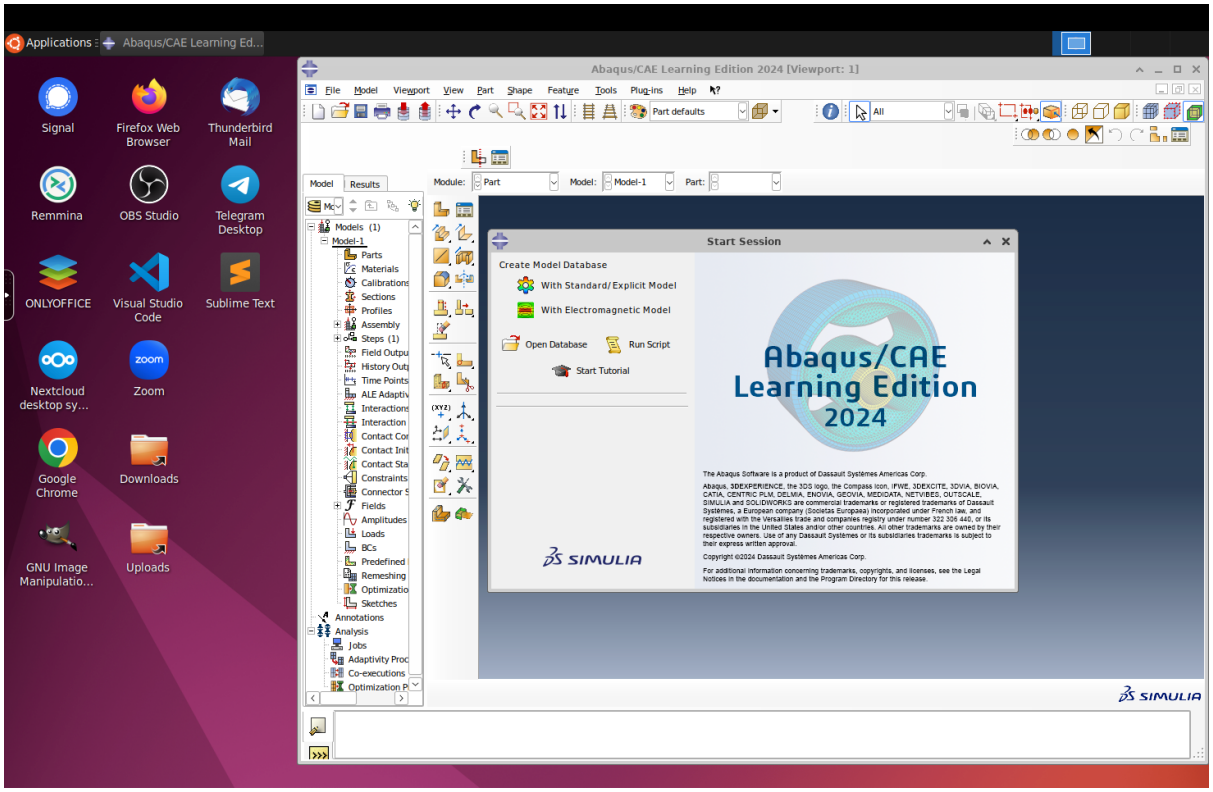
- **Automation Deficit:** The LE version lacks the standard command-line flexibility of the Professional Linux version, making it difficult to trigger via API.
- **Resource Constraints:** Abaqus typically requires high-performance hardware (8GB+ RAM). To keep cloud costs viable for a student project/research agent, we needed it to run on "low-tier" hardware (4GB RAM).

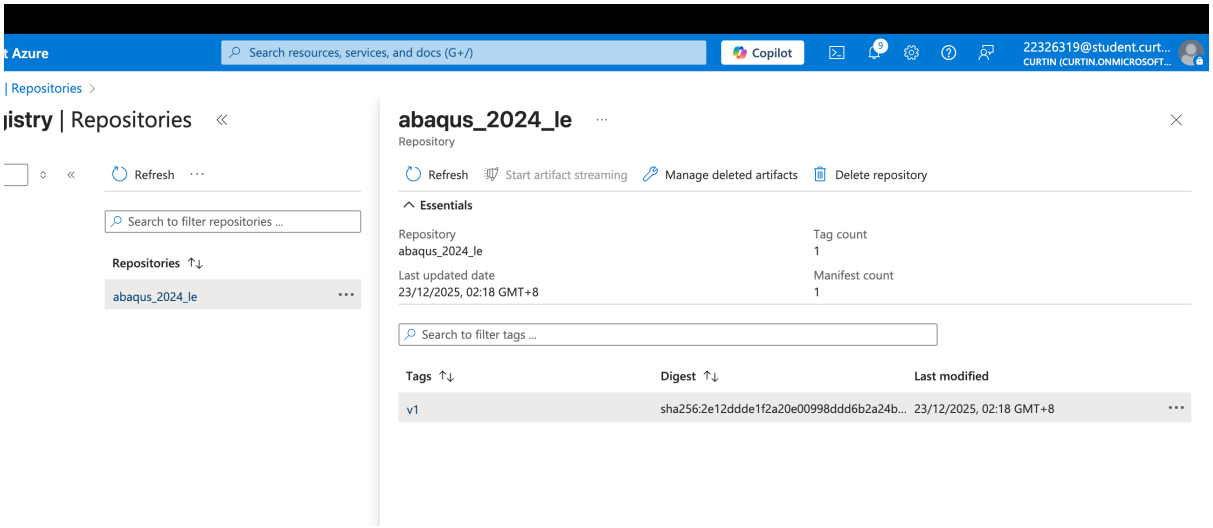
3. The Solution: "The Gold Image" Architecture

We engineered a custom Docker container that "tricks" the software into thinking it is running on a high-end Windows machine while remaining fully scriptable via Linux.

Technical Stack:

1. **Host:** Azure Ubuntu VM (Standard B-Series).
2. **Container Base:** Kasm/VNC Ubuntu Jammy Desktop (provides a virtualized display buffer for the GUI requirements).
3. **Compatibility Layer: Wine 9.0+** configured with a custom Windows 10 prefix.
4. **Abaqus Core:** Custom installation based on the *mwierszycki* research, adapted for 2024 LE.
5. **Graphics:** Mesa-based software rendering to allow GUI-based verification without a physical GPU.





4. Production Workflow & Scalability

In the actual production environment, the complexity of Wine and GUI virtualization is **abstracted away**. The system now functions as a "Worker Layer":

- **Registry-Based Deployment:** The environment is stored as a 20GB "Gold Image" in an **Azure Container Registry (ACR)**.

- **On-Demand Instantiation:** When the AI Agent generates an .inp file, the orchestrator spins up a container, executes the job, and shuts down.
- **Hardware Efficiency:** We successfully optimized the environment to pass all Abaqus verification tests on a **2-core, 4GB RAM** instance—a significant reduction in typical resource overhead.

5. Next Steps

1. **Full Scripting/Docker Compose:** Finalizing the docker-compose file to link the Orchestrator, the FEA Worker, and the Blob Storage.
2. **Batch Processing:** Implementing the agent logic to handle multiple .inp configurations simultaneously.
3. **Model Upgrade:** Transitioning from the "mini" model to a more capable LLM via MCP (Model Context Protocol) to improve the quality of the NLP-to-FEA conversion.

Final Technical Note

While this research utilized the LE version for proof-of-concept, the entire pipeline is **version-agnostic**. In a commercial or large-scale academic deployment, the "Worker" container would simply be swapped for a licensed Linux-native Abaqus image. The orchestration logic remains identical, proving the robustness of the current architecture.