High Availability Containerized Application Deployment Project

<https://github.com/DallasAD/wikijs-ha-gcp-deployment>

04/20/2025

COP 3604.01

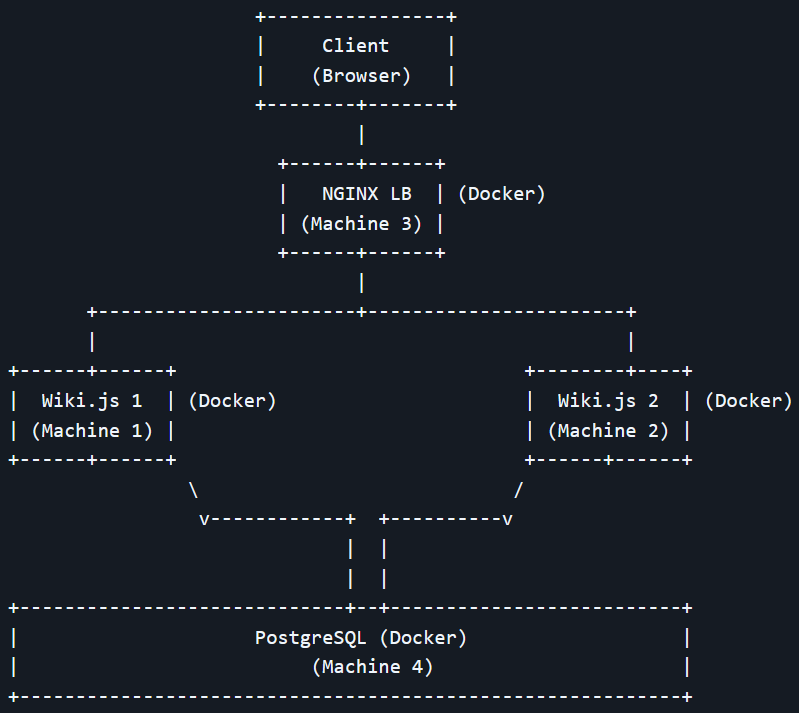
Dallas Desimone, Connor Hardee, Brenna Holloway, Justis Nazirbage

**Introduction**

This project aims to utilize a variety of industry-standard tools to provide high-availability deployment of a web application. We demonstrate the high-availability deployment of containerized instances of Wiki.js across multiple Google Cloud Platform (GCP) virtual machines. By splitting up the work of this modular system, we were able to accomplish this task in parallel. The Project Lead was [\_\_\_], who lead everyone in staying on track, providing updated to-do lists, assigning tasks, and assisting in the implementation. The Infrastructure & Automation Lead was [\_\_\_], who lead setting up the virtual machines and making sure the firewall rules and subnet configuration was properly set up. The Application Deployment Lead was [\_\_\_], who lead writing the compose files and setting up the Wiki.js instances. The Data Management Lead was [\_\_\_], who lead setting up the database, LVM, and load balancing.

**System Architecture & Design**

The architecture includes two dockerized instances of Wiki.js, which were deployed on separate GCP VMs, an NGINX load balancing container to distribute incoming traffic, a centralized PostgreSQL database, a Linux Logical Volume Manager (LVM) which was used for persistent storage of database data, automation scripts to avoid manual orchestration and promote infrastructure-as-code best practices, and secrets managed securely via environment variables.

This deployment provides horizontal scalability, failover tolerance, and persistent storage in a multi-tiered architecture, which is suitable for real-world cybersecurity operations labs or student infrastructure projects. A diagram of the architecture is shown to the right.

**Implementation Details**

Application Deployment

We used Wiki.js for our application, which is a powerful and extensible open source wiki software. It is compatible with PostgreSQL, which is what we decided to use. We simply deployed two containers, each in their own GCP VM. The compose file for the first container is shown below. The “.env” file allows the application to connect to the PostgresSQL database using the user and password specified.

A screen shot of a computer

AI-generated content may be incorrect.

High Availability

As mentioned above, two containers were used to handle the load of the HTTP requests. The application is made highly-available by ensuring the application can still run even if one instance fails, since there are multiple instances of it.

Load Balancing

To ensure that our deployment was highly-available, we utilize an NGINX load balancer so HTTP traffic can be distributed among the Wiki.js containers evenly. This prevents failures, and handles failures if they arise. The compose file for the load balancer container is shown below, as well as the config file.

A screenshot of a computer program

AI-generated content may be incorrect. A computer screen with white text and black background

AI-generated content may be incorrect.

Persistent Storage

Automation

The application achieved automation through Docker Swarm. This allowed for one instance to be a dedicated manager of all of the Docker containers, which were also separated into their own GCP VMs. In doing so, a single compose file was created for the manager node to use. To handle things running in order, scripts were written to check for readiness. Below is the single compose file, as well as some scripting that was used.

A screen shot of a computer program

AI-generated content may be incorrect. [INSERT SCRIPTING FILES ONCE DEVELOPED]

Secrets

Secrets were secured using environment variables in a hidden “.env” file. This helps to maintain the security of our deployment by restricting what is shown and hosting the “.env” file locally. This file is shown below.

A computer screen with white text

AI-generated content may be incorrect.

Database Setup

[database setup blurb] The compose file is shown below.

A screenshot of a computer program

AI-generated content may be incorrect.

GCP Infrastructure

**Problems Faced**

An initial problem we faced was with using TheHive as our application. There were several issues with this approach, so we switched to using Wiki.js, which helped with our horizontal scaling.

Another problem we faced was with communication between our virtual machines. We realized that one VM was configured to a different region than the others, so the internal IP addresses could not work. An additional problem we faced relating to the VM’s was being unable to access each other’s files in each VM. This was because of a lack of Identity and Access Management (IAM).

**Testing Results**

**What Was Learned**

Dallas Desimone

Connor Hardee

Brenna Holloway

Justis Nazirbage

This project was great at reinforcing the material we learned in class. At times in class, I would be confused about how certain tools fell into the “big picture” of containerization. This project really cleared that up for me as my teammates and I worked on the several components that make up a highly-available containerized deployment of an application.

I was able to dive deeper into learning about compose YAML files when my teammate developed them. The load balancing was something that I talked about in my “Stand for Linux,” relating to Netflix’s use of it, so seeing how my teammate implemented it into our project was great. I was able to set up the Swarm manager and its nodes, and I contributed to the scripting that was involved in making sure they ran in the correct order. Overall, this project was a great learning experience, and my teammates were fantastic to work with.

**Individual Contribution Table**

|  |  |  |
| --- | --- | --- |
| Name | Activity Type | Effort |
| Dallas Desimone |  | % |
| Connor Hardee |  | % |
| Brenna Holloway |  | % |
| Justis Nazirbage |  | % |

**Conclusions & Future Work**

From working on this project, the team was able to learn and practice setting up the GCP infrastructure, setting up the database, deploying highly-available applications, implementing load balancing, utilizing automation, ensuring security, and working with persistent storage.

This project serves as a great foundation and opportunity for expansion into more complex and customized applications. A future project might be an e-commerce store, where customers typically send many HTTP requests. Through containerization and high-availability techniques, we can ensure that the application stays up and the deployment is secure.