Perspecta Situational Awareness Command and Control

Utilizing Automation and Orchestration technologies to have highly available services with no loss of Availability

Student Team: Mohamed Ahmed, Lithe Abushaikha, Ali Sharaf, Ali Alktebi, Ammar Al-Kahfah

Mentorship Team: Pete Schmidt (Perspecta), Zachary Estes (Perspecta), Joseph Landino (Perspecta), Rock Sabetto (GMU)

Introduction

Perspecta, on behalf of the **Department of Defense**, has **mission-critical** applications running on servers all over the world. In the event of a compromised server, the DoD/Perspecta is unable to move an application with its data and dependencies in a timely manner. The objective of our senior design project is to **develop** an **application/solution** that will be able to **securely move software** from a **compromised** server to a **non-compromised** one. By **containerizing**, **automating** and **orchestrating** the mission-critical **applications** on the current servers, the DoD/Perspecta will be able to achieve this efficiently and securely.

Find a solution that's
- Lightweight
- Efficient/Fast
- Secure
- Resilient
- Scalable

DevOps as a method for
- Collaboration
- Managing tasks
- Sharing code, ideas and resources
- Building and Testing
- Automation

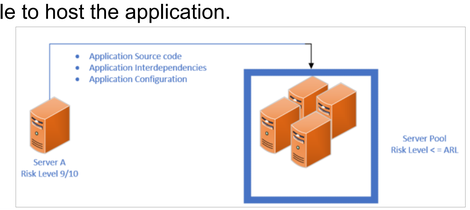
Experiment with new technologies
- Source Control & CI/CD: Gitlab
- Containers: Docker
- Orchestration: Kubernetes
- Configuration Automation: Ansible

Aim

Our goal in this research is to propose solutions on how we can **move applications** with all **interdependencies** from one server to another. The transfer should occur when the reporting server hosting the application has an unacceptable risk level.

CRL (Current Risk level) > ARL (Acceptable risk level)

Risk level should be determined by an infrastructure level Vulnerability scanner or manually reported by Cybersecurity personnel if needed. The transfer, setup, and hosting of an application should be fully automated with minimal human interaction. For example: Assuming that the ARL is 6/10 and Server A CRL is 9/10, the applications and interdependencies on Server A will be packaged and transferred to another server that will be able to host the application.

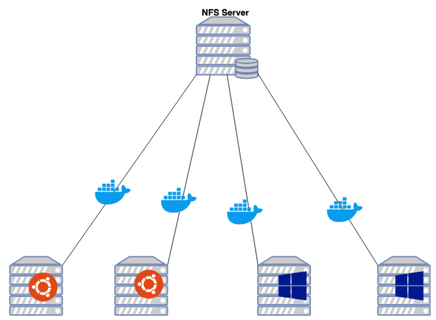


Method 1 (Brute force)

- **Gitlab-runner + Docker**
  - Have GitLab-runner agent listening to jobs
  - When data is committed to repository, a job is triggered
  - The YAML file is interpreted and a set of pre-defined commands are used to delete the old container mapped data in the running application and copy/map the newly pushed data into the right directory
  - As a server goes down, on shutdown the new data will be committed triggering the GitLab job and listening GitLab-runner will be able to receive and run the job conducting the fully application move with data persistence

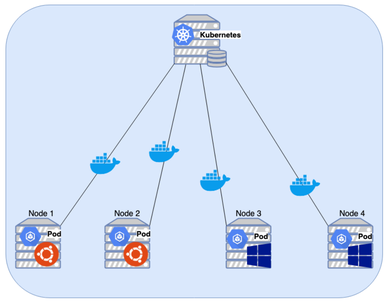
Method 2 (Optimal)

- **Docker + central NFS server**
  - Setup a central NFS server in which we'll map and configure all container volumes onto it
  - If one container dies, we'll be able to spawn another container of the same application while having our data mapped and persisted onto the NFS server



Method 3 (Unfavourable)

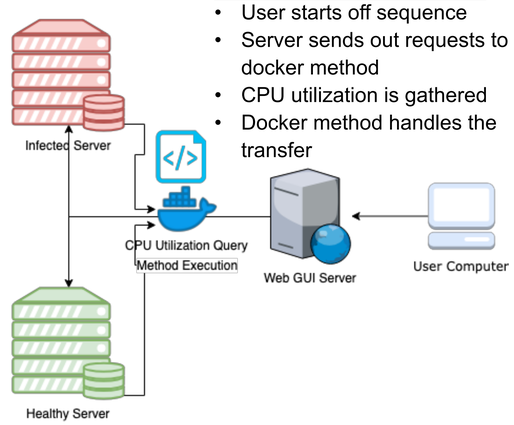
- **Kubernetes**
  - Orchestrate Method 2
  - Have a master for configuration and deploying our application onto worker nodes
  - Have an underlying NFS server for data persistence across all nodes/pods
  - As nodes/pods fail or get deleted, Kubernetes will schedule new pods onto other available worker automatically without any human intervention.
  - Data will persist as all deployments will be using an NFS persistent volume claim that'll map the data into one central location



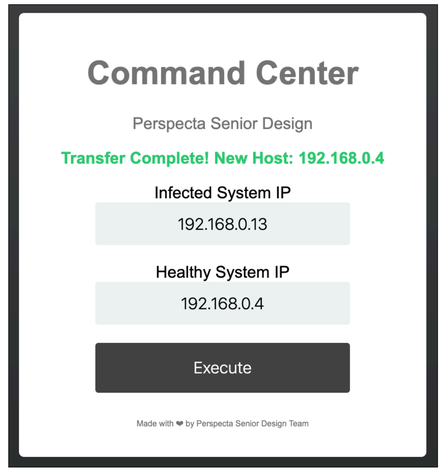
Web Interface

-Web Interface Architecture

- User starts off sequence
- Server sends out requests to docker method
- CPU utilization is gathered
- Docker method handles the transfer



Web Interface Cont.



Conclusion

The focus of this project is to **minimize risk while maintaining optimal performance, availability and minimal cost in the use of mission-critical applications**. Through the use of containerization, automation and orchestration technologies this can be implemented to efficiently and securely achieve this functionality. The methods provided are all proven to be working in environments similar to Perspecta's. We provided a thorough report detailing all three methods along with how to implement them, and the pros and cons of each method. As mentioned earlier, Perspecta is going to use this product on behalf of the United States Department of Defense, and due to the nature of their work that Perspecta is going to apply their method of choice to, we were unable to assist with the actual implementation of the methods.

References

[1] R. Morabito, "Power Consumption of Virtualization Technologies: An Empirical Investigation," 2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC), Limassol, 2015, pp. 522-527.

[2] J. Zhang, X. Lu and D. K. Panda, "Performance Characterization of Hypervisor-and Container-Based Virtualization for HPC on SR-IOV Enabled InfiniBand Clusters," 2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), Chicago, IL, 2016, pp. 1777-1784.

[3] A. M. Joy, "Performance comparison between Linux containers and Virtual machines," Computer Engineering and Applications (ICACEA), 2015 International Conference on Advances in, Ghaziabad, 2015, pp. 342- 346.

[4] W. Felter, A. Ferreira, R. Rajamony and J. Rubio, "An updated performance comparison of virtual machines and Linux containers," Performance Analysis of Systems and Software (ISPASS), 2015 IEEE International Symposium on, Philadelphia, PA, 2015, pp. 171-172.