Fermilab GlideinWMS Deployment: Automation with Docker, Kubernetes, and Helm

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Aidan Himley

Spring 2023

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Rosanne Vrugtman, Department of Computer Science

Fermilab GlideinWMS Deployment: Automation with Docker, Kubernetes, and Helm

CS4991 Capstone Report, 2023

Aidan Himley
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
ah8uqq@virginia.edu

ABSTRACT
Fermi National Accelerator Laboratory's (Fermilab's) Glidein Workflow Management System (GWMS) is a complex software system that requires expertise to deploy and operate. To streamline the deployment process, I created tools to automate the installation of the software and simplify the configuration options. I used Docker to create portable container images of various parts of GWMS, Kubernetes to automatically set up a group of containers and the networking between them, and Helm to easily customize the Kubernetes cluster. I also made extensive use of shell scripting within the containers to automate features such as credentials management that could not be pre-defined at the image level. My project resulted in easily-deployed containers for the Frontend and Central Manager parts of the system. Further work is needed to containerize the other components of GWMS and to ensure the configuration options provided are sufficient to fit the users' use cases.

1. INTRODUCTION

GWMS is used by the High Energy Physics community to facilitate large amounts of data processing by matching computational jobs with worker machines in a distributed computing network (Fermilab, n.d.). It is built on top of another system, HTCondor, which provides the basic features necessary to submit jobs, add worker machines to the pool of available resources, and find suitable matches between them (HTCondor, n.d.). The role of GWMS is to extend the functionality of HTCondor through the use of placeholder jobs called glideins. Use of GWMS can improve on HTCondor alone by offering increased automation of worker machine discovery and management, as well as enhanced job management features such as reordering and monitoring.

The installation process for GWMS is long and complex. It requires detailed knowledge of the configuration options and authentication methods available, and it must be installed on a particular operating system. By automating and containerizing the installation process, I aimed to enable new users to quickly deploy GWMS and reduce the dependency on specific operating systems, overall lowering the barrier to entry.

2. RELATED WORK

Among the prior work done on GWMS containerization, Davila (2022) provided a basic Docker container for the GWMS Frontend. I built upon this framework by adding installation steps and network configuration.

I also referenced the technical documentation for the software tools I used to build my solution. I used the tutorials and reference materials from the Kubernetes documentation

to learn how to create networked clusters of containers (Kubernetes, n.d.). Similarly, I used the Helm documentation to learn how to create customizable templates for Kubernetes clusters (Helm, n.d.).

3. PROJECT DESIGN

For several components of the GWMS architecture, my work comprised the development of Docker images, a Kubernetes cluster, and a Helm chart.

3.1. Overview of GWMS Architecture

Both HTCondor and GWMS consist of several separate components, usually installed on separate devices. HTCondor components include submit hosts, where users submit their jobs; worker nodes, which run the jobs; and the Central Manager, which collects a pool of jobs and worker nodes. The GWMS components are the Frontend, which queries the Central Manager to discover user jobs; and the Factory, which generates glideins and submits them to worker nodes. The main components and flow of information are illustrated in Figure 1. The installation of any one of these components is highly complex – for example, the Frontend installation instructions are many pages long (Open Science Grid, n.d.).

A major reason for the complexity of installation is that all components must communicate over a network with at least one other – for example, the Frontend must communicate with the Central Manager to find jobs and the Factory to request glideins, and glideins must communicate with the Central Manager to receive jobs. Therefore, each component must be configured to know where the others are. In addition, the network between components cannot be considered secure, so components must also be able to verify the authenticity of messages from others. During the setup process, components must be told which peers to trust and which

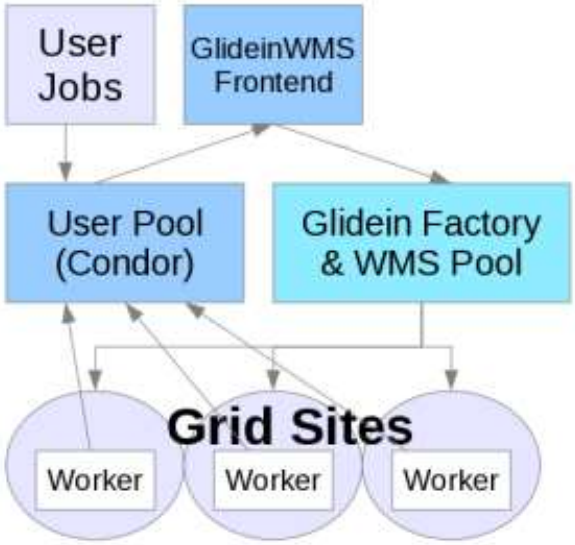

Figure 1. Overview of GWMS Architecture (Fermilab, n.d.)

authentication methods to use, and credentials must be shared between components.

My general approach to the solution was to place GWMS and HTCondor components in containers, specifically the GWMS Frontend and HTCondor Central Manager. Because much of the setup required involves networking and authentication, I also built tools with Kubernetes and Helm to automatically manage groups of containers.

3.2. Docker Images

Docker is a software system that allows programmers to bundle their software with all of its dependencies, including the operating system, in a software object called a container. From the perspective of the software running in the container, the container is the computer on which it is running, which it has entirely to itself. To create containers, programmers write a text file specifying an operating system to use and a sequence of commands to apply to set up the environment. Docker applies these instructions to create a pre-built file called an image that represents the starting state of a container. Images are completely static and

are instantiated to create running containers, which can then be modified by the software running in them.

The first step in my project was to provide images for the HTCondor Central Manager and GWMS Frontend. An image for the Central Manager had already been created by the Open Science Grid, which I used unmodified. A GWMS Frontend prototype image had been provided by Davila (2022), which I modified to further automate deployment. I added directories in the filesystem to hold credentials that would be generated at runtime, added a credentials management utility in the installation instructions, and removed a pre-generated password. Because images are static, the password must be generated at container creation instead of image creation to avoid all container instances using the same password. I also uploaded the resulting image file to a public repository.

3.3. Kubernetes Cluster

My next step was to create an automated way to set up the networking between components and container configuration that cannot be done at the image layer, including credentials management. For this step, I used Kubernetes, a tool for managing clusters of containers and their interface with the host machine (the computer on which the containers run). To use Kubernetes, programmers describe the cluster configuration in text files called manifests, which Kubernetes reads to create a running cluster.

To enable networking between components, I wrote manifest files to connect the network ports used by the Frontend and Central Manager to the same ports of the host machine. That way, any messages to the host machine intended for a Frontend or Central Manager would be forwarded to the appropriate container, enabling the two containers to communicate with each other and any components outside the host machine.

To perform the container configuration, I used Kubernetes features that allow for running arbitrary shell scripts (sequences of instructions) within the containers at startup. For each component, I created a setup script that generates a random password for use by HTCondor, then uses that password to create idtokens, a method of authenticating with other components. Finally, the setup script places the generated idtokens in a directory which is configured in the manifest files to be shared between containers, so that each component can trust the other.

3.4. Helm Chart

Certain values in the Kubernetes manifest files are guaranteed to change from user to user, most notably the IP address and hostname of the host machine. However, Kubernetes manifests are entirely static, with to specify configuration variables. To provide an easy way to configure these values, I used Helm, a software system in which programmers write templates for Kubernetes manifests and a variables file that specifies values used to fill in the fields of the templates. I replaced every appearance of the relevant values in the Kubernetes manifests with a Helm template variable, then wrote a Helm variables file with those values to provide a single place where users can configure the most common options.

4. RESULTS

My work produced a Helm chart for installing a Kubernetes cluster with a GWMS Frontend and HTCondor Central Manager. The components are as capable as traditional installations at communication with outside hosts and submitting jobs.

The time required to install the cluster is dependent on the user's familiarity with Kubernetes. In the best case, Kubernetes is already installed on the host machine, and the installation process is very easy, consisting of a single download, a few straightforward configuration options, and a single command to install. In the worst case, the user must also install Kubernetes, a process which is nontrivial but which allows the user to quickly deploy the cluster any number of times once completed. In either case, the amount of manual credentials configuration required is significantly reduced, from dozens of steps to only two, with the rest automated.

5. CONCLUSION

I produced a Docker image, Kubernetes cluster, and Helm chart to automate most of the installation and setup of the GWMS Frontend and HTCondor Central Manager. These tools significantly reduce the complexity of the deployment process, allowing end users in the High Energy Physics community to use an important tool for data processing while investing less time and technical expertise getting the tool to work.

6. FUTURE WORK

Further work is needed to containerize the other components of GWMS, in particular the GWMS Factory and HTCondor worker node. The HTCondor submit host requires more manual interaction by end users than the other components, so it is not as good of a candidate for containerization. Research is needed to determine the best way to deploy a submit host that can communicate with containerized components and makes sense for the end users.

In addition, more communication with end users is needed to ensure the supplied solution fits their infrastructure and configuration needs. Modifications may be necessary to allow deployment across multiple physical devices, alternative authentication methods, and easier access to commonly used configuration options.

REFERENCES

Davila, D. (2022, April 20). GWMS Containers. Retrieved February 25, 2023, from GitHub website: https://github.com/glideinWMS/containers/tree/51e422dd34f8cfb90e9e18f24bb6a20d9d7a6b63/frontend

Fermilab. (n.d.). GlideinWMS. Retrieved February 25, 2023, from https://glideinwms.fnal.gov/doc.prd/index.html

Helm. (n.d.). Docs Home. Retrieved February 25, 2023, from https://helm.sh/docs/

HTCondor. (n.d.). HTCondor Manual. Retrieved February 25, 2023, from https://htcondor.readthedocs.io/en/latest/

Kubernetes. (n.d.). Kubernetes Documentation. Retrieved February 25, 2023, from Kubernetes website: https://kubernetes.io/docs/home/

Open Science Grid. (n.d.). GlideinWMS VO Frontend Installation. Retrieved February 25, 2023, from https://osg-htc.org/docs/other/install-gwms-frontend/