Automated Container Image Generation for Seamless Integration with GlideinWMS

CERN Summer Student 2024

Fatemah Alrasheed

Department of Computer Engineering, Kuwait University

A project report submitted in partial fulfillment of the requirements for the Summer Student Program

Project Advisor

Marco Mascheroni

CERN

Geneva, Switzerland

August, 2024

Table of Contents

Abstract

The CMS experiment at CERN is the basis of high-energy physics research, and it relies on an extensive and distributed computational infrastructure to process large amounts of data for simulations and analyses. The goal of this project is to improve the GlideinWMS system's computing resource integration for the CMS experiment at CERN. By creating a Docker-based solution, we enable automated and efficient job management within a distributed computing environment. The Docker setup and associated scripts facilitate the deployment, configuration, and execution of GlideinWMS jobs, significantly simplifying the process of joining and utilizing the CMS computing pool. It is essential to anticipate and evaluate this infrastructure's scalability constraints to meet the constantly increasing demand for computer resources. This approach aims to enhance resource scalability and simplify the process of joining the CMS computing infrastructure.

Keywords: CMS, GlideinWMS, Docker, HTCondor, Containerization, Computing Resources, Kubernetes

Acknowledgment

I would like to extend my gratitude to my advisors and the submission infrastructure team for their guidance and support throughout this project. Special thanks to Marco Mascheroni and Florian Von Cube for their expert advice and to the CERN team for providing the resources and infrastructure essential for the project's success. Their insights and assistance were critical in developing a solution that improves the efficiency of resource management within the CMS experiment.

Introduction

The Compact Muon Solenoid (CMS) experiment at CERN is one of the most significant high-energy physics experiments in the world. It requires large amount of computational power. Hundreds of thousands of CPUs are needed for the experiment, and at its peak, over a million cores spread throughout the globe are used. [1] Using the global computing grid, we examine the volume of data. It is spread over about 170 data centers globally, with each location pledging a specific amount of resources to CMS (see Figure 1).

Figure 1: Worldwide Computing Grid

To manage these distributed computing resources efficiently, CMS and other experiments like DUNE utilize the GlideinWMS workload management system. GlideinWMS allows these experiments to harness computing power from various sources, creating a flexible and scalable environment for their workloads. The CMS Global Pool is a centrally managed HTCondor [2] pool that dynamically scales in size by submitting GlideinWMS pilot jobs, known as glideins, to the computing elements of different resource providers supporting CMS. The goal is to integrate new computing resources into the existing infrastructure in an easier way, ensuring that the process is both efficient and scalable. [3] This involves components such as the Access Point, Central Manager, GlideinWMS Frontend, GlideinWMS Factory, Execute Point, and Pilot as illustrated in Figure 2. These components are essential for managing GlideinWMS operations within CMS.

The Docker-based approach will facilitate easier configuration, execution, and integration of new resources, ultimately enhancing the CMS computing infrastructure's flexibility and effectiveness. This solution takes care of the immediate integration issues. It also puts the CMS experiment in a better position to meet demands in the future. This ensures that the distributed computing environment remains stable. It also improves the system's ability to adapt to the changing requirements of high-energy physics research.

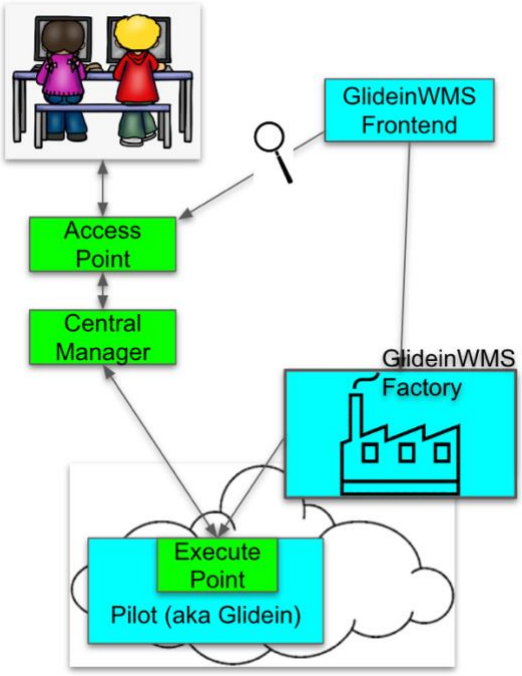

Figure 2: CMS Pool Components

Objectives

The main goal of this project is to enhance the integration of computing resources into the GlideinWMS system used by CMS. The project's goal is to simplify the deployment, configuration, and management of GlideinWMS processes in a distributed computing environment. This involves creating Docker images that has the necessary scripts and configurations to facilitate seamless job execution and resource integration. Another key goal is to simplify the process of joining and utilizing the CMS computing pool, thereby improving resource scalability and reducing operational complexity. The project seeks to

make the integration process more efficient and adaptable. This ensures that the CMS experiment can effectively leverage its distributed computing resources. The goal is to meet the demands of high-energy physics research.

To achieve this, a new container image will be specifically designed for seamless integration with the GlideinWMS system, which will be published on DockerHub. Site administrators will be able to easily deploy this container image on their worker nodes, significantly simplifying the process of expanding computational resources. Additionally, a comprehensive user manual and detailed operator instructions will be developed as integral components of the project, ensuring smooth adoption and operation across various sites.

Tools and Technologies

- **Bash Scripting and Linux Command Line**: Used for writing startup and job configuration scripts.
- **Docker**: For creating, managing, and deploying container images. [4]
- **Podman**: An alternative to Docker for building and running containers.
- **HTCondor**: A job scheduler for managing and executing jobs on distributed resources.
- **GlideinWMS**: Manages job submissions and resource integration for distributed computing environments.

Outcome

This project effectively integrates computing resources with the GlideinWMS system using Docker containers. First, there is the Docker image that contains all the necessary tools and scripts to support GlideinWMS.[4] Scripts were also developed for managing container startup and job execution, making the whole process smoother and more automated. The `myexe` Condor job submission script has been configured to define job requirements clearly. The container_startup.sh and glidein_startup.sh scripts ensure that jobs are set up

and run efficiently within the container. To assist users, a detailed guide has been provided, covering everything from building and running the Docker container to submitting jobs. This comprehensive setup simplifies the integration process and makes it easier to join and utilize the CMS computing resources effectively.

The new architecture as shown in Figure 3 shows how the system is simplified by eliminating the need for certain components (like the GlideinWMS Factory). This simplification is especially useful for integrating new resources, such as High-Performance Computing (HPC) systems, where installing a Compute Element can be challenging. The direct connection between the Central Manager and the Execute Points allows for easier and more efficient integration of these resources.

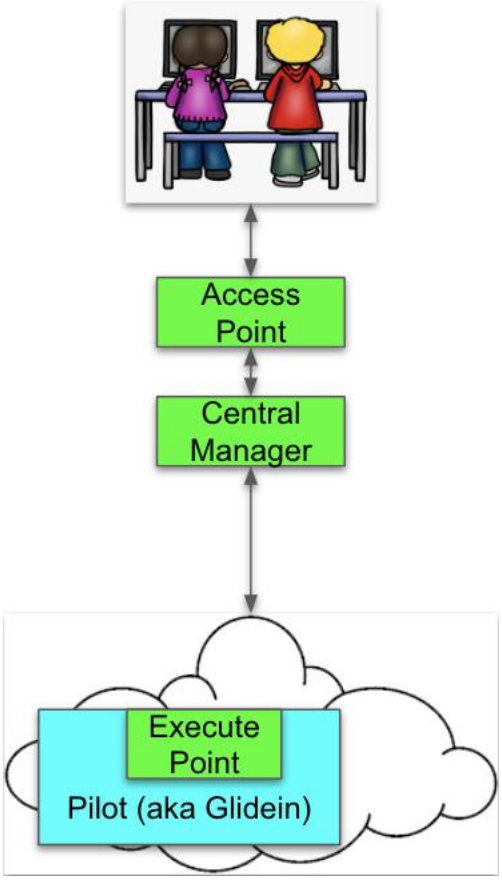

Figure 3: New Architecture

Conclusion

This project successfully addresses the challenges of integrating distributed computing resources into the GlideinWMS system by creating Docker containerization. The creation of custom Docker images, along with associated scripts and detailed documentation, significantly simplifies the process for site administrators to join the CMS computing pool, reducing operational complexity and enhancing resource scalability.

By automating key aspects of job management and resource integration, this project not only improves the current infrastructure but also positions the CMS experiment to meet the growing computational demands of high-energy physics research. The work done here prepares for more efficient and adaptable resource management, ensuring that the CMS infrastructure remains stable and responsive to future challenges. The success of this effort establishes guidelines for similar projects in other large-scale scientific studies and highlights the need of containerization in high-energy physics.

Reference

[1] CERN, "The CMS experiment," [Online]. Available: https://home.cern/science/experiments/cms. [Accessed: Aug. 21, 2024].

[2] "HTCondor," University of Wisconsin-Madison, [Online]. Available: https://research.cs.wisc.edu/htcondor/. [Accessed: Aug. 22, 2024].

[3] GlideinWMS, "GlideinWMS Documentation," [Online]. Available: https://glideinwms.fnal.gov/doc.prd/index.html. [Accessed: Aug. 21, 2024].

[4] Docker, "Docker Documentation," [Online]. Available: https://docs.docker.com/. [Accessed: Aug. 21, 2024].

[5] CMS O&C, "Data Management Monitoring," [Online]. Available: https://cms-dm-monitoring.cern.ch/. [Accessed: Aug. 21, 2024].

Appendix A: User Manual

Prerequisites

1. Podman/Docker Installed: Ensure that Podman or Docker is installed on your system.

2. HTCondor Knowledge: Familiarity with HTCondor is recommended for submitting jobs.

3. Token Access: Obtain a valid token for authentication, requested from the submission infrastructure team at CERN.

Step-by-Step Instructions

1. Pull the Docker Image

- First, pull the Docker image from the repository:

$ *podman pull* <*image_repository*>

2. Build the Docker Image

- If you need to build the image locally, use the following command:

$ *podman build -f Dockerfile --build-arg TOKEN=[] -t image_name .*]

```
[root@mmascher-vault my_docker_project]# podman build -f Dockerfile --build-arg TOKEN=eyJhbGciOiJIUzI1NiIsImtpZCI6Iml0Yl
9wb29sX2tleSJ9.eyJleHAiOjE3MjU3MTkxMjcsImlhdCI6MTcyMDUzNTEyNywiaXNzIjoiY21zZ3dtcy1pdGIuY2Vybi5jaCIsImp0aSI6IjdkODQ4ZWQ1O
GEzOTIzMDIwMjczZjg1YzU3Y2JhN2IwIiwic2NvcGUiOiJjb25kb3I6XC9BRFZFUlRJU0VfTUFTVEVSIGNvbmRvcjpcL0FEVkVSVElTRV9TVEFSVEQgY29uZ
G9yOlwvQURWRVJUSVNFX1NDSEVERCBjb25kb3I6XC9SRUFEIGNvbmRvcjpcL1dSSVRFIiwic3ViIjoibW1hc2NoZXIt -t project_image .
STEP 1/9: FROM opensciencegrid/software-base:23-el9-testing
STEP 2/9: RUN yum install -y wget perl-File-Copy python3-condor ca-certificates
--> Using cache ac80801e03d9640f5105cdb28115bcab41793c0ccb0a753d0e4031c127f66c7d
--> ac80801e03d9
STEP 3/9: WORKDIR /home/condor
--> Using cache ffe6cbfe503229cfb24fa6998b660172a9d9b3f8a57b0647cc717faea17e9fad
--> ffe6cbfe5032
STEP 4/9: COPY ./glidein_startup.sh /home/condor/glidein_startup.sh
--> Using cache 805febd80097146530e8593f8fdc4fc19ab20e89a225c57c1a39287a0b688059
--> 805febd80097
STEP 5/9: COPY ./container_startup.sh /home/condor/container_startup.sh
--> 7233e19eee3b
STEP 6/9: COPY ./manual_glidein_startup /home/condor/manual_glidein_startup.sh
--> 4a30089ab26a
STEP 7/9: RUN chown -R condor:condor /home/condor &&     chmod 755 /home/condor/container_startup.sh /home/condor/glidei
n_startup.sh /home/condor/manual_glidein_startup.sh
--> 493ae9079afd
STEP 8/9: USER condor
--> 4d690eee9d8a
STEP 9/9: CMD [ "/home/condor/container_startup.sh" ]
COMMIT project_image
--> 52098e3807c0
[Warning] one or more build args were not consumed: [TOKEN]
Successfully tagged localhost/project_image:latest
52098e3807c0c16084b166e55f6445a4016feafa6e5f61d43e2019424294b325
```

Figure 4: podman build

- Replace `TOKEN=[]` with your actual token, and `image_name` with your preferred name for the image.

3. Run the Docker Container

- Start the container with the following command:

$ *podman run -e TOKEN=[] --rm --name=container_name -v /cvmfs:/cvmfs:shared --security-opt=unmask=ALL -it image_name*

- Replace `TOKEN=[]` with your actual token, `container_name` with a name for the container, and `image_name` with the name of your Docker image.

4. Submit an HTCondor Job

- Open a new shell on your system, and submit a job using the following command:

$ *_condor_SEC_TOKEN_DIRECTORY=/tmp/token/ condor_submit -pool vocms0808.cern.ch -name vocms0809.cern.ch myexe*

Ensure that the `myexe` file is properly configured with your job parameters.

5. Monitor Condor Status

- To watch the status of the Condor system, use the following command:

$ *watch _condor_SEC_TOKEN_DIRECTORY=/tmp/token/condor_status -pool vocms0808.cern.ch*

```
Every 2.0s: _condor_SEC_TOKEN_DIRECTORY=/tmp/token/ condor_status -pool vocms0808.cern.ch                    mmascher-vault.cern.ch: Fri Aug 16 14:24:16 2024

Name                                  OpSys      Arch   State     Activity LoadAv Mem   ActvtyTime

slot1@glidein_5_47261668@1cf37df31691 LINUX      X86_64 Unclaimed Idle      0.000 6986  0+00:00:00

               Total Owner Claimed Unclaimed Matched Preempting  Drain Backfill BkIdle

  X86_64/LINUX     1     0       0         1       0          0      0        0      0

         Total     1     0       0         1       0          0      0        0      0
```

This command continuously monitors the Condor status, allowing you to keep track of the job submissions and the overall system status.

*You can override default arguments, such as CPUs, Memory, and Sitename, using `--override-args` when submitting jobs.