Autoscaling for HPC Runners

Tri Nguyen[1,2], Haiying Xu[1], Brian Vanderwende[1]

National Center Atmospheric Research[1], Indiana University Bloomington[2]

I. Abstract

This project extends self-hosted CI/CD server using autoscaling runners to promote reliable and rapid testing on HPC platforms. Although traditional self-hosted runners were created for earlier initiatives, they are difficult to put up widely for organizations. Our goal is to develop a centralized CI/CD server, featuring autoscaling, PBS scheduler integration and robust authentication security.

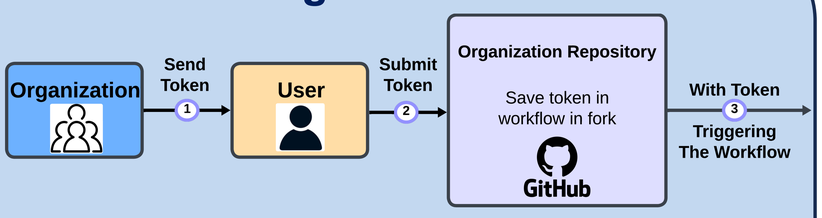
II. What is CI/CD?

Continuous Integration (CI): practice of frequently integrating code changes into shared repository
Continuous Delivery (CD): automating release of validated code and ready for deployment

Why CI/CD?
- Minimize manual effort by automating repetitive task
- Detect bugs early making them easier to fix

Kubernetes: a orchestration tool for scaling, managing and deploying the containers
Helm: a package manager of Kubernetes to install, upgrade applications inside Kubernetes

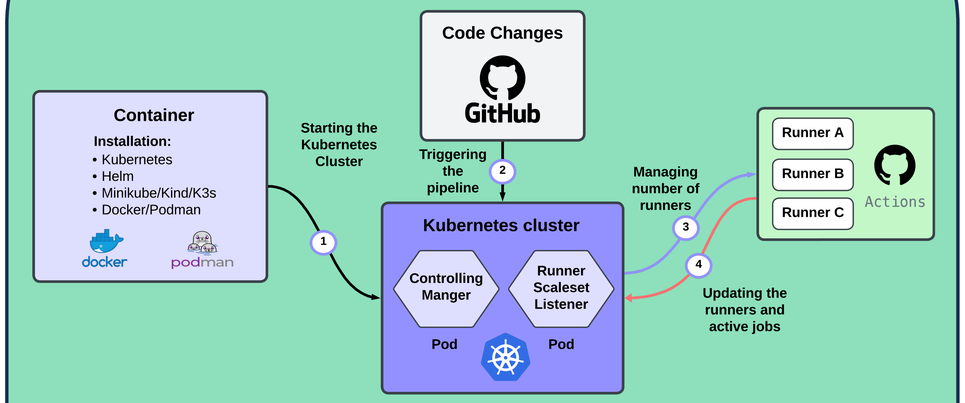
III. Using tokens for forks



1. NCAR organization provide token/secret to authorized users
2. Users add token/secret to workflow inside fork repo
3. Users who make pull requests will trigger the runner pipeline to build and test their code

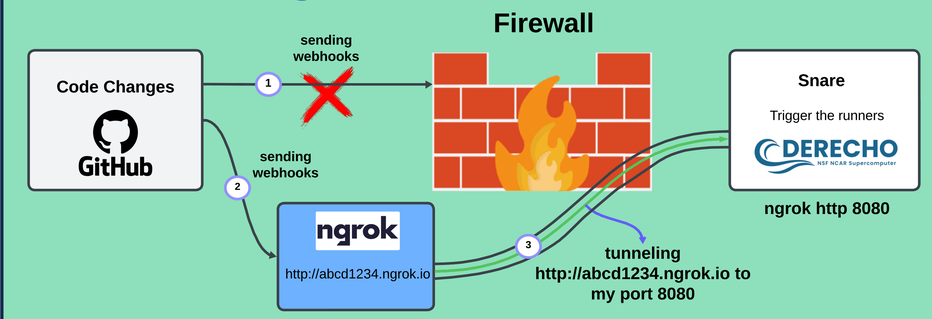
IV. Methods for scaling HPC runners

1. Container in Container



Container: A container is a lightweight, portable and self-sufficient software package that includes code, runtime and systems to run applications in different computing environments.
Container in Container: container engines run inside the container
Why?
- Isolating build configuration from host environment
- Standardizing complex installation of Kubernetes and Helm

2. Autoscaling with Webhooks



Webhooks: when some events happens in repository (Pull Request), Github will send an HTTP request to specified URL
Snare: a simple program that listens to Github webhook events and runs Unix command based on them
Ngrok: a reverse proxy allowing webhooks to enter in your localhost through secure tunneling during development

V. Challenges

- Cgroup v2 required for rootless Kubernetes.
- Installing Kubernetes, Helm, and managing a cluster within a rootless presents challenges
- Github API response time is highly variable between 10 seconds to 10 minutes

VI. Future work

- Develop robust authentication mechanisms for user mapping in PBS schedulers
- Explore Github Teams and Enterprise for infrastructure
- Further investigate starting Kubernetes cluster in rootless environment

Acknowledgement

I would like to extend my gratitude to my mentors, Haiying Xu and Brian Vanderwende for their invaluable guidance. I also appreciate the technical support from Nick Cote. Thank you to SIParCS 2024 for a great summer of research.

Connect with me !