Ultrascale Systems Research Center

Using CI/CD to Automate Cluster Setup and Initialization

Kevin Pelzel

Mentor: Paul Peltz, HPC-DES

Los Alamos NATIONAL LABORATORY EST. 1943

Project Overview

CI/CD stands for Continuous Integration / Continuous Delivery. It’s a tool used to run automated tasks on a developers code whenever a change is made to a remote repository. Continuous Integration usually refers to testing the quality of code with unit testing or other forms of tests, while Continuous Delivery usually involves sending that code onward to be processed in some way such as building, packaging, and deployment. The goal of this project was to find a CI/CD tool that best fits our needs and to use that tool to setup and configure a bare metal cluster with as little human intervention as possible. After researching CI/CD tools, it was determined that Gitlab CI would fit the needs of this project suitably. Below is a general comparison between Gitlab CI and a very popular CI/CD tool, Jenkins.

VS

Gitlab CI

Pros:

- Uses “Runners” to execute various tasks
- Written in GO
- Runners can be easily created and destroyed
- Runners can use different executors: SSH, VirtualBox, Docker, Shell, etc.
- Jobs are configured in a YAML file at the root of the project repository
- Open source

Cons:

- General functionality not as extensive as Jenkins

Jenkins

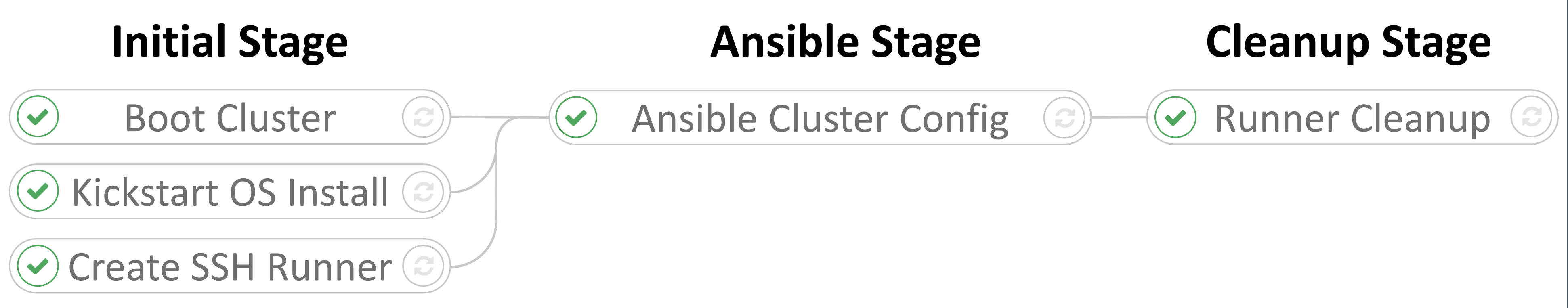
Pros:

- Has extensive plugin library
- Current industry standard in continuous integration
- Open source

Cons:

- Current Gitlab plugin doesn’t offer as much functionality as Gitlab CI
- Teardown and setup not as simple as Gitlab runners
- Managed and controlled by a web UI separate from the repository

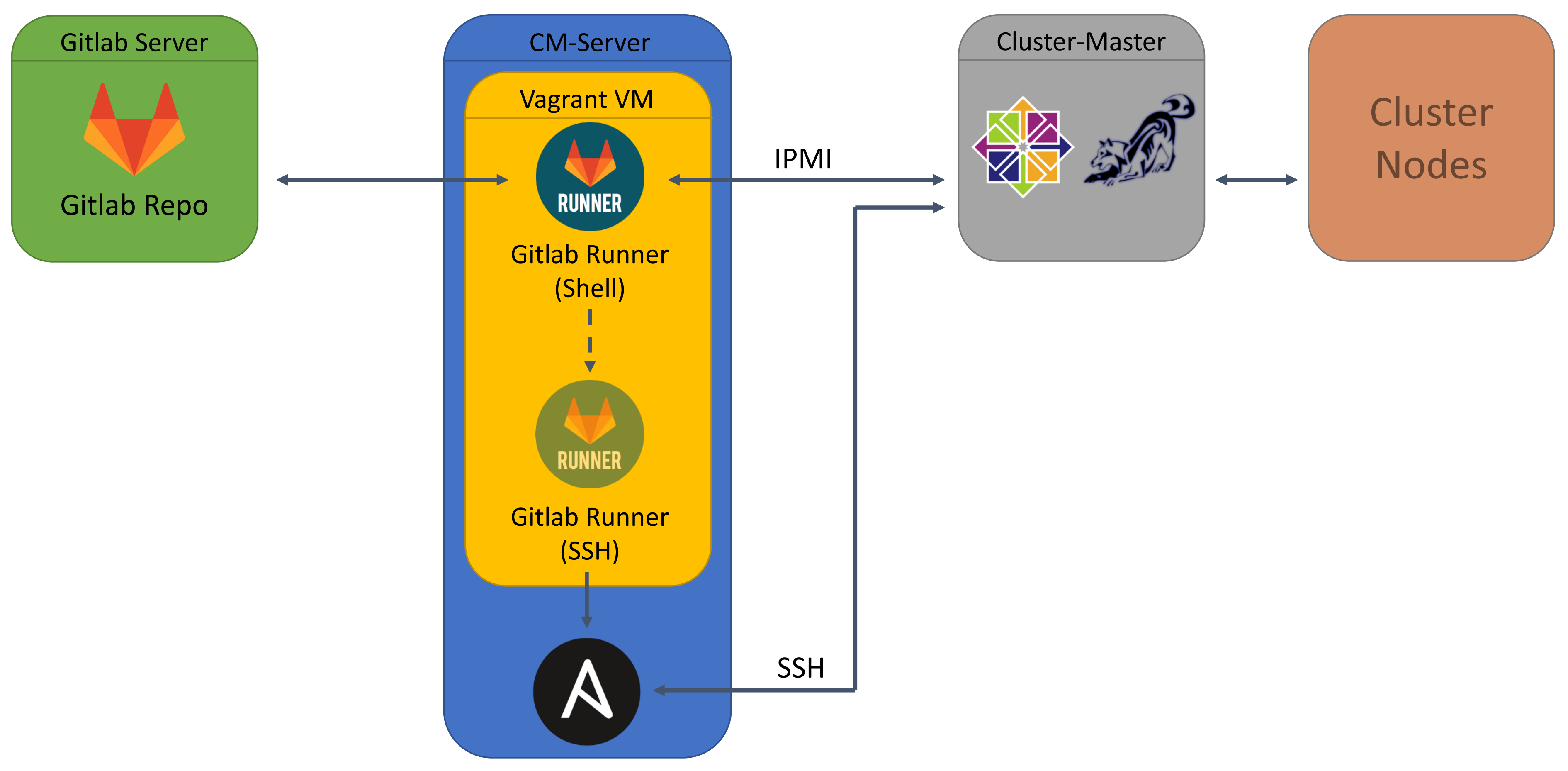
Cluster Initialization Pipeline



- The first stage handles the initial setup of the cluster
  - The boot job uses IPMI to trigger PXE boot or regular boot
  - The OS installs using a PXE server and kickstart file
  - A temporary SSH runner is created to run the ansible playbook
- The ansible stage runs the ansible playbook against the specified cluster
- The cleanup stage removes the temporary SSH runner
- Jobs can have many configuration options, such as:
  - Execute manually
  - Execute on failure
  - React to certain conditions being met
- Each Job has a live log output that can be viewed from the web UI

Runner Configuration

The diagram below shows the entire setup of how Gitlab runners fit into the workflow from repository to bare metal. It starts with code getting committed to the Gitlab repository. As long as there is a Gitlab CI YAML file in the repository, Gitlab will trigger a runner to execute the defined stages.


Gitlab Server
Gitlab Repo
CM-Server
Vagrant VM
RUNNER
Gitlab Runner (Shell)
RUNNER
Gitlab Runner (SSH)
IPMI
SSH
Cluster-Master
Cluster Nodes