EPJ Web of Conferences **214**, 07032 (2019) https://doi.org/10.1051/epjconf/201921407032
CHEP 2018

CVMFS: Stratum 0 in Kubernetes

David Schultz[1,*], *Heath* Skarlupka[1], *Vladimir* Brik[1], and *Gonzalo* Merino[1]

[1]Wisconsin IceCube Particle Astrophysics Center, University of Wisconsin-Madison, 222 W Washington Ave, Suite 500, Madison, WI 53703, USA

Abstract. IceCube is a cubic kilometer neutrino detector located at the south pole. CVMFS is a key component to IceCube's Distributed High Throughput Computing analytics workflow for sharing 500GB of software across datacenters worldwide. Building the IceCube software suite across multiple platforms and deploying it into CVMFS has until recently been a manual, time consuming task that doesn't fit well within an agile continuous delivery framework. Within the last 2 years a plethora of tooling around microservices has created an opportunity to upgrade the IceCube software build and deploy pipeline. We present a framework using Kubernetes to deploy Buildbot. The Buildbot pipeline is a set of pods (docker containers) in the Kubernetes cluster that builds the IceCube software across multiple platforms, tests the new software for critical errors, syncs the software to a containerized CVMFS server, and finally executes a publish. The time from code commit to CVMFS publish has been greatly reduced and has enabled the capability of publishing nightly builds to CVMFS.

1 Introduction

The IceCube detector [1] is located at the geographic South Pole and was completed at the end of 2010. It consists of 5160 optical sensors buried between 1450 and 2450 meters below the surface of the South Pole ice sheet and is designed to detect interactions of neutrinos of astrophysical origin. The IceCube Collaboration's worldwide computing grid consists of many heterogeneous clusters, backed by CernVM-FS [2] (hereafter referred to as CVMFS) for software distribution. The CVMFS architecture is divided into a central Stratum 0 server (the source of the repository), multiple Stratum 1 servers (repository replicas with high network bandwidth), HTTP proxy servers local to clusters, and CVMFS clients on each node. IceCube manages its own Stratum 0 server and uses Stratum 1 servers and HTTP proxies from the Open Science Grid [3] and other partners for distribution.

The IceCube CVMFS repository, located at /cvmfs/icecube.opensciencegrid.org/, is a common software base for simulation, data processing, and user analysis code. Typically, one major version is released per year. This gets built on ~4-6 different Linux platforms to support the various clusters in the collaboration. Each release includes basic software such as Python, Boost, and ROOT, as well as more custom open source physics software. Standard releases of internal IceCube software are also added to the latest base release at the time, to support simulation production and data processing. The final piece added to CVMFS is a set of common files – test data, lookup tables, reference files, and other things used by many people.

* Corresponding author : david.schultz@icecube.wisc.edu

As for sizes, the common files account for 32GB of data, slowly growing over time. Each software base version is roughly 15GB per version per platform. We are now on our 4th version. This gives a total raw size of about 500GB, which gets compressed and deduplicated by CVMFS to 150GB.

2 Kubernetes stratum 0 design

As part of a modernization project, the CVMFS stratum 0 system was chosen for an upgrade. The previous setup used virtual machines and a NFS filesystem as shared disk. This shared disk was fairly slow, especially in iops, which was felt every time a software build was run. VM provisioning was also fairly difficult, as it needed to be kickstarted and puppetized across several OS types and versions.

2.1 Motivations for Kubernetes

One strong goal of the new design is to use containers as a base. They can be easily built and tested on a laptop, then shipped to the production cluster. It is easy to upgrade or roll back the containers, as they are versioned. And since kickstart/puppet is not involved, new OS platforms can be rolled out in hours instead of days.

Kubernetes [4] was chosen as an orchestration platform for two main reasons. First, it seemed to be winning the container orchestration wars in industry. Using software with wide industry support is a strong win, since it usually means better testing and robustness of the solution. Also, we had just installed a Kubernetes cluster locally, so it was available for this project.

2.2 Build automation

The build process can also be automated, similar to continuous integration software practices. As part of the migration to Kubernetes, the existing build scripts were taken a step further, automating the entire process from source code to published software on CVMFS.

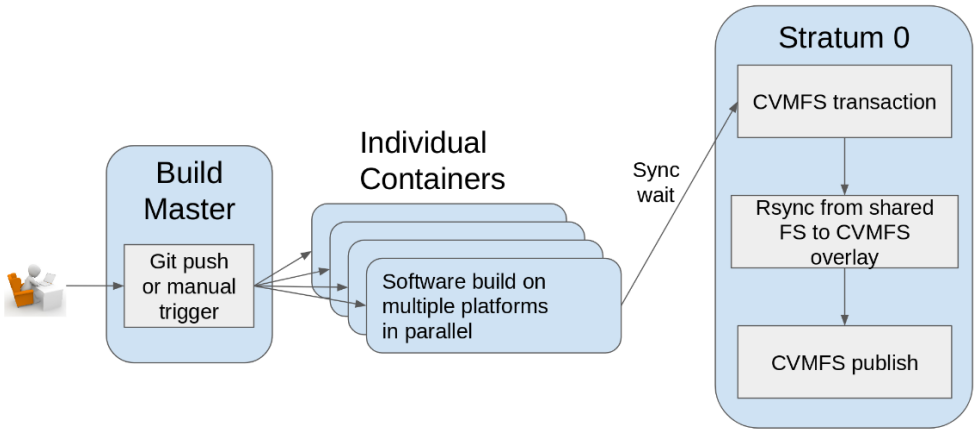

Fig. 1. The build workflow for adding software to IceCube CVMFS, proceeding from left to right. Individual containers are in blue.

Fig. 2. The waterfall view in Buildbot, showing a parallel software build with a synced publish on the stratum 0. An hourly user space rsync and publish is also visible, as well as periodic whitelist signing.

2.2.1 Workflow

The build workflow can be seen in Figure 1. It starts with a trigger action, either triggered by action on a git repository for the build scripts, or manually by a human. This initiates multiple software builds in parallel on each supported platform. The workflow waits for all builds to succeed, then initiates a CVMFS transaction on the stratum 0, rsyncs the updated software to the overlay filesystem, and publishes the update.

2.2.2 Buildbot

Buildbot [5] was chosen because of its flexibility, and because we had prior experience with it. All configuration lives in Python files, so any build step you can program in Python is possible. It natively handles source triggers, dependencies, locking, and other common situations. You can see an example of buildbot in Figure 2.

2.3 Container setup

Kubernetes was set up as in Figure 3, with underlying Docker [6], Kubernetes, and Ceph [7] storage. The Buildbot master runs in its own container, with all necessary configuration and shared secrets. Buildbot workers run for each platform as independent containers. The CVMFS stratum 0 runs in a separate container, with both the stratum 0 HTTP server and a

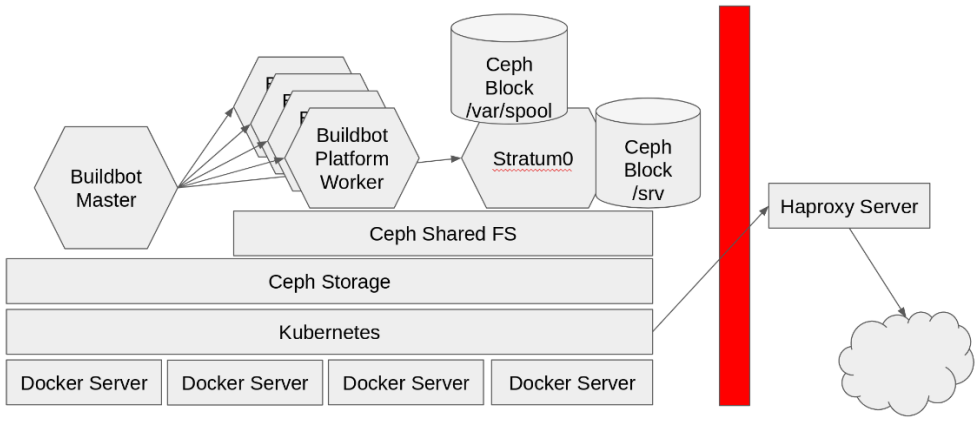

Fig. 3. A diagram of the Kubernetes cluster setup, with 4 physical servers running Kubernetes and Ceph. The red bar is a firewall, with an HAProxy load balancer to redirect external traffic to the correct node in the cluster to reach the right container.

Buildbot worker for automation. A cephfs shared filesystem is mounted in each builder and the stratum 0. The stratum 0 also has two ceph block volumes for local storage.

HAProxy [8] is used as an external load balancer and proxy, to route stratum 0 HTTP traffic to the stratum 0 container, and for the Buildbot master dashboard. Externally, the stratum 0 looks no different than a single server installation. We merely had to put a DNS CNAME to the proxy server and traffic was directed from the old server to the new system.

3 Usage experiences

The new CVMFS stratum 0 system has been running for about 9 months as of this writing. The most successful feature has been the increased build automation. Nightly builds of the latest internal software happen automatically. New releases are a simple matter of adding the release name to the build script repository and pushing the manual build trigger. No need to log in to every machine and shepherd the process.

The one main flaw in the system is that if Kubernetes is down for any reason, the stratum 0 is down. This has happened several times as we learn more about Kubernetes and its limits. The good news is that the stratum 0 and build containers come back to life automatically when Kubernetes is restored. While this is not a large issue locally, OSG's Stratum 1 monitoring has registered the repeated outages as unusual and commented to us about it.

For some of the strange problems that occasionally crop up, it is very nice to be able to delete a container and have it respawn, like a reboot on a normal machine. This has so far fixed every issue found, and is a simple and fast procedure for other system administrators to follow.

4 Conclusions

The modernization project to put IceCube's CVMFS stratum 0 into Kubernetes was a success. The infrastructure is easier to manage, and the automation improvements alone are worth the invested effort. Future work will focus on integrating more testing before publishing changes, and other automated procedures now that a build system is in place.

We acknowledge the support from the following agencies: U.S. National Science Foundation-Office of Polar Programs, U.S. National Science Foundation-Physics Division.

References

1. M. G. Aartsen et al, JINST **12**, P03012 (2017)
2. J. Blomer, P. Buncic, T. Fuhrmann, Proc. of the 1st int. workshop on Network-aware data management (NDM'11) 49–56 (2011)
3. R. Pordes et al, J. Phys.: Conf. Ser. **78**, 012057 (2007)
4. Kubernetes project, "Kubernetes" [software], version 1.12.0, 2018. Available from https://github.com/kubernetes/kubernetes/releases/tag/v1.12.0 [accessed 2018-10-03]
5. Buildbot project, "Buildbot" [software], version 1.4.0, 2018. Available from https://github.com/buildbot/buildbot/releases/tag/v1.4.0 [accessed 2018-10-03]
6. D. Merkel, Linux J. **239**, 2 (2014)
7. Ceph project, "ceph" [software], version 13.2.2, 2018. Available from https://github.com/ceph/ceph/releases/tag/v13.2.2 [accessed 2018-10-03]
8. HAProxy project, "HAProxy" [software], version 1.8.14, 2018. Available from http://www.haproxy.org/download/1.8/ [accessed 2018-10-03]