FERMILAB-CONF-21-053-SCD

FTS3: Data Movement Service in containers deployed in OKD

Lorena Lobato Pardavila[1], *Burt* Holzman[1] , *Edward* Karavakis[2], *Lincoln* Bryant[3] and *Steven* Timm[1]

[1] Fermi National Accelerator Laboratory, Batavia, IL, United States
[2] IT Department, European Organization for Nuclear Research (CERN), Geneva, Switzerland
[3] Enrico Fermi Institute, University of Chicago, Chicago, IL, United States

Abstract. The File Transfer Service (FTS3) is a data movement service developed at CERN which is used to distribute the majority of the Large Hadron Collider's data across the Worldwide LHC Computing Grid (WLCG) infrastructure. At Fermilab, we have deployed FTS3 instances for Intensity Frontier experiments (e.g. DUNE) to transfer data in America and Europe, using a container-based strategy. In this article we summarize our experience building docker images based on work from the SLATE project (slateci.io) and deployed in OKD, the community distribution of Red Hat OpenShift. Additionally, we discuss our method of certificate management and maintenance utilizing Kubernetes CronJobs. Finally, we also report on the two different configurations currently running at Fermilab, comparing and contrasting a Docker-based OKD deployment against a traditional RPM-based deployment.

1 Introduction

1.1 FTS3

The File Transfer Service (FTS3) [1] distributes the majority of the Large Hadron Collider (LHC) [2] data across the Worldwide LHC Computing Grid (WLCG) [3] infrastructure. It is integrated with experiment frameworks such as Rucio [4], PhEDEx [5] and DIRAC [6] and it is used by more than 35 experiments at CERN and in other data-intensive sciences outside of the LHC and even outside the High Energy Physics (HEP) domain. In 2020 alone, the centrally monitored FTS3 instances transferred more than 1 billion files and a total of 1 exabyte of data with global transfer rates regularly exceeding 60 GB/s.

FTS3 is a low-level data management service, responsible for scheduling reliable bulk transfer of files from one site to another while allowing participating sites to control the network resource usage. FTS3 provides simplicity by allowing easy user interaction for submitting transfers. FTS3 also has the WebFTS [7] portal, which is a web-based file transfer and management solution allowing users to invoke reliable, managed data transfers on distributed infrastructures from within their browser. FTS3 offers a rich monitoring system for real-time and accounting purposes [8], and a Web Admin interface to be able to modify the internal settings of the service such as to configure access rights and limits on storages and links.

It also provides reliability by ensuring data integrity with checksum comparison and the retrial of failed transfers. It is also flexible and scalable with multiprotocol support (WebDAV/HTTPS [9], GridFTP [10], XRootD [11] and SRM [12]), diversity in the way that clients can access the service (REST APIs, python bindings, CLI), transfers from and to different storage systems (EOS [13], DPM [14], Object Storages such as S3/SWIFT, STORM [15], dCache [16], CASTOR [17] and CTA [18]) and support for tapes with the bringonline component. Finally, one of the biggest advantages of FTS3 is its ability to be run without link and channel configuration with parallel transfer scheduling and optimisation to get the most from the network without saturating the storage systems, with support for intelligent priorities, activity shares and VO shares for classification of transfers.

This manuscript has been authored by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics.

1.2 SLATE

Building, operating, and maintaining a distributed computing facility in support of scientific collaborations is difficult for a number of reasons. One aspect of this difficulty is that the downstream operators of an application are often far removed from the software's developers. For example, each WLCG site operates and maintains a collection of gatekeepers, proxy caches, and storage elements, generally with independent system administrators, despite the services themselves being largely uniform across collaborations. To simplify operations for sites, the Services Layer At The Edge (SLATE) [19] project was designed to insert a ubiquitous and programmable layer of abstraction such that developers of software, especially collaboration-specific software, could take a more direct role in its deployment and simplify operations across the board. To that end, SLATE has taken a page from industry to provide a common, Kubernetes-based platform to deploy and operate services at sites. It adds a federation layer, essentially a privilege model to provide secure access to trusted members of central production teams which have expertise in operating globally distributed systems.

Having considered risks in containers and container registries (as identified by NIST SP 800-190 [20] among others), the SLATE team chose to implement a curated catalog of applications not unlike an "app store" commonly found on phones, or tablets for instance. The catalog contains application definitions (Dockerfiles) and templated Kubernetes configurations (Helm Charts) for which artifacts are built and scrutinized by the SLATE team and hosted publicly. Indeed, the SLATE approach to application packaging and deployment has been carefully crafted for composability. Institutions with security postures that restrict deployment and operation of collaboration services by centralized expert teams (often, e.g., national laboratories) may still reap benefits of the SLATE platform by using its open repository of Helm packages and Docker containers independent of a SLATE federation.

The containerized version of the FTS3 software described in this paper is one such successful use of this feature.

2 Architecture

The Fermilab FTS3 configuration presented here involves two public instances. Both have been containerized and deployed in OKD, the community distribution of Kubernetes that powers Red Hat OpenShift [21].

2.1 Overview

The implementation relies on different parts, which will be explained in the following sections:

- Docker images: there are three Docker images, of which two are customized for the Fermilab installation:
  - A standard MariaDB image for the backend database (provided by OKD and used without modification)
  - An image from the frontend FTS3 server, based on an image provided by the SLATE project and adapted to manage grid certificates in a container volume
  - A fetch-crl image, for the purpose of fetching certificate authorities and updating certificate revocation lists
- OpenShift OKD: the underlying infrastructure in which the project is deployed.
- Harbor image registry: docker images are stored in a centralized private Fermilab image registry (imageregistry.fnal.gov).

2.2 Docker images

As mentioned in the previous section, we have created two custom Docker images: one for the *FTS3 server*, and the other for *fetch-crl* which updates the CRLs.

Both images were built and stored in the image registry described in section 2.4.

2.2.1 FTS3 server

The FTS3 server container runs the FTS3 server executable, associated daemons, and httpd. The image is based on one provided by the SLATE project and has been adapted to run services as a non-privileged user (required in the OKD environment).

The dockerfile used to build this image handles database schema evolution when the environment variable DATABASE_UPGRADE is set to "yes". When this variable is set, the fts-database-upgrade script is executed, which creates and/or updates the database schema based on the FTS3 server version.

2.2.2. fetch-crl

This image is based on the Scientific Linux distribution and installs EPEL and OSG repositories. They are needed to install - and keep up to date - the packages required to manage the CA certificates and get the certification revocation lists (CRLs).

On deployment, the image executes a custom script which initializes and manages the certificate directory, renewing the CA certificates when needed. This differs from installation of services on a bare metal host, where the certificates would be managed by a crontab entry run as root.

2.3 OpenShift OKD

Fermilab deploys OKD, the community distribution of Kubernetes that powers Red Hat OpenShift. It is a solution for Kubernetes container management that provides a secure, multi-tenant infrastructure. The OKD security model has non-root containers by default - suggested as good practice - and role-based access control (RBAC) that organizes users into projects that provide multi-tenancy out of the box.

OKD adds a number of additional features to upstream Kubernetes to help with deploying and managing services. For example, where vanilla Kubernetes has Deployments, OKD has DeploymentConfigs that among other things can respond to triggers that automatically update the deployment with a variety of strategies for rolling out new versions of a service. Another useful feature is that OKD has primitives for building Docker images (BuildConfigs) and for managing both locally built images (via BuildConfigs) and external images (ImageStreams). This is in contrast to upstream Kubernetes which has no built-in feature for triggering container redeployment when an underlying image has changed.

2.4 Image Harbor

The two docker images mentioned above are built and pushed to/pulled from a Fermilab image registry (imageregistry.fnal.gov). The image registry is based on Harbor [https://goharbor.io/], which is an open-source and private Docker-like registry which secures the images across cloud platforms. It is backed by Gluster storage for persistent volumes, which will be explained later on.

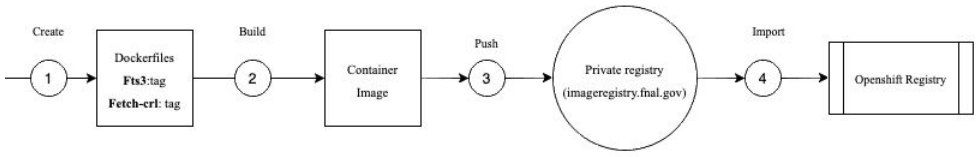

Fig. 1. Image registry workflow

As observed in ***[Fig. 1]***, both container images are first created and built with Docker. Afterwards, they are tagged with the corresponding FTS3 versions (e.g. 3.8, 3.9, 3.10). Finally, they are pushed to the local Harbor image registry, where an SHA code is assigned to them. In order to connect the registry from Harbor with OKD, the images are imported in the OKD

registry by providing the name and tag of the image with the corresponding SHA code. In this way, every time that the container is started, it will pull the image from the registry if it is not already locally cached.

3 Certificate Management and Maintenance - Kubernetes CronJobs

In this section, the method of certificate management and maintenance using Kubernetes CronJobs is discussed. The docker images provided by the FTS3 developers assume Certificate Authorities (CAs), certificate revocation lists (CRLs), and certificates (certs) to be in a common area. Additionally, it employs an automated process to acquire a host certificate that only works for machines inside CERN's network.

In contrast, we configure the OSG repository which maintains the integrity of sites and services, including the dependencies mentioned above. For example, the certificate authorities (CAs) provide the trust roots for the OSG public key infrastructure, and also the repository helps to have CRLs up to date on the hosts.

There are two ways to get CA certificates from OSG: either by an RPM installation or through scripts. With the RPM solution, we would need to ensure docker images were continually kept up to date: as soon as OSG releases an update, we would need to update the docker images too. In the case of scripts, we would run a cron job periodically, checking for an update and downloading the certificates automatically without installing or updating the RPM. We chose to deploy scripts, leveraging the native support for Kubernetes CronJobs.

4 FTS3 public instances running at Fermilab

As described in the introduction, the FTS3 service consists of two primary components: a web services based frontend, and a database backend. We run each in a separate Kubernetes pod. The backend deploys a standard vanilla MariaDB docker image provided by OKD, while the frontend deploys the FTS3 images described above in section 2.2. The database access credentials are stored as Kubernetes secrets and published as environment variables to both the MariaDB and frontend pod.

The frontend pod, in addition to deploying an FTS3 container, also deploys two containers ("initContainers") that only start on pod initialization, and are guaranteed to run to completion before the FTS3 container begins. The fetch-crls-first-time initContainer downloads the certificate authorities and initial certificate revocation lists (as described in section 3) - these are needed by the FTS3 server. The wait-for-ftsdb initContainer is a simple loop which sleeps until the database pod is running, since the FTS3 server requires database access on boot.

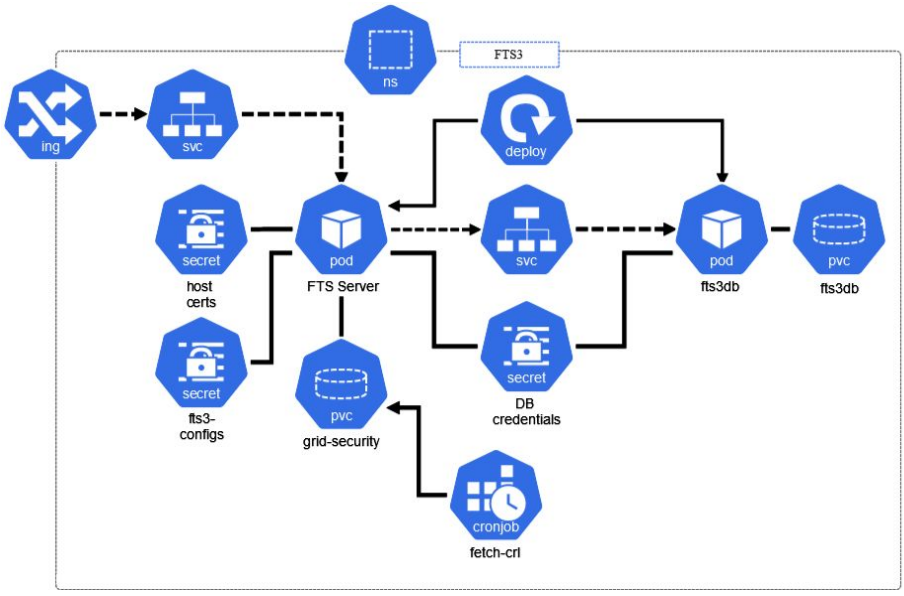

Fig. 2. Containerized configuration deployed in OKD

We provide two volumes of persistent storage via 1 GiB persistent volume claims (PVC) on the GlusterFS file system provided by OKD. One volume stores the data from the database; the other contains the certificate authorities and certificate revocation lists. These volumes were performant enough for our needs, but a busier database might require node-local storage or a caching layer such as Redis or memcached. In order to provide a performance baseline for the database, we have configured a second option, where we run only the frontend pod and connect it to a dedicated MariaDB 10.3 server running on a physical instance. Comparative measurements are in progress.

5 Future Plans

The future work involves the testing of JWT tokens [22] supported by the latest FTS3 version (3.10 at the moment of writing this article), in addition to the integration of FTS3 with other services such as Rucio.

The FTS3 authorization and authentication is currently done only with X.509 proxy certificates. Users may find the process of obtaining an X.509 certificate, creating a proxy, and placing it properly in the environment to be time-consuming. With the latest FTS3 version, JWT tokens can be used to authenticate to FTS3 and to authorize transfers to the various storage elements authorizing transfers on the storages. These tokens can be provided by two different Entity Providers: tokens for Fermilab-based VOs are provided by a CILogon [23] issuer, while tokens for the CMS experiment will be issued by WLCG IAM [24].

5.1 Rucio

The DUNE experiment uses the Rucio file replication system [4] to track its file replicas and schedule and manage file transfers between sites. FTS3 is used by Rucio as its transport layer. DUNE originally requested an FTS3 server be installed at Fermilab to have better flexibility to access US-based high performance computing centers. We now use the Fermilab-based FTS3 server for all intra-US file transfers including to Brookhaven and NERSC. Some preliminary load tests have been conducted; we expect to transfer petabytes of data in the next few months. In the longer term, transfers will expand to other high performance computing facilities.

5.2 DUNE tests

In addition to the Rucio use, DUNE will also use FTS3 as the transport from the experimental halls of ProtoDUNE and eventually the DUNE Far Detector in South Dakota. The NP02 ProtoDUNE experiment at CERN already uses this method and the NP04 ProtoDUNE experiment will shortly shift to use this method.

6 Conclusions

The choice of a containerized configuration for FTS3 came from the desire to have a public FTS3 instance available at Fermilab for experiments to use as a testbed. As the initial usage by DUNE and other experiments is expected to be small – but increasing over time – a containerized deployment allows more flexible use of server resources by enabling horizontal scaling according to demand. The containerized deployment at Fermilab runs on an OKD instance that supports multiple tenants enabling sharing of the cluster among multiple services.

Another benefit of running services on Kubernetes or OKD is resilience to hardware or application faults, by restarting the server automatically in case of failure, or transparently migrating pods to other physical servers

DUNE has successfully submitted many test transfers via Rucio, and plans to move the development containerized deployment of FTS3 on OKD to production shortly. We expect to open the service to additional experiments in the future.

7 References

[1] E Karavakis et al, "FTS improvements for LHC Run-3 and beyond", 2020 EPJ Web of Conferences 245, 04009 doi:10.1051/epjconf/202024504016

[2] L. Evans and P. Bryant, *LHC Machine*, JINST **3** S08001 (2008)

[3] I Bird, Computing for the Large Hadron Collider, Annual Review of Nuclear and Particle Science 61 :99-118 (2011)

[4] M. Barisits, T. Beermann, F. Berghaus et al, *Rucio: Scientific Data* Management, Comput Softw Big Sci **3**: 11 (2019)

[5] M. Giffels, Y. Guo, V. Kuznetsov et al, *The CMS Data Management System*, J. Phys.:Conf. Ser. **513** 042052 (2014)

[6] S K Paterson and A Tsaregorodtsev, *DIRAC optimized workload management*, J. Phys.: Conf. Ser. **119** 062040 (2008)

[7] A. Kiryanov, A. A. Ayllon and O. Keeble, *FTS3 / WebFTS – A Powerful File Transfer Service for Scientific Communities*, Procedia Computer Science **66** 670-678 (2015)

[8] E Karavakis et al, "Unified Monitoring Architecture for IT and Grid Services", 2017 J. Phys.: Conf. Ser. 898 092033 doi:10.1088/1742-6596/898/9/092033

[9] G. Bernabeu et al, *Experiences with http/WebDAV protocols for data access in high throughput computing*, J. Phys.: Conf. Ser. **331** 072003 (2011)

[10] W. Allcock et al, *The Globus Striped GridFTP Framework and Server*, SC '05: Proceedings of the 2005 ACM/IEEE Conference on Supercomputing, Seattle, WA, USA, 2005, pp. 54-54, doi: 10.1109/SC.2005.72 (2005)

[11] A. Dorigo et al, *XROOTD - a highly scalable architecture for data access*, WSEAS Transactions on Computers, 4(4):348--353 (2005)

[12] L. Abadie et al, *Storage Resource Managers: Recent International Experience on Requirements and Multiple Co-Operating Implementations*, 24th IEEE Conference on Mass Storage Systems and Technologies (MSST 2007), San Diego, CA, 2007, pp. 47-59, doi: 10.1109/MSST.2007.4367963 (2007)

[13] A. J. Peters and L. Janyst, *Exabyte Scale Storage at CERN*, J. Phys.: Conf. Ser. **331** 052015 (2011)

[14] A. Alvarez et al, *DPM: Future Proof Storage*, J. Phys.: Conf. Ser. **396** 032015 (2012)

[15] A. Carbone et al, *Performance Studies of the StoRM Storage Resource Manager*, Third IEEE International Conference on e-Science and Grid Computing (e-Science 2007), Bangalore, pp. 423-430 (2007)

[16] P. Fuhrmann and V. Gülzow, *dCache, Storage System for the Future*, Euro-Par 2006 Parallel Processing. Euro-Par 2006. Lecture Notes in Computer Science, vol **4128** (2006)

[17] G Lo Presti et al, *CASTOR: A Distributed Storage Resource Facility for High Performance Data Processing at CERN*, 24th IEEE Conference on Mass Storage Systems and Technologies (MSST 2007), San Diego, CA, 2007, pp. 275-280 (2007)

[18] E. Cano et al, *CERN Tape Archive: production status, migration from CASTOR and new features*, EPJ Web Conf. Volume 245, 2020, https://doi.org/10.1051/epjconf/202024504013

[19] Gardner, R., Breen, J., Bryant, L., & McKee, S., *SLATE and the Mobility of Capability*, Science Gateways 2017

[20] Souppaya, M., Morello, J., & Scarfone, K. (2017). Application container security guide. National Institute of Standards and Technology. https://doi.org/10.6028/nist.sp.800-190

[21]OKD: http://www.okd.io

[22] JWT: https://jwt.io/

[23] J. Basney, Jim, T. Fleury, J. Gaynor, (2013). CILogon: a federated X.509 certification authority for CyberInfrastructure logon. Concurrency and Computation: Practice and Experience. https://doi.org/10.1145/2484762.2484791

[24] WLCG IAM: https://indigo-iam.github.io/wlcg-docs/