Deploying FTS in Kubernetes Environment

CERN Summer Student Programme 2021

Supervisor Mihai Patrascoiu *
Student Panagiota Angeliki Karpatou **

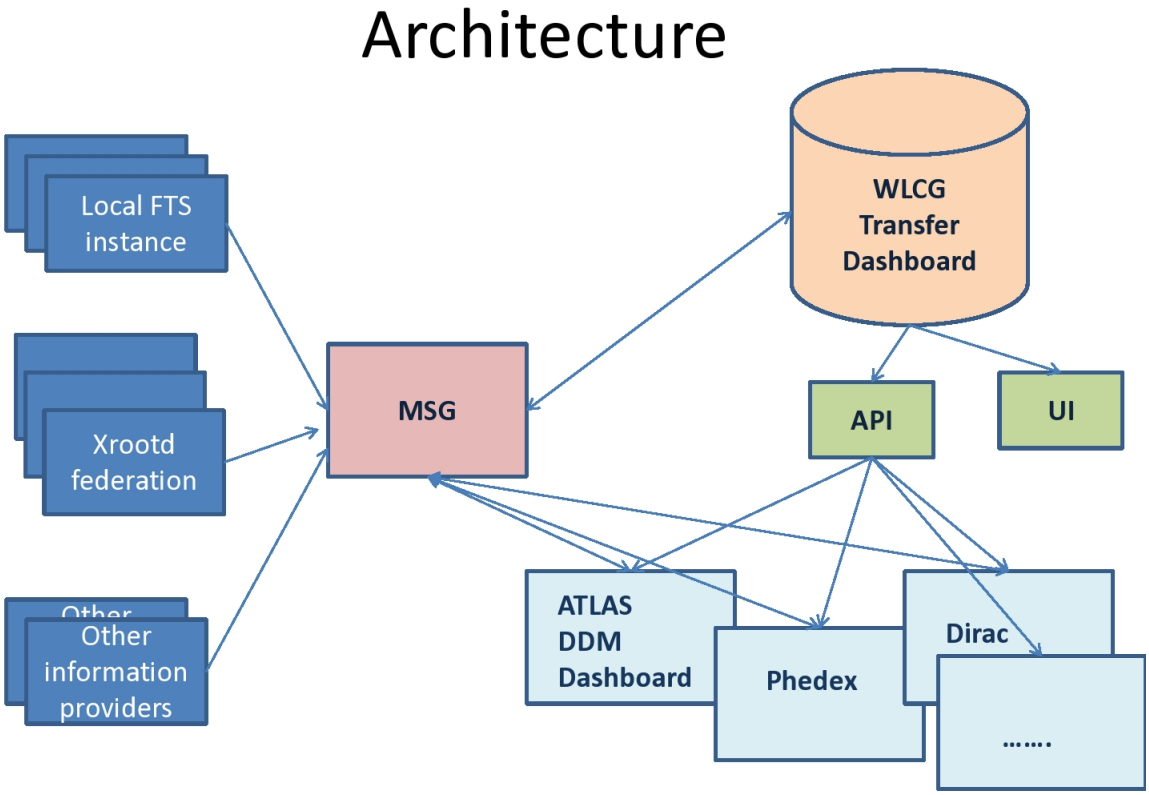
Abstract

FTS3 is the service responsible for globally distributing the majority of the LHC data across the WLCG infrastructure. It is a low level data movement service, responsible for reliable bulk transfer of files from one site to another while allowing participating sites to control the network resource usage. In this project, it will be explored whether it is feasible to deploy FTS in Kubernetes environment.

* CERN, Geneva, Switzerland
** University of Piraeus, Piraeus, Greece

Contents

1 Introduction



A file transfer service (FTS) is a service that is responsible for the central management of data files. The purpose of an FTS is to allow computers on the same network to access the files managed by it. The aforementioned service adds value to any infrastructure by providing users with the ability to share information over a network.

Specifically, FTS3 is the service responsible for globally distributing the majority of the LHC data across the WLCG infrastructure. It is a low level data movement service, responsible for reliable bulk transfer of files from one site to another while allowing participating sites to control the network resource usage.

2 Motivation

As previously mentioned, FTS3 is the service responsible for globally distributing the majority of the LHC data across the WLCG infrastructure. Therefore, due to its nature, FTS3 has high computational resource needs, even though the aforementioned resources often remain idle.

Consequently, a need has arisen to explore whether it is feasible to utilize the Kubernetes environment in order to automating deploy, scale, and manage the corresponding applications.

3 Project Implementation

The process of deploying FTS in Kubernetes environment will be explored in the following chapters.

3.1 Prerequisites

3.1.1 VOMS Client

The Virtual Organization Membership Service (VOMS) enables Virtual Organization (VO) access control in distributed services. It is at the core of the WLCG authorization stack and it is used daily to authorize access to storage and computing resources.

Membership to a VO is proven via temporary proxy certificates, which are signed by the VOMS server and certify that a user belongs to the corresponding VO group. Particularly, once having obtained a Grid User Certificate from the CERN Certification Authority and having joined the group dteam, a user is able to create a temporary proxy for the aforementioned group.

```
Contacting voms2.hellasgrid.gr:15004 [/C=GR/O=HellasGrid/OU=hellasgrid.gr/CN=voms2.hellasgrid.gr] "dteam"...
Remote VOMS server contacted succesfully.


Created proxy in /tmp/x509up_u141868.

Your proxy is valid until Mon Aug 02 03:06:38 CEST 2021
```

Similarly, it is expected that a Grid Host Certificate must be obtained from the CERN Certification Authority, which should be accessible at the /etc/grid-security/ directory, in order for the FTS to function properly on the next stages of the process.

In the spirit of contribution to open source initiatives, a VOMS Client Setup Guide has been also created as part of the File Transfer Service project. *

* See Appendix A

3.1.2 MySQL Database

MySQL is an open source SQL database management system.

A remotely accessible MySQL database has been created by means of the CERN Database On Demand service.

3.1.3 Repositories & Packages

A number of packages is needed in order to deploy FTS3, most of which can be found on the FTS repository and DMC repository.

Additionally, Python 2.7 and pip are required for most components, whereas PHP is required for the WebFTS component.

Moreover, given that Python 2 has reached end of life and several packages are no longer available for it, they can only be installed by specifying the desired version for each of them, such as `pip install "zipp==3.1.0"`.

Certainly, another major component of FTS3 is the Grid File Access Library (GFAL2), a C library providing an abstraction layer of the grid storage system complexity.

3.2 FTS Components

3.2.1 FTS3-REST

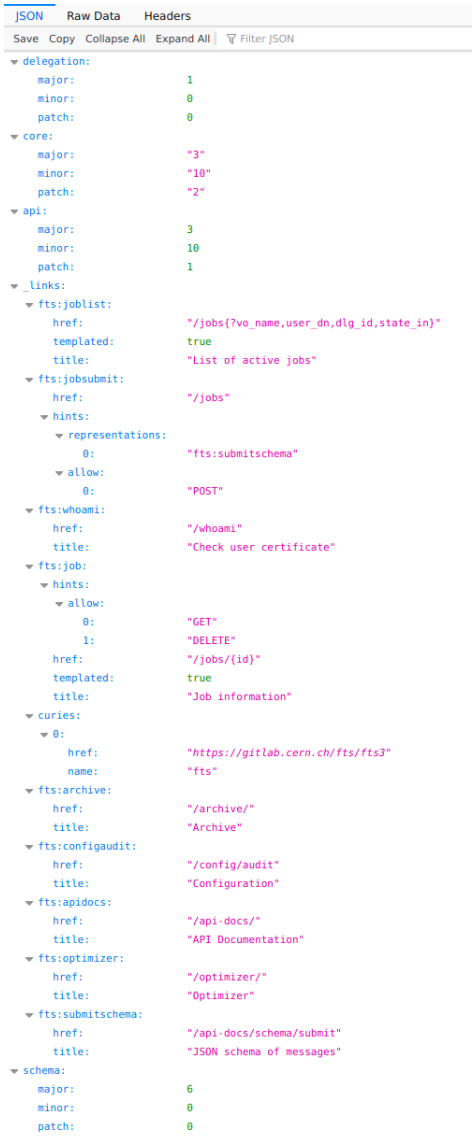
FTS-REST provides a Python API for easy integration with frameworks and a CLI for copying files from one site to another.

Its installation is based on the aforementioned packages and the existence of a MySQL database, whereas its configuration revolves around the /etc/fts3/fts3config file.

After having properly configured said file, opened the corresponding port and restarted the httpd service, executing the command fts-rest-whoami -s "https://host:port" should provide certain details regarding the server, given that those values would have been replaced with the desired ones.

Moreover, accessing the aforementioned URL via a browser should provide the information shown on the left.

It should be specified that the default port for this service is 8446.



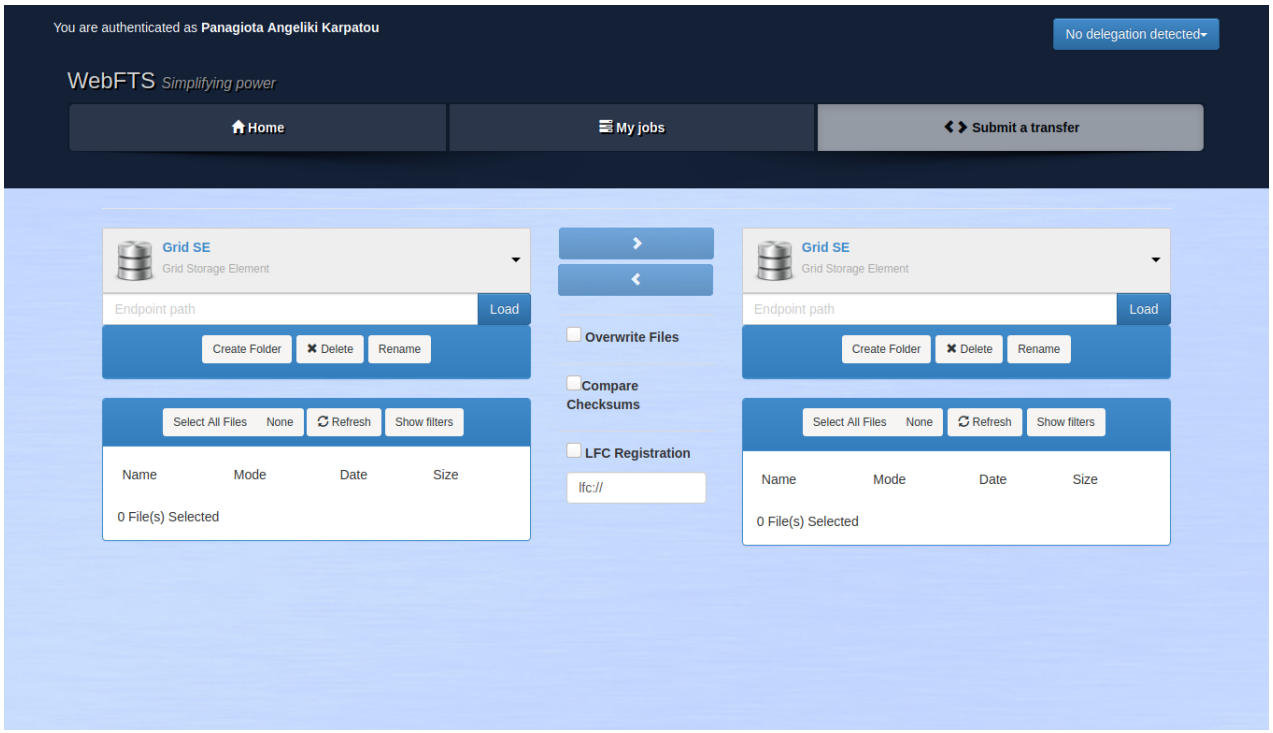
3.2.2 WebFTS

WebFTS is a web interface that provides a file transfer and management solution in order to allow users to invoke reliable, managed data transfers on distributed infrastructures.

Its configuration revolves around the /etc/httpd/conf.d/webfts.conf file, which can be originally found in the /var/www/webfts/conf/ directory.

After having properly configured said file, opened the corresponding port and restarted the httpd service, accessing the corresponding URL via a browser should provide the user interface shown below.

It should be specified that the default port for this service is 8444.



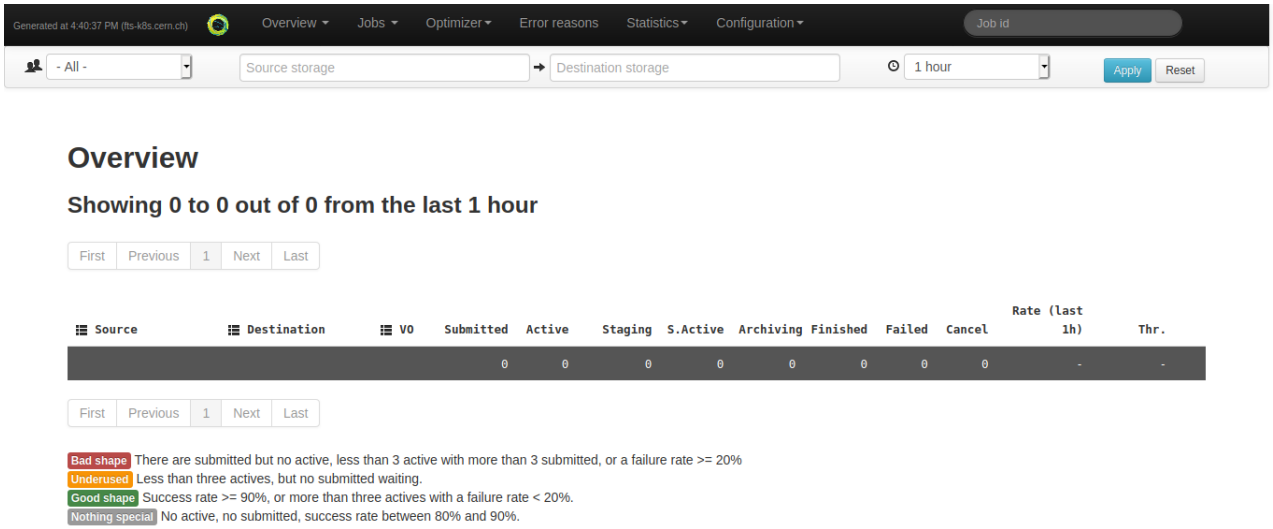
3.2.3 Web Monitoring

FTS provides monitoring for several profiles: General monitoring for end users, Discovery Data for researches and Service Specific for service managers.

Its configuration revolves around the /etc/fts3web/fts3web.ini file.

After having properly configured said file, opened the corresponding port and restarted the httpd service, accessing the corresponding URL via a browser should provide the user interface shown below.

It should be specified that the default port for this service is 8449.



3.3 Migration to Kubernetes

Kubernetes, is an open source system for automating deployment, scaling, and management of containerized applications. It groups containers that make up an application into logical units for easy management and discovery.

3.3.1 Kubernetes Components

3.3.1.1 Cluster

Kubernetes is used to coordinate a highly available cluster of computers that are connected to work as a single unit. The abstractions in Kubernetes allow users to deploy containerized applications to a cluster without tying them specifically to individual machines.

3.3.1.2 Application

Upon creation of a running Kubernetes cluster, the user can deploy containerized applications on top of it by creating a Kubernetes Deployment configuration. The Deployment instructs Kubernetes how to create and update instances of the application.

Once a Deployment is created, the Kubernetes control plane schedules the application instances included in that Deployment to run on individual Nodes in the cluster.

Once the application instances are created, a Kubernetes Deployment Controller continuously monitors those instances.

3.3.1.3 Pod

Once a Deployment is created, Kubernetes will create a Pod to host the application instance. A Pod is a Kubernetes abstraction that represents a group of one or more application containers, and some shared resources for those containers.

3.3.1.4 Node

A Pod always runs on a Node. A Node is a worker machine in Kubernetes and may be either a virtual or a physical machine, depending on the cluster. Each Node is managed by the control plane.

A Node can have multiple pods, and the Kubernetes control plane automatically handles scheduling the pods across the Nodes in the cluster. The control plane's automatic scheduling takes into account the available resources on each Node.

3.3.2 FTS Deployment

Upon creation of a cluster and eventually, deployment of FTS in Kubernetes environment, it will consist of the following applications:

- FTS Server
- FTS-REST
- FTS Web Monitoring

3.3.2.1 FTS Server

Once having created an image of a server running the FTS Server, the corresponding application can be deployed.

It is important to also create the volumes /etc/fts3, which should contain the fts3config and fts-msg-monitoring.conf files, and /etc/grid-security, which should contain the Grid Host Certificate from the CERN Certification Authority, along with the corresponding keys and the VOMS files.

The containerized application would then be automatically assigned to a pod, and could be hosted by a node.

Please note that an FTS Server Docker container registry is already hosted on the File Transfer Service Project at GitLab.

3.3.2.2 FTS-REST

Once having created an image of a server running the FTS-REST, the corresponding application can be deployed.

It is important to also create the volumes /etc/fts3, which should contain the fts3config file, and /etc/grid-security, which should contain the Grid Host Certificate from the CERN Certification Authority, along with the corresponding keys and the VOMS files.

The containerized application would then be automatically assigned to a pod, and could be hosted by a node.

It should be specified that the default port for this container is 8446.

Please note that an FTS-REST Docker container registry is already hosted on the File Transfer Service Project at GitLab.

3.3.2.3 FTS Web Monitoring

Once having created an image of a server running the FTS Web Monitoring, the corresponding application can be deployed.

It is important to also create the volumes /etc/fts3, which should contain the fts3config and fts3rest.ini files, and /etc/grid-security, which should contain the Grid Host Certificate from the CERN Certification Authority, along with the corresponding keys.

The containerized application would then be automatically assigned to a pod, and could be hosted by a node.

It should be specified that the default port for this container is 8449.

Please note that an FTS Web Monitoring Docker container registry is already hosted on the File Transfer Service Project at GitLab.

4 Acknowledgements

I would like to sincerely thank CERN for offering me the opportunity to participate in the Summer Student Programme, and most of all, my supervisors, colleagues, lecturers and coordinators for their patience and eagerness to teach and help throughout the implementation of such an exciting project.

5 References

[1] CERN OpenStack Private Cloud Guide, https://clouddocs.web.cern.ch/

[2] Access using Kerberos, https://linux.web.cern.ch/docs/kerberos-access/

[3] EGI IGTF Release, https://wiki.egi.eu/wiki/EGI_IGTF_Release

[4] File Transfer Service, https://fts.web.cern.ch/fts/

[5] FTS3 Documentation, https://fts3-docs.web.cern.ch/fts3-docs/

[6] Access to Kubernetes cluster from outside CERN, https://codimd.web.cern.ch/vjC8BHbTS7etHwJve-K2Uw

[7] Kubernetes, https://kubernetes.io/

[8] Apache ActiveMQ, https://activemq.apache.org/

Appendix

A VOMS Client Setup Guide

Once you have obtained your **Grid User Certificate** in the form of a PKCS #12 file with a .p12 extension, you will be able to create both your **certificate** and **private key** in the form of PEM files:

```
$ openssl pkcs12 -in "usercred.p12" -out "usercert.pem" -clcerts -nokeys
$ openssl pkcs12 -in "usercred.p12" -out "userkey.pem" -nocerts -nodes
```

You should then move said files in the following locations and change their permissions accordingly:

```
$ mv -t $HOME/.globus/ usercred.p12 usercert.pem userkey.pem

$ sudo chmod 600 $HOME/.globus/usercred.p12
$ sudo chmod 644 $HOME/.globus/usercert.pem
$ sudo chmod 400 $HOME/.globus/userkey.pem
```

After having set up properly your certificates, you will be ready to install the packages required for the creation of a **VOMS Client**:

```
$ sudo yum install voms-clients3
$ sudo yum install voms-clients-java
```

Specifically, for the **ca-policy-egi-core** meta-package, you will have to create the following repository, according to the https://wiki.egi.eu/wiki/EGI_IGTF_Release:

```
$ sudo vim /etc/yum.repos.d/EGI-trustanchors.repo
```

You will then have to include there the **configuration** below:

```
[EGI-trustanchors]
name=EGI-trustanchors
baseurl=http://repository.egi.eu/sw/production/cas/1/current/
gpgkey=http://repository.egi.eu/sw/production/cas/1/GPG-KEY-EUGridPMA-RPM-3

gpgcheck=1
enabled=1
```

After that, you will be able to install the meta-package **ca-policy-egi-core**:

```
$ sudo yum install ca-policy-egi-core
```

Last but not least, you will also need a few more files for both **vomses** and **vomsdir**, which are already available via LXPLUS and can be obtained via SFTP:

```
$ sftp lxplus.cern.ch
sftp> cd /etc/vomses
sftp> get dteam*
Fetching /etc/vomses/dteam-voms2.hellasgrid.gr to dteam-voms2.hellasgrid.gr

/etc/vomses/dteam-voms2.hellasgrid.gr 100% 112 135.3KB/s 00:00
sftp> quit

$ sftp lxplus.cern.ch
sftp> cd /etc/grid-security/vomsdir/dteam
sftp> ls
voms2.hellasgrid.gr.lsc
sftp> get *
Fetching /etc/grid-security/vomsdir/dteam/voms2.hellasgrid.gr.lsc to voms2.hellasgrid
/etc/grid-security/vomsdir/dteam/voms2.hellasgrid.gr.lsc 100% 129 148.6KB/s
00:00
sftp> quit
```

You should then move said files in the following locations:

```
$ sudo mkdir /etc/vomses
$ sudo mv dteam-voms2.hellasgrid.gr /etc/vomses/
$ sudo mkdir /etc/grid-security/vomsdir/dteam
$ sudo mv voms2.hellasgrid.gr.lsc /etc/grid-security/vomsdir/dteam/
```

Congratulations, you will be now able to create your **VOMS Proxy**:

```
voms-proxy-init -voms dteam
```

If everything has worked properly, you will get an output similar to the one below:

```
Contacting voms2.hellasgrid.gr:15004 [/C=GR/O=HellasGrid/OU=hellasgrid.gr/CN=voms2.he
"dteam"...
Remote VOMS server contacted succesfully.

Created proxy in /tmp/x509up_u141868.
Your proxy is valid until Mon Jul 26 13:10:42 CEST 2021
```

Please note that the aforementioned guide has been published on the official CERN FTS3 Documentation.