Deploying Container Based FTS on Openstack Magnum

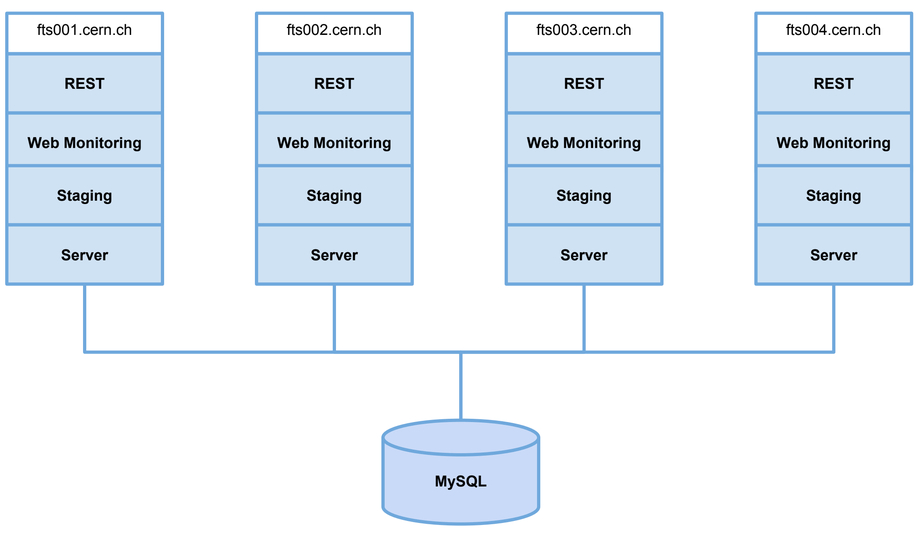
Andrea Manzi, Alejandro Álvarez Ayllón, María Arsuaga Ríos, Ricardo Brito Da Rocha

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

Current deployment of FTS

FTS has been designed to be easy to scale horizontally just adding more VMs to a server. Each instance is a clone running the exact same set of services, with state persisted via a shared MySQL database.

The service splits the load evenly across VMs, and when a node enters or exits, the queue is readjusted accordingly, so each instance runs a similar amount of transfers.



Issues with the current deployment

While we scale all services in parallel, the fact is that the scaling needs of each one is different.

For instance, we may need to add a new VM to sustain the amount of parallel transfers required to process the queue, but there may be no staging operations pending, or the amount of HTTP requests may actually remain constant.

Therefore, we are scaling services that do not need to scale. These additional services will consume resources (mainly RAM), that could be used to run more transfers in parallel.

Here we evaluate the containerization of FTS as a possible way of independently scaling services, making the overall operation cheaper in terms of computing resources.

Magnum

Magnum is an OpenStack API service making container orchestration engines such as Docker Swarm, Kubernetes, and Apache Mesos available as first class resources in OpenStack.

Magnum uses Heat to orchestrate an OS image which contains Docker and Kubernetes and runs that image in either virtual machines or bare metal in a cluster configuration.

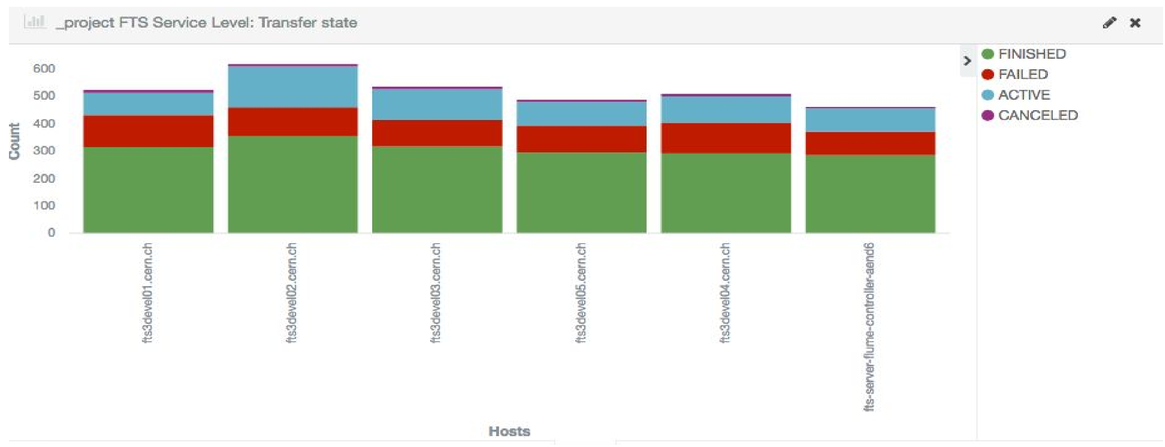
The installation at CERN is using VMs for the moment and TLS support. It has been extended with specific customizations for HEP, like CVMFS and EOS support (under integration)

Links

https://gitlab.cern.ch/fts/docker-files/
https://gitlab.cern.ch/fts/fts-rest/container_registry
https://gitlab.cern.ch/fts/fts3/container_registry
https://gitlab.cern.ch/fts/fts-monitoring/container_registry

Dockerized FTS

A single FTS VM deployment has been split in 3 different containers (Server/Staging, REST, Web Mon) automatically built via GitLab CI and published to GitLab Registry available at CERN.
Both Server/Staging and REST containers have been integrated with the Monitoring infrastructure at CERN, based on Flume/ES/Kibana, so logs are automatically published and metrics are available on dedicated dashboards.



FTS is one of the pilots for the Magnum deployment at CERN. 2 clusters have been deployed for evaluation: Swarm and Kubernetes. Mesos is also available since September, but has not been evaluated yet.

Docker Swarm

First tests were performed with Docker 1.10 and Docker Swarm 1.2.3 deployed via Magnum. FTS3 configuration and x509 certificates have been installed directly on the Bays nodes and mounted via Docker volumes.
We managed to run the 3 FTS container types without issues, but we missed some functionalities implemented only in Docker 1.12 not yet available at CERN via Magnum (Docker Service in particular). Therefore we concentrated our testing to Kubernetes

Kubernetes

A Kubernetes (v 1.2.0) cluster deployed vía Magnum allowed us to:

- perform scaling of the FTS Server/Staging component according to our needs. A first component in Python was developed integrating Kubernetes REST API to autoscale based on the FTS transfer queues.
- set up a unique REST API entry point via Kubernetes Services and HA-Proxy. The FTS REST containers could be scaled over the cluster transparently to users
- write log to Persistent Volumes (locally but EOS integration is also foreseen)
- test rolling upgrades

Future Work

The scaling at the level of the cluster VMs performed in Magnum via Heat is the next item on the list, together with new tests with Swarm when the latest features will be available. A mixed VM + Docker deployment is now possible but we aim first at reaching the same confidence at operating our VM cluster before eventually move to a full dockerized FTS service in production.
Future developments FTS side, which will allow further isolation of the Server, Staging, Optimizer and Scheduler functionalities are foreseen: a micro-service architecture is crucial in order to benefit from the Swarm/Kubernetes Orchestration.