Table of Contents

CMS Singularity Support

Introduction

If present and working, CMS pilots will automatically utilize the singularity container system to launch payloads. This provides CMS with:

- **Isolation**: the payload cannot interact with other payloads or pilots. When singularity is used, the use of `glexec` (which also provides isolation and traceability) is disabled.
- **Portable OS environments**: CMS will start the payload in the appropriate OS environment (RHEL6 and RHEL7), regardless of the host OS. As the container is provided by CVMFS, we keep the number of runtime dependencies for your site minimal.

Planning your Site Deployment

The CMS pilot likely can startup on any RHEL-like OS environment; it is possible it would function on Debian (we'd love to hear from testers!). Particularly, **RHEL6 is fully supported**; there are some known kernel / autofs bugs we work around in the pilot wrapper.

Singularity will utilize CVMFS to distribute files; however, since CMS uses a very minimal number of binaries from the OS, you do not need to plan additional space in your local disk cache.

Known limitations:

- Due to a bug in CRAB3, pilots need to run on RHEL6 in order to run CRAB3 jobs. This will be fixed in the next CRAB3 release.
- SAM tests have not been updated for testing RHEL7; they will need to be run in a RHEL6 environment. No estimate is available for fixing SAM.
- Singularity is not recursive: if CMS will use singularity for payloads, then the site cannot start the CMS pilot inside singularity. *However*, with modest effort, one can run Singularity inside Docker (not yet documented; contact Nebraska for more details on the setup).

Install RPM

Install the RPM using the OSG instructions. The same RPM is also available in the WLCG yum repository which is probably more convenient for those who get their Linux distribution from CERN.

The configuration file is in `/etc/singularity/singularity.conf`; no changes are necessary to support CMS.

One way to minimize risk inherent to using a `setuid` binary is to disable unnecessary features. For CMS, there are two possible steps:

1. Only install the `singularity-runtime` package (removes the ability to create new images).
2. CMS does not use loopback-based image mounts, so it is OK to disable them as described in the OSG instructions.

While this does reduce the risk, these are not required and may disrupt other `singularity` users at your site.

Verify CVMFS and SITECONF

CVMFS must be functioning on the host OS; currently, the images live inside the `/cvmfs/singularity.opensciencegrid.org` repository. This repository must be available on your host (either through `autofs` or static mounts). If you are unsure if this is available, you can simply type `ls /cvmfs/singularity.opensciencegrid.org` to see if any errors occur. Also make sure that `/cvmfs/oasis.opensciencegrid.org/` is available as well.

The contents of `/cvmfs/cms.cern.ch/SITECONF/local` *must* be visible inside the container. If you configure CVMFS such that the `local` symlink evaluates to a directory outside CVMFS, then three options exist:

1. Switch to distributing the contents of `SITECONF/local` via to CVMFS.
2. Singularity will automatically bind-mount `/etc/cvmfs/SITECONF` into the container if it exists. Copy your SITECONF files there (maintain with Puppet or similar) and update the `local` symlink to `/etc/cvmfs/SITECONF`.
3. If it is prohibitive to move SITECONF (i.e., lives on NFS), then **bind mount** the existing location to `/etc/cvmfs/SITECONF` (a symlink will not work) and update the `local` symlink to `/etc/cvmfs/SITECONF`.

Configure POSIX Storage Elements

If CMSSW reads files at your site through the POSIX APIs (i.e., `file://` protocol), then the POSIX storage must also be accessible inside the container *as the pilot user*.

Two potential changes will be needed:

- Change the permissions for CMS files such that the CMS pilot user can read CMS data.
- Bind mount the storage to `/cms` and update `storage.xml` accordingly. To be visible, a *bind mount* must be used (symlinks will not suffice).

Stageout will occur as the pilot UID. Given the user / production separation requirements for CMS, this implies you will need to stageout through a grid service (GridFTP, Xrootd, HTTPS) instead of using POSIX-based cp). Check your site-local-config.xml and change plugin in local/fallback stageout sections if needed. The cmspilot user should not have generic write access to all local files.

Testing

Each pilot will test for a functioning `singularity` at startup. It will log any failures to its stdout. If you are unable to read pilot stdout for jobs running at your site, then you may access the pilot logs from this location:

http://submit-3.t2.ucsd.edu/CSstoragePath/FactoryLogsGlobalPool/sdsc/

Pilot logs are typically copied to that web directory 45 minutes after the pilot exits.

This topic: Main > CmsSingularity
Topic revision: r10 - 2018-05-04 - BrianBockelman