Docker experience at INFN-Pisa Grid Data Center

2015 J. Phys.: Conf. Ser. 664 022029

(http://iopscience.iop.org/1742-6596/664/2/022029)

Journal of Physics: Conference Series **664** (2015) 022029 doi:10.1088/1742-6596/664/2/022029

Docker experience at INFN-Pisa Grid Data Center

E. Mazzoni[1], **S. Arezzini, T. Boccali, A. Ciampa, S. Coscetti**
INFN Sezione di Pisa, Pisa (ITALY)

D. Bonacorsi
University of Bologna and INFN Sezione di Bologna, Bologna (ITALY)

Abstract. Clouds and virtualization offer typical answers to the needs of large-scale computing centers to satisfy diverse sets of user communities in terms of architecture, OS, etc. On the other hand, solutions like Docker seems to emerge as a way to rely on Linux kernel capabilities to package only the applications and the development environment needed by the users, thus solving several resource management issues related to cloud-like solutions. In this paper, we present an exploratory (though well advanced) test done at a major Italian Tier2, at INFN-Pisa, where a considerable fraction of the resources and services has been moved to Docker. The results obtained are definitely encouraging, and Pisa is transitioning all of its Worker Nodes and services to Docker containers. Work is currently being expanded into the preparation of suitable images for a completely virtualized Tier2, with no dependency on local configurations.

1. Introduction

Docker [1] is an open platform to build, ship and run distributed applications. It is a container based virtualization framework that uses Linux containers (LXC [2]) as the core technology, with a large set of tools allowing for large-scale container handling, shipping and deployment. It allows for easy container contextualization and inter-container communications. Docker utilization is becoming the de-facto standard for container based virtualization, and is the recommended solution endorsed by most heavy weight IT providers.
Some examples are Spotify, that uses Docker for continuous delivery, service testing and deployment; Baidu, that uses Docker as a Platform-as-a-Service (PaaS), profiting of its flexibility for many framework and applications; eBay, that uses Docker for its easy application deployment, and for continuous integration process; and many more. Recently, Docker has reached 120k lines of code, about 10k commits and roughly 600 core contributors, of which about 95% work outside Docker.

Virtualization can be achieved in several ways, depending on the desired (or acceptable) level of invasiveness. Setting aside solutions where the whole system hardware is emulated (like QEMU [3]), the industry reference technologies are full system virtualization (OpenStack [4], OpenNebula [5]), where each virtualized machine has its own running kernel, and kernel based process isolation, which has been a possibility in Linux since the beginning, via *chroot*, and more recently via *cgroups* and projects like the Linux Container (LXC).
The former technology has a broader scope, being able to utilize different operating systems on the same physical host (e.g. mixing Windows and Linux machines), but pays a price in terms of resource utilization (mainly RAM, to a smaller extent CPU reduced efficiency) and access to high performance devices. It is considered an overkill in HEP, where more or less all the scientific computation is carried out on Linux machines. Linux Containers, at the heart of the Docker technology, instead of simulating complete machines, just provide process separation and sand-boxing over a common kernel all the machines share. This is particularly adequate in an environment where system standardization has

1enrico.mazzoni@pi.infn.it

already been achieved to a large extent, for example by the adoption of grid middleware operated on WLCG resources.
Docker provides CPU, memory and file system isolation, and can run different processes on the same kernel but with completely different runtime (e.g. one can run SLC5 on SLC6 or on Ubuntu). Docker adds on top of LXC a complete series of tools for image storing, deployment, testing and contextualization, as well as for container handling (*start/stop/...*) and communications; derived images can be based on existing images, with Docker taking care to save only the deltas between them. It also add a public repository (Docker Hub, *hub.docker.com*), which allows for safe image storing and easy deployment for open software projects, which provides already cooked images for most of the widespread Linux based operating systems. Currently Docker Hub hosts 45k publicly accessible images, and starting a new container on a Linux based machine is as easy as a single line command:

```
$ docker run -it centos:centos7
$ cat /etc/redhat-release
CentOS Linux release 7.0.1406 (Core)
```

The images can be built using so called *Dockerfiles*, e.g.:

```
FROM cmssw/slc6-vanilla
    RUN yum -y update && yum -y install rubygems ruby-devel gcc ruby193
    RUN echo "gem: --no-ri --no-rdoc" > ~/.gemrc && \
       gem install puppet && \
       gem install librarian-puppet -v 1.0.9
    CMD /bin/bash
```

All images are layered, there is no need to re-download previously downloaded layers (e.g. each of the above statements is a layer). Additionally, it is relatively straightforward to set-up Docker on one's own operating system of choice. Docker comes pre-packaged on most modern distributions, including SLC6:

```
sudo yum install docker-io
sudo service docker start
```

2. The INFN-Pisa Computing Center

Virtualization technologies are the best choice when dealing with users with diverse needs in term of computing resources. While Linux is the de-facto standard for most scientific uses, precise hardware and software needs can still vary to the extent of not allowing for a catch-all solution. Software from previous generation experiments is often not validated for the latest-greatest Linux versions, for example, and thus cannot coexist with recent experiments, which instead prefer the best performance coming with newer releases.
The CMS Pisa Tier-2 center is one of the biggest Italian scientific computing center: it is a Tier-2 in WLCG, supporting CMS, but CMS is not the biggest among the users, which include also ATLAS, LHCb, and 20 other Virtual Organizations, a National Theoretical Physics computing center, a 2000 (and more) cores fluidodynamical cluster (used in industry related researches). In terms of resources, it consists of 8k cores, >2 PB of high performance storage and highly heterogeneous WN hosts (some are 1 GbE, some 10 GbE, some Infiniband). Also the OS requests are heterogeneous (some still SL5, even SL4 up to some months ago; some prefer very recent releases, generally OpenSUSE).
The only common points between such a diverse user-community are:

- GPFS [6] is used to serve data for all the use-cases supported on-site;
- AFS [7] is used for user areas;
- LSF [8] is used for resource access, also interactive.

The main issue in Pisa recently was to understand how to provision the correct environment to all these diverse resources. Virtual Machines (e.g. OpenStack) were considered as an option, but Pisa also has older machines that are low in RAM. Infiniband connectivity, moreover, seems to lose performance in a completely virtualized environment. The solution identified up to recently so far was a very light virtualization via *chroot*. Every host machine (with the very latest OpenSUSE kernel) was starting sand-boxed machines as tar files containing complete SL6 (or SLx) systems, via *chroot*. Site-wide file-systems (CVMFS/GPFS/AFS) were mounted by the host, and then seen by the machines as local file systems using the *mount --bind* option. Every machine had pre-installed as many tar files as the possible environments are: these could be un-tarred and "started" on demand via a set of in house developed scripts. Each tar had a preset LSF client configuration, ultimately deciding to which LSF pool the machine had to join. This solution works and it has been in production for Pisa since 3 years. On the other hand, Docker is a viable solution, too. It has the same overhead (virtually 0) as *chroot*, and comes with the additional advantage of easier image storing, deployment and control.
These are the main points in favor of Docker adoption in Pisa:

- no more tar files and complicated management;
- the adoption of a git-like image management, where every action on the repository is properly logged, and you can branch, pull, etc;
- the very easy deployment of a local image repository, to avoid exposure of sensitive information;
- the extremely easy conversion of an existing *chroot*'s tar image to a Docker image, via a single command:

```
sudo tar -C ExistingChrootBaseDir -c . | sudo docker import - PisaWN
```

A Tier-2 on-demand approach was initially planned, where no dependency on the underlying physical architecture was foreseen, but currently it has been decided to prioritize ease of deployment and performance over flexibility. Hence Pisa decided to mount CVMFS/GPFS/AFS on the host and pass them via "-v" option, instead of mounting them inside the container. The solution also allows for the sharing of CVMFS/AFS caches in case of multiple running machines, and does not force the containers to run in privileged mode, which would not be possible e.g. in opportunistic sites. LSF runs within the container, and automatically connects the machine to the proper LSF queue, with a start command as follows. A Worker Node instance is started just via

```
docker run -v /cvmfs:/cvmfs -v /afs:/afs \ -v /gpfs/ddn:/gpfs/ddn \
-v /chrootlfs/home:/home/grid –d \
-t localregistry.pi.infn.it:5000/enricomazzoni/testwn:0.3 \
/etc/sysconfig/docker-pi/start
```

The complete workflow of operations is shown in Figure 1: admin's action selects and starts the appropriate container, which then joins the correct LSF queue and becomes available for processing (via batch system and interactive). Pisa moved 10% of the WNs to Docker so far, following an R&D process of just a couple of weeks. No user actually experienced any difference. The rest of the center will migrate as well in the next few weeks. At this point, the idea would be to move all the services used by CMS via Docker, namely:

- Squid: just a Linux standard machine;
- Computing Elements (CEs): one already moved without problems, starting from a *chroot*-ed test we had;
- User Interfaces (UIs): similar to WNs, can share most of their image;
- PhEDEx (CMS transfer agents on a User Interface): a simple modification of the latter;
- Xrootd redirectors (for CMS data federations): Pisa moved it from *LXC* and it is already running now with Docker.

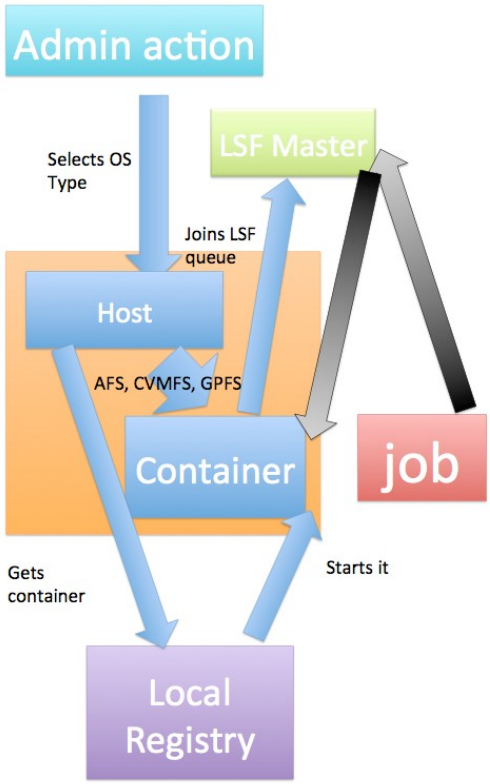

Figure 1: Workflow for starting a docker container, and attach it to the Batch infrastructure.

3. **A more site independent Tier2 on demand approach**

As explained in Section 2, the setup used in Pisa heavily depends on the knowledge of the local system: the example command line there assumes AFS, CVMFS and GPFS are available to the host, and are just passed to the container as-is. While the solution is optimal for Pisa, there pre-requirements cannot in general be assumed to hold for a generic Linux site willing to contribute its resources to any of the users Pisa hosts.

The optimal solution would instead be to depend on nothing but a compatible Linux kernel (currently anything 64 bit greater than 2.6.x is generally ok); in particular one cannot assume any special file system to be present on the host (AFS is not a real issue, CVMFS is). For what concerns access to input data (present on GPFS in Pisa), the simple solution is to access them via a remote streaming protocol (either Xrootd or WebDAV), which most of our biggest users already allow as a fallback to local access.

The naive solution of mounting a plain CVMFS in the container does not work, since this eventually needs to interact with the host kernel, which is not allowed but when using Docker in “privileged mode”, which would not be accepted easily by resource providers, since it removes most of the process separation and sand-boxing Docker provides. A solution to the problem exists since long, originally developed for the CDF experiment at FNAL, and is called Parrot [9]: it provides user space access to various file systems, without any need of privileged operations, and CVMFS is among these.

The solution the Pisa center is testing involves a specifically cooked container, containing

- A recent CentOS image;
- The full WLCG User level middleware stack;
- A working Parrot setup, configured for CVMFS and the WLCG experiments;

- A local squid (CMVFS still needs a squid; while a site level one is preferred, a local one can still work for small setups);
- A user level setup suitable for WLCG authentication.

The image, publicly available as

tommasoboccali/wnondemand

is currently tested as the baseline for a site independent container setup. The container has shown to be able and run a standard CMS analysis application, using software from Parrot/CVMFS and remote data access. We plan to suggest this image for people performing casual analysis activities on their private machine; eventually we would like to test this is a large environment on opportunistic resources.

With a simple command:

```
docker run -t -i -e "SQUIDPROXY=cmssquid.pi.infn.it:3128" \
      -v /Users/tom/Globus:/root/Credentials \
      tommasoboccali/wnondemand \
      /root/startGridUser.sh
```

a non root shell is provided, with the Globus credentials as in the local area (/Users/tom/Globus), and CVMFS is activated using the external PROXY. If the

```
      -e "SQUIDPROXY=cmssquid.pi.infn.it:3128”
```

is not provided, a local squid is started, and CVMFS attaches to that.

At this point a simple:

- voms-proxy-init –voms cms
- source /cvmfs/cms.cern.ch/cmsset_default.sh
- scram list

gets the user a completely working CMS environment (but it would be the same with ATLAS, LHCb and many others).

4. **Conclusions**

In summary, Docker offers interesting opportunities to INFN-Pisa Computing center. It can ship entire CMSSW distributions and do benchmarking easily. It offers plenty of desired features and flexibility at a site level, and all this at no cost in terms of performances (as from the tests done so farm, at least). The transition in itself is very easy, indeed matter of days, admittedly maybe also because Pisa was already at *chroot* point - but other works in CMS show that starting from bare metal is as easy, and Docker provides most of the specific scripts a site may need for image processing, download history, starting, stopping. In a nutshell, it is a neat tool, and we will continue testing it (and possibly integrating it) in our activities.

The present work is partially funded under program PRIN “/STOA-LHC 20108T4XTM/”, /CUP: I11J12000080001,/.

5. **Bibliography**

[1] https://www.docker.com/
[2] https://linuxcontainers.org/lxc/introduction/
[3] http://www.qemu.org/
[4] https://www.openstack.org/
[5] http://opennebula.org/
[6] *"GPFS: A Shared-Disk File System for Large Computing Clusters"*, Proceedings of the FAST 2002 Conference on File and Storage Technologies Monterey, California, USA January 28-30, 2002.
[7] http://www.openafs.org/
[8] http://www-03.ibm.com/systems/platformcomputing/products/lsf/
[9] Douglas Thain and Miron Livny, *"Parrot: Transparent User-Level Middleware for Data Intensive Computing"*, Workshop on Adaptive Grid Middleware at PACT, January, 2003.