2017 25th Euromicro International Conference on Parallel, Distributed and Network-Based Processing

# Coherent Application Delivery on Hybrid Distributed Computing Infrastructures of Virtual Machines and Docker Containers

Germán Moltó, Miguel Caballer, Alfonso Pérez, Carlos de Alfonso, Ignacio Blanquer
*Instituto de Instrumentación para Imagen Molecular (I3M).*
*Centro mixto CSIC - Universitat Politècnica de València - CIEMAT*
*Camino de Vera s/n, 46022 Valencia, España*
*Email: gmolto@dsic.upv.es,{micafer1,caralla,alpegon,iblanque}@i3m.upv.es*

***Abstract*—There is an opportunity for Distributed Computing Infrastructures (DCIs) to embrace container-based virtualisation to support efficient execution of scientific applications without the performance penalty commonly introduced by Virtual Machines (VMs). However, containers (e.g. Docker) and VMs feature different image formats and disparate procedures for deployment and management, thus hindering the adoption of hybrid DCIs (HDCIs) comprised of those kind of resources. This paper describes a workflow based on open-source tools and standards to introduce coherent application delivery on HDCIs in which applications require to be deployed on both VMs and Docker containers. Leveraging and extending the TOSCA standard to describe application requirements, and adopting DevOps practices, resulted in the coherent creation of the artifacts required for the execution of the applications on different platforms. The paper features the adoption of this approach in the INDIGO-DataCloud project.**

***Keywords*-Cloud Computing; Distributed Infrastructures; Containers; Docker; Scientific Computing**

## I. Introduction

Scientific applications typically require Distributed Computing Infrastructures (DCIs) that can satisfy their execution requirements. With the advances of the hypervisor technologies Cloud Computing raised as a computing paradigm in which pools of shared computing resources are offered for multiple tenants using Virtual Machines (VMs) as the unit for application isolation and performance encapsulation. VMs allowed the (virtual) infrastructure to be adapted to the application and not viceversa. This simplified the process of application execution on other physical infrastructures at the expense of a performance penalty introduced by virtualisation [1].

As technology evolves, container-based technology have gained significant traction in the last years due to its near-native performance (see [2] for details) for application execution while featuring application isolation (though not as comprehensive as for VMs) for multi-tenant environments [3]. Among the different container technologies available (e.g. OpenVZ, LXC, Rkt), Docker[1] managed to widespread container adoption and, thus, a huge ecosystem of tools that adopted Docker containers blossomed. In particular Docker is significantly suitable for application delivery and there are examples in the literature that use container-based computing for application deployment and the composition of micro-services (see for example [4] or [5]).

Indeed, cloud-based infrastructures currently coexist with container-based infrastructures. Examples of the former are the infrastructures made available on a pay-as-you-go basis by public Cloud providers such as Amazon Web Services (AWS) or Google Cloud Platform (GCP) and on-premises clouds created by Cloud Management Platforms (CMPs) such as OpenNebula and OpenStack. Examples of the latter are the infrastructures created by Container Orchestration Platforms (COPs) such as Kubernetes, Docker Swarm or Apache Mesos. Even more, it is not uncommon to create container-based infrastructures on top of cloud-based ones, as is the case of the Amazon EC2 Container Service (EC2) which provides a highly scalable, fast, container management service for Docker containers.

Unfortunately, cloud-based infrastructures are not seamlessly compatible with container-based infrastructures. As an example, Virtual Machine Images (VMIs) created for a specific hypervisor (KVM, Xen or VMware) cannot be directly transformed into a Docker image, required to deploy a Docker container. However, Docker images have considerable benefits for application delivery across a range of multiple Operating Systems (OSs) and multiple hardware architectures platforms. This introduces build-once-run-anywhere capabilities for applications, a feature that is very much required on heterogeneous computing platforms, such as DCIs.

In this paper we use the term Hybrid DCIs (HDCIs) to refer to a collection of sites that offer computational capabilities in the shape of either VMs or Docker containers to support the execution of the different applications required by a set of scientific communities. There are other uses of the term hybrid in the computing literature such as the one employed by Moca et al. [6] in which a hybrid DCI comprises Internet Desktop Grid, Clouds and Best Effort Grids. Therefore, to clarify the scope of the paper, Figure 1 describes the different types of computational sites of a HDCI, as considered in this paper. Firstly, Figure 1.a)

[1]Docker - http://www.docker.com

2377-5750/17 $31.00 © 2017 IEEE
DOI 10.1109/PDP.2017.29

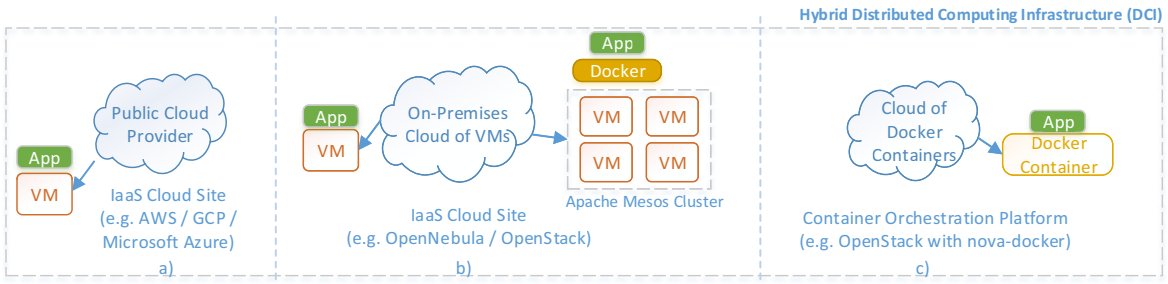


Figure 1. A sample hybrid Distributed Computing Infrastructure (DCI) composed of IaaS Cloud sites (both on-premises and public) and Container Orchestration Platforms.

depicts a public Cloud provider, which provide computing and storage capacity on a pay-as-you-go basis. This can be used by scientific communities for several reasons including, but not limited to: i) the ability to access computing power without requiring external approval by a scientific committee, as it typically happens when applying for supercomputing resources; ii) the ability to perform Cloud burst or to elastically supplement existing computational facilities with resources from a public Cloud in order to cope with increased workload and iii) the ability to provision an increased amount of simultaneous co-located resources to perform parallel execution of a scientific application by means of a virtual cluster. Secondly, Figure 1.b) depicts an on-premises Cloud site supported by a Cloud Management Platform such as OpenNebula or OpenStack on which VMs can be provisioned to support the execution of scientific applications. This is the case of the EGI Federated Cloud[2] a grid of academic private clouds and virtualised resources, built around open standards that supports the resource requirements of scientific communities. Users can provision individual VMs on which applications are executed.

Nowadays, there is a major trend in the use of DevOps approaches to support Infrastructure as Software (IaS). As Fitzgerald et al. [7] states, IaS proposes, among others: i) *version controlled infrastructure*, allowing rollback of changes not working as expected; ii) *immutable infrastructure*, where changes to the infrastructure are only made through version control to know the precise configuration of the infrastructure; when changes are applied to an infrastructure, the components are terminated an provisioned again rather than being dynamically changed while running; iii) *test-driven development*, i.e., the ability to consistently test infrastructure changes via automated testing and iv) *Continuous Faster Deployment*, by allowing automated, reproducible deployments without introducing manual configuration of resources.

Indeed, there exists an inherent gap that prevents scientific applications from being effectively deployed in hybrid DCIs composed by both VM-based infrastructures and container-based infrastructures. Different deployment strategies are currently required for VMs and Docker containers. Therefore, scientific communities require effective strategies to package and deploy applications on such diverse computational infrastructures. This is one of the goals of the INDIGO-DataCloud project[3] an H2020 EU project that involves 26 partners across Europe to develop an open-source data and computing platform targeted at scientific communities, deployable on multiple hardware and provisioned over hybrid, private or public, e-infrastructures. In particular, this paper summarises the procedure, tools and lessons learned to perform application delivery in hybrid DCIs based on both VMs and Docker containers, highlighting the approach adopted in the INDIGO-DataCloud project.

The remainder of the paper is structured as follows. First section II describes the proposed architecture to support coherent application delivery on different types of virtual resources. Then, section III describes an use case in order to highlight the benefits of the proposed architecture. Finally, section IV summarises the main achievements of the paper pointing to future work.

## II. Architecture for Application Development

This section identifies the components of the architecture employed to support the deployment of scientific applications that intend to be executed on HDCIs composed of Virtual Machines and Docker containers via a standard-based fully open-source approach, show in Figure 2.

In the figure, a distributed version control and Source Code Management (SCM) hosting service (GitHub) is used as a central code repository providing versioning support for all the files and traceability of the changes. It includes repositories for: i) the open-source scientific application,

[2]EGI Federated Cloud - https://www.egi.eu/infrastructure/cloud

[3]INDIGO-DataCloud - https://www.indigo-datacloud.eu

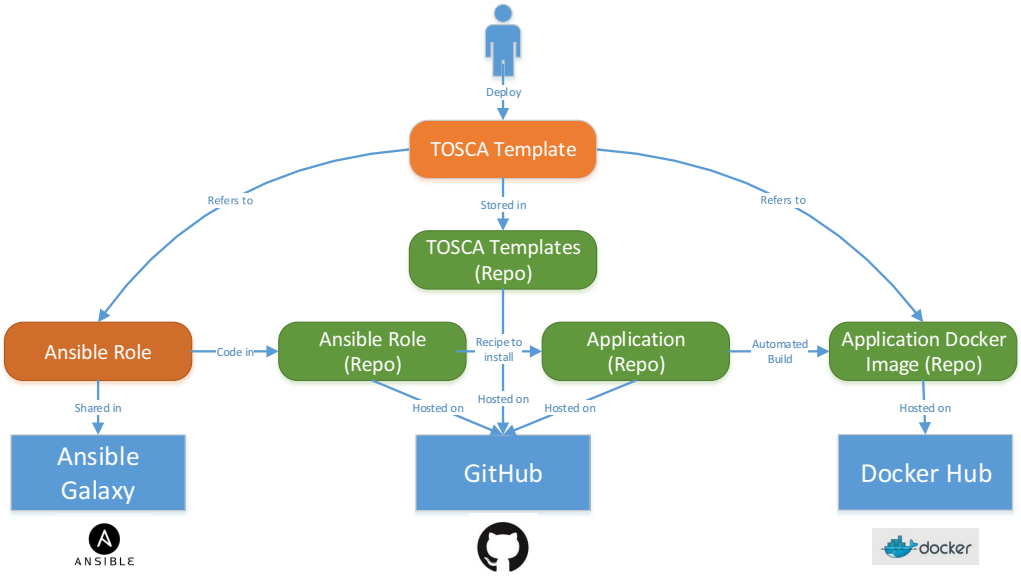


Figure 2. Overview of the platforms and their interrelation for application deployment support.

which also includes a Dockerfile to describe the application deployment procedure in a container; ii) an Ansible [8] Role with the recipe to install the application on the underlying platform (regardless of a Docker container or a VM) and; iii) the TOSCA templates that allow users to describe applications to be deployed on a HDCI. TOSCA (Topology and Orchestration Specification for Cloud Applications) [9] is an OASIS standard to describe Cloud application architectures as a topology template. In particular, we adopted the TOSCA Simple Profile in YAML version 1.0 [10] specification and included new non-normative types to support additional capabilities and applications. A TOSCA template is employed to describe the resources and configuration required to deploy a certain application.

Therefore, we rely on a single approach to describe application deployment both for VMs and for Docker containers. The very same Ansible Role is used to: i) dynamically deploy the application at runtime on a vanilla VM and ii) create the Docker image that will contain the application. This avoids having two different approaches for application deployment on VMs and Docker containers, thus easing maintenance. The Ansible Role is registered into Ansible Galaxy [11] so that it can be easily retrieved and shared. This way, Dockerfiles are simplified to the extent that they just retrieve the role from Ansible Galaxy and execute it.

The Ansible Roles are designed to be multi-platform (in particular, INDIGO-DataCloud supports both Ubuntu 14.04 and CentOS 7) so that the application can be seamlessly deployed under different Operating Systems. Also, they are specifically designed so that if the application to be deployed is already installed in the node all the recipe steps are rapidly skipped. This way, if the Ansible Role is executed on a container spawned out of the already-configured Docker image, no extra time is dedicated to its configuration.

The Docker images must encapsulate application code that has been tested to comply with the expected functionality. For this, a Continuous Integration (CI) strategy has been adopted so that proper SQA that involves code style, unit testing and integration testing, is assessed prior to creating the deployment artifacts (i.e. the Docker images), which are automatically built.

## III. Case Study: INDIGO-DataCloud

The aforementioned approach has been adopted in INDIGO-DataCloud, supporting the deployment of applications from specific user communities on an HDCI. This section provides further details by showcasing one of the applications supported in the area of bioinformatics: the Galaxy Portal [12], a web-based platform for data intensive biomedical research. Other research areas addressed by the project are Environmental and Earth Science and Physics and Astrophysics.

### A. The Galaxy TOSCA Template

The TOSCA template, available in GitHub[4], describes the deployment of a single node that will host the Galaxy Portal. The TOSCA template receives as inputs the number of CPUs

[4]TOSCA Template for Galaxy (single node): https://github.com/indigo-dc/tosca-templates/blob/master/galaxy.yaml

and memory required for the execution of application. This information is used by the Orchestration components of the PaaS layer of INDIGO-DataCloud to choose the appropriate node on which to perform the execution of the application. Further information about these components is available in a public deliverable [13].

In particular, the TOSCA template references two non-normative node types: *tosca.nodes.indigo.GalaxyPortal* and *tosca.nodes.indigo.LRMS.FrontEnd.Local* to specify that the front-end node will be used for the execution of jobs.These node types are defined in the file that includes the non-normative types for INDIGO-DataCloud[5]. Indeed, different non-normative types are defined for the different applications supported in project. Each non-normative type for an application references a specific Ansible Role registered in Ansible Galaxy. In this particular case it is the indigo-dc.galaxycloud[6] Ansible Role.

The TOSCA node types provide the abstraction and the Ansible Roles the recipe to deploy the application on multiple platforms. More complex TOSCA templates are available in the *indigo-dc/tosca-templates* GitHub repository, such as *galaxy_elastic_cluster.yaml* which deploys the Galaxy Portal on a front-end node of a virtual elastic cluster that features horizontal elasticity to provision additional working nodes on demand depending on the number of jobs queued up at the LRMS, supporting the SLURM batch job manager.

### B. HDCI Testbed

An evaluation has been carried on a testbed composed by:

- A Container Orchestration Platform managed by OneDock [14]. OneDock introduces support for OpenNebula to create Docker containers as if they were VMs. The very same OpenNebula APIs and interfaces can be employed since Docker is supported as an additional hypervisor. This way, support for lightweight virtualization by means of containers is introduced while maintaining compatibility with existing applications that require resource provisioning from an IaaS Cloud site.
- An on-premises Cloud managed by OpenNebula, to deploy VMs on a cluster of physical nodes.

### C. Deployment of Application on HDCIs

Figure 3 describes the deployment process of applications on a HDCI. First of all, the Docker images required to support the scientific applications are pulled from Docker Hub and registered in the site's catalog of virtual images. This has a double intention. First of all, OneDock uses the Docker image repository in the front-end of the cluster as a cache so that the Docker containers executed in the working

[5]Non-normative types for INDIGO-DataCloud: https://github.com/indigo-dc/tosca-types/blob/master/custom_types.yaml

[6]The indigo-dc.galaxycloud Ansible Role is available at: https://galaxy.ansible.com/indigo-dc/galaxycloud/

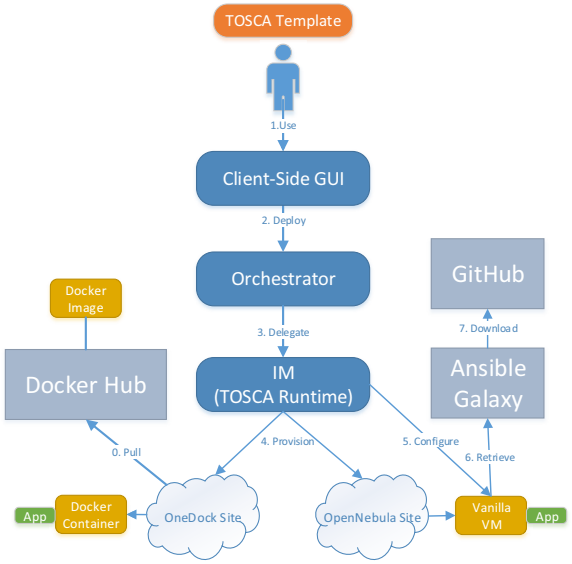


Figure 3. Deployment of applications on HDCIs.

nodes of the Cloud site are rapidly spawned. Also, the site is autonomous to decide which images, and in turn which scientific communities, are supported on that site.

The TOSCA templates are submitted to the Orchestrator which delegates (after choosing the right site) to the Infrastructure Manager (IM) [15] which is an open-source tool to deploy and customise virtual infrastructures on IaaS Clouds. The IM has been extended in INDIGO-DataCloud to support the TOSCA 1.0 Simple Profile in YAML together with the additional non-normative types developed in the context of the project. Therefore, the IM is a TOSCA runtime.

In the case of provisioning resources from an IaaS Cloud site based on VMs, the IM proceeds, as part of the deployment process of the TOSCA template, to contextualize and configure the VM by retrieving the corresponding Ansible Role for the application specified in the TOSCA template from Ansible Galaxy, and in turn downloading the code for the Ansible Role from GitHub, and executing the role inside the VM. Notice that, in the case of VMs, vanilla VMs (e.g. plain Ubuntu 14.04 and plain CentOS 7) have to be available and pre-registered in the site's image repository. Once the execution of the Ansible Role has finished, the IM reports back to the Orchestrator that the infrastructure is ready to be used. The user may access the infrastructure with the information specified in the "outputs" section of the TOSCA template. This is typically the IP of the node, to be accessed via SSH, or the endpoint to be accessed using a web browser.

In the case of provisioning resources from an IaaS site based on OneDock, a Docker container out of the specified image in the TOSCA template is created. In this case, the

configuration step from the IM is analogous as with the VMs (to maintain consistency) but the execution of the Ansible Role is skipped since the application is already available in the container. This speeds up the deployment procedure since no installation occurs at runtime.

### D. Execution Results

The same Galaxy Portal TOSCA template has been submitted to the OpenNebula site providing VMs and to the OneDock testbed. In the first case the average time to get the VM instance fully configured and ready to be accessed by the users was 20:48 (minutes:seconds). In the OneDock case the time needed was 7:25. As expected, the time needed in OneDock is reduced, due to the advantages of the container technology. The time needed to create and boot the instance is reduced from 1:27 in the VM case to 0:20 in case of containers. Also, the time to install the Galaxy portal is reduced, since in the container case it is already installed and the Ansible playbook only checks that the application is installed as expected: from 19:21 to 7:05. It is important to point out that the OneDock testbed is not running directly on bare metal but over VMs, thus introducing an extra overhead and preventing a further reduction in the configuration time. Notice that, with this approach, users can run their applications either on VMs or on Docker containers with the same execution environment across infrastructures.

## IV. Conclusions and Future Work

This paper has described the experience gained when simultaneously harnessing computing platforms based on Virtual Machines and based on Docker containers by means of open standards, open-source tools and publicly available platforms. The adoption of Ansible Roles together with the Automated Build capabilities of Docker Hub has introduced the ability to simultaneously create Docker images to be distributed to container-based infrastructures. At the same time, applications can be automatically deployed via a DevOps approach at runtime on vanilla VMs. Future works involves further extensions to the TOSCA Simple Profile in YAML version 1.0 with the inclusions of new non-normative types for additional support to other scientific communities.

## Acknowledgments

The authors would like to thank the European Commission for the support through the Horizon 2020 INDIGO-DataCloud project under grant agreement RIA 653549 and to the Spanish "Ministerio de Economía y Competitividad" for the project CLUVIEM (TIN2013-44390-R).

## References

[1] L. Ramakrishnan, R. S. Canon, K. Muriki, I. Sakrejda, and N. J. Wright, "Evaluating Interconnect and Virtualization Performance for High Performance Computing," *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, no. 2, p. 55, oct 2012. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2381056.2381071

[2] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, "An updated performance comparison of virtual machines and Linux containers," in *2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, mar 2015, pp. 171–172. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7095802

[3] T. Bui, "Analysis of Docker Security," Tech. Rep., jan 2015. [Online]. Available: http://arxiv.org/abs/1501.02967

[4] G. M. Tihfon, S. Park, J. Kim, and Y.-M. Kim, "An efficient multi-task PaaS cloud infrastructure based on docker and AWS ECS for application deployment," *Cluster Computing*, vol. 19, no. 3, pp. 1585–1597, sep 2016. [Online]. Available: http://link.springer.com/10.1007/s10586-016-0599-0

[5] J. Stubbs, W. Moreira, and R. Dooley, "Distributed Systems of Microservices Using Docker and Serfnode," in *2015 7th International Workshop on Science Gateways*. IEEE, jun 2015, pp. 34–39. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7217926

[6] M. Moca, C. Litan, G. C. Silaghi, and G. Fedak, "Multi-criteria and satisfaction oriented scheduling for hybrid distributed computing infrastructures," *Future Generation Computer Systems*, vol. 55, pp. 428–443, 2016.

[7] B. Fitzgerald, N. Forsgren, K.-J. Stol, J. Humble, and B. Doody, "Infrastructure Is Software Too!" *SSRN Electronic Journal*, 2015. [Online]. Available: http://www.ssrn.com/abstract=2681904

[8] "Ansible." [Online]. Available: https://www.ansible.com/

[9] J. Crandall and P. Lipton, "OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA) TC." [Online]. Available: https://www.oasis-open.org/committees/tc{_}home.php?wg{_}abbrev=tosca

[10] D. Palma, M. Rutkowski, and T. Spatzier, "TOSCA Simple Profile in YAML Version 1.0," Tech. Rep., 2016. [Online]. Available: http://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.0/TOSCA-Simple-Profile-YAML-v1.0.html

[11] "Ansible Galaxy." [Online]. Available: https://galaxy.ansible.com/

[12] "Galaxy Project." [Online]. Available: https://galaxyproject.org

[13] INDIGO-DataCloud, "Design Document and Work Plan for the PaaS Architecture - D5.4," Tech. Rep. [Online]. Available: https://www.indigo-datacloud.eu/documents/design-document-and-work-plan-paas-architecture-d52

[14] "OneDock." [Online]. Available: https://github.com/indigo-dc/onedock

[15] M. Caballer, I. Blanquer, G. Moltó, and C. de Alfonso, "Dynamic management of virtual infrastructures," *Journal of Grid Computing*, vol. 13, no. 1, pp. 53–70, 2015. [Online]. Available: http://link.springer.com/article/10.1007/s10723-014-9296-5