Digital repository as a service: automatic deployment of an Invenio-based repository using TOSCA orchestration and Apache Mesos

Marica Antonacci[1,*], *Alberto* Brigandì[2], *Miguel* Caballer[3], *Eva* Cetinić[4], *Davor* Davidovic[4], *Giacinto* Donvito[1], *Germán* Moltó[3], and *Davide* Salomoni[5]

[1]INFN Bari, Via E. Orabona 4, 70125 Bari, Italy
[2]Santer Reply S.p.A., Via Robert Koch 1/4, 20152 Milano, Italy
[3]Instituto de Instrumentación para Imagen Molecular (I3M). Centro mixto CSIC - Universitat Politècnica de València. Camino de Vera s/n, 46022, Valencia, Spain
[4]Ruđer Bošković Institute, Bijenicka cesta 54, 10000 Zagreb, Croatia
[5]INFN-CNAF, Viale Berti Pichat 6/2, 40127 Bologna, Italy

Abstract. In the framework of the H2020 INDIGO-DataCloud project, we have implemented an advanced solution for the automatic deployment of digital data repositories based on Invenio, the digital library framework developed by CERN. Exploiting cutting-edge technologies, such as Docker and Apache Mesos, and standard specifications to describe application architectures such as TOSCA, we are able to provide a service that simplifies the process of creating and managing repositories of various digital assets using cloud resources. An Invenio-based repository consists of a set of services (e.g. database, message queue, cache, workers and frontend) that need to be properly installed, configured and linked together. These operations, along with the provisioning of the resources and their monitoring and maintenance, can be challenging for individual researchers or small-to-moderate-sized research groups. To this purpose, the INDIGO-DataCloud platform provides advanced features for orchestrating the deployment of complex virtual infrastructures on distributed cloud environments: it is able to provision the required resources automatically over heterogeneous and/or hybrid cloud infrastructures and to configure them automatically ensuring dynamic elasticity and resilience. This approach has been successfully adapted to support the needs of the researchers and scholars in the domain of the Digital Arts and Humanities.

1 Introduction

The Arts and Humanities have seen an exponential growth in digital research materials, especially in the last decade, as a result of new, born-digital materials and large digitization efforts. Because humanities disciplines nowadays generate and analyse an increasing amount of data, parts of their research process become more and more data-intensive and have to be supported by emerging research infrastructures. Moreover, the long-term accessibility and public access to research data have been given increasing importance in order to allow

*e-mail: marica.antonacci@ba.infn.it

the sharing and re-use of results from different research groups. While other research areas, particularly Natural Sciences, already made extensive use of e-Infrastructure services, the Arts and Humanities are not yet exploiting these facilities at the same level. Considering the insufficient experience with the use of e-Infrastructure resources among researchers in this area, our work addresses the needs of those individual researchers and small research groups that find it difficult to setup and manage a digital repository. This requires particular skills and includes activities such as installing and configuring software components, operating and maintaining services, as well as providing and managing resources and eventually scaling the resources depending on the number of users and the generated data.

The European H2020 project INDIGO-DataCloud [1] provides a set of open-source tools and solutions that allow building services that exploit cloud resources in heterogeneous and hybrid environments. Starting from these tools we have implemented a service that simplifies the process of creating and managing data repositories using cloud resources. Compared to the installation, configuration and management of the repository on a local infrastructure, the INDIGO solution simplifies the process of repository creation and configuration on multiple cloud sites, thus making it more robust for different cloud vendors. In addition, it provides an easy management and dynamic scaling of the resources (compute and storage capacities) making it very easy to operate from the system administration point of view. In our solution, we deploy repositories based on Invenio [2]. Invenio is a mature, open-source, repository framework and is well-known in digital arts and humanities communities, thus the learning curve for using the service would be greatly decreased. Furthermore, Invenio provides a user-friendly graphical interface, as well as user and collection management system, pre-defined metadata sets for different records types (video, document, photos, etc.) and supports DOI (Digital Object Identifier) integration.

2 Digital Repository as a Service

The implemented system is sketched in Fig. 1: a user can request the deployment of an Invenio-based repository using a template document written in the TOSCA [3] language. TOSCA (Topology and Orchestration Specification for Cloud Applications) is an open language to describe the components, together with their relationships, of an application architecture to be deployed in a Cloud. The user submits the TOSCA template to the INDIGO PaaS Orchestrator [4]: this service allows to coordinate the provisioning of cloud resources and the deployment of virtual infrastructures on heterogeneous cloud environments such as private clouds (OpenStack, OpenNebula) and public clouds (Amazon Web Services, Microsoft Azure). The selection of the cloud providers is carried out by the Orchestrator on the basis of some criteria such as the user Service Level Agreements (SLA), the availability and performance of the services offered by the cloud providers or the data location. Once the deployment of the repository is completed, the user can access an administrative dashboard for monitoring, configuring or scaling the service instances. The repository endpoint is provided and can be accessed by the members of the user community.

Under the hood the submission of the TOSCA template triggers a complex workflow that creates the compute and storage resources in the selected cloud site, installs and configures an Apache Mesos [5, 6] cluster on top of those virtual machines, deploys the different components of the Invenio repository as Docker containers running on top of the Mesos cluster.

2.1 Managed services through Marathon and Apache Mesos

The Invenio-based repository consists of a set of components: the load-balancer (based on HAProxy), the web frontend, the web backend, the task queue (Celery) workers, the cache

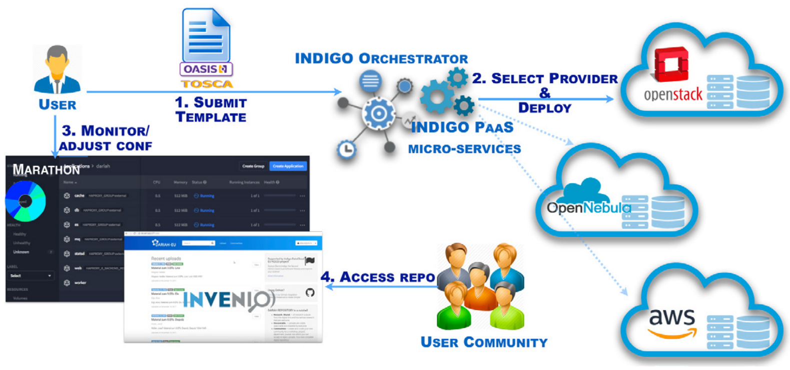


Figure 1. Digital Repository as a Service. The system allows to automate the provisioning of the cloud resources and the installation and configuration of the Invenio-based repository. The user submits the deployment request, described in the TOSCA templating language, to the INDIGO PaaS Orchestrator that selects the best cloud provider and coordinates the provisioning and configuration of the needed resources. At the end of this complex deployment workflow, the user gets the endpoint to access the repository and the administrative dashboard.

(redis), the message queue (rabbitmq), the database, and the search engine (elasticsearch). Docker images for these services are publicly available and can be used to ease the deployment of the data repository and to start a development instance in case the user wants to customize and/or extend the base service. Docker containers simplify the process of developing, deploying and running services using a DevOps approach. Indeed we have used containers both for the testing phase and the production environment, but with two different deployment models. The testing environment (Fig. 2) was setup using a single virtual machine running all the components via *docker-compose* [7], a tool for defining and running complex applications with Docker in the scope of a single host, whereas in the production environment (Fig. 3), the Invenio components were automatically deployed using our system. The Invenio components are executed as long-running services on top of an Apache Mesos cluster through its framework, Marathon [8], to ensure fault tolerance and scalability.

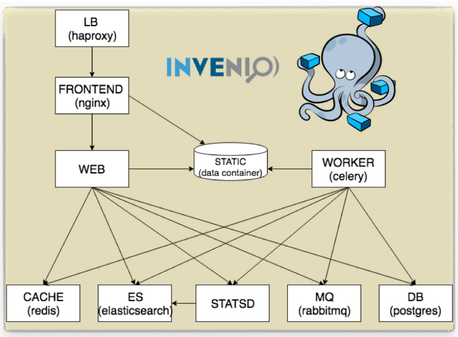


Figure 2. Test environment: single machine deployment using docker-compose. All the components of the Invenio-based repository are run as Docker containers on the same machine. This setup is useful for full-stack testing in a simple and repeatable environment.

Apache Mesos [5][6] is a cluster manager that provides efficient resource usage, fault tolerance, high-availability and scalability. Marathon [8], a container orchestration framework that runs on top of Mesos, ensures that the dockerized services are always up and running. We have developed a set of Ansible [9] roles that allow the automatic deployment of a production-ready Mesos cluster consisting of a set of master nodes (the leader election is managed by ZooKeeper [10]) that coordinate the work of a set of slave nodes that are in charge of executing the tasks. The cluster includes also two edge nodes configured as cluster load-balancers used to access the services deployed on the cluster from the external network. The failover of the load balancer is ensured through Keepalived [11] installed on both nodes. The service discovery is implemented through Consul [12] that provides also DNS functionalities. An important feature added in the context of the INDIGO project is the cluster elasticity: the INDIGO plugin for CLUES [13] monitors the workload on the Mesos cluster and is able to add or remove slave nodes depending on the needed resources. Another essential aspect that we have addressed is the possibility to provide persistent storage to the deployed Docker containers: since data stored in the Docker containers are volatile, stateful applications need to store data on persistent volumes. We have tested and integrated two different solutions: 1) installing and configuring a distributed file system (NFS) among the cluster nodes and binding local persistent volumes to the Marathon applications; 2) using an external storage service such as OpenStack Cinder [14] and the Docker volume driver REX-Ray [15]. Although the second solution can profit from the reliability and fault-tolerance of the external storage service, the first approach is more general since it can be applied also in those sites that do not provide one of the storage backends supported by the volume driver; moreover, the sharing of the volumes among different Docker containers is not fully supported by some storage services (e.g. OpenStack Cinder) whereas it can be easily enabled through the NFS.

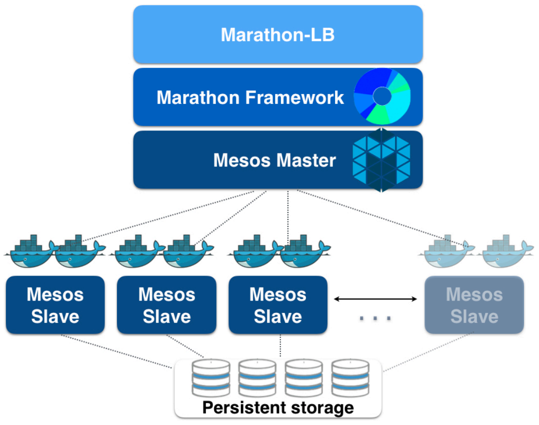
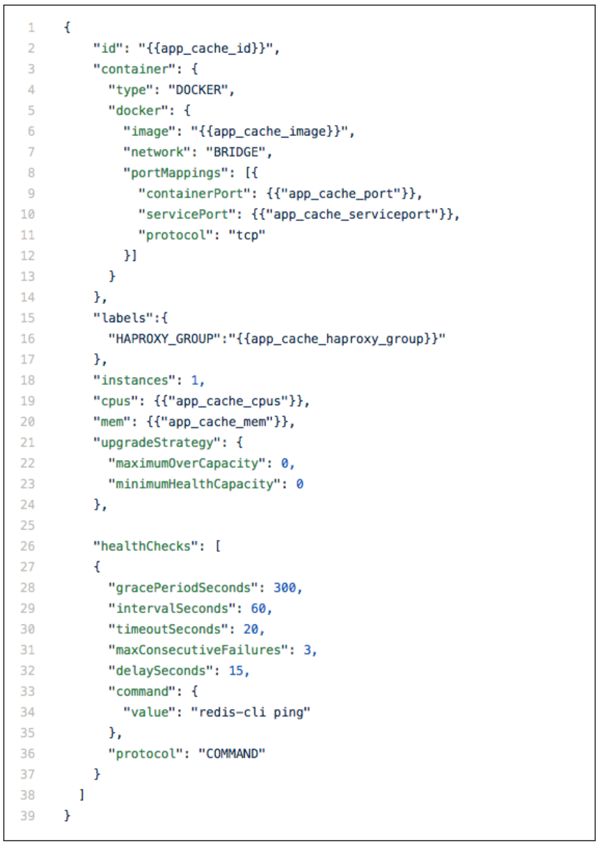


Figure 3. Production environment: distributed deployment using an Apache Mesos cluster. The different components of the digital repository (the web application, the database, the message queue, etc.) are deployed as Marathon long-running services.

The Invenio services are automatically deployed on the Mesos cluster through REST APIs calls to the Marathon endpoint. The request body is built starting from a template JSON file included in the Ansible role: for example, the JSON template used for starting the cache application on Marathon is shown in Fig. 4.

We have exploited the health checks in order to monitor the status of the deployed services and restart them automatically in case of failure. The label HAPROXY_GROUP is used



```
{
    "id": "{{app_cache_id}}",
    "container": {
      "type": "DOCKER",
      "docker": {
        "image": "{{app_cache_image}}",
        "network": "BRIDGE",
        "portMappings": [{
          "containerPort": {{"app_cache_port"}},
          "servicePort": {{"app_cache_serviceport"}},
          "protocol": "tcp"
        }]
      }
    },
    "labels":{
      "HAPROXY_GROUP":"{{app_cache_haproxy_group}}"
    },
    "instances": 1,
    "cpus": {{"app_cache_cpus"}},
    "mem": {{"app_cache_mem"}},
    "upgradeStrategy": {
      "maximumOverCapacity": 0,
      "minimumHealthCapacity": 0
    },

    "healthChecks": [
    {
      "gracePeriodSeconds": 300,
      "intervalSeconds": 60,
      "timeoutSeconds": 20,
      "maxConsecutiveFailures": 3,
      "delaySeconds": 15,
      "command": {
        "value": "redis-cli ping"
      },
      "protocol": "COMMAND"
    }
  ]
}
```

Figure 4. JSON application definition file used to deploy the cache (redis) Docker container. It is stored as a jinja2 template in the Ansible role that we have developed to automate the deployment of the Invenio service on Marathon.

to expose the service on the cluster load-balancer that is based on the Marathon-LB [16] service, another component from the Mesos ecosystem, based on HAProxy [17]. Marathon-LB subscribes to the Marathon's event bus and updates the HAProxy configuration in real time. When an application starts, stops, relocates or has any change in health status, Marathon-LB will automatically regenerate the HAProxy configuration and reload HAProxy service.

2.2 INDIGO-DataCloud Platform as a Service (PaaS)

The INDIGO Platform as a Service (PaaS) [1] allows the users to deploy virtual infrastructures with complex topologies (such as clusters of virtual machines or dockerized applications) using a standardized interface based on the TOSCA specification. The adoption of the TOSCA standard ensures the portability of the deployment topology description across different cloud providers. The core component of the PaaS layer is the Orchestrator: it collects high-level deployment requests and translates them into actions to coordinate resources interacting with the underlying cloud infrastructures. The PaaS layer features advanced federation and scheduling capabilities ensuring the transparent access to different cloud environments, both the “traditional” cloud management frameworks, such as OpenStack, OpenNebula, AWS and Azure, and container orchestration platforms, for example Apache Mesos. The selection of the best cloud provider to fulfill the user request is performed by the PaaS Orchestrator taking into account criteria such as the user's SLAs, the services availability and the data location.

The cloud resources are provisioned automatically through the INDIGO Infrastructure Manager (IM) [18] that is able to orchestrate different cloud management frameworks (OpenStack, OpenNebula, AWS, Azure, etc.). The IM is in charge of creating all the needed resources (VMs, storage volumes, networks/security groups, etc.) contacting directly with the provider selected by the PaaS Orchestrator. Once deployed, the virtual machines are automatically configured running dedicated Ansible Roles publicly shared on Ansible Galaxy (*indigo-dc* namespace [19]). The IM gets VMs IP information, and uses cloud provider contextualization tools (cloud-init [20] in most of the cases) to create an admin user to access the VMs via SSH. Then, it configures Ansible in the deployed VMs, enabling the execution of the artifacts specified in the TOSCA template. Those artifacts install and configure all the components needed to deploy the user's application (Docker, Mesos, Marathon, etc.). Once configured, the TOSCA outputs are returned to the PaaS orchestrator that shows them to the user.

2.3 TOSCA description of the digital repository deployment

An example of TOSCA template that can be used to deploy the Invenio-based repository is provided in the tosca-templates GitHub repository [21]. The TOSCA template describes the entire repository stack: from the infrastructure to be provisioned, through the middleware tier, up to the application level. The description of the infrastructure tier includes the definition of the hosts, networks and storage volumes; the middleware tier defines the software components that need to be installed on each node in order to build up the Mesos cluster. The hosts are divided in three main categories: master nodes, slave nodes and load-balancers, as foreseen by the Mesos cluster architecture. The last tier, the application level, defines the services that will be deployed on the Mesos cluster to setup the data repository. The logical diagram in Fig. 5 shows the relationships and dependencies among the application (tosca-type *tosca.nodes.indigo.DariahRepository*) and the middleware components (tosca-type *tosca.nodes.indigo.MesosMaster*); the lifecycle of the application is managed through the Ansible role that receives the values for the input variables from the template itself.

Indeed, the template provides many input parameters (default values are provided as well) so that the users can easily customize their digital repository instance. For example, one can change the Docker images used to run the different services and the resources (memory, cores, storage size) assigned to each service.

3 The DARIAH use-case

A typical use-case coming from the DARIAH (Digital Research Infrastructure for the Arts and Humanities) involves a small research group or a project generating a number of records (digital objects) which they want to store and make available to a wider audience. In this case, a DARIAH member (project/institute manager, research group leader, or individual researcher) can access the INDIGO's Digital Repository as a Service through a user-friendly web-portal and makes an initial request to create a new repository. Subsequently, this user is referred to as "Repository Manager". Upon the successful authorization of the user (by the DARIAH IdP), a new, empty repository instance is launched. Once the requested repository is up and running the repository manager defines the repository, collections, items and fills it in with data. These actions usually include: providing additional repository information,

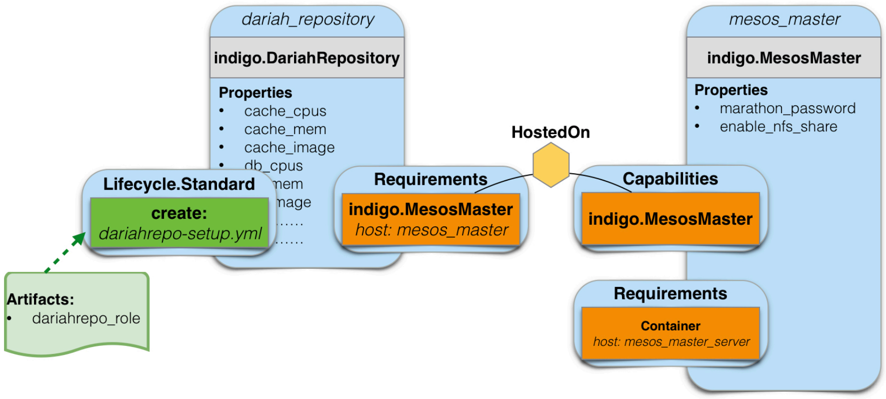


Figure 5. TOSCA template logical diagram: the relationships and dependencies among the application (tosca-type *tosca.nodes.indigo.DariahRepository*) and the middleware components (tosca-type *tosca.nodes.indigo.MesosMaster*) are shown. For the sake of simplicity, the components for the Mesos slave and load-balancer nodes have been omitted.

defining repository/collection items' metadata, creating new collections, upload data, upload existing collections from other/local repositories, defining the lifetime of the repository, storage properties, etc. In addition, the manager can set up visibility properties, invite new users and grant permissions to access, browse, search, download and/or upload records. Once the repository is setup up and filled, the end-user can access, browse, search, upload and update the records of the new repository, depending on their given access rights.

4 Conclusions and future work

In this paper we described the implementation of the Digital Repository as a Service application based on the solutions developed in the INDIGO-DataCloud project. Our efforts were directed at providing a service that simplifies the process of creating and managing repositories of various digital assets using cloud resources. The automatic deployment of the Invenio-based data repository includes the provisioning of the needed compute and storage resources, the installation and configuration of a highly-available Mesos cluster on top of which the repository services are deployed as dockerized long-running services. In this way, the management of the services is delegated to the Mesos framework, Marathon, that automatically monitors the health of the applications and takes care of restarting the containers in case of failures. A demonstrator has been developed for supporting the Arts and Humanities Research use-case and is now being extended for the DARIAH Thematic Service [22] in the EOSC-HUB project. Future work includes the integration with advanced data management solutions, for example Onedata [23], from the INDIGO project, and the EUDAT [24] storage services, in order to provide further functionalities such as metadata management, transparent data transfer, archive and replication.

Acknowledgments

The authors would like to acknowledge the support of the INDIGO-DataCloud project, funded by the European Commission's Horizon 2020 Research and Innovation Programme (grant agreement RIA 653549).

References

[1] Salomoni D. et al., INDIGO-DataCloud: a platform to facilitate seamless access to e-infrastructures, Journal of Grid Computing **16**, 381-408 (2018). https://doi.org/10.1007/s10723-018-9453-3

[2] Invenio, https://invenio-software.org

[3] TOSCA Simple Profile in YAML Version 1.0. Edited by Derek Palma, Matt Rutkowski, and Thomas Spatzier. 21 December 2016. OASIS Standard. http://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.0/os/TOSCA-Simple-Profile-YAML-v1.0-os.html. Latest version: http://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.0/TOSCA-Simple-Profile-YAML-v1.0.html

[4] INDIGO-DataCloud PaaS Orchestrator [software]. Version 2.1.1. Available from https://github.com/indigo-dc/orchestrator/releases/tag/v2.1.1-FINAL [accessed 2019-02-11]

[5] Hindman, B. et al. Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center. In NSDI, Vol. 11, No. 2011, pp. 22-22.

[6] Apache Mesos, Apache Software Foundation, http://mesos.apache.org/ (accessedOct2018)

[7] Docker compose. https://docs.docker.com/compose/

[8] Marathon, A container orchestration platform for Mesos and DC/OS. https://mesosphere.github.io/marathon/

[9] Ansible documentation. https://docs.ansible.com/ansible/latest/index.html

[10] Apache ZooKeeper. https://zookeeper.apache.org/

[11] Keepalived, Load balancing and High-Availability. http://www.keepalived.org/

[12] Consul by Hashicorp. https://www.consul.io/

[13] C. De Alfonso, M. Caballer, F. Alvarruiz, V. Hernández "An energy management system for cluster infrastructures". Computers & Electrical Engineering, Volume 39, Issue 8, Pages 2579-2590, 2013

[14] Cinder, the OpenStack Block Storage Service. https://docs.openstack.org/cinder

[15] REX-Ray. https://rexray.readthedocs.io

[16] Marathon-LB. https://github.com/mesosphere/marathon-lb

[17] HAProxy, The Reliable, High Performance TCP/HTTP Load Balancer. http://www.haproxy.org/

[18] Miguel Caballer, Ignacio Blanquer, Germán Moltó, and Carlos de Alfonso. "Dynamic management of virtual infrastructures". Journal of Grid Computing, Volume 13, Issue 1, Pages 53-70, 2015, ISSN 1570-7873, DOI: 10.1007/s10723-014-9296-5.

[19] Ansible roles developed by INDIGO project. Available from https://galaxy.ansible.com/indigo-dc [accessed 2019-02-11]

[20] cloud-init, the standard for customising cloud instances. https://cloudinit.readthedocs.io

[21] TOSCA templates developed by INDIGO project. Version 3.0.0. Available from https://github.com/indigo-dc/tosca-templates/releases/tag/v3.0.0 [accessed 2019-02-11]

[22] EOSC-HUB - DARIAH Science Gateway. https://eosc-hub.eu/catalogue/DARIAH%20Science%20Gateway

[23] Onedata, global data access solution for science. https://onedata.org/#/home

[24] EUDAT service catalogue. https://www.eudat.eu/catalogue