Using continuous integration principles for managing virtual laboratories

Dejan Stamenov, Dimitar Venov, Goran Petkovski, Boro Jakimovski, and Goran Velinov

Faculty of Computer Science and Engineering, Skopje, Republic of North Macedonia, Ss. Cyril and Methodius University
stamenov.dejan@outlook.com, dimitar.venov@students.finki.ukim.mk, {goran.petkovski,boro.jakimovski,goran.velinov}@finki.ukim.mk

Abstract. Researchers around the world are involved in the process of managing and processing large volumes of data while using a plethora of different tools. This increases the need for virtual laboratories that will allow scientists to create an environment for rapid data processing and analyses, using the latest available technology without spending too much time on configuration. In this paper, we present the Virtual Laboratory System (vLab) for managing laboratories on the cloud. The system is based on the service-oriented architecture, built on top of Spring Framework and hosted on the EGI Federated Cloud (FedCloud). It provides an interface for managing clusters of virtual machines that are created by the service, providing tools for data processing and analysis, along with the required datasets for these operations. The architecture of the system and the tools used to achieve successful management of virtual laboratories are also presented with the details of the cloud architecture where the vLab system is hosted.

Keywords: FedCloud · Virtual laboratory · Virtual organization · Web service.

1 Introduction

The data scientists around the world come from established fields such as statistics, engineering, physics, machine learning, business intelligence, neuroscience and more, having special competences (skills and knowledge) that enables them to extract value from digital data. The scientific communities are involved in creating and maintaining data products, in which the end users contribute towards improving the product. Users are increasingly interacting with real-time data and services, thus generating lots of data. The science implies a holistic approach and uses scientific methods to extract value from data. This raises the need for virtual laboratories that will allow scientists to create an environment for rapid data processing and analyzes, using the latest available technology.

In this paper we present our Virtual Laboratory System (vLab), a service-oriented architecture hosted on the EGI Federated Cloud (FedCloud) [3] that

provides a service for creating and managing virtual laboratories. OpenNebula is used as a cloud computing platform on the data center in our solution, implementing the Open Cloud Computing Interface (OCCI) for managing and monitoring cloud resources. Our system features an Application Programming Interface (API), which provides an automation in the process of managing computational resources to the end users. Up until this point, it was required for a cloud administrator to manually set up the cluster of virtual machines, and then apply proper configuration before it can be used by researches. With this service, we provide the possibility for each data scientist to create clusters of virtual machines in the cloud without need for a dedicated cloud administrator. The clusters of virtual machines that are created by the service provide tools for data processing and analyses, along with the required datasets. The virtual machines in the cluster are automatically provisioned and configured using Puppet [7]. The Hadoop [1] open-source software framework is used in the virtual machines for distributed storage and processing of big data.

The paper structure is as follows. Related work is presented in Section 2. The FedCloud structure is provided in Section 3, while the implementation of the vLab service is explained in detail in Section 4. The Section 5 describes the process of automating the deployment of Hadoop cluster using Puppet. Our use case application is provided in Section 6, while concluding the paper contribution and presenting the future work in Section 7.

2 Related work

Atlas Tier 3 Cluster Tool (at3c) consists of remote repository for checking cluster parameters and configuration of hardware environment based on real-world use scenarios. Main advantage is quick recovery from undesirable situations (failed hardware, shortage of sustainable environment, etc.) [4], as opposed to our system which provides automatic configuration only of Hadoop environments.

Another collaborative tool for building common e-Infrastructures is CHAIN-REDS project. Their existing grid infrastructure contains materials from researches around the world, with mission to provide computational resources to scientists and organizations. Its integration with EGI (European Grid Infrastructure) and FedCloud has managed to produce functional solution for storing cloud images and templates. Our service-oriented architecture shares similarities with this solution, having CHAIN-REDS with one advantage: creation and management of mini-clouds in different VRC (Virtual Research Communities) from all continents in the world [11], with difference to our system which give access only to EGI's infrastructure.

3 FedCloud structure

EGI Federated Cloud represents heterogeneous research community with aim to provide distributed collection of computational resources. Integrates public and community clouds into a scalable computing platform for data and/or compute

driven applications and services. Each cloud can provide Infrastructure as a Service (IaaS) features for users, i.e. CPU/GPU computing, storage and network configuration. There are Virtual Organizations (VOs) formed within the EGI Federated Cloud. Each VO of the cloud federation is a resource allocation for a specific community or purpose. The cloud providers who support a given community join the VO dedicated to that community. Users and their applications can consume cloud resources from the VO after becoming members of the same VO. X.509 certificates are used to control the access to the cloud resources.

4 vLab service

The vLab service is the main part of the cloud architecture which provides computational services to the end users. It provides services for creation of computational resources[1], along with services to remove these resources and listings of available computational resources for each end user. Each computational resource is created based on a configuration, which includes different specifications for the virtual machines that are being created (in terms of computational power, storage and memory), along with the number of virtual machines being created (these virtual machines form independent cluster) and required configuration files by which the virtual machines in the cluster will be configured to be aware of the existence of other machines in the cluster itself.

4.1 Technical overview

The API (Application Program Interface) is built with Java technology and Spring Framework [12], using the REST architecture (used over HTTP protocol). vLab system API does not directly communicate with the cloud; an intermediate framework jOCCI [5] is used, which provides the functionality of accessing different instances of the cloud when doing computational services on user's request. In addition to this, PostgreSQL [6] is used as a main database of the API, storing virtual machine configurations, user's tokens and the computational resources that are created for each token. The token represents unique set of characters for each user of the cloud, which is sent to the API when requesting a service by the end users. Because every virtual machine created by the API needs additional configuration to become part of a cluster, the Git repository management system - GitLab is used to publish configuration data. This repository is managed by the API itself; when new computational resources are created, configuration files are published in the repository. When resources are removed from the cloud, the configuration data for these resources is removed from the repository. Puppet is used as an orchestrational tool for automatic deployment and configuration of parameters in Hadoop cluster. Description is stored in Git repository, so when users apply the puppet template it's automatically pulled from the repository

[1] "Computational resources" or "virtual machines", both terms have the same meaning further in the paper.

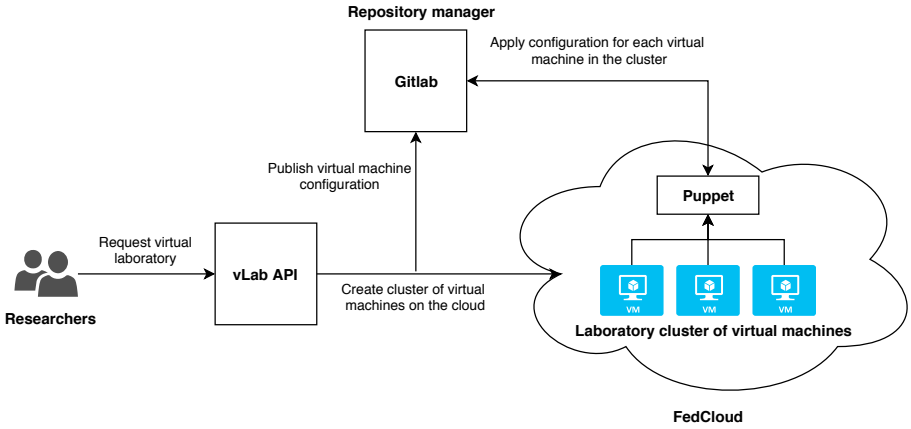

Fig. 1. vLab system - architectural overview.

and provision into FedCloud. In this way, users from different background can create custom-related scenarios for using cluster management tool (in our paper is described Hadoop cluster) with management of related git repository. The FedCloud VMs are developed on Open Nebula Cloud infrastructure. On Fig. 1 is given the architectural overview.

4.2 API interface

The vLab system API provides multiple interface calls to meet the requirements for management of computational resources on the cloud.

The most important API call is the one that creates clusters of virtual machines on the cloud - **POST: resource/createresource**. This API call has additional required parameters:

- **userToken** - unique set of characters for each user of the cloud, used to identify the user,
- **configuration** - computational specification for each virtual machine that is being created as part of a cluster (in terms of computational power, storage and memory), and
- **key** - SSH key which will be used by the user to connect to the virtual machine.

After the virtual machines are created, the API call - **GET: resource/listresources** can be used to retrieve the IP addresses of the computational resources that were created. This API call has the following required parameters:

- **userToken** - used to identify which virtual machines are created for the given user token, and then returns the IP addresses of these computational resources, and

– **configuration** - one user may have different number of virtual machines created on the cloud. Each virtual machine is created based on a given configuration; this parameter is used to filter the machines which IP addresses will be returned by the API call.

When the computational resources created by the user are not needed, the API call - **DELETE: resource/deleteresource** will delete all the resources, based on the required parameter: **userToken**.

4.3 PostgreSQL database design

PostgreSQL is used as the main database of the API, storing configuration data about the clusters of virtual machines and also, keeping track of the user's cloud resources based on their tokens. On Fig. 2 is presented an overview of the database design.

The *VM_CONFIGURATION* table is used to store data about different cluster setups. For example, here we keep data on how we need to setup all the virtual machines in a given cluster when the configuration of these virtual machines is based on Hadoop. This allows us to define different setups in the future, so that the API will be able to provision virtual machines in the cluster based on the provided data in the table.

The *USER_TOKENS* table is used to store each token that is sent by the user, while also tracking down the number of the resources that are being used by the user. The *RESOURCE_DETAILS* table references the *USER_TOKENS* table, while also storing data about the virtual machines that are created for the user. This is done to lower the load of the cloud; if we don't store this data in the database and then retrieve it when needed, we would have to query the cloud for each user to get the details about each virtual machine based on user's request. The integrity of this data is maintained based on each action of the user, meaning that when the user removes the cluster or creates a new one, all the data stored in the database is also removed or new data is added.

5 Hadoop deployment with Puppet

In this section we describe the process of automating the deployment of Hadoop cluster using the Puppet configuration management tool. First, in Subsection 5.1 we give short introduction of Hadoop, then in Subsection 5.2 we give short introduction of Puppet and finally, in Subsection 5.3 we provide the details of the deployment process.

5.1 Hadoop

Hadoop [1] is an open source software framework for large scale distributed data storage and data analyses [13]. The framework is designed to be able to scale from single machine to thousands of machines, where each machine offers local

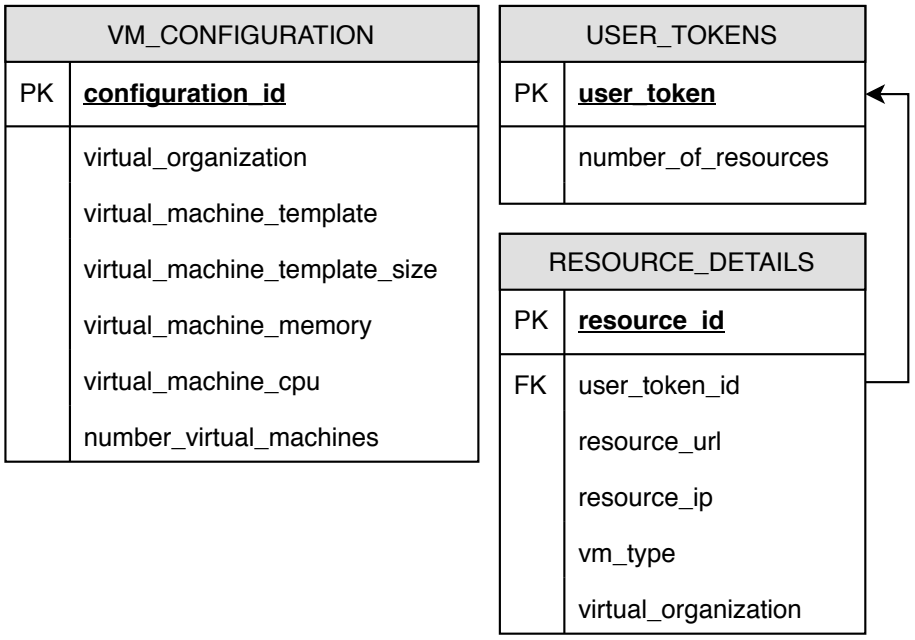

Fig. 2. vLab API database design.

storage and computation. Hadoop follows the master-slave architecture, where a cluster is composed of one NameNode (the master) and one or more DataNodes (the slaves). The NameNode manages the file system namespace and regulates access to files by clients, and the DataNodes manage storage and computation.

The main goals of Hadoop are quick detection of faults and automatic recovery from them, high throughput of data access rather than low latency of data access, support for large files in the order of terabytes and support for tens of millions of files in a single instance [2].

The framework includes the following modules:

- **Hadoop Common** - Common utilities that support the other modules,
- **Hadoop Distributed File System (HDFS)** - A distributed file system that provides high-throughput access,
- **Hadoop YARN** - A framework for job scheduling and cluster resource management, and
- **Hadoop MapReduce** - Implementation of the MapReduce programming model for processing large data sets.

5.2 Puppet

Puppet [7] is a configuration management tool which is written in the Ruby [10] programming language and allows us to control and automate the configuration of all elements comprising an IT system, such as hosts, installed software,

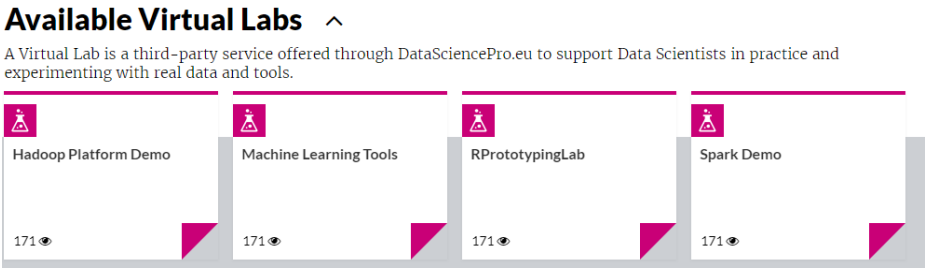

Fig. 3. vLab system - web interface overview.

users, running services, configuration files, scheduled tasks, storage, monitoring and security. Through Puppet, we can describe the system desired end state either using Ruby or a Ruby domain specific language (DSL). Puppet based infrastructure generally consists of a master server with agents installed on each client/node (master-slave architecture).

5.3 Automatic Deployment of Hadoop With Puppet

Deploying Hadoop consists of unpacking the desired version of Hadoop on all of the machines in the cluster in the desired place. Then, setting up the configuration files to include the addresses of the NameNode (the master), and the addresses of the DataNodes (the slaves), as well as some additional configuration parameters. Then, we need to enable communication without password between the NameNode and all of the DataNodes, which include a creation of a special user for Hadoop. Finally, we need to format HDFS and start Hadoop from the NameNode.

To be able to automate this, we have developed a Puppet module which is composed of multiple Puppet classes that are supposed to deploy Hadoop on all of the machines in the cluster. One of the challenges that we faced was synchronization between all of the Puppet agents. In order to start Hadoop, we have to first deploy Hadoop on all of the nodes, and then start it. But, because Puppet catalogs are executed on Puppet's agents independently, the NameNode Puppet agent doesn't know whether Hadoop has been deployed on all of the other servers. To achieve this synchronization, we have used PuppetDB [9] with Facter [8], and through PuppetDB they are accessible to all of the Puppet agents. The NameNode Puppet agent is reading these facts and as soon as all of the DataNodes are set up, it continues with its execution, which includes formating HDFS and starting the cluster.

6 Use case application

As part of developing and contribution to the EGI project, we have developed web application which allows the data scientists to create virtual laboratory

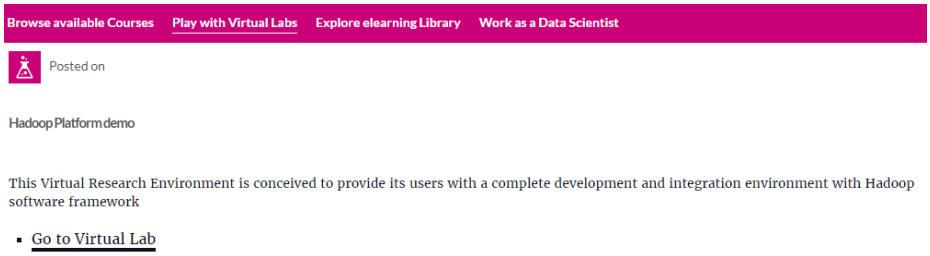

Fig. 4. vLab system - available clusters.

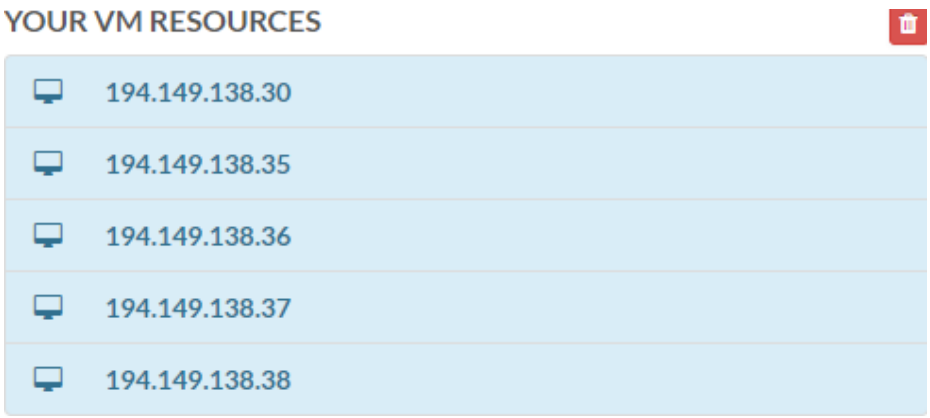

Fig. 5. vLab system - IP addresses from the machines in the cluster.

environments using Hadoop, while effectively managing its instances. Users are authenticated with x509 certificates that are used in EGI services.

On Fig. 3 is displayed the default interface of the application. On Fig. 4 is given the option for selection of virtual laboratories provided by the web application. When the type of virtual laboratory is chosen, the application will ask for the public SSH key which will be used for signing in the created virtual machines. The users are able to choose the desired scale of the cluster in the cloud. Finally, after successful provision of machines, the web application is providing the IP addresses of every instance part of the created cluster, as shown on Fig. 5. These IP addresses are public, which means the users can sign in from every device which has Internet connectivity. After finishing with their work, the cluser can be shut down with graceful poweroff signal sent to the interface of OpenNebula. Then, the virtual machines are terminated and get deleted from the list of available machines.

7 Conclusion and future work

In this paper we presented our service-oriented architecture for managing virtual laboratories - Virtual Laboratory System (vLab), hosted on the EGI Federated Cloud (FedCloud). The virtual machines created by the API provide tools for data processing and analysis. The details of the API were provided, along with the Puppet configuration, which provides automation in the process of deployment of the Hadoop machines. The EGI Federated Cloud structure was also provided.

In future we are planning to conduct researches where we would test if the Virtual Laboratory System (vLab) is capable of being deployed on other cloud providers like Google Cloud Platform, Elastic Compute Cloud and Microsoft Azure, and if it is not, what changes would be required to make the system cross-platform. Also, we are planning to extend our service-oriented architecture to include other laboratories, for example: Apache Spark, and to provide more configuration options for all of the existing laboratories.

References

1. Apache: Hadoop documentation (2019), http://hadoop.apache.org/docs/current/
2. Borthakur, D.: The hadoop distributed file system: Architecture and design (2008)
3. Foundation, T.E.: The egi federated cloud (2019), https://www.egi.eu/federation/egi-federated-cloud/
4. Hendrix, V., Benjamin, D., Yao, Y.: Scientific cluster deployment and recovery–using puppet to simplify cluster management. In: Journal of Physics: Conference Series. vol. 396, p. 042027. IOP Publishing (2012)
5. Kimle, M.: jocci api client library (2019), https://github.com/Misenko/jOCCI-api
6. PostgreSQL: Postgresql documentation (2019), https://www.postgresql.org/docs/
7. Puppet: Puppet documentation (2019), https://puppet.com/docs/
8. Puppet: Puppet facter documentation (2019), https://docs.puppet.com/facter/
9. Puppet: Puppetdb documentation (2019), https://docs.puppet.com/puppetdb/
10. Ruby: Ruby documentation (2019), https://www.ruby-lang.org/en/documentation/
11. Ruggieri, F., Andronico, G., Barbera, R., Matyska, L.: Introduction to the chain-reds project (objectives and achievements). In: e-Infrastructures for e-Sciences 2013 A CHAIN-REDS Workshop organised under the aegis of the European Commission. vol. 199, p. 001. SISSA Medialab (2014)
12. Software, P.: Spring framework (2019), https://spring.io/
13. Taylor, R.C.: An overview of the hadoop/mapreduce/hbase framework and its current applications in bioinformatics. In: BMC bioinformatics. vol. 11, p. S1. BioMed Central (2010)