Cloud Environment Automation: from infrastructure deployment to application monitoring

IOP Conf. Series: Journal of Physics: Conf. Series **898** (2017) 082016 doi:10.1088/1742-6596/898/8/082016

Cloud Environment Automation: from infrastructure deployment to application monitoring

C. Aiftimiei[1,2], A. Costantini[1], R. Bucchi[1], A. Italiano[3], D. Michelotto[1], M. Panella[1], M. Pergolesi[4], M. Saletta[5], S. Traldi[6], C. Vistoli[1], G. Zizzi[1] and D. Salomoni[1]

[1]INFN CNAF, Bologna, Italy
[2]IFIN - "Horia Hulubei", Bucharest - Magurele, Romania
[3]INFN Bari, Bari, Italy
[4]INFN Perugia, Perugia, Italy
[5]INFN Torino, Torino, Italy
[6]INFN Padova, Padova, Italy

E-mail: cristina.aiftimiei@cnaf.infn.it, alessandro.costantini@cnaf.infn.it

Abstract. The potential offered by the cloud paradigm is often limited by technical issues, rules and regulations. In particular, the activities related to the design and deployment of the Infrastructure as a Service (IaaS) cloud layer can be difficult to apply and time-consuming for the infrastructure maintainers. In this paper the research activity, carried out during the Open City Platform (OCP) research project [1], aimed at designing and developing an automatic tool for cloud-based IaaS deployment is presented. Open City Platform is an industrial research project funded by the Italian Ministry of University and Research (MIUR), started in 2014. It intends to research, develop and test new technological solutions open, interoperable and usable on-demand in the field of Cloud Computing, along with new sustainable organizational models that can be deployed for and adopted by the Public Administrations (PA). The presented work and the related outcomes are aimed at simplifying the deployment and maintenance of a complete IaaS cloud-based infrastructure.

1. Introduction

Cloud computing is a way to provide and enable the use of distributed computing, storage resources and services that have been developed, thoroughly tested and adopted by industry, science and government. For these actors, however, the acceptance of the models offered by the Cloud, even if they represents a strong opportunity, is often limited both by technical-scientific issues and by rules and regulations that impose specific behaviurs to protect providers and consumers of public services (citizen or company).

On such premises, the Open City Platform (OCP) project [1] intends to research, develop and test new technological solutions that are open, interoperable and usable on-demand on the Cloud, as well as innovative organizational models that can be sustainable over time. The aim of the project is to innovate, with scientific results and new standards, the delivery of services by Local Government Administrations (LGA) and Regional Administrations to citizens, Companies and other Public Administrations (PA). Therefore the main areas of research conducted by the OCP are related to:

(i) scientific and technological challenges:
- Federated management of heterogeneous cloud platforms
- Integrated monitoring and support to billing systems
- Design and reengineering of Cloud applications
- Disaster recovery as a Service
- Integration of PaaS components in particular PaaS for eGov
- Open data and the Open Service and integration into business models
- Federated identity management and its trust relationship

(ii) legal, organizational, functional challenges and new business models that are necessary to ensure a concrete feasibility within government scope of the results achieved by the project:
- Define new organizational models and public governance where regions have the role of infrastructure intermediaries
- Decouple the exercise of administrative functions (public role) from the Information and Communications Technology (ICT) instrument (private role) on which the function is performed
- Adherence to new models regulations
- Accountability and attribution of precise responsibilities ensuring mutual protection among those involved in the chain of service

The OCP architecture has been designed as a scalable, multilayer and interoperable platform which consists of the following main components (see Figure 1 for details):

- Infrastructure as a Service (IaaS) platform based on OpenStack, suitably configured by capitalizing on the pioneering experiences made in-house on the INFN Cloud-infrastructure [2] as well as from previous cloud-related projects [3, 4].
- Platform as a Service (PaaS) platform that allows to use heterogeneous IaaS sites and easily manage the deployment and execution of new applications in a cloud environment.
- Software as a Service (SaaS) platform that consists of an Application Store and a set of new services that allow customers to choose the application of interest, to configure it, to suit their needs and execute it on cloud-infrastructures.

Besides the aforementioned components, there are other layers aimed to create a complete integration between the different Cloud services:

- The OCP Platform engine module, able to implement business logic. It provides support for the automation of process workflows and complex services composition based on the orchestration of simple services with reference to specific activities.
- Identity management module, that represents an authentication framework for the management of digital identities, giving the ability to control and manage the access to the infrastructure and related layers and services made avilable in the cloud environment.
- Open Data and Open Service Engine aimed at providing a full support to the generation and use of data from homogeneous data sources.
- A marketplace where cloud-based services and components provided by the Public Administations and made available through the cloud and offered to citizens and providers to interact with the PA the first and provide innovative services to the community the latter.

After the initial development and integration of the main components of the OCP platform, including Iaas, PaaS and SaaS services, we had to cope with an increasing number of requests to deploy new testbeds at the LGAs interested to use the OCP platform, starting from the IaaS level. In this contribution we present in particular the solution developed in through the OCP

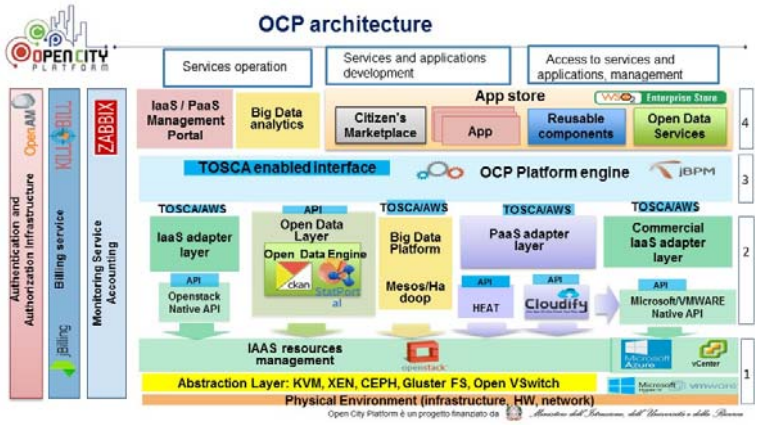

Figure 1. OCP platform architecture and main components.

project for the automatization and standardization of installation and configuration procedures for the IaaS layer.

The paper is organized as follow. In Section 2 the common installation methods for IaaS and their limits are described. In Section 3 the new semi-automatic installation method is proposed and the developed features are highlighted. Finally, Section 4 concludes the paper presenting directions to future work.

2. IaaS and installation scenarios

The Open City Platform project makes available a Cloud Computing platform open source, flexible, scalable and compliant to international standards that consists of a set of components organized in different layers, fully integrated with each other, that can already be installed and used for activation of on demand services and access data in the Data Centers where the experimentation take place. The platform is continuously maintained and updated with additional components developed through the project in order to progressively meet the various specific needs of PA. In its first phase, the project made available the services related to the IaaS platform layer by evaluating two different installation methods already available during the experimenting phase:

- manual installation and configuration ('hardcore').
- fully-automated installation via Graphical User Interface (GUI) using Fuel [5].

2.1. Manual Installation and Configuration

This method consists of different steps aimed at (i) installing the OpenStack middleware based on the use of the package manager and the repositories available for the chosen distribution of the Operating System; (ii) tuning the configuration of the individual services by editing the corresponding configuration files; (iii) verifying the installation and configuration steps by checking the state of the processes involved, the status of the connections between related services, the messages collected in the logs, etc.

In the case of deploying multi-node in High Availability environment (HA), services are first installed in standalone mode and then clustered. With respect to this approach some advantages were identified:

- better understanding of OpenStack dependencies between components, of the possible choices in the deployment of services

- improved control over configurations, the possibility to enable advanced functionalities like: multi-region support, SSL support, Identity management service and so on.

On the other hand, the following disadvantages were also identified:

- advanced knowledge of Linux OS, bash and network configuration is required
- error-prone - the method is subject to typo errors in particular for multi-node HA deployments
- time-consuming, many of the operations are repetitive

This method is widely used in many cloud infrastructures both for pre-production as well as for test and production environments. Moreover, the greater flexibility and control in the choice of configuration are the main strengths. On the opposit side, instead, the method requires additional effort to keep aligned configurations on more servers, expecially in the case of HA multi-node deployments.

2.2. Fully-automatic installation using Fuel

The other method evaluated for the OCP project is a full automatic installation and configuration method using Fuel [5]. Fuel is a framework for the installation and management of an OpenStack-based infrastructure. It allows to perform the installation and configuration of one or more OpenStack environments through both a Web-GUI interface or a classical Command Line Interface (CLI). Fuel comes with a series of interesting functionalities such as (i) the automatic host (physical or virtual) discovery; the presence of a step-by-step wizzard for an easy installation; (iii) the adoption of different roles that can be assigned to each node; (iv) the possibility to consult via a GUI the logs of the installation and configuration, providing also control of their verbosity; (v) offers the possibility to perform tests regarding the health of the services and the correctness of network configuration.

As for the previous method, also in this case some advantages have been identified:

- Easy installation through its GUI
- It enables new deployments and subsequent (limited) modifications on running infrastructures (modifing services and add or remove nodes)

On the contrary, the following disadvantages have been singled out:

- The initial configuration cannot be changed (eg. move from a simple to a high-availability setup)
- The OpenStack regions are not yet supported.
- The various components of the Cloud Controller (eg. Keystone, Glance, Horizon, Neutron) are installed on the same node
- Custom configuration cannot be applied.

Among the already mentioned disadvantages we have to stress the fact that, at the time of the tests, the latest OpenStack version supported for Fuel was Icehouse [6] on Centos 6.5 and on Ubuntu 12.04.4. Specifications that were in contrast with the OCP project requirements advertised to provide a full support to OpenStack version Juno [7] on Ubuntu 14.04 LTS [8].

3. The AutomaticOCP IaaS deployment tool

The investigated methods already described in the above Section present some important drawbacks that are here recapped:

- The manual installation and configuration method can be time-consuming for the infrastructure maintainers due to the repetitive operations they have to perform in order to keep configurations correctly aligned among different servers and nodes.

- The automatic installation method based on Fuel tool [5], even if it solves many of the disadvantages identified in the manual method, does not permit a full tuning of the infrastructure being not flexible enough to cope with the architectural requirements of the PA.

For the above mentioned reasons, a semi-automatic installation and configuration tool (hereafter called AutomaticOCP) has been designed. The AutomaticOCP tool is able to take the advantages of the methods presented in the previous Section, including the use of a Web-GUI, and, in the same time, be flexible enough to meet the architectural requirements of the OCP project (advertised to use OpenStack version Juno [7] on Ubuntu 14.04 LTS [8]) as well as the ones of the Data Center where the OpenStack Infrastructure will be deployed.

The solution proposed is leveraging two of the most popular open source automation tools, namely Foreman [9] and Puppet [10], making use as much as possible of the official OpenStack Puppet modules [11], as well as of other community supported Puppet modules for services like MySQL/Percona [12], Ceph [13], and others.

3.1. The Puppet method: Roles and Profiles

According to the Puppet documentation [14], the use of **roles and profiles method** is the most reliable way to build reusable, configurable, and refactorable system configurations. Following this model, we developed our Puppet code in three main layers:

- Component modules: these are the normal modules that configure specific pieces of technology (like apache, mysql, etc)
- Profiles: Wrapper classes that use multiple component modules to configure a technology (like Wordpress, Jenkins, etc)
- Roles: Wrapper classes that use multiple profiles to build a complete system configuration. In our case the Roles moduels contain the variables that have to be set in order to deploy the OCP Infrastructure.

In the present work the Component modules are those available in the official Puppet repos and provided by the community members. Unfortunately, some of them, like the ones for Ceph and Percona/MySQL have been slightly modified (due to some missing features), and have been provided as internal OCP modules, made available via the INFN Gitlab repository [15].

Roles and profiles for the different services have been developed to make the whole infrastructure flexible enough and to cope with the technical and architectural requirements to which the Data Center have to comply with.

An overview of the infrastructure architecture can be seen in Figure 2 where all the main components and services are present and here briefly described:

- **Master Node**: it hosts the configuration management services, like Foreman and Puppet, used to install and configure the whole OCP-IaaS. At present, the configuration management tools have to be manually configured. The same node can eventually host a Zabbix [16] monitoring server for resources and application monitoring. The Zabbix server configuration has been implemented through a Puppet Role named **monit_server**. In the same way, the Zabbix agents configuration on the OCP-IaaS nodes has been implemented through a Puppet Role named **monit_agent**.
- **RHMK Nodes**: they provide external OpenStack services such as: (i) database cluster services (Percona/Mysql [12] and MongoDB [17]); (ii) a messaging system implementing the AMQP protocol to let the Openstack services to connect and horizontally scale in case of an increase of demands (RabbitMq [18]) and (iii) a set of services for the High Availability and Load Balancing (HAproxy [19], Zookeeper [20] and Keepalived [21]). All the RHMK

services configurations have been identified in Puppet with a Role named **rhmk**, while the configuration of each individual mentioned service is represented by a Profile.

- **Storage Nodes**: Ceph, version Hammer [22], has been chosen as a distributed block storage platform whereby a minimum of three nodes are configured to ensure data replication. To cope with the project requirements, Ceph was also chosen as object storage for the PaaS layer by configuring the Ceph Object Gateway [23]. Ceph Object Gateway is an object storage interface built on top of *librados* to provide applications with a RESTful gateway to Ceph Storage Clusters. In particular Ceph makes use of the RADOSGW daemon, a FastCGI module that provides interfaces compatible with OpenStack Swift [24] and Amazon S3 [25]. The Ceph storage service has been configured through a Puppet Role named **Storage**.
- **Controller and Network Nodes**: they contain some common OpenStack services which are defined as Puppet Profiles. The configuration variables are hosted in the Puppet Role named **Controller&Network**. The service is designed to run in a High Availability setup but it can be deployed on a single node also. **Controller** and **Network** services can be split in different nodes if required by architectural needs and each service is designed to run as single server as well as part of a HA cluster. OCP Network currently support the network configuration as from the OpenStack documentation [26].
- **Compute Node**: it contains the Nova services which are configured throught Puppet Profiles. The configuration variables are hosted in the Puppet Role named **Compute** that permits to deploy the node and add it to the OpenStack infrastructure at any time.

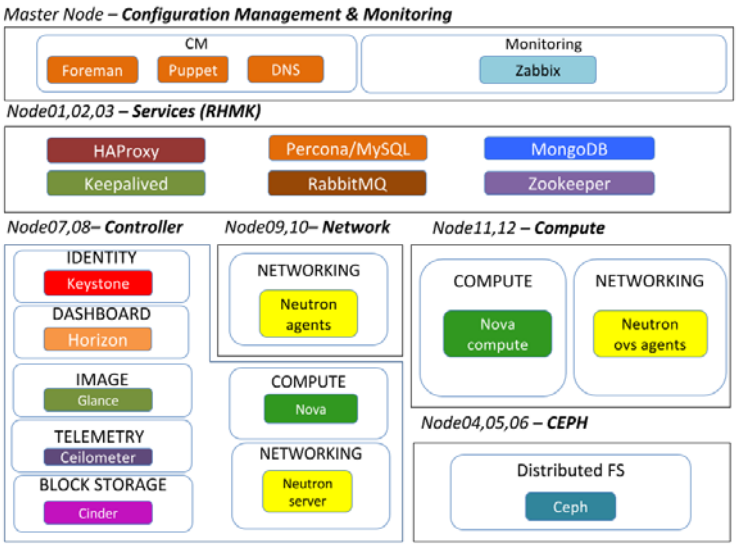

Figure 2. Overview of the OCP Infrastructure architecture and related Services.

3.2. Infrastructure deployment with Foreman GUI

As already mentioned, the proposed solution is leveraging on the Foreman [9] automation software used to provide a Web-GUI for the IaaS installation and configuration processes. Foreman is an open source complete life cycle systems management tool for provisioning, configuring and monitoring of physical and virtual servers. Foreman has deep integration with configuration management software like Puppet to automate tasks and application deployment.

Foreman provides a web user interface, API, for an easy interaction but also a CLI, for more experienced users.

The version of Foreman available and suitable for the project purposes was the 1.10.1. Foreman has been used as it is, leveraging on its native features. In particular there has been a massive use of the HostGroups, a collection of user-selected classes and parameters needed for the installation and configuration of the OCP IaaS. As shown in Figure 3, a main Host Group called *OCP-BASE* has beed defined associating to it only those Puppet classes used by all the Roles. Starting from the *OCP-BASE* Host Group, a set of sub-Host Group has been defined (one for each Puppet Role) and the related class parameters have been assigned and populated.

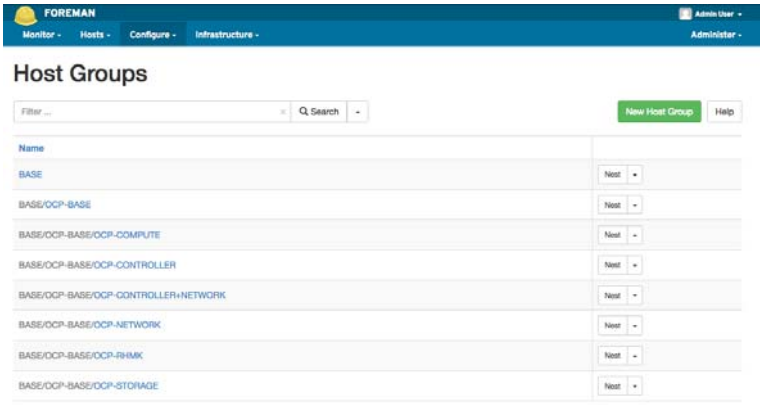

Figure 3. Foreman Web-GUI: OCP Host Group definition.

After the OCP-IaaS installation and configuration process finished and all the services are up and running on the related hosts, Foreman can be used to control and maintain the configuration status of the nodes as shown in Figure 4. Moreover, Foreman can be used also to reconfigure a selected node in case of errors due to misconfiguration, including therein the worst case where a complete reinstallation is needed.

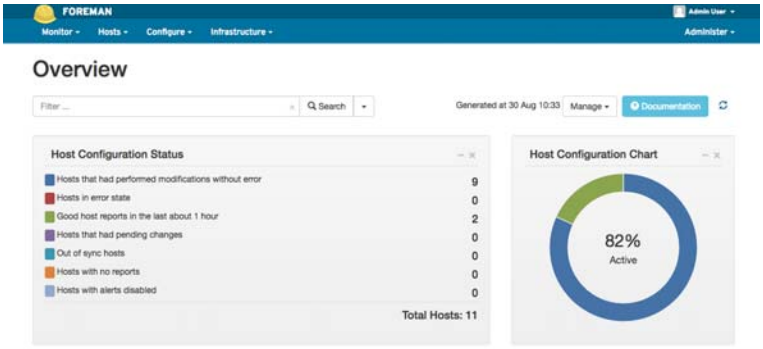

Figure 4. Foreman Web-GUI: A way to control hosts configuration status.

4. Future work

In the present paper a new semi-automatic installation method for IaaS has been presented. The method has proved to be flexible enough to meet the architectural requirements dicted by the OCP project as well as of the Data Center where the OpenStack Infrastructure will be deployed.

With our solution we tried to address the different requirements and realities that we met during the collaboration with PAs, including, among others (i) the configuration of multiple external networks; (ii) full management of the configuration of the network layer giving the ability to merge or split the configuration of the various OpenStack networks (management, data, public and external); (iii) use of Ceph both as block and object storage backend by configuring the Ceph Object Gateway in order to expose Swift APIs; (iv) fine grained variable configuration through the use of the Foreman GUI allowing site-admins to specify the values of all service specific parameters.

The positive results obtained and the experience gained during the testing phase, led us to investigate new semi-automatic procedures able to install and configure a complete OCP-IaaS providing also a full support to the OpenStack Identity API v3 [27]. As a challenge, we are now designing the semi-automatic tool to be used for performing the upgrade of the OCP-IaaS layer to a new OpenStack version. Moreover, automated tools devoted to the installation and configuration of the PaaS layer (the so called CloudFormation as a Service) are under investigation and will be part of future enhancements.

References

[1] OpenCityPlatform Project, *http://www.opencityplatform.eu/*
[2] C. Aiftimiei, R. Bucchi, A. Costantini, D. Michelotto, M.Panella, D. Salomoni and G. Zizzi: Cloud@CNAF - the road to Juno. INFN-CNAF Annual Report 2015, pp 134.139 (2016) ISSN 2283-5490
[3] PRISMA project,*http://www.ponsmartcities-prisma.it/*
[4] MCloud project,*http://www.ecommunity.marche.it/AgendaDigitale/MCLoud/Obiettivi/tabid/206/Default.aspx*
[5] Fuel framework, *https://www.mirantis.com/products/mirantis-openstack-software/*
[6] OpenStack Icehouse, *https://www.OpenStack.org/software/icehouse/*
[7] OpenStack Juno, *https://www.OpenStack.org/software/juno/*
[8] Canonical, *https://wiki.ubuntu.com/TrustyTahr/ReleaseNotes/14.04*
[9] Foreman framework, *http://theforeman.org/*
[10] Puppet, *https://puppetlabs.com/*
[11] OpenStack Puppet, *https://wiki.openstack.org/wiki/Puppet*
[12] Percona cluster, *https://www.percona.com/software/mysql-database/percona-server*
[13] Ceph storage platform, *http://ceph.com/releases/v0-94-hammer-released/*
[14] Puppet Documentation, *https://docs.puppet.com/pe/2016.2/r_n_p_intro.html*
[15] OCP Internal Repository, *https://baltig.infn.it/groups/ocp-tools*
[16] Zabbix monitoring service, *http://www.zabbix.com/*
[17] MongoDB no SQL database, *https://www.mongodb.com/*
[18] RabbitMQ messaging service, *https://www.rabbitmq.com*
[19] Haproxy TCP/HTTP Load Balancer, *www.haproxy.org*
[20] Apache ZooKeeper, *https://zookeeper.apache.org/*
[21] Keepalived, *www.keepalived.org/*
[22] Ceph V0.94 (Hammer), *http://ceph.com/releases/v0-94-hammer-released/*
[23] Ceph object storage, *http://docs.ceph.com/docs/master/radosgw/*
[24] OpenStack Swift, *https://wiki.openstack.org/wiki/ReleaseNotes/Juno#OpenStack_Object_Storage_.28Swift.29*
[25] Amazon Simple Storage Service, *https://aws.amazon.com/it/s3/*
[26] OpenStack Networking documentation, *http://docs.openstack.org/security-guide/networking/architecture.html*
[27] OpenStack API v3, *http://developer.openstack.org/api-ref/identity/v3/*