CloudTUI: a multi Cloud platform Text User Interface

Massimo Canonico, Andrea Lombardo and Irene Lovotti
DiSIT - Computer Science Institute
University of Piemonte Orientale
Italy
massimo.canonico@unipmn.it, {andrea.lombardo,irene.lovotti2}@studenti.unipmn.it

ABSTRACT

Most of the online services that we use everyday are provided by Cloud Computing infrastructures. This widespread is due to the substantial effort spent to make all these services as much user-friendly as possible. Unfortunately, the usage of the most popular Cloud Computing infrastructures is not easy at all and it requires high computer science skills. In this paper, we propose CloudTUI: a multi Cloud Computing platform text-based interface able to make the interaction with Cloud systems easy and intuitive even for users without any prior experience concerning the Cloud Computing paradigm.

1. INTRODUCTION

The NIST defines Cloud Computing as a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [20]. Nowadays, most of the Web services that we use (such as, webmail, Internet search, online storage, social networks, ...) are provided by a Cloud Computing infrastructure. The huge interest in this recent computing paradigm is testified by the large number of Cloud Computing platforms developed both commercial (i.e., *Amazon's EC2* [2], *Microsoft Azure* [5], *IBM Blue Cloud* [4], *RackSpace* [16], ...) and non-commercial (i.e., *Eucalyptus* [8], *OpenNebula* [13], *OpenStack* [15], *Nimbus* [12], ...). The common functions of these systems regard the provisioning of Virtual Machines (VMs) to users. In particular, a Cloud Computing platform has to provide the mechanisms to start-up/shut-down VMs, to create/destroy storage volumes and to attach/detach them to/from VMs. Due to the lack of standardization in Cloud Computing, the above common functions have completely different syntax depending on which Cloud platform is chosen. Moreover, sometimes even the terminology is different. For example, a running VM is called *instance* in Eucalyptus, *workspace* in Nimbus, and so on. This lack of standardization leads to the following issues: a) interoperability between different Cloud platforms is difficult to realize and b) users of Cloud platforms have to learn all the different commands, syntax and terminology used by each Cloud system. *CloudTUI* aims at solving these issues by providing a text based interface intuitive and easy to use. In particular, with CloudTUI, the user can interact with different Cloud Computing platforms and perform the actions mentioned above without knowing the details concerning the APIs of each Cloud platform.

The rest of this paper is organized as follows. Section 2 discusses the related works concerning the various user interfaces for Cloud platforms while in Section 3 we present CloudTUI features in detail. Conclusions and future works are discussed in Section 4.

2. RELATED WORKS

Most of the open source Cloud Computing platforms provide two different user interfaces: command-line and Web-based interface . From our point of view, both interfaces are not user-friendly for many reasons.

First of all, command-line interfaces require an advanced skills on using the terminal emulator in Linux environment. Moreover, the parameters for the various tasks are related to identifiers (such as, user-id, instance-id, image-id, volume-id, size-id, ...) that are not based on mnemonic names, so the user must always retrieve the identifier for each entity involved in the Cloud platform. For example, in order to start an instance in Eucalyptus, the minimum amount of parameters that a user must know consists in i) the image identifier related to the VM which (s)he wants to start-up, ii) the identifier of the key pair that (s)he will use to login into the running VM and iii) the size identifier of the VM image that (s)he is going to use.

The syntax of existing command-line interfaces are complex and not intuitive. Furthermore, as discussed in Section 1, each Cloud Computing platform has developed its own tool. In particular, Eucalyptus uses *euca2ool*[7], Nimbus has its *nimbus-cloud-client*[11], while OpenStack uses *nova-client*[14], just to name a few. Moreover, most of these command-line interfaces are not able to notify the user when a running VM changes its status. For example, in order to figure out when a VM has become *running* (that means *"ready to use"*), the user must repeatedly execute the *"print VM status"* command.

In the last years, thanks to the proliferation of Web-based interfaces, some of the problems mentioned before are partially solved. As a matter of fact, Web-interfaces hide the

VALUETOOLS 2013, December 10-12, Torino, Italy

DOI 10.4108/icst.valuetools.2013.254413

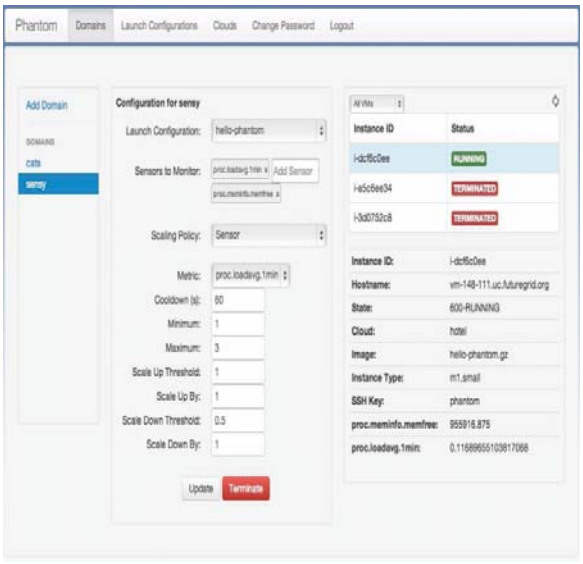

Figure 1: Nimbus Phantom: launching a domain

syntax of the command and help out the user to retrieve the identifiers of VMs/volumes/key pairs and of all other entities involved in a Cloud system. The main problem with Web-based interfaces consists in the fact that they are limited (they do not provide all the features that command-line interfaces do) and there is no way to automate or to customize the tasks to perform (conversely, command-line interfaces can be integrated in a script). Moreover, the Web-based interfaces need advanced configuration settings even for basic tasks such as start-up a VM. For example, in order to use the Nimbus Web-based interface, it is necessary to create and configure a user profile, then a *launch configuration* must be created. After that, it is necessary to select which Cloud will execute the VM, setup a *contextualization* and finally the selected VM can be launched. Fig. 1 is a screenshot of this last step: we can observe how many parameters have yet to be defined. Due to space constraints, we cannot go in detail of each step previously mentioned; the interested reader can find more information of each step in [6]. Finally, another problem comes from the terminology used in the command-line interface that could be different with respect to the terminology used in the Web-based interface. This is true not only considering different Cloud platforms (a running VM is called *workspace* in Nimbus, while in Eucalyptus/OpenStack it is called *instance*) but also in the same Cloud platform. For example, Nimbus uses the term *domain* for a running VM in its Web-based interface, while, as mentioned before, a running VM is called *workspace* in the Nimbus command-line interface. As one could easily imagine, this terminology choice may confuse the user.

One of the first Web-based interface for different Cloud platforms was *Hybridfox* [9]: an add-on of the Firefox Web browser that attempts to provide a single interface to switch seamlessly between multiple Cloud accounts. Unfortunately this software has not been updated recently and it is not fully compatible with the latest versions of Cloud platforms. So users have to learn how to use a specific Web-based interface for each Cloud platform: *User Console* [3] for Eucalyptus, *Phantom* [6] for Nimbus, just to name a few.

Conversely to all limitations and problems mentioned before for command-line and Web-based interfaces, our tool provides an easy and intuitive text-based interface that allows users to start-up/shut-down/monitor running VMs on various Cloud Computing platforms. Moreover, our tool allows users to create/delete volumes and attach/detach them to/from a running VM. CloudTUI covers also security aspects by showing/creating/deleting key pairs necessary to log into running VMs. Finally, CloudTUI uses the same terminology for all Cloud platforms in order to make the user experience as simple as possible.

3. CLOUDTUI AND CLOUDTUI-ADVANCED

In this paper, we present two tools written in Python: *CloudTUI* and *CloudTUI-advanced*. CloudTUI provides an easy interface for the basic tasks available on each Cloud Computing platform, while CloudTUI-advanced exploits the *Boto* library [18] in order to provide not only the basic tasks but also scale-up/scale-down policies to take advantage of the Cloud Computing dynamism. Both tools proposed are modular and open source so anyone can contribute to improve the code.

We put much effort in trying to make any aspect of our tools as simple as possible. As a matter of fact, in order to run our tools, the users has just to unpack the archive and launch the tool. After that, the tool will ask where the credentials file is stored (the credentials file contains confidential information, such as the private key, that are necessary to interact with the Cloud platform); this allows our tools to automatically import all the necessary information without asking anything else to the user. Moreover, our tools are able to remember where the credential information is stored, so that the user can immediately start to use the Cloud system selected after the first run. Since Cloud Computing involves many hardware and software components, the risk that something can go wrong is high. For this reason, we pay much attention to prevent any possible failure and consequently we have implemented different fault tolerant mechanisms in order to overcome from a failure as fast as possible. For example, if a VM fails to start-up, the CloudTUI provides to the user an alternative VM with the same virtual hardware characteristics (such as number of cores, memory capacity, and so on).

We decided to keep separate our two tools CloudTUI and CloudTUI-advanced for different reasons. The first one is related to the user skills. CloudTUI is for beginner users that want to interact with different Cloud platforms by using only the basic functions. Moreover, CloudTUI has minimal software requirements: it just needs Python library installed and nothing else. The code of CloudTUI is stable and perfectly working thanks to the fact that it is a sort of wrapper based on the native Cloud clients of each Cloud platform. Conversely, CloudTUI-advanced is dedicated to advanced users which want to fully exploit Cloud platforms potentialities by managing also scale-up and scale-down policies. Moreover, the software requirements for CloudTUI-advanced impose to install only the last versions of both Python and Boto (version 2.7.5 and version 2.13.3 respectively). This is due to the fact that we have experimentally tried and true that older versions are not fully compatible with some Cloud platforms. Moreover, CloudTUI-advanced could not be considered stable since Boto is still a working in progress project especially with respect to the open source

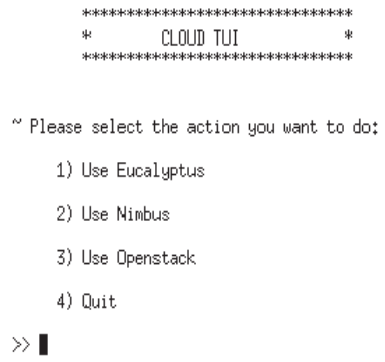

Figure 2: CloudTUI: first request

Cloud systems. As a matter of fact, from our experiments, we can assert that only Nimbus is fully compatible with Boto libraries. From our point of view, this problem regards both sides: on one side, Boto documentation and support must be improved in order to help Cloud platform developers to make their products more Boto compatible and, on the other side, Cloud platform developers must face with compatibility issues between Cloud system API and Boto API. For all of these reasons CloudTUI-advanced needs a constant upgrade/update to follow up the improvements of Boto and its interaction with Cloud platforms.

Documentation, source code, videos and screenshots concerning our tools are available in the CloudTUI project website [19]. Finally, our tools have been experimentally tested on *FutureGrid* [17]: a national-scale Grid and Cloud testbed that includes a geographically distributed set of heterogeneous computing systems. CloudTUI and CloudTUI-advanced are already being used by FutureGrid, OpenStack, Nimbus and Eucalyptus user communities and the positive feedback received encourages us to keep this project going on. In the following two sections, we describe in detail the features of our tools.

3.1 CloudTUI

With CloudTUI tool the user can perform the most common tasks available in a Cloud Computing platform such as: a) to show the VM images/volumes/key pairs available, b) to create/delete any running VM/volumes and c) to attach/detach volumes to/from a specific running VM. After launching CloudTUI, the user can select which Cloud Computing platform wants to use between Eucalyptus, Nimbus and OpenStack (see Fig. 2). The second step is to enter the absolute path where the user credentials are stored. In particular, for Eucalyptus/OpenStack the tool will ask the path where eucarc/novarc file is stored, while for Nimbus the tool will ask the path where the Nimbus cloud-client is installed. After that, the user is ready to decide which task to perform. In Fig. 3, we show the CloudTUI menu for Eucalyptus with all possible actions that a user can perform. Due to space constraints, we cannot describe in detail the workflow of each action available. For this reason, in the CloudTUI project website [19], we provide documentation, more screenshots and short videos that show how easily a user can interact with our tool.

3.2 CloudTUI-advanced

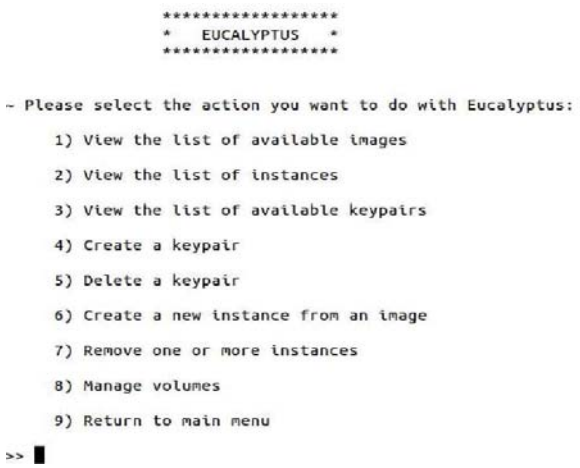

Figure 3: CloudTUI: Eucalyptus menu

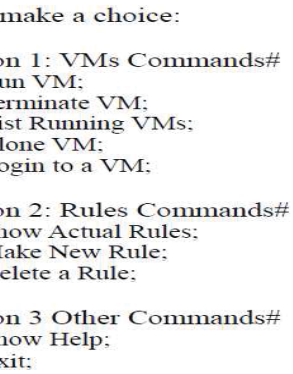

Figure 4: CloudTUI-advanced: main menu

With CloudTUI-advanced, the users can improve their interaction with Cloud Computing platforms by exploiting the potentiality of Boto [18]. Boto is an integrated interface to infrastructural services compatible with private and public Cloud systems. With Boto is possible to build out scripts to automate Cloud Computing tasks without knowing the specific syntax of the Cloud client provided by the selected Cloud platform.

As a matter of fact, besides the basic tasks mentioned in the previous section, CloudTUI-advanced allows users to configure scaling-up and scaling-down policies. For example, it is possible to activate the following policy: "if the CPU load of a specific VM has been higher than 70% in the last 5 minutes, then create a clone of this VM". This policy can be useful in order to balance the workload between different VMs that are providing the same service. Another example policy could be "if memory occupation is lower than 20%, then shut-down the VM". This policy can be useful in order to minimize energy consumption by powering off useless services. CloudTUI-advanced helps users during the policies creation by asking which metric must be monitored (i.e., "CPU load", "memory occupation", "storage level", "bandwidth available", ...), what is the threshold and the sign (i.e., "<30", ">=50", ...), and finally the action to perform (i.e., "clone", "start-up", "shut-down",...). Of course, anytime it is possible to add new metrics to monitor and cre-

ate new policies.

CloudTUI-advanced is compatible with any monitoring tool that provides a TCP connection with the monitor server. As a matter of fact, CloudTUI-advanced has a monitoring module able to interact with a monitor server by setting just three input parameters: a) hostname or IP address of the monitor server, b) port number where the monitor server is listening, and c) the "request string" that is the message to send to the monitor server in order to obtain the metrics requested. Once these parameters are set, CloudTUI-advanced will periodically send a TCP request to the monitor server asking the actual value of the metrics to monitor. For example, supposing to use *Munin* [10] as monitoring tool, in order to obtain information concerning CPU usage, CloudTUI-advanced will send "fetch cpu" string to the monitor server and it will wait for a number as answer (that is, the actual value of the CPU load). Fig. 4 shows all possible tasks allowed by CloudTUI-advanced. Due to space constraints, we cannot discuss the workflow for all possible tasks provided by CloudTUI-advanced. For this reason, we created a video showing how easy and intuitive is the interaction with our tool. This video, and many other information such as screenshots, documentation are available in the website dedicated to our tools [19].

4. CONCLUSION AND FUTURE WORKS

In this paper, we present two text-based tools able to interact with different Cloud Computing platforms by an easy and intuitive interface. In particular, we propose two different versions of our tool: CloudTUI which helps users to perform the basic tasks on different Cloud platforms and CloudTUI-advanced which also implements policies to scale-up/scale-down the Cloud systems by monitoring metrics specified by users. Both CloudTUI and CloudTUI-advanced are currently being used by Cloud users communities belonging to FutureGrid, Nimbus, Eucalyptus and OpenStack projects. The positive feedback received from these communities encourage ourselves to keep this project going on.

Regarding possible future works there are some interesting activity that can be carried on. First of all, we plan to increase the number of Cloud Computing platforms supported by CloudTUI tools considering also commercial projects such as Azure and RackSpace. In the next CloudTUI release, we will include federation capabilities in order to allows users to create virtual infrastructures that overlap different Cloud providers. Moreover, we will be focused on CloudTUI-advanced compatibility issues since much work has to be done concerning the interaction between Boto and OpenStack/Eucalyptus. As a matter of fact, from our experience, Boto is fully compatible only on Nimbus Cloud platform. Finally, we plan to improve the monitoring module of CloudTUI-advanced by using *CloudWatch* [1]: a monitor service able to collect and track metrics of the running VMs. CloudWatch is a very promising project since it can also send alarms and notifications or automatically make changes to the resources monitored.

5. REFERENCES

[1] Amazon CloudWatch. http://aws.amazon.com/cloudwatch. [Online; accessed 30-Sept-2013].

[2] Amazon Elastic Cloud. http://aws.amazon.com/ec2. [Online; accessed 30-Sept-2013].

[3] Eucalyptus User Console. http://www.eucalyptus.com/eucalyptus-cloud/iaas/user-console. [Online; accessed 30-Sept-2013].

[4] IBM Blue Cloud project. http://www.ibm.com/cloud-computing. [Online; accessed 30-Sept-2013].

[5] Microsoft Azure. http://www.microsoft.com/azure. [Online; accessed 30-Sept-2013].

[6] Nimbus Platform Phantom. http://phantom.nimbusproject.org. [Online; accessed 30-Sept-2013].

[7] The Euca2ools command line tool. http://www.eucalyptus.com/download/euca2ools. [Online; accessed 30-Sept-2013].

[8] The Eucalyptus project. http://www.eucalyptus.com. [Online; accessed 30-Sept-2013].

[9] The Hybridfox project. http://code.google.com/p/hybridfox. [Online; accessed 30-Sept-2013].

[10] The Munin project. http://munin-monitoring.org. [Online; accessed 30-Sept-2013].

[11] The Nimbus cloud client. http://www.nimbusproject.org/docs/2.10.1/clouds/cloudquickst [Online; accessed 30-Sept-2013].

[12] The Nimbus project. http://www.nimbusproject.org. [Online; accessed 30-Sept-2013].

[13] The OpenNebula project. http://www.opennebula.org. [Online; accessed 30-Sept-2013].

[14] The OpenStack command line tool. https://wiki.openstack.org/wiki/NovaClientCLI. [Online; accessed 30-Sept-2013].

[15] The OpenStack project. http://www.openstack.org. [Online; accessed 30-Sept-2013].

[16] The RackSpace project. http://www.rackspace.org. [Online; accessed 30-Sept-2013].

[17] G. Fox et al. *FutureGrid - a reconfigurable testbed for Cloud, HPC, and Grid Computing*. Chapman & Hall, 2013.

[18] M. Garnaat. Boto. http://boto.readthedocs.org. [Online; accessed 30-Sept-2013].

[19] M. Canonico, A. Lombardo, I. Lovotti. CloudTUI. https://portal.futuregrid.org/projects/314. [Online; accessed 30-Sept-2013].

[20] Peter Mell and Tim Grance. The NIST Definition of Cloud Computing. http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf, 2013.