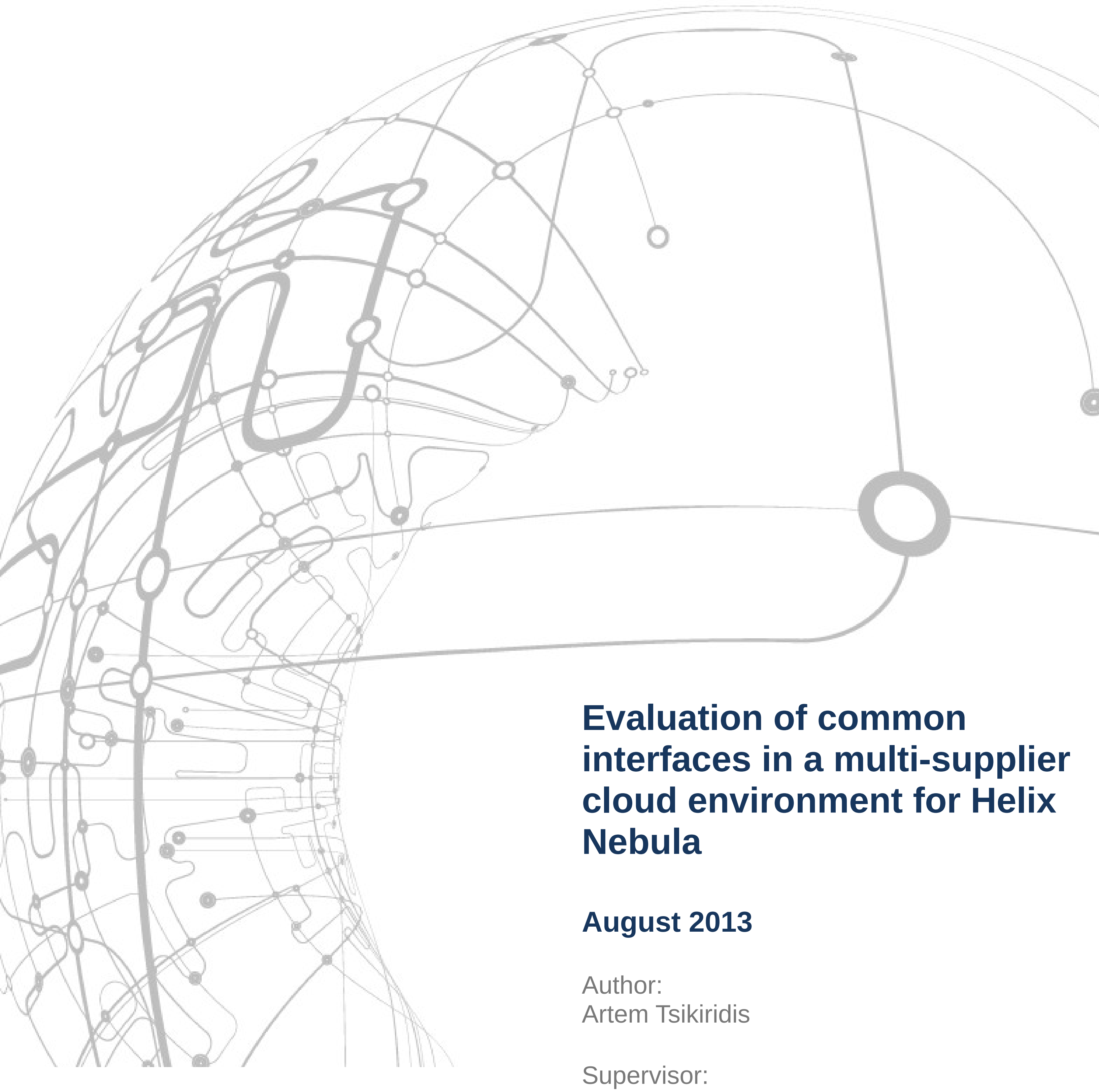


Evaluation of common interfaces in a multi-supplier cloud environment for Helix Nebula

August 2013

Author:
Artem Tsikiridis

Supervisor:
Mattia Cinquilli

CERN openlab Summer Student Report 2013

Acknowledgement

I would like to acknowledge Mattia Cinquilli (CERN, IT-SDC-OL) and Cristovao Jose Dominguez Cordeiro (CERN, IT-SDC-OL) for their great help and valuable insights they provided me concerning the structure and content of this report.

Moreover, I would like to take the opportunity and thank Mattia Cinquilli for his continuous guidance and support as my CERN supervisor during my stay as an Openlab Summer Student.

Project Specification

Partner/group	IT-SDC-OL
Title	Integration of LHC experiments resource and tools with Helix Nebula
Description	In the context of the Helix Nebula Science Cloud, ATLAS and CMS experiments are collaborating to define, evaluate and integrate cloud standards, services and resources into the existing distributed computing infrastructure. The final goal of the initiative is to create a European federated cloud infrastructure for science. In this project, the student will work on topics related to deployments of LHC experiments resources in the cloud, focusing in the interfacing between the Helix Nebula Science Cloud and experiment tools. Specific work will include some or all of the following: * Testing BlueBox solutions (EnStratus, SlipStream) and related interfaces, against various cloud providers. * Work in the dynamic allocation of resources through the BlueBox solution. * Participate in the evaluation of remote cloud facilities through both production and analysis jobs for ATLAS and CMS experiments.
Supervisor	Mattia Cinquilli

Abstract

The Helix Nebula initiative is a partnership between leading European IT-intense scientific research organisations (CERN, EMBL and ESA) and leading IT cloud providers. Its goal is to form a federated Cloud Computing infrastructure that will satisfy the growing demand of scientists for computing power. The concept of the federated cloud requires a standard framework named BlueBox so that current and future cloud providers are able to smoothly interface their system to the existing multi-cloud infrastructure. Section one acts as an introduction to the reader, highlighting the importance of cloud computing in general and at CERN specifically, and provides a brief overview of the Helix Nebula initiative. Section two contains information about the functionality and performance of SlipStream, an interface that has been evaluated as a potential BlueBox solution. Section three contains the analysis of a *de facto* standard in cloud computing such as the EC2 interface, which is very useful for Helix Nebula and multi-cloud environments in general. Finally, in section 4, you may find listed the key conclusions about SlipStream, based on the evaluation that was carried out along with several other remarks about multi-cloud operations.

Table of Contents

Contents

1 Introduction

1.1 Cloud Computing

In recent years, cloud computing has gradually become a dominant technology and business model for corporate environments and other data-intensive organizations such as governmental and research institutions. Organizations use cloud computing services on different levels depending on the intensity of their computing activity, while in the same time taking into account factors such as cost and organizational policy. These services are accessible to the end-user through interfaces or clients and may broadly be divided into three categories: Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS).

Software as a Service (SaaS) is a software delivery model in which software and applications are hosted on the cloud. SaaS is typically accessed by users using a web browser or a thin client (client that heavily relies on the server to perform any computations). It is becoming a very common delivery model for business applications. Companies who adopt this model are aiming to benefit from reduced costs. This comes from the fact that supporting a self-owned IT-infrastructure is not required [1].

Platform as a Service (PaaS) is the level of cloud computing service that allows the end-user to use tools and libraries that are provided by the infrastructure in order to develop his own software and applications. Development environments and Database Management Systems are common examples [2].

Infrastructure as a Service (IaaS) providers offer the end-user the physical or virtual machines which are called instances. Usually, IaaS providers supplement their service with APIs to handle images (collections of operating systems with appropriate additional software packages), load-balancers, firewalls and file-based storages [3]. In the scope of this report, we are going to focus on this level of service for reasons that are going to be stated below.

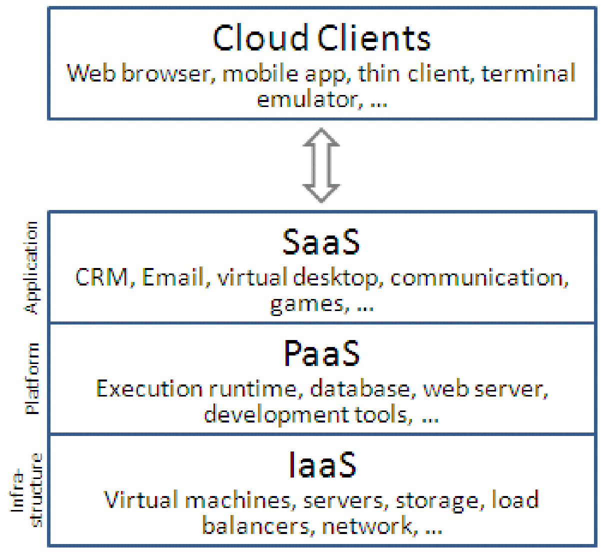

Cloud Clients
Web browser, mobile app, thin client, terminal emulator, ...
SaaS
CRM, Email, virtual desktop, communication, games, ...
PaaS
Execution runtime, database, web server, development tools, ...
IaaS
Virtual machines, servers, storage, load balancers, network, ...
Application
Platform
Infra-structure

Figure 1 Layers of service in cloud computing (source: Wikipedia)

1.2 Cloud Computing at CERN

1.2.1 The present: Worldwide LHC Computing Grid (WLCG)

CERN is a research institution that has extensive needs in the analysis of vast amounts of data in order to support the LHC-based and other experiments and achieve stability in the daily IT life of the organization. Due to that, a computing model dictating that all data are being processed solely on-site (CERN data centre) is unrealistic in terms of computing infrastructure and organisational constraints. Therefore, since 2002 the Worldwide LHC Computing Grid has been deployed to cope with the approximately 15 petabytes that are generated per year by the LHC. WLCG is collaboration between more than 150 computer centres across the globe which are separated into tiers: Tier 0, Tier 1 and Tier 2. It has been noted for achieving excellent performance during the first phase of LHC execution (2010-2013) by offering high throughput to operations while on the same time being highly available and transparent to users.

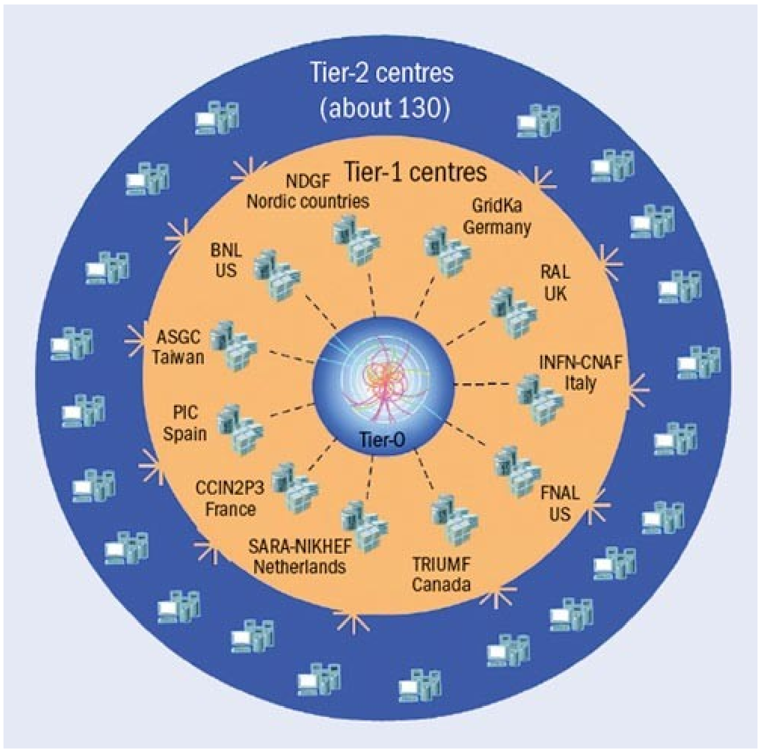


Figure 2 Worldwide LHC Computing Grid (WLCG) (source: wlcg.cern.ch)

1.2.2 Further scaling of WLCG with Cloud Computing

In recent years, the computing requirements of CERN experiments have increased due to the increasing amount of data produced and shall increase further in the future. Therefore, new technologies are being evaluated by the WLCG as extensions for the Grid to ensure that scalability is achieved. Cloud computing is one of the most prominent technologies that are being evaluated. We are specifically interested in Infrastructure as a Service solution, due to the flexibility that IaaS offers and due to the fact that IaaS is a more immediately attainable option than building other PaaS or SaaS services for the end users.

The main reasons why cloud computing is important at the level of IaaS for WLCG are the following:

1. Transparency (the user is agnostic on all cloud operations apart from the provided interface).

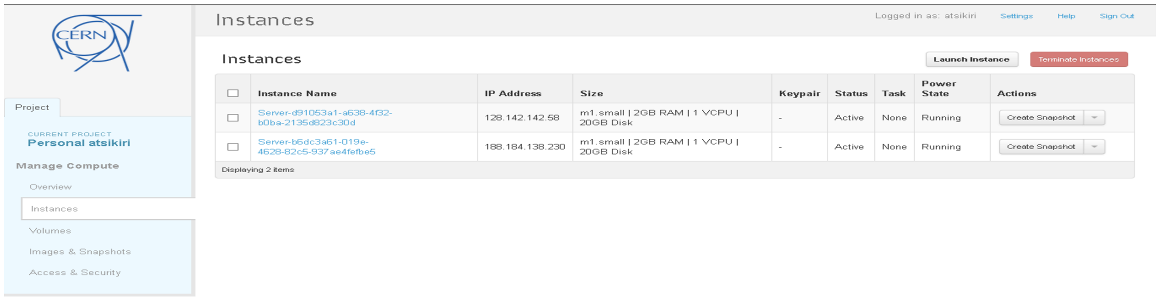
2. Usage optimization and dynamic resource provisioning.

3. Optimisation of data centre operations through virtualization technologies (radically speeds up data centers operations).

4. Many candidate standard tools and protocols supported by industry. For example, most cloud providers rely on the HTTP protocol for operations while many provide an interface usually considered a *de facto* standard such as the Amazon EC2 interface (more in section 3).

1.2.3 CERN Private Cloud

The various WLCG sites are evaluating the possibility of exploring these new technological opportunities. The first data facility migrating to cloud computing is Tier 0 hosted at the CERN computing center. CERN computing center has decided to gradually migrate its data center and operation toolsets, to a cloud model collectively referred to the CERN agile infrastructure. Therefore, since July 25 2013, the CERN private Cloud is fully operational and available for users requiring cloud-based data analysis or jobs in a distributed computing environment (https://openstack.cern.ch/dashboard). It is based on Openstack Cloud Computing software and offers its users two APIs (EC2 and Nova) and a dashboard. Generally, it is possible to deploy fully functional virtual machines in the cloud with multiple operating systems being supported. These features allow the CERN private cloud to interact seamlessly with the rest of the infrastructure of WLCG and in some scenarios in multi-cloud environments.



Instances

☐	Instance Name	IP Address	Size	Keypair	Status	Task	Power State	Actions
☐	Server-d91053a1-a638-4f32-b0ba-2135d823c30d	128.142.142.58	m1.small | 2GB RAM | 1 VCPU | 20GB Disk	-	Active	None	Running	Create Snapshot
☐	Server-b6dc3a61-019e-4628-82c5-937ae4fefbe5	188.184.138.230	m1.small | 2GB RAM | 1 VCPU | 20GB Disk	-	Active	None	Running	Create Snapshot

Displaying 2 items

Figure 3 CERN private Cloud is based on Openstack Cloud Computing software

The last observation about multi-cloud environments leads us to some other use cases that have to be examined. Occasionally, exceptional tasks that require a large amount of resources in an on-demand basis may have to be dealt with. Limitations may be reached regarding the usage of the CERN private cloud (user quotas, actual resources limits, peaks of load etc.) and the demand for resources may still not be fully satisfied. In such a case, direct access to additional resources is necessary and in order to achieve this in an on-demand basis, CERN has to collaborate with other Cloud providers (commercial, private clouds and hybrid clouds). However, this statement by itself is incomplete as interfacing of specific experiment-related tools (for instance) as well as the support for several important functions such as contextualization and authentication are integral features for such a system. Moreover, the transition between providers should be as transparent as possible to the user to ensure maximum efficiency. Thus, a framework must be implemented to allow cloud providers to smoothly interface their systems while on the same time it should not compromise end-user usability and ease of use. For example, the use-case in which an end-user requires a virtual machine in a specific cloud provider should not be more complicated than picking the desired provider from a list of cloud providers.

The above concept is a description of a multi-cloud environment that is also known as a federated cloud.

1.3 Helix Nebula and the federated cloud

Figure 4 Helix Nebula logo

Helix Nebula is a federated cloud (as described in 1.2) and is the collaboration between several European scientific organisations and commercial cloud service suppliers to create a cloud for Science. Helix Nebula aspires to support a framework to transparently deploy and manage instances of multiple cloud providers to achieve further scalability. In order to achieve this functionality, a common interface must be established to seamlessly interface providers in the present and in the future. This common interface may be described in what is called the BlueBox interface. Several solutions have been shortlisted, such as:

- Bonfire
- Computenext
- enStratus
- Nephos
- OpenNebula
- SlipStream

Generally, we are interested in evaluating the following topics:

1. Authentication mechanisms

2. Full support to perform a VM lifecycle (that is to run VMs, list VMs and terminate VMs)
3. Contextualization mechanisms
4. Consistency with native provider tools
5. Multi-Cloud deployments
6. Transparency of dashboard

During my stay as an Openlab Summer student at CERN in the summer of 2013 I was involved in the evaluation of SlipStream in terms of functionality and performance. You may find the results along with quantified performance results in section two.

1.4 Other Challenges in multi-cloud environments

Cases may be presented when all providers participating in a multi-cloud environment support common *de facto* interfaces such as the Amazon EC2. In those cases, deployment in multi-cloud environments may become possible without many drawbacks. However, one must understand and always take into account several mismatches that may occur due to alterations in the implementation of this common interface. For example, error messages and code numbers may vary between the Amazon EC2 implementation and the EC2 interface of an Openstack cloud. Moreover, several Amazon-specific functionalities may not be supported. With these in mind, it is useful to identify these mismatches and develop an automated and extendable suite that shall test whether an EC2 interface at an arbitrary cloud endpoint is compliant to a specific EC2 implementation. More about EC2 interfaces and a developed test-suite may be found in sections 5 and 6.

2 SlipStream Functionality Report

SlipStream is a web interface developed by Sixsq that is currently evaluated by the Helix Nebula initiative as a potential Blue Box solution. The SlipStream application has two main workflows: Image Creation and Deployment.

The Deployment workflow allows the deployment of several virtual machines together, as part of a consistent system. For example, a 3-tier system can be deployed, composed of a client, a server and a database, all running on different machines. The number of machines running clients, servers and databases can easily be altered, since they are parameterised. A deployment may be performed via a *deployment module*. In the following graphic are all the phases of a deployment in SlipStream:

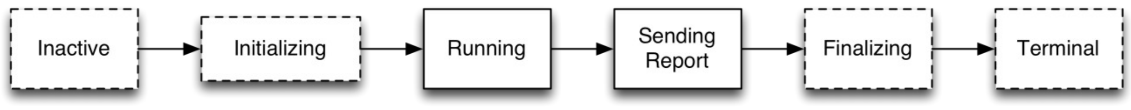


Figure 5 Deployment phases of SlipStream (reference: https://bb.sixsq.com/html/reference-manual.html)

The building blocks of a deployment are machine images (i.e. virtual images), which can also include disk images (i.e. mountable persistent storage for machine images). The creation of these building blocks is performed using the *Image Creation* workflow. The specification of how a node (Instance) will be configured and which image it will get are specified in a *Project.* An important aspect of *Projects* is the property of inheritance that is supported. For example a Project may get several properties from another Project and modify other parameters on the current Project. These relationships make the manipulation of images highly customizable.

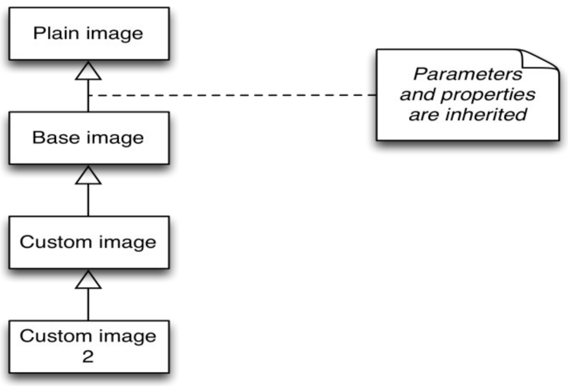
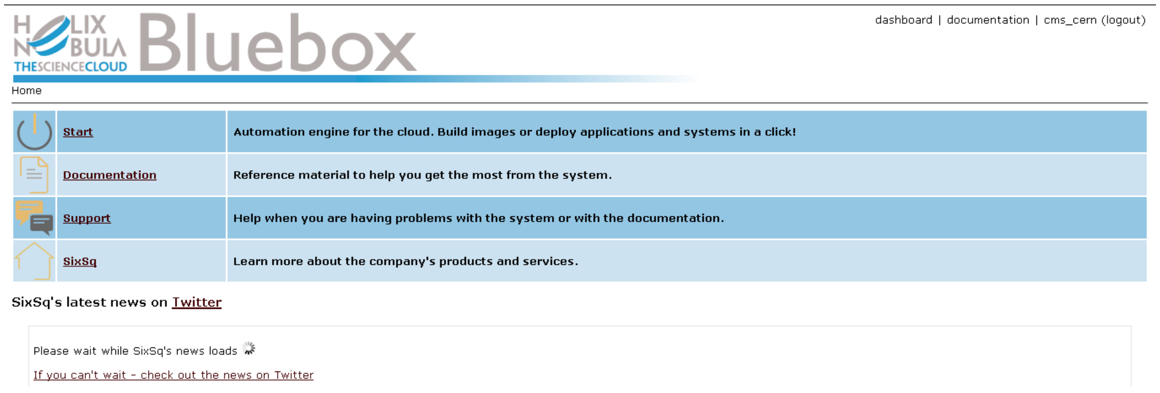


Figure 6 SlipStream image inheritance (source: https://bb.sixsq.com/html/reference-manual.html)

For a more in depth overview of SlipStream, refer to https://bb.sixsq.com/html/reference-manual.html. In this section results of the functionality evaluation will be presented, according to the specifications set by the Helix Nebula initiative.


HELIX NEBULA THESCIENCECLOUD Bluebox
dashboard | documentation | cms_cern (logout)
Home

Start	Automation engine for the cloud. Build images or deploy applications and systems in a click!
Documentation	Reference material to help you get the most from the system.
Support	Help when you are having problems with the system or with the documentation.
SixSq	Learn more about the company's products and services.

SixSq's latest news on Twitter

Please wait while SixSq's news loads

If you can't wait - check out the news on Twitter

Figure 7: The main page of SlipStream

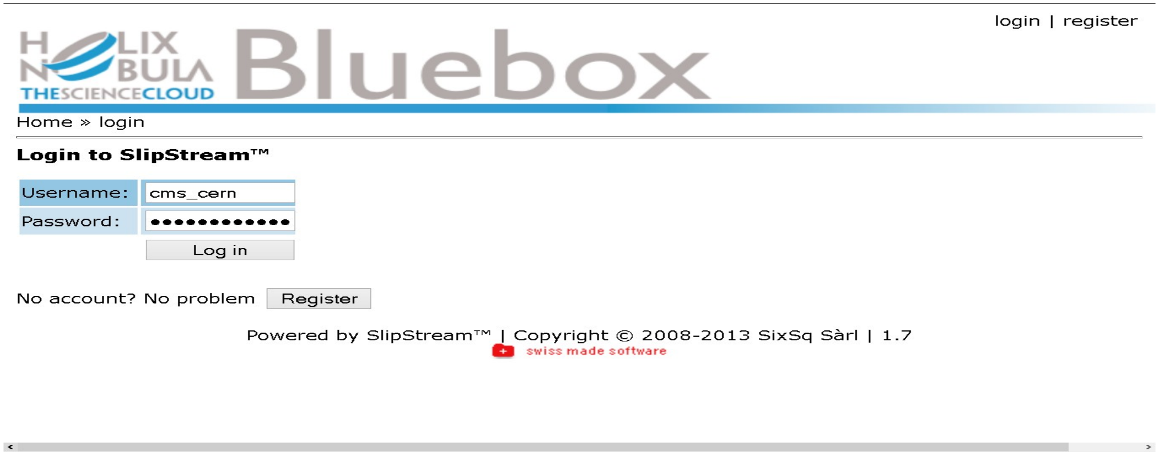
In the current prototype, apart from the interface itself we have three cloud providers that complete the current picture. These are: Atos, CloudSigma and Interoute. When applicable, the native tools (APIs, Web GUIs) of these interfaces are used to perform comparisons and get familiar to native operations. Ideally, however, the BlueBox should be completely independent to these tools because only then it is transparent and provider-independent.

2.1 Authentication mechanisms

A BlueBox, as defined by Helix Nebula, must grant the end-user the appropriate access every time. Thus, in a federated cloud, all authentication parameters must be passed to many service providers. In this sub-section, the authentication parameters offered by SlipStream are described.

Authentication of end-user to SlipStream interface

The end-user logs in with a unique log-in username to the SlipStream interface after registering for the service. After this procedure, he is able to build Deployment modules and Projects (handle images). However, it is not possible for him to actually run a deployment yet as he has not provided any authentication mechanisms on how should SlipStream or the user himself access the virtual machines, which are also mandatory.



Login to SlipStream™

Username:	cms_cern
Password:	●●●●●●●●●●●●

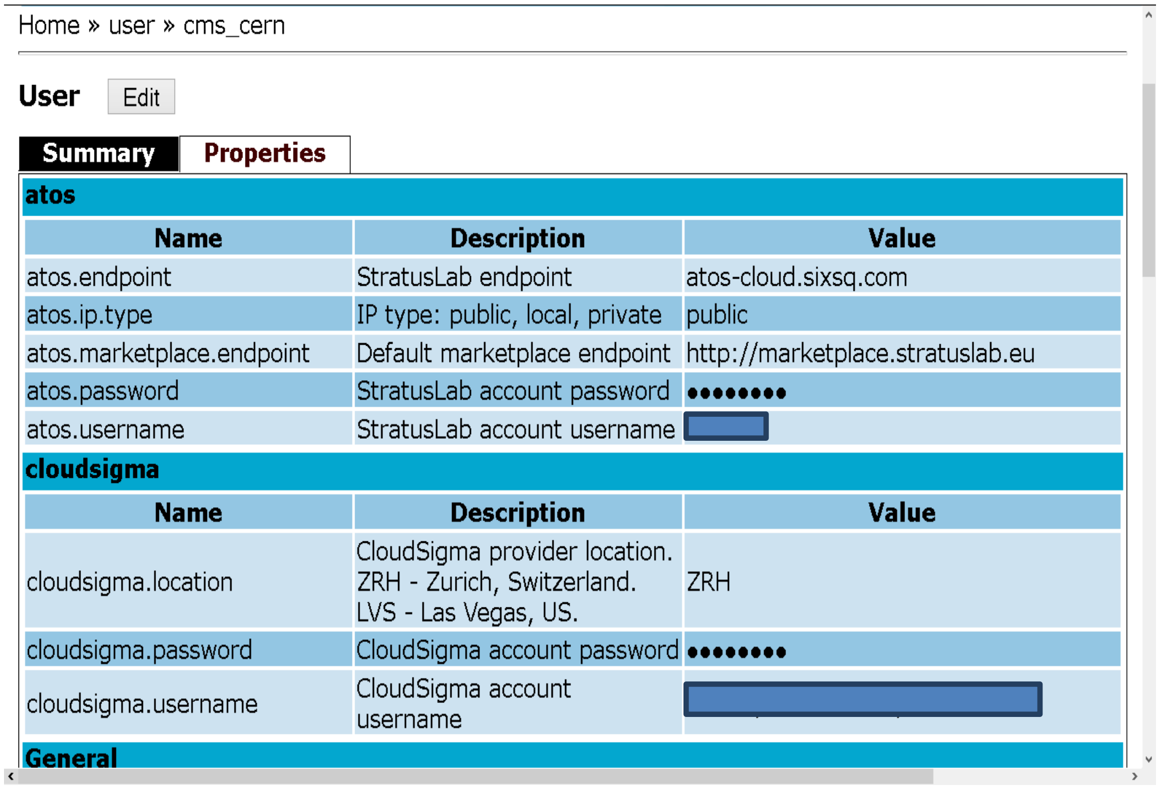
No account? No problem

Powered by SlipStream™ | Copyright © 2008-2013 SixSq Sàrl | 1.7

Figure 6: Logging in to the interface

Cloud Provider Credential

SlipStream requires the user, after logging in, to provide appropriate credentials for the Cloud providers that are taking part in the deployment. These credentials are usually in the form of username and password. It is also very important that the correct cloud endpoint is specified. After completing this level of authentication it is possible to perform a deployment in one or multiple cloud providers.



Home » user » cms_cern

User Edit

Summary Properties

atos

Name	Description	Value
atos.endpoint	StratusLab endpoint	atos-cloud.sixsq.com
atos.ip.type	IP type: public, local, private	public
atos.marketplace.endpoint	Default marketplace endpoint	http://marketplace.stratuslab.eu
atos.password	StratusLab account password	••••••••
atos.username	StratusLab account username	[redacted]

cloudsigma

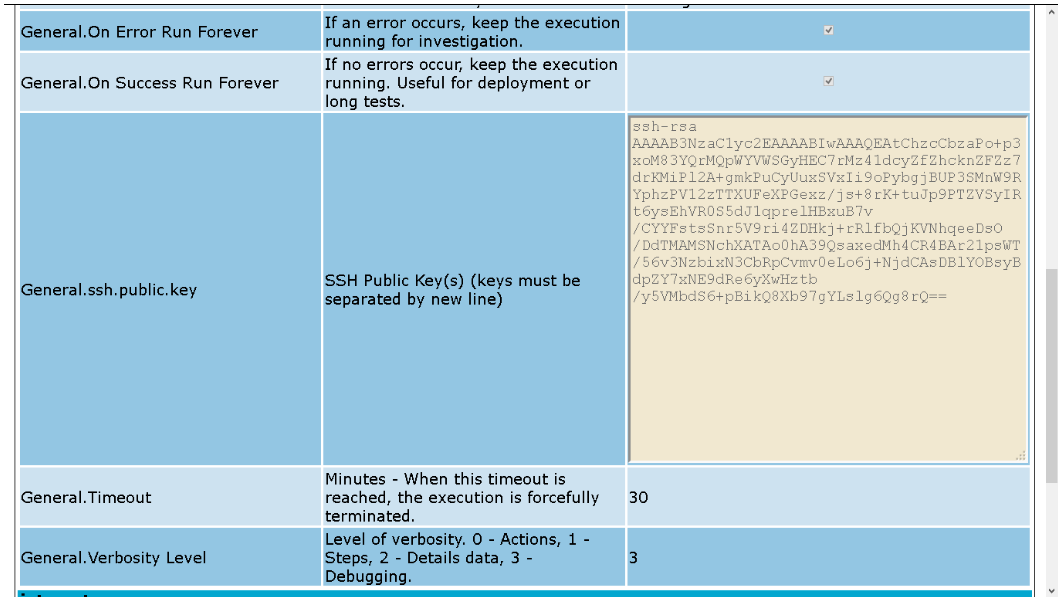
Name	Description	Value
cloudsigma.location	CloudSigma provider location. ZRH - Zurich, Switzerland. LVS - Las Vegas, US.	ZRH
cloudsigma.password	CloudSigma account password	••••••••
cloudsigma.username	CloudSigma account username	[redacted]

General

Figure 7: Providing Cloud providers Credentials

Virtual Machines Credentials

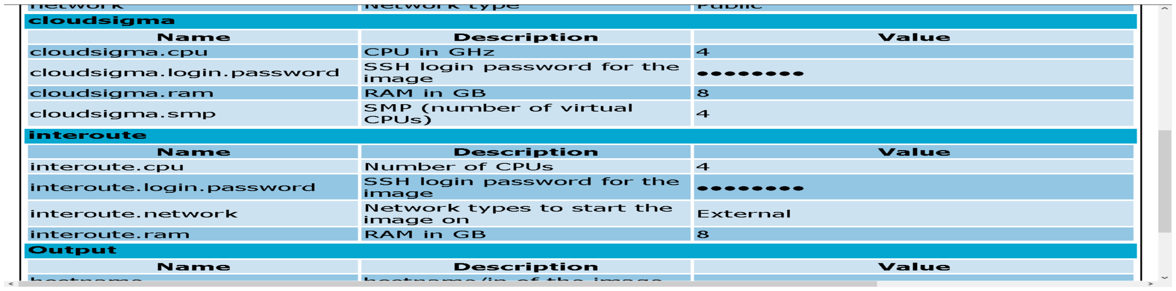
In order to have full control of the virtual machines deployed by SlipStream, it is important to be able to log-in to the VMs via SSH. SlipStream offers public/private key authentication and also a predefined root password for deployments. The public key must be passed to the specific text area box in the user properties page. Once an instance is deployed, the public key is automatically passed by SlipStream's mechanisms. After that, it is immediately possible to perform an SSH authentication by providing the secure private key. This feature is fully functional with all three currently supported providers.



General.On Error Run Forever	If an error occurs, keep the execution running for investigation.	✓
General.On Success Run Forever	If no errors occur, keep the execution running. Useful for deployment or long tests.	✓
General.ssh.public.key	SSH Public Key(s) (keys must be separated by new line)	ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAtChzcCbzaPo+p3xoM83YQrMQpWYVWSGyHEC7rMz41dcyZfZhcknZFZz7drKMiPl2A+gmkPuCyUuxSVxIi9oPybgjBUP3SMnW9RYphzPVl2zTTXUFeXPGexz/js+8rK+tuJp9PTZVSyIRt6ysEhVR0S5dJ1qprelHBxuB7v/CYYFstsSnr5V9ri4ZDHkj+rRlfbQjKVNhqeeDsO/DdTMAMSNchXATAo0hA39QsaxedMh4CR4BAr21psWT/56v3NzbixN3CbRpCvmv0eLo6j+NjdCAsDBlYOBsyBdpZY7xNE9dRe6yXwHztb/y5VMbdS6+pBikQ8Xb97gYLslg6Qg8rQ==
General.Timeout	Minutes - When this timeout is reached, the execution is forcefully terminated.	30
General.Verbosity Level	Level of verbosity. 0 - Actions, 1 - Steps, 2 - Details data, 3 - Debugging.	3

Figure 6: User screen in SlipStream. SSH Public Key is passed here.

The root password may also be passed as a parameter for each node. However, it will be utilized only by deployment modules to perform orchestrations of virtual machines (one orchestrator per provider). This functionality is not available for Atos Cloud, but only for CloudSigma and Interoute.



cloudsigma		
Name	**Description**	**Value**
cloudsigma.cpu	CPU in GHz	4
cloudsigma.login.password	SSH login password for the image	••••••••
cloudsigma.ram	RAM in GB	8
cloudsigma.smp	SMP (number of virtual CPUs)	4
interoute		
Name	**Description**	**Value**
interoute.cpu	Number of CPUs	4
interoute.login.password	SSH login password for the image	••••••••
interoute.network	Network types to start the image on	External
interoute.ram	RAM in GB	8
Output		
Name	**Description**	**Value**

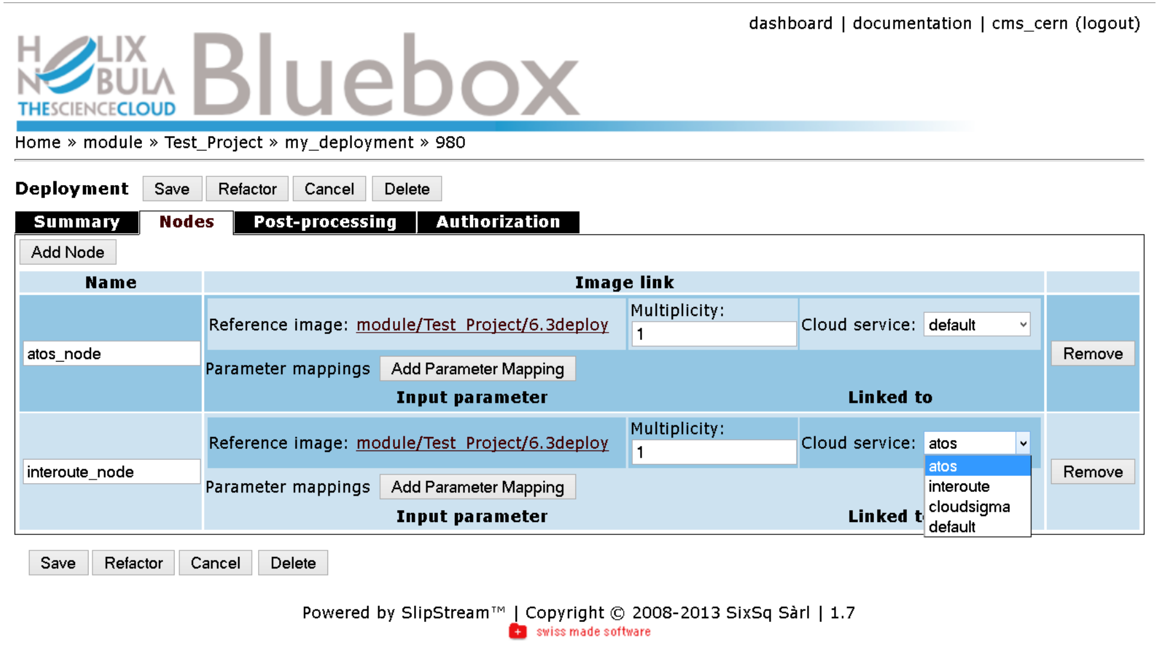
Figure 7: Root Password may be set for each Node.

2.2 Support to perform a VM lifecycle

A BlueBox solution must be totally independent regarding all the matters that concern the management of a virtual machine. Thus, the concepts of booting VMs, listing the available VMs with consistent states and terminating VMs without requiring any interactions with any tool (GUI or other API) apart from BlueBox.

Running an instance

SlipStream allows us to run deployment modules and Projects of images directly (only one VM will be booted in this case). This can happen by clicking the *Run* button either on a Project or a deployment module. Notice that the VM in question will be booted in the default provider that the user sets at his Account Parameters page. However, if this is a deployment module the user can specify which cloud provider he would want to use overriding the default provider setting.



dashboard | documentation | cms_cern (logout)

Home » module » Test_Project » my_deployment » 980

Deployment Save Refactor Cancel Delete

Summary | Nodes | Post-processing | Authorization

Add Node

Name	Image link			
atos_node	Reference image: module/Test_Project/6.3deploy	Multiplicity: 1	Cloud service: default	Remove
	Parameter mappings Add Parameter Mapping			
	Input parameter		Linked to	
interoute_node	Reference image: module/Test_Project/6.3deploy	Multiplicity: 1	Cloud service: atos	Remove
	Parameter mappings Add Parameter Mapping			
	Input parameter		Linked to	

Save Refactor Cancel Delete

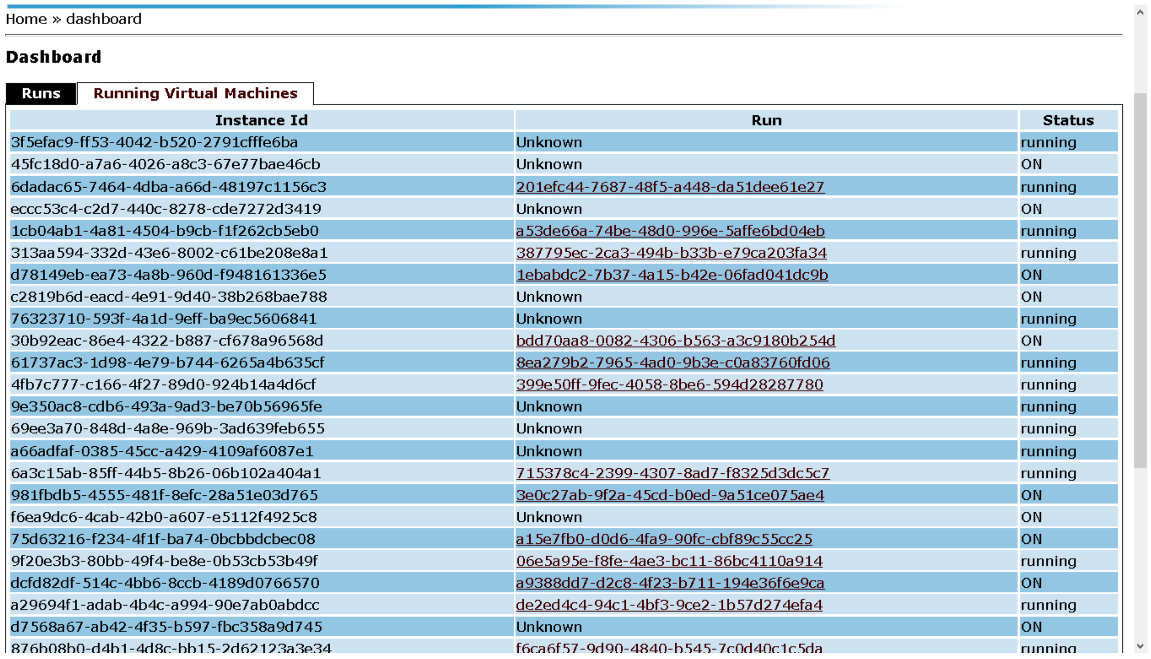
Powered by SlipStream™ | Copyright © 2008-2013 SixSq Sàrl | 1.7

Figure 8: Choosing cloud provider in a deployment module

Describing the currently running VMs

It is possible to list all currently running machines by viewing the Running Virtual Machines page of the dashboard screen. Notice that the listed VMs are not only the VMs that have been booted via SlipStream, but the currently running VMs that are affiliated to the respective cloud provider account. The first column of the page indicates unique Instance IDs of the VMs. The second column contains a value that is generated by SlipStream and acts as a unique ID for each discrete run. However, this column gets the value “unknown” when the run of the instance is a single-Project run and not a deployment. Finally, the third column indicates the status of the VM. The values obtained are from the respective native cloud provider (e.g. ON, running, Running).

Note: In order to make the dashboard appear more homogeneous in next versions of SlipStream the values in the latter column should be replaced with a single word that will reflect the current situation of each provider.

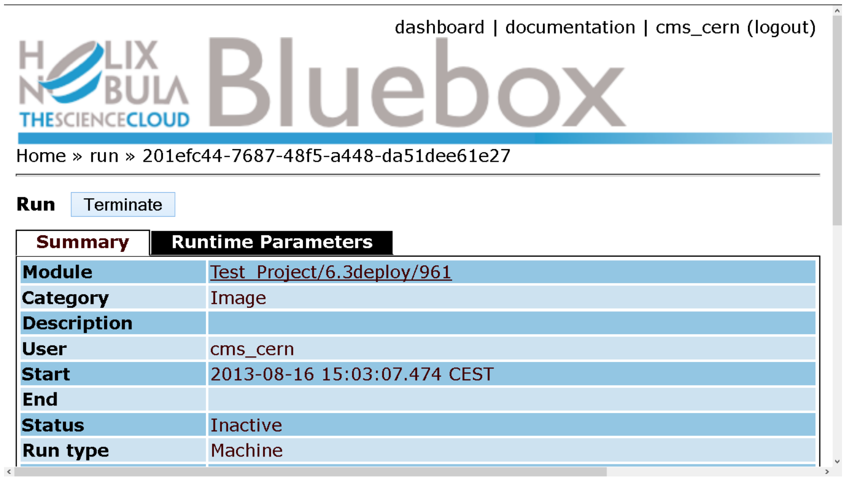


Instance Id	Run	Status
3f5efac9-ff53-4042-b520-2791cfffe6ba	Unknown	running
45fc18d0-a7a6-4026-a8c3-67e77bae46cb	Unknown	ON
6dadac65-7464-4dba-a66d-48197c1156c3	201efc44-7687-48f5-a448-da51dee61e27	running
eccc53c4-c2d7-440c-8278-cde7272d3419	Unknown	ON
1cb04ab1-4a81-4504-b9cb-f1f262cb5eb0	a53de66a-74be-48d0-996e-5affe6bd04eb	running
313aa594-332d-43e6-8002-c61be208e8a1	387795ec-2ca3-494b-b33b-e79ca203fa34	running
d78149eb-ea73-4a8b-960d-f948161336e5	1ebabdc2-7b37-4a15-b42e-06fad041dc9b	ON
c2819b6d-eacd-4e91-9d40-38b268bae788	Unknown	ON
76323710-593f-4a1d-9eff-ba9ec5606841	Unknown	running
30b92eac-86e4-4322-b887-cf678a96568d	bdd70aa8-0082-4306-b563-a3c9180b254d	ON
61737ac3-1d98-4e79-b744-6265a4b635cf	8ea279b2-7965-4ad0-9b3e-c0a83760fd06	running
4fb7c777-c166-4f27-89d0-924b14a4d6cf	399e50ff-9fec-4058-8be6-594d28287780	running
9e350ac8-cdb6-493a-9ad3-be70b56965fe	Unknown	running
69ee3a70-848d-4a8e-969b-3ad639feb655	Unknown	running
a66adfaf-0385-45cc-a429-4109af6087e1	Unknown	running
6a3c15ab-85ff-44b5-8b26-06b102a404a1	715378c4-2399-4307-8ad7-f8325d3dc5c7	running
981fbdb5-4555-481f-8efc-28a51e03d765	3e0c27ab-9f2a-45cd-b0ed-9a51ce075ae4	ON
f6ea9dc6-4cab-42b0-a607-e5112f4925c8	Unknown	ON
75d63216-f234-4f1f-ba74-0bcbbdcbec08	a15e7fb0-d0d6-4fa9-90fc-cbf89c55cc25	ON
9f20e3b3-80bb-49f4-be8e-0b53cb53b49f	06e5a95e-f8fe-4ae3-bc11-86bc4110a914	running
dcfd82df-514c-4bb6-8ccb-4189d0766570	a9388dd7-d2c8-4f23-b711-194e36f6e9ca	ON
a29694f1-adab-4b4c-a994-90e7ab0abdcc	de2ed4c4-94c1-4bf3-9ce2-1b57d274efa4	running
d7568a67-ab42-4f35-b597-fbc358a9d745	Unknown	ON
876b08b0-d4b1-4d8c-bb15-2d62123a3e34	f6ca6f57-9d90-4840-b545-7c0d40c1c5da	running

Figure 9: Running VMs as described by SlipStream

Terminating VMs

In order to terminate a VM or an already running deployment of VMs one must click on the *Terminate* button. Alternatively, it is possible to use the ss-abort (Boolean) command to programmatically order an instance to terminate during execution (contextualization) and reporting phase.


dashboard | documentation | cms_cern (logout)
HELIX NEBULA THESCIENCECLOUD
Bluebox
Home » run » 201efc44-7687-48f5-a448-da51dee61e27
Run
Terminate
Summary
Runtime Parameters

Module	Test_Project/6.3deploy/961
Category	Image
Description	
User	cms_cern
Start	2013-08-16 15:03:07.474 CEST
End	
Status	Inactive
Run type	Machine

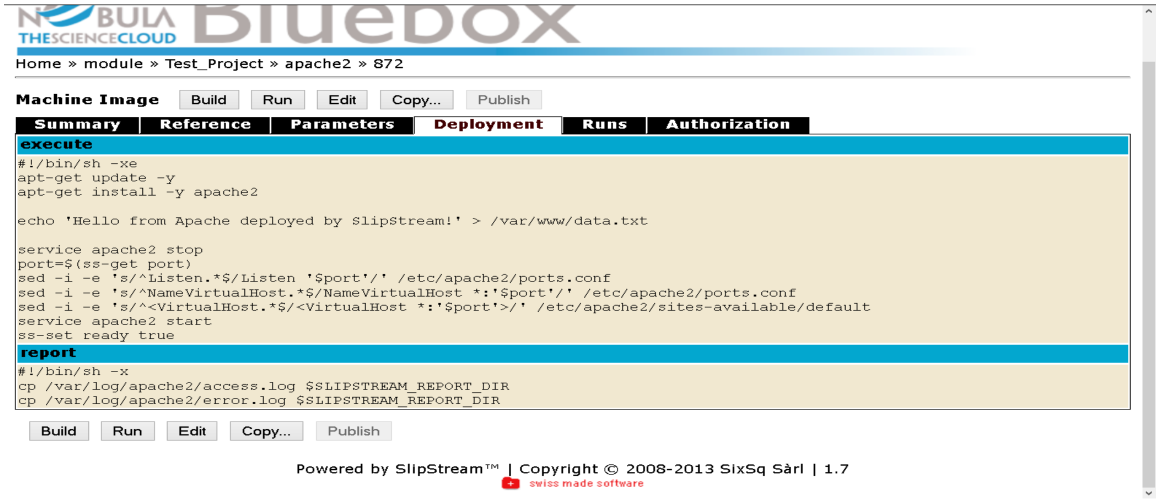
Figure 10: Terminating VMs with SlipStream

2.3 Contextualization Mechanisms

Contextualization of Virtual Machines in the cloud is a common practice that allows instances to learn about their cloud environment and obtain an early configuration in order to run properly and in an automated way. Contextualization is useful as it can minimize user interventions to the Virtual Machines (for example SSH log-in).

SlipStream offers an area where deployment scripts may be specified at the “Deployment” page of a Project. For example, for a Linux system, bash scripts that will be run as soon as the instance is booted on the provider can be defined. The deployment page is split into two parts: *Execute* and *Report.* Scripts of the *Execute* field will run as soon as the deployment is in a *Running* phase, while scripts located in *Report* when run when deployment is in *Reporting* phase. Notice however, that these mechanisms are only applicable to deployment modules and not to single runs of instances.

Finally, the SlipStream python client (ss-* commands) is one more tool that SlipStream offers us and may be used for the better utilization of the interface. At this point it must be clarified that the ss-* python clients are not an alternative to the Web GUI. This means that it cannot be used to administrate cloud operations in a way that is performed by Nova or EC2 (administrator outside the cloud orchestrates all the VMs). However, it is very useful for managing VM-VM interactions and making use of Parameters. In the example below, the apache web server below will use the port we specify in the Parameters screen.



execute

```
#!/bin/sh -xe
apt-get update -y
apt-get install -y apache2

echo 'Hello from Apache deployed by SlipStream!' > /var/www/data.txt

service apache2 stop
port=$(ss-get port)
sed -i -e 's/^Listen.*$/Listen '$port'/' /etc/apache2/ports.conf
sed -i -e 's/^NameVirtualHost.*$/NameVirtualHost *:'$port'/' /etc/apache2/ports.conf
sed -i -e 's/^<VirtualHost.*$/<VirtualHost *:'$port'>/' /etc/apache2/sites-available/default
service apache2 start
ss-set ready true
```

report

```
#!/bin/sh -x
cp /var/log/apache2/access.log $SLIPSTREAM_REPORT_DIR
cp /var/log/apache2/error.log $SLIPSTREAM_REPORT_DIR
```

Figure 11: Performing Contextualization via SlipStream. In the example, the VM will act as a web-server as Apache Web server is being installed.

2.4 Consistency with native provider tools

An important aspect of a Cloud Computing interface that handles many providers is the consistency between the actual state of the instances (as provided from the native clients) and the information displayed by the interface. Therefore, it is important to understand what is displayed during the different phases of the deployment.

SlipStream offers us two statuses for a VM when it is deployed: the "State" of a VM and the "cloud state".

States of VM phases are defined by SlipStream and are the following:

Inactive -> Initializing -> Running -> Sending -> Report -> Finalizing -> Terminal

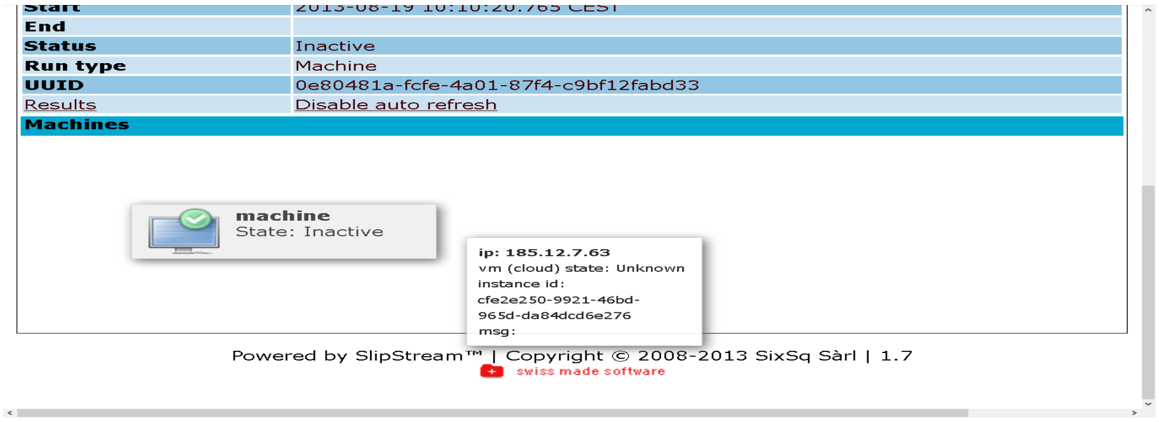
If the deployment is successfully completed the run ends specifying "Done". If "Terminate" button is pressed they are heading to "Aborting" phase. Notice that “VM State” messages are only applicable to deployment modules (orchestrations). Otherwise, they are always Inactive.

The "cloud state" field gets its values from the provider's API or web GUI. If this is not possible the cloud state is "unknown".

Here are some remarks about client consistency for each one of the providers:

- Atos Cloud: Atos Cloud Instances are generally very consistent to the native Stratus lab Client. Orchestrations are carried out smoothly and VM States are progressing accordingly to the phases specified above. When single VMs are booted (no orchestrator) the Cloud States are synchronized with the Stratus Client.
- CloudSigma: Deployments are generally progressing well. However, when virtual machines are deployed solely, the cloud states remain unknown and are not updated as they should be.
- Interoute: Cloud states are always unknown despite the fact that IP addresses and other vital information are displayed properly.

a



Start	2013-08-19 10:10:20.765 CEST
End	
Status	Inactive
Run type	Machine
UUID	0e80481a-fcfe-4a01-87f4-c9bf12fabd33
Results	Disable auto refresh
Machines	

machine
State: Inactive

ip: 185.12.7.63
vm (cloud) state: Unknown
instance id:
cfe2e250-9921-46bd-965d-da84dcd6e276
msg:

Powered by SlipStream™ | Copyright © 2008-2013 SixSq Sàrl | 1.7
swiss made software

Figure 12: An Interoute single VM launch always has unknown VM (cloud) state, meaning it does not get data from the Interoute cloud provider properly.

2.5 Multi-Cloud Deployments

A multi-cloud deployment is a functionality that allows the user to orchestrate in one single deployment resources from different cloud providers. This is a concept which would be very desirable to the Helix Nebula initiative for potential BlueBox solutions. SlipStream offers this functionality by allowing the user to create a deployment module that uses nodes where at least one virtual machine is not in the default cloud provider. For example, below is a deployment of two Centos Instances where one is in Atos Cloud and other is in Interoute.

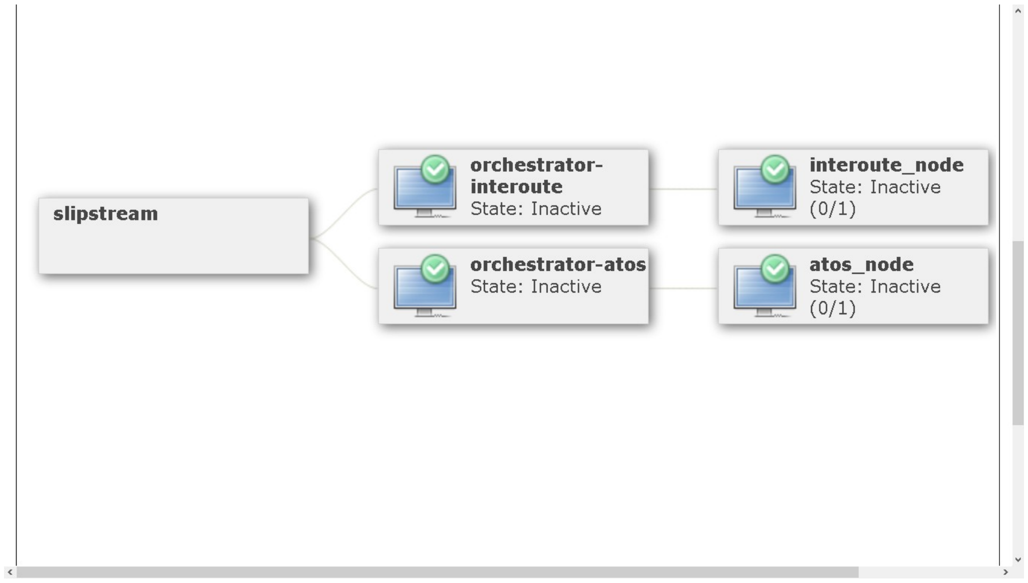


Figure 10: Interoute and Atos deployment.

Notice that when we request 2 instances we also get one orchestrator instance per Cloud provider. Therefore, with such a query, 4 instances are running. The orchestrator acts as the instance responsible to pass all necessary data that the user has on SlipStream (authentication tokens, contextualization scripts) and some necessary SlipStream software (ss-client).

2.6 Performance Tests

In this section you may find the results of performance tests. These tests indicate operations delays between various use cases of the interface and (in some cases) measure monitoring unreliability. Comparisons have been made when applicable with native clients and GUIs.

It is important to examine for each provider both single *Project* runs and *Deployment* runs.

The following tables are an overview of the results that were gathered:

Tests Cases / Cloud Providers	Atos	Atos (depl.)	CS	CS (depl)	IR	IR (depl)
Start BB – Booting Provider	7s	7s, 92s	68s	100s, 255s	21s	21s, 200s
Start BB – Runs Provider	80s	7s, 92s	70s	100s, 270s	23s	25s,200 s
Start BB – Runs BB	80s	7s, 92s	70s	100s, 285	40s	40s,205 s
Start Provider – Runs on provider	80s	-	-	-	20s	-
Start provider – runs on BB	82s	-	-	-	30s	-

Tests / Cloud Providers	Atos	Atos (depl.)	CS	CS (depl)	IR	IR (depl)
Delete on BB – Deleting on Provider	0	0	-	-	-	-
Delete on BB – Delete on Provider	2s	2s	1s	1s	15s	20s
Delete on BB – Deleted on BB	2s	2s	1s	2s	Never	Never
Run and Delete on BlueBox – Deleting on Provider	2s	2s	5s	5s	-	-
Run and Delete on Provider – Deleted in BB	4s	4s	4s	4s	-	-

Notes:

- For deployment runs two numbers are specified: time of booting an orchestrator and time of booting the actual instances.
- At the time of testing, it has not been possible to terminate Interoute instances via SlipStream.

3 De facto standards and Amazon EC2

3.1 SlipStream EC2 Bridge

A prototype of an EC2 bridge is currently being implemented at: https://github.com/SlipStream/SlipStreamEC2. Currently, this prototype supports the following EC2 Actions:

- **RunInstances**: Starts an instance. Multiplicity not supported yet.
- **DescribeInstances**: Lists the instances that are currently deployed (very basic, mapping of UUIDs and Images).
- **TerminateInstances**: Terminates an instance. Multiplicity not supported yet.

However, two minor changes had to be made in order to make the SlipStream-EC2 Bridge fully functional. One issue that was resolved involved modifications to the build system (Maven dependencies) while the other change ensured that the final HTTP request was functional for DescribeInstances command.

Pull request here: https://github.com/SlipStream/SlipStreamEC2/pull/1

Or clone: https://github.com/atsikiridis/SlipStreamEC2

Please view the README file for installation instructions.

3.2 EnStratius EC2 Bridge

Enstratius is another potential BlueBox solution that is being evaluated by the partners of the Helix Nebula initiative. Although, the evaluation of Enstratius is out of the scope of this report and project, here are some remarks about the EC2 interface which were briefly examined:

- The interface has one endpoint per cloud provider and not one endpoint for the whole interface. Two cloud providers are currently supported by EnStratius: T-Systems Cloud and CloudSigma. The EC2 endpoints are: ts.ess.helix-nebula.eu and cs.ess.helix-nebula.eu. This is different from the SlipStream EC2 prototype that is described in 3.1 mainly because the EC2 endpoint is pointing to the provider(s) and not the interface.
- The EC2 credentials are generated by the interface by providing a name for each pair.
- Generally, this interface requires a more specific configuration in order to run and the queries created are not compatible with the EC2 de facto standard.~~-~~For example, the standard authentication parameters for an EC2 interface are the access key (EC2_ACCESS_KEY) and the secret key (EC2_SECRET_KEY). After testing the T-Systems Cloud endpoint the required value of the secret key for a valid cloud operation is “NotSet” (hard-coded value). This means that the secret key (EC2_SECRET_KEY) is not really utilized. In the same time, the access key has an **accesskey:secretkey** format, thus including both necessary credentials. While this approach does not compromise the functionality of the EC2 Bridge, it is not standard for EC2 interfaces, as the structure of an http query may be significantly different (see the appendix for more information about HTTP queries to EC2 interfaces). ~~An example is the use non usage of the Secret key with a predefined not set value~~.

3.3 CERN private cloud

The CERN private cloud based on Openstack supports an EC2 interface. However, several commands are not supported when using known EC2 APIs such as Eucalyptus euca2ools (you may find the commands supported by euca2ools here: http://docs.openstack.org/developer/nova/runnova/euca2ools.html). In this section you may find a functionality evaluation of direct HTTP queries against the CERN private cloud. The reason that the HTTP query API was preferred is in order to understand all interactions and error messages that the endpoint issues.

Please view the Appendix for a complete reference of the commands: https://its.cern.ch/jira/browse/CSCLOUD-92

Note: All tests were performed against: http://ibex.cern.ch

4 A Test Suite for EC2 Interfaces

4.1 Motivation

In this section you may find details about the implementation of a test-suite that performs black-box testing to a Cloud Endpoint. The test results reflect whether the cloud endpoint is compliant against specific implementations of EC2 interfaces (AWS, Openstack, OpenNebula).

Currently, our team has developed a prototype that ensures that basic commands can be performed against the Openstack standard EC2 API. However, the test-suite is extensible and can easily support other flavours. The test suite is developed in Python and may be found here:

https://github.com/atsikiridis/EC2TestSuite

4.2 Example Usage

1) Source into your system the variables EC2_ACCESS_KEY, EC2_SECRET_KEY and EC2_URL.
2) $ sudo python setup.py install
3) Install prettytable in your preferred way.
4) 4) $ec2test -f openstack -i *yourimage*

The last command has two required arguments: the EC2 implementation standard (flavor) against which we test the cloud endpoint and a valid imageId. The result is a functionality report of the endpoint. A –debug flag enables debugging information.

```
INFO:Ec2TestSuite:

+------------------------+--------------+

| Configuration parameter |    Value     |

+------------------------+--------------+

|         debug          |    False     |

|         flavor         |  openstack   |

|        imageId         | ami-00000181 |

+--------------------+--------------------+---------------+

|       Action       |  Parameters used   | Is Functional |

+--------------------+--------------------+---------------+

|   DescribeImages   |                    |      Yes      |

| DescribeInstances  |                    |      Yes      |
|   RunInstances   |                  |     Yes     |
| TerminateInstances |                  |      No     |
+-------------------+-------------------+---------------+
```

5 Conclusions

SlipStream is a potential BlueBox solution that generally abides with the specifications set by the Helix Nebula initiative. Although some parts are not always consistent or fully functional, it is a result of the fact that the interface is still a prototype. Moreover, the performance measurements heavily rely on parameters regarding cloud computing providers' functionalities and are thus outside the scope of SlipStream.

The EC2 interface is an important functionality for a BlueBox and despite the fact that SlipStream is not yet totally EC2 compatible, the prototype provided by sixsq is now in a working state. Our needs to evaluate EC2 interfaces (Helix Nebula, Openstack, Amazon, OpenNebula, Enstratius) underlines the importance of an automated testing mechanisms for EC2 interface compliance and has led us to develop a prototype testsuite which is described in section 4.

6 References

[1] http://cloudtaxonomy.opencrowd.com/taxonomy/software-as-a-service/

[2] "The NIST Definition of Cloud Computing". National Institute of Science and Technology. Retrieved 24 July 2011.

[3] Amies, Alex; Sluiman, Harm; Tong, Qiang Guo; Liu, Guo Ning (July 2012). "Infrastructure as a Service Cloud Concepts". *Developing and Hosting Applications on the Cloud*. IBM Press. ISBN 978-0-13-306684-5.

[4] SlipStream Reference manual: https://bb.sixsq.com/html/reference-manual.html

[5] J. de la Mar, P. Evans, M. Symonds, R. Jenkins, M. E. Begin, J. Graham, P. Parsons : Helix Nebula – The Science Cloud: Blue Box Approach , version 1.6p

7 Appendix

EC2 OpenStack Complete Command Reference

In this summary you can find all possible API calls that can be performed on AgileInfrastructure (http://ibex-cloud-controller.cern.ch:8773/services/Cloud). All calls were tested using http requests on the endpoint of AI following the format indicated here: http://docs.aws.amazon.com/ElasticLoadBalancing/latest/DeveloperGuide/using-query-api.html.

Signing Process and valid HTTP requests

Note: The boto library, euca2ools and other APIs use this method to interact with the cloud's endpoint. The purpose of this summary is to replicate these requests manually with none of these APIs in order to understand the functionalities supported by an EC2 interface, an openstack cloud and AI specifically.

In order to prepare a valid http request for AI, several authentication parameters (AUTHPARAMS) must be set to the following values:

- URL=http://ibex-cloud-controller.cern.ch:8773/services/Cloud
- AWSAccessKeyId=****yourEC2accesskey****
- SignatureVersion=2
- SignatureMethod=HmacSHA256
- Version=2010-08-31 (or newer)
- Timestamp=****current**** (Format: %Y-%m-%dT%H:%M:%S.000Z)
- Signature=****computed****

In order to compute the signature follow this guide: http://docs.aws.amazon.com/general/latest/gr/signature-version-2.html

Very useful scripts available in python and perl that work perfectly to produce valid http requests and signatures and issue them to the Cloud's endpoint are in this github project: https://github.com/jeffk/ec2_signer.

GET is used by default. The request method may by altered in the code.

Actions and Parameters supported

The list of actions supported by AI is a subset of Actions supported by AWS (found here: http://docs.aws.amazon.com/AWSEC2/latest/APIReference/query-apis.html).

Each of the actions has several associated parameters. Here are all valid Actions to be performed on AI:

Action	Parameters (r= required, nr= not required, c=conditional)	Description
AllocateAddress	Domain=vpc(r)	Allocate a floating IP from a given floating ip pool.
AssociateAddress	PublicIp(r), InstanceId(r), PrivateIpAddress(nr), AllowReassociation(nr)	Associate floating IP to instance.
AttachVolume	VolumeId(r), InstanceId(r), Device(r)	Attach Volume to running instance with the name of device.
AuthorizeSecurityGroupIngress	GroupName(r), IpPermissions.n.IpProtocol(r), IpPermissions.n.FromPort(r), IpPermissions.n.ToPort(r), IpPermissions.n.Groups.m.GroupName(c), IpPermissions.n.IpRanges.m.CidrIp(c)	Adds one or more ingress rules to a security group.
CreateImage	InstanceId(r), Name(r), Description(nr), NoReboot(nr), BlockDeviceMapping params(nr)	Creates an image.
CreateKeyPair	KeyName(r)	Creates a new 2048-bit RSA key pair with the specified name.
CreateSecurityGroup	GroupName(r), GroupDescription(r)	Creates a security group.
CreateSnapshot	VolumeId(r), Description(nr)	Creates a snapshot of a volume.
CreateVolume	AvailabilityZone(r), SnapshotId(c), Size(nr), VolumeType(nr), Iops(c)	Creates a volume that can be attached to an instance.

DeleteKeypair	KeyName(r)	Deletes a keypair.
DeleteSecurityGroup	GroupName(r)	Deletes a security group.
DeleteSnapshot	SnapshotId(r)	Deletes a Snapshot.
DeleteVolume	VolumeId(r)	Deletes a volume.
DeregisterImage	ImageId(r)	Deregisters an AMI.
DescribeAddresses	PublicIp.n(nr),AllocationId.n(nr), filter params(nr)	http://ibex-cloud-controller.cern.ch:8773/services/Cloud/?Action=DescribeAddresses&AUTHPARAMS
DescribeAvailabilityZones	ZoneName.n(nr), filter params(nr)	http://ibex-cloud-controller.cern.ch:8773/services/Cloud/?Action=DescribeAvailabilityZones&AUTHPARAMS
DescribeImageAttribute	ImageId(r), Attribute(r)	Describes an attribute of an image.
DescribeImages	ImageId.n(nr), ExecutableBy.n(nr), Owner.n(nr), filter params(nr)	Describes Images.
DescribeInstanceAttribute	InstanceId(r), Attribute(r)	Describes an attribute of an image.
DescribeInstances	InstanceId.n(nr), filter params(nr)	Describes images.
DescribeKeyPairs	KeyName.n(nr), filter params(nr)	Describes keypairs.
DescribeRegions	RegionName.n(nr), filter params(nr)	Describes Regions.
DescribeSecurityGroups	GroupName.n(nr), filter params(nr)	Describes security Groups.
DescribeSnapshots	SnapshotId.n(nr), Owner.n(nr),	Describes snapshots.

	RestorableBy(nr), filter params(nr)	
DescribeVolumes	VolumeId.n(nr), filter params(nr)	Describes Volumes.
DetachVolume	VolumeId(r),InstanceId(nr), Device(nr), Force(nr)	Detaches Volume from running instance.
DisassociateAddress	PublicIp(r)	Disassociates IP from instance.
GetConsoleOutput	InstanceId(r)	Retrieves console output for the specified instance.
ImportKeyPair	KeyName(r), PublicKeyMaterial(r)	http://ibex-cloud-controller.cern.ch:8773/services/Cloud/?Action=ImportKeyPair&InstanceId=someInstance&AUTHPARAMS
RebootInstances	InstanceId.n(r)	Reboots one or more instances.
RegisterImage	ImageLocation(c), Name(r),Description(nr),Architecture(nr),KernelId(nr), RamdiskId(nr), RootDeviceName(nr, Block Device params(nr)	Registers an Image.
ReleaseAddress	PublicIp(r)	Releases Adddress from available floating IPs.
RunInstances	ImageId(r), MinCount(r), MaxCount(r), KeyName(r), SecurityGroup.n(nr), UserData(nr), InstanceType(nr), Placement.AvailabilityZone(nr), Placement.GroupName(nr), Placement.Tenancy(nr), KernelId(nr), RamdiskId(nr), Blocking params(nr), Monitoring.Enabled(nr), DisableApiTermination(nr), InstanceInitiatedShutdownBehavior(nr)	Runs instances.

StartInstances	InstanceId.n(r)	Starts one or more instances.
StopInstances	InstanceId.n(r)	Stops one or more instances.
TerminateInstances	InstanceId.n(r)	Shuts down one or more instances.