Current IaaS/SOI APIs

Foreword:

This is draft version of this IaaS API review. There will be further updates and revisions of this document, especially around the areas of storage and networking. There maybe links that are inaccessible if you do not have SLA@SOI project membership, but if you have any queries in respect to the content pointed to by those links please contact Andy Edmonds (andrewx.edmonds@intel.com).

Below is a listing of Interfaces offered by IaaS providers. The API review does not account for providers that offer Storage as a Service, Integration (middleware) as a Service or IaaS management service (e.g. auto-scaling, image creation/management). Also the review is currently only focused on the aspect of each interface that pertains to the creation of compute resources. It does not review means for attaching storage, specifying network configuration. These details will be added in future revisions. This is to allow us to focus on what functionality the SLA@SOI A4 Infrastructure Management work package needs to offer and what interfaces task TA4.2 needs to support for the 1st year. What's described here are APIs related to Core IaaS that is providers that do not resell infrastructure offered by other providers (e.g. rightscale who act as proxy to EC2). Data schemas and models related to infrastructure provisioning can be found under the TA4.1 page.

Scope

Any infrastructure as a service provider that publishes an API and corresponding schema (data model) that allows the creation of compute resources. Storage and networking aspects are not reviewed for now.

General Requirements

- Create, update, delete, display attributes of a resource. The resource can be either virtual or physical.
  - Update should imply reconfiguration in the relevent context
- Perform basic life cycle operations on provided resources.
  - start, stop, suspend, resume
- Group resources logically - a cluster.
- Group groups of resources logically - a domain or "data centre".
- Requests to the interface should be functionally complete - document-oriented. These requests will form the agreement between customer and provider and in effect establish a SLA.
- Resources that are provided should be monitorable and verifiable (e.g. you get what you ask for)
- Limit functional leakage in the interface. For example, seperate the management of hyervisors and the management of resources created by a hypervisor. For example don’t allow the listing of all VMs under a hypervisor. This is something that is done by the likes of OpenNebula and Enomaly.
- Like with the Sun Cloud API, there should be one single point of entry that allows the discovery of all other resources belonging to the customer. This is an aspect of REST but one that can be simulated in other approaches.

Evaluation Criteria

General:

API Reference: a link to the canonical location of the API

C**ommunication Pattern**: {synchronous, asynchronous}

Web Service Technology: {RPC, Document, SOAP, REST}

SLA: reference/description of the offered SLA by the provider. Is {automated, manual}.

Data Schema review: link to it

By Staging and Image Management Function:

- Function name, description, interface, supported virtual images and applicances

By Runtime-function:

- **Compute-resource** oriented operations
  - Function name, description, interface
  - This will be the first area subject to the API review
- **Storage-resource** oriented operations
  - Function name, description, interface
- **Network-resource** oriented operations
  - Function name, description, interface
- **Ancillary-resource** oriented operations
  - Resource type, function name, description, interface

Orthogonals

- Security
- Configuration
- Other?

Core IaaS/SOI APIs

Amazon EC2

- API: http://docs.amazonwebservices.com/AWSEC2/latest/DeveloperGuide
- Communication Pattern: Synchronous
- Web Service Technology: Primary is by WSDL interface over HTTP-SOAP. There is an interface that can be used somewhat like a REST-style approach
- **SLA:** http://aws.amazon.com/ec2-sla/ User is responsible for providing evidence of violations. SLA is manual.
- Data Schema review

Compute-resource oriented operations	
Operation Name	Description
runInstances	The runInstances operation launches a specified number of instances.
terminateInstances	The TerminateInstances operation shuts down one or more instances.
rebootInstances	Reboots one or more specified instances.
describeInstances	The DescribeInstances operation returns information about instance representations that you own.
confirmProductInstance	The ConfirmProductInstance operation returns true if the specified product code is attached to the specified instance. Related to windows installations and their licensing (to confirm)

Comments / Observations from SLA@SOI perspective

- (-) EC2 interface explicates the types of machines that can be provisioned via the runInstances method. This does not allow for custom-defined machines such that we will have in SLA@SOI.
- (-) The describeInstances specifies an instance's representation in a non-standard fashion - this could perhaps return the instance representation using OVF.
- (+) An almost defacto standard in infrastructure as a service providers
- (-) No means to specifiy non-functional requirements
- (+) Document oriented interface

- CPU/Mem Sizing: Fixed by CPU/Mem combo, T shirt model (Small, Medium, Large), compute intense/mem intense

The Sun Cloud API

- API: http://kenai.com/projects/suncloudapis/pages/Home
- Communication Pattern: Synchronous
- Web Service Technology: REST interface over HTTP. The base URI to resources is X.cloud.sun.com, where 'X' is an unique virtual data center ID. Provisioned resources are grouped in clusters. TO BE CONFIRMED
- SLA: No SLA is currently available as there is only the API, no service
- Data schema review

Compute-resource oriented operations	
Operation Name	**Description**
POST /clusters/clusterX	Post the VM representation to that cluster's URI. The VM representation does not need to be complete; only the core information should be present. Note at this stage, the resources are not deployed. The VM however will be associated with the cluster.
GET /clusters/123456	Gets information about a specific cluster.
PUT /clusters/123456	Updates information about a cluster.
DELETE /clusters/123456	Deletes a cluster and all the resources grouped by it.
POST /cluster/deploy?name=XYZ	Performs the control action "deploy" on the named cluster. The full list of actions that can be executed on a cluster is: • deploy, undeploy, start, stop Each action is associated with a state of the same name.
GET /vms/XYZ	Gets the representation of a particular VM, identified by XYZ that may be associated with a cluster.
PUT /vms/XYZ	Updates the named VM with new attributes.
DELETE /vms/XYZ	Deletes the named VM
POST /vms/XYZ?control=start	Performs the control action "start" on the named VM. The full list of actions that can be executed on a VM is: • deploy, start, stop, undeploy, reboot, hibernate, resume Each action is associated with a state of the same name.

Comments / Observations from SLA@SOI perspective

- (+) The seperation of requesting resources and deploying them (executing the request) is useful as this pushes control to the client and enables the client control when to deploy (future timed provisioning).
- (+) This API's notion of a cluster of VMs maps very well to the current SLA@SOI ProvisioningRequest, which is a collection of virtual machines, configured (network) in a particular manner.
- (-) Does not include any notion of machine readable SLAs

Flexiscale

- API: http://api.flexiscale.com
- Communication Pattern: Synchronous. Those methods that return a job ID can issue a call back check to wait notification of job completion (ref method: WaitForJobs(jobID)).
- Web Service Technology: WSDL interface over HTTP-SOAP
- SLA: http://www.flexiscale.com/FlexiScale_FAQ.pdf "We offer 99.99% uptime."
- Data schema review

Compute-resource oriented operations	
Operation Name	Description
ListServers	Lists customers servers and returns servers machine representations as a name-value map.
CreateServer	Creates a server based on a name-value map of hardware and OS parameters. Returns a Job ID.
StartServer	Starts a server with the given server ID. Returns a Job ID.
StopServer	Stops a server with the given ID and the stop method type (hard, soft). Returns a Job ID.
StopStartServer	Stops and starts a server with the given ID. Returns a Job ID
ModifyServer	Allows a server be reconfigured including the basics; ram, cpu, disk etc. The response is of the new machine representation. A StopStartServer call is then required to execute the new configuration
RebootServer	Reboots a server with the given server ID. Returns a Job ID.
DestroyServer	Deletes a server, returning true or false to indicate success or not

Comments / Observations from SLA@SOI perspective

- (-) No inbuilt capability to deal with a grouping of servers in a request on their service side. This is possible through repeated calls to their interface.
- (-) The createServer/listServers specifies an server's representation in a non-standard fashion - this could perhaps return the instance representation using OVF.
- (+) Very simple API to use. No complex types.
- (-) No means to specify non-functional requirements
- CPU/Memory: Proportional, more Mem means more CPU automatically. Storage is dealt with separately

ElasticHosts

- API: http://www.elastichosts.com/products/api
- Communication Pattern: Synchronous
- SLA: http://www.elastichosts.com/benefits/peace-of-mind (SLA = peace of mind that we know what we are doing)
- Web Service Technology: REST interface over HTTP supporting only GET and POST verbs. The base URL to resources is https://X.api.elastichosts.com, where 'X' is an availability zone.

Compute-resource oriented operations	
Operation Name (operation, resource)	Description
GET /servers/list	For each server, a list of {SERVER, CPU, MEM}.
GET /servers/SERVER-UUID/info	Key-value pairs that represent the server specs also the cumulative received byte count for NICs and the cumulative transmitted byte count for NICs.
POST /servers/create	Key-value pairs for server configuration and returns the same data representing the instantiation of requested server.
POST /servers/SERVER-UUID/set	Key-value pairs that represent the server attributes to change and the name of the server to make the changes to.
POST /servers/SERVER-UUID/destroy	Returns a HTTP 204 No Content. Will do a hard shutdown of said server and delete.
POST /servers/SERVER-UUID/shutdown	Returns a HTTP 204 No Content status code. Will then perform a soft ACPI shutdown.
POST /servers/SERVER-UUID/reset	Returns a HTTP 204 No Content and soft reboots the server.

Comments / Observations from SLA@SOI perspective

- (-) No way to dynamically change your availability zone
- (-) No means to specifiy non-functional requirements
- (-) No elasticity per server - to have elastic behaviour one needs to utilise their API directly
- (-) some inconsistencies with REST-style architecture e.g. the HTTP verb DELETE should be used to destroy a server
- CPU/MEN Sizing: GUI based interface, Any size or prefined templates

GoGrid

- API: http://wiki.gogrid.com/wiki/index.php/API
- Communication Pattern: Synchronous
- Web Service Technology: REST interface over HTTP supporting only GET and POST verbs.
- **SLA:** GoGrid provides a SLA with hard values that their services aim to adhere to.

Compute-resource oriented operations	
Operation Name	Description
grid.server.list	Lists all the servers in the system.
grid.server.get	Retrieves one or many server representations from your list of servers.
grid.server.add	Adds a single server representation to your grid.
grid.server.delete	Deletes a single server representation from your grid.
grid.server.power	This call will issue a power command to a server object in your grid. Supported power commands are: • power on (start), power off (stop), power cycle (restart)

Comments / Observations from SLA@SOI perspective

- (-) The list method returns all servers available, however it doesn not group them by application-grouping.
- (-) The delete method is inconsistent with a REST-style approach. It should use the HTTP DELETE verb.
- (-) Appropriate HTTP verbs for each method are not specified in the documentation

Enomaly - Enomalism

- **API:**
  - Internal API: http://enomalism.com/api/enomalism/
  - REST API: http://enomalism.com/api/enomalism/enomalism_rest_api.pdf
- Communication Pattern: Synchronous
- Web Service Technology: REST interface over HTTP
- **SLA:** Enomalism is a product and enomaly does not offer a service with a corresponding SLA

Compute-resource oriented operations	
Operation Name	Description
GET or POST /rest/machines or /rest/machine	Lists the machines associated with an account. Returns a list of machine IDs
GET or POST /rest/machine/<UUID>	Perform an update on the machine (configuration?) or retreive its machine representation
GET or DELETE /rest/hypervisor/<UUID>	Perform an action on the named hypervisor
GET or POST /rest/machine/<UUID>/hypervisors	Gets a listing of all the hypervisors installed on a machine.
GET or POST /rest/machine/<UUID>/clusters	Gets a list of clusters to which the named machine belongs to.
POST /rest/machine/<UUID>/actions/unlock	Unlocks a locked machine
GET or POST /rest/machine/<UUID>/actions	Gets a list of actions available on a machine.
POST /rest/machine/<UUID>/actions/suspend	Suspends the named machine
POST /rest/machine/<UUID>/actions/resume	Resumes the named machine
POST /rest/machine/<UUID>/actions/shutdown	Shutsdown the named machine
POST /rest/machine/<UUID>/actions/poweroff	Powersoff the named machine
POST /rest/machine/<UUID>/actions/reboot	Reboots the named machine
POST /rest/machine/<UUID>/actions/create	Creates the named machine
POST /rest/machine/<UUID>/actions/store_xml	Stores a machine definition file to be used by the create action
POST /rest/machine/<UUID>/actions/manage	Makes the machine "manageable"
POST /rest/machine/<UUID>/actions/totaldelete	Completely deletes the referenced machine
POST or PUT /rest/machine/<UUID>/actions/provision	Provisions the named machine

Comments / Observations from SLA@SOI perspective

- (-) From a REST point of view the machine listing action is broken/confusing. The 2nd action listed is also broken/confusing. In fact, the whole "REST" interface seem loosly REST.
- (-) Confusing interface - to create a VM; do I have to call; store_xml, create and provision?
- (?) Allows access to the hypervisor
- On a public facing API actions related to the hypervisor or virtual machine manager (marked in italics above) should not be visible.
- Interface is quite granular as a result of mixing hypervisor and resource management
- (-) Heavily dependent on libvirt

OpenNebula

- API: http://www.opennebula.org/doku.php?id=documentation:rel1.2:api
- Communication Pattern: Synchronous
- Web Service Technology: XML-RPC
- **SLA:** OpenNebula is a product and has no corresoponding service with a SLA

Compute-resource oriented operations	
Operation Name	Description
one.vmallocate	Allocates a virtual machine description from the given template string
one.vmdeploy	Initiates the instance of the given vmid on the target host.
one.vmaction	Submits an action to be performed on a virtual machine.
one.vmmigrate	Migrates one virtual machine (vid) to the target host (hid).
one.vmget_info	Gets information on a virtual machine.
one.hostallocate	Adds a host to the host list.
one.hostinfo	Retrieves the information of the given host.
one.hostdelete	Deletes a host from the host list.
one.hostenable	Enables or disables a given host.

Comments / Observations from SLA@SOI perspective

- Might be better placed in the review of current virtualisation management APIs.
- (-) On a public facing API actions related to the hypervisor or virtual machine manager (marked in italics above) should not be visible.
    - Methods that start with *one.vm** are the ones most suitable to be exposed as external methods to allow customers manage their resources. Those with *one.host** are particular to internal infrastucture management from a provider's perspective.
- (-) How can "actions" be discovered at runtime?
- (-) Session tracking is used through out all methods.

Slicehost

- **API:** http://www.slicehost.com/docs/Slicehost_API.pdf
- Communication Pattern: Synchronous
- Web Service Technology: REST interface over HTTP. Capistran and ActiveResource (RoR) pattern. HTTP PUT verb and Ruby wrappers
- **SLA:** State a best effort type of SLA.

Compute-resource oriented operations	
Operations (root URL:https://apikey@api.slicehost.com)	Description
s = Slice.find(xxx)	
/slices/xxxx/hard_reboot.xml (s.put(:hard_reboot))	Hard Reboot
/slices/xxxx/reboot.xml (s.put(:reboot))	Soft Reboot
/slices/xxxx/rebuild.xml?image_id=x (s.put(:rebuild, :image_id => x))	Rebuild Image
/slices/xxxx/rebuild.xml?backup_id=x (s.put(:rebuild, :backup_id => x))	Backup Image
slice = Slice({'image_id':1, 'flavor_id':1, 'name':'example'}).save()	Create a Slice. Saving it (ActiveResource pattern) (It actually "persist it")
slice.name = 'example.org' slice.save()	Updating the name of the Slice

Comments / Observations from SLA@SOI perspective

- (+-) Xen Based
- (-+) Pure Restful approach
- (-+) No cpu choice, proportional to amount of memory (cpu "cycles" = 4 times the ammount of mem)
- (+) Basic DNS management, private IP networking (slice 2 slice comms), Shared IP (HA)
- (+-) Authentication for the API uses standard HTTP Authentication which uses the customers unique
  password as part of the URL
- Software management: Capistrano is suggested for Slice management (software deployment) with a frontend with svn to manage software on the slices.

Globus Nimbus

- API: EC2 WSDL. See above evaluation of EC2. A WSRF interface is also supported.
- Evaluation WSRF - TODO

Eucalyptus

- API: EC2 WSDL. See above evaluation of Amazon EC2
- SLA: No SLA as this is a product.
- Seems that Eucalyptus will be officially integrated with next Ubuntu to simplify the deployment of services between private infrastucture and EC2 compatible providers (via Eucalyptus)

AppNexus

API: https://wiki.appnexus.com/display/documentation/APIs

As AppNexus do not offer an easily accessible programmable API (only command line tools), it has been excluded.

F5.com

API: http://devcentral.f5.com/wiki/default.aspx/iControl/GlobalLB.html (note: requires free login :-()

Apache Tashi (Incubator)

- **API:** http://opencirrus.intel-research.net/tashi
- Communication Pattern: Synchronous, RPC (No much details in manager documentation)
- Web Service Technology: Thrift RPC (full details not available for how clusters are managed; cluster = group of nodes). Functionality available for one client machine.
- **SLA:** Tashi is a product and as such does not offer a SLA

Compute-resource oriented operations (tashi-client)	
Operation Name	Description
pauseVm	Pauses a virtual machine instance (supply instance ID)
suspendVm	--instance <value> --destination <value>
unpauseVm	Resumes a VM instance from a paused state (supply instance id)
getUsers	
getHosts	
getInstances	
createVm	[--userId <value>] --name <value> [--type <value>] --disks <value> [--n... --type 9 --disks mpi-hardy.qcow2:True,...
shutdownVm	--instance <value>
getUsers	
resumeVm	[--userId <value>] --name <value> [--type <value>] --disks <value> [--n...
migrateVm	--instance <value> --targetHostId <value>

Comments / Observations from SLA@SOI perspective

- Might be better placed in the review of current virtualisation management APIs.
- (-) Methods do not suit that of an external cloud API. Very much a framework to cater for internal management of virtual machines hosted within a provider.
- (-) Light on documentation
- (+) Lightweight implementation
- (+) KVM live migration (xen?)
- (-) Only supports KVM currently
- (+ -) Not clear how they do VM cpu usage (KVM style scheduling - 50-50, 33 - 33 -33,...)
- (-) Not much management functionality for clusters
- (-) clustering functionality not described (yet)
- (+) Usage of Thrift as "IDL" will make easy to integrate with other approaches via RPC

CohesiveFT

API:
http://www.cohesiveft.com/components/com_mambowiki/index.php/Elastic_Server_Manager_API_-_Alpha
Note: For the management of a VM (Virtual Server) and not the host running the VM

Communication Pattern: Synchronous

Web Service Technology: SOAP/WSDL. For management of the VM

Management Operations. A Resource/Group of Resources (source)	
Operation Name	Description

• 1.2 getCredentials
• 1.3 getLoadAverages
• 1.4 getUptime
• 1.5 getMemInfo
• 1.6 getDiskInformation
• 1.7 getLog
• 1.8 getSysLog
• 1.9 restartRubymin
  ○ 1.9.1 Networking Functions
• 1.10 getHostIPAddress
• 1.11 getNetworkInfo
• 1.12 setNetworkInfo
• 1.13 getNetworkStats
• 1.14 resetNetwork
• 1.15 getHostname
• 1.16 setHostname
  ○ 1.16.1 Timezone Functions
• 1.17 getTimezoneList
• 1.18 setTimezone
• 1.19 getTimezone
  ○ 1.19.1 Nameserver Functions
• 1.20 getNameservers
• 1.21 addNameserver
• 1.22 delNameserver
  ○ 1.22.1 NTP Functions
• 1.23 addNTPServer
• 1.24 deleteNTPServer
• 1.25 setDefaultNTPServer
• 1.26 getDefaultNTPServer
• 1.27 getNTPServerList
• 1.28 runNTPServer
• 1.29 setNTPUpdateTime
• 1.30 getNTPLog
• 1.31 clearNTPLog
  ○ 1.31.1 Appliance/System Control Functions
• 1.32 shutdown
• 1.40 enableAppliance
• 1.41 disableAppliance
• 1.42 getApplianceStatus
	Managing one VM appliance (Virtual Server)

• 1.33 stopAppliances • 1.34 startAppliances • 1.35 restartAppliances • 1.36 resetAppliance • 1.37 getEnabledApplianceList • 1.38 getApplianceList • 1.39 getEnabledApplianceList	Managing Several

Comments / Observations from SLA@SOI perspective

- (-) Not really an IaaS player but a Virtual Server (Appliance) builder with their own package management for configuring the VMs with software bundles.
- (-) (+) "Middle man" providing software packaging inside VMs rather than infrastucture
- (-) Compute resource provisioning is handled by the UI, not the API
- (+) Different choice of Virtualization technologies
- (+) Possibility of downloading the VMs for running in own infrastructure
- (-) Deployment on external infrastucture (Amazon, Elastic Hosts)
- (-) Api seems evolving and the semantics of the operations does not appear finalised. Managing groups of resources does not appear to be addressed yet.
- (+) VPN-Cubed allows to weave several servers in different infrastructures in one unique network. (Different product)
- (-) No explicit SLA (up to the external providers)
- (+) Could be useful input to SLA@SOI A3 Software Management Workpackage

Providers not considered

The following are not considered for evaluation as they do not offer a public programmable API. However some do offer a user interface to interact, configure and create infrastructure.

- Joyent Accelerators
- Mosso
- Cassatt Active Response
- 3Tera
- NewServers

References

http://groups.google.ca/group/cloud-...s-and-enablers
http://cloudwiki.fzi.de/index.php/Cloud_ecosystem

Attached Files

Attached Images