OpenStack API Extensions

Jorge L. Williams

rackspace® HOSTING

Agenda

- The Problem
- Extensions
- Extensions in REST
- Promoting Extensions to new Features
- Extensions in OpenStack
- Questions?

The Problem

Standardization vs Innovation and Differentiation

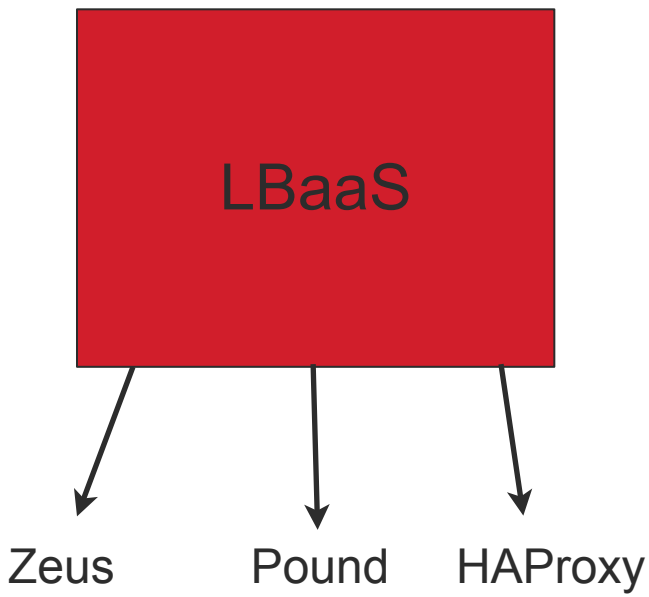
- We want our APIs to be Open Standards...
  - Defining Standard APIs good for OpenStack, and our Customers
  - We want to encourage others to implement our APIs
  - Standards need to be stable
    - Hard to develop against something that's in constant flux
  - Standards need to be general
    - May be impossible for someone to adopt our standards if they contain niche functionality.
  - The more general and stable the API, the more likely others will adopt it.
- We want to innovate
  - Quickly add features that differentiate one implementation from other implementations
    - Without breaking our clients
    - Without going through an approval process
  - We want to allow others to also make changes to the API
    - More likely to adopt OpenStack APIs if they can be modified
    - We may all benefit from these changes
  - Developers should feel free to experiment and develop new features without worrying about the implications to the API as a standard.

There are a number of problem that we're trying to solve: One is there always seems to be a conflict between

One goal is to create this ubiquitous cloud technology

Pluggability

- OpenStack services should be pluggable.
  - One OpenStack Service...Many Backends:
- Backends contain a set of shared capabilities, widely applicable to all deployments
- However, each backend may contain special features and niche functionality
- How do we provide access to special features, while still abstracting away the details of the backend?



hypervisors in compute, network switches in NaaS, etc.

Implemented via drivers

Open Source

- Open source presents another interesting challenge: Others can make changes to the code.
  - OpenStack stock Compute API vs.
    - Rackspace Version vs.
    - Other Modified versions.
  - What does OpenStack Compute API 1.1 mean if we have different implementations all with different capabilities?
  - How do we ensure compatibility among the different versions?

Extensions

Case Study: OpenGL

- The problem we're facing is not new.
  OpenGL faced a similar problem in the 90's
  - How do you define an open graphics library that:
    - Is considered a standard specification
    - Allows vendors to differentiate their products by adding special features
    - And yet is a governed spec
      - An architecture review board (ARB):
        - » Proposes and approves specification changes
        - » Makes new releases
        - » Ensures conformance testing
  - The solution was to allow extensions in the specification
    - Vendors can define special features as extensions
  - A very successful strategy
    - The core OpenGL API is general and uncluttered and an accepted standard.
    - Over 500 extensions have been defined over OpenGL's lifetime
      - Best become standard features; others abandoned
      - Different extensions for the same feature? Let the best one win.
      - Many innovations came via the extension process: vertex and fragment shaders, etc.
      - Extensions have been defined by many different vendors: NVidia, ATI, Apple, IBM, Intel, …

Extensions

So what are extensions...

- Extensions add capability to the API beyond those of the specification
- An API specification must be written to allow for extensibility
  - We need flexibility in the contract to allow for new data elements, actions, states, headers, parameters, and resource types.
  - The core API specification defines the extension mechanism, but extensions themselves are not part of the core.
- Implementors are only required to implement the core API
  - But they must implement the core API completely, this way clients can expect a minimum level of functionality.
- Extensions can be promoted
  - Extensions may follow a promotion path, at the end of which an extension may become part of the next version of the core API.
  - Niche extensions, may never be promoted to the core.

If 90% of calls return unimplemented then what's the point of having a core API?

Calls will constantly keep failing

Extensions vs. Versions

Versions	Extensions
Centralized: Versions are maintained by the entity that controls the API Spec: The Project Technical Lead (PTL) in collaboration with the Project Policy Board (PPB). Only the PTL can create a new version, only the PTL defines what OpenStack Compute 1.1 means.	**Decentralized:** Extensions are maintained by third parties: Rackspace, OpenStack developers, etc. Anyone can create an Extension.

Extensions vs. Versions

Versions	Extensions
Centralized: Versions are maintained by the entity that controls the API Spec: The Project Technical Lead (PTL) in collaboration with the Project Policy Board (PPB). Only the PTL can create a new version, only the PTL defines what OpenStack Compute 1.1 means.	**Decentralized:** Extensions are maintained by third parties: Rackspace, OpenStack developers, etc. Anyone can create an Extension.
Deal with **Core Functionality**	Deal with **Specialized/Niche/ Incomplete Functionality**

Can be applicable to a wide degree of backends...many hypervisors...many load balancers...etc.

Extensions vs. Versions

Versions	Extensions
Centralized: Versions are maintained by the entity that controls the API Spec: The Project Technical Lead (PTL) in collaboration with the Project Policy Board (PPB). Only the PTL can create a new version, only the PTL defines what OpenStack Compute 1.1 means.	**Decentralized:** Extensions are maintained by third parties: Rackspace, OpenStack developers, etc. Anyone can create an Extension.
Deal with **Core Functionality**	Deal with **Specialized/Niche Functionality**
Appear infrequently: Versions provide a stable platform on which to develop.	**Appear frequently:** Extensions bring new features to the market quickly, and in a compatible manner.

Extensions vs. Versions

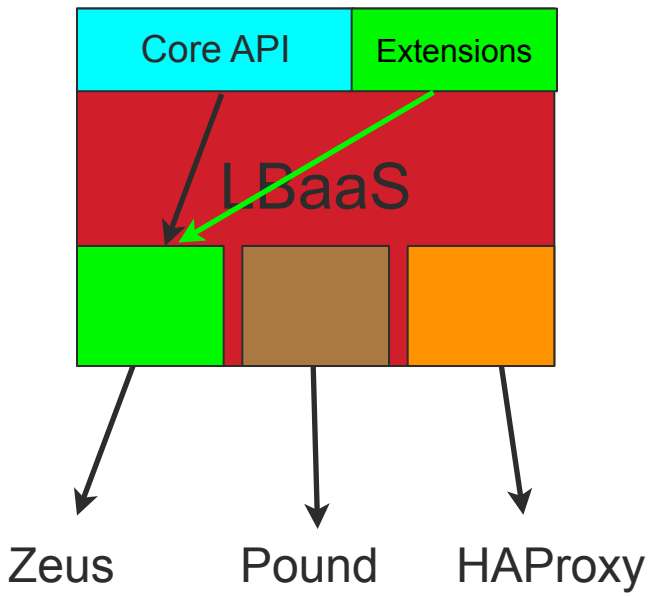
Versions	Extensions
Centralized: Versions are maintained by the entity that controls the API Spec: The Project Technical Lead (PTL) in collaboration with the Project Policy Board (PPB). Only the PTL can create a new version, only the PTL defines what OpenStack Compute 1.1 means.	**Decentralized:** Extensions are maintained by third parties: Rackspace, OpenStack developers, etc. Anyone can create an Extension.
Deal with **Core Functionality**	Deal with **Specialized/Niche Functionality**
Appear infrequently: Versions provide a stable platform on which to develop.	**Appear frequently:** Extensions bring new features to the market quickly, and in a compatible manner.
Are Queryable: You can programmatically tell what versions are available by doing a GET on the base URL (/) of the API endpoint.	**Are Queryable:** You can programmatically tell what extensions are available by doing a GET on the extensions resource (/v1.1/extensions).

Extensions vs. Versions

- Our APIs should be both Extensible and Versionable

Extensions and Plug-ability go Hand in Hand

Each driver may expose functionality that a client may access via an extension.



Extensions in REST

What can be extended

Extension status:

A new verb on an existing resource. How do we handle this?

- Extensions may define:
  - New elements/attributes
  - New actions
  - New headers
  - New parameters
  - New states
  - New resources
  - New mime-types (support for Atom representation or plist representation)
  - New capabilities (the ability to edit an otherwise un-editable attribute)

Extensions are queryable

- You should be able to tell what extensions are available by making a single call.
- If you use extensions as a client this is the first call you'll probably make

Sample Extension Query

- Extensions are queryable via */extensions*

```
<extensions xmlns="http://docs.openstack.org/api-specs/v1.0"
            xmlns:atom="http://www.w3.org/2005/Atom"
            >
    <extension name="Public Image Extension"
               namespace="http://docs.rackspacecloud.com/servers/api/ext/pie/v1.0"
               alias="RAX-PIE"
               >
        <atom:link rel="describedby" type="application/pdf"
                   href="http://docs.rackspacecloud.com/servers/api/ext/cs-pie-20111111.pdf"/>
        <atom:link rel="describedby" type="application/vnd.sun.wadl+xml"
                   href="http://docs.rackspacecloud.com/servers/api/ext/cs-pie.wadl"/>
        <description>
            Adds the capability to share an image with other users.
        </description>
    </extension>
    <extension name="Cloud Block Storage"
               namespace="http://docs.rackspacecloud.com/servers/api/ext/cbs/v1.0"
               alias="RAX-CBS"
               >
        <atom:link rel="describedby" type="application/pdf"
                   href="http://docs.rackspacecloud.com/servers/api/ext/cs-cbs-20111201.pdf"/>
        <atom:link rel="describedby" type="application/vnd.sun.wadl+xml"
                   href="http://docs.rackspacecloud.com/servers/api/ext/cs-cbs.wadl"/>
        <description>
            Allows mounting cloud block storage volumes.
        </description>
    </extension>
</extensions>
```

Sample Extension Query

- Human Readable Name and Description

```
<extensions xmlns="http://docs.openstack.org/api-specs/v1.0"
            xmlns:atom="http://www.w3.org/2005/Atom"
            >
    <extension  name="Public Image Extension"
                namespace="http://docs.rackspacecloud.com/servers/api/ext/pie/v1.0"
                alias="RS-PIE"
                >
        <atom:link rel="describedby" type="application/pdf"
                   href="http://docs.rackspacecloud.com/servers/api/ext/cs-pie-20111111.pdf"/>
        <atom:link rel="describedby" type="application/vnd.sun.wadl+xml"
                   href="http://docs.rackspacecloud.com/servers/api/ext/cs-pie.wadl"/>
        <description>
            Adds the capability to share an image with other users.
        </description>
    </extension>
    <extension name="Cloud Block Storage"
                namespace="http://docs.rackspacecloud.com/servers/api/ext/cbs/v1.0"
                alias="RS-CBS"
                >
        <atom:link rel="describedby" type="application/pdf"
                   href="http://docs.rackspacecloud.com/servers/api/ext/cs-cbs-20111201.pdf"/>
        <atom:link rel="describedby" type="application/vnd.sun.wadl+xml"
                   href="http://docs.rackspacecloud.com/servers/api/ext/cs-cbs.wadl"/>
        <description>
            Allows mounting cloud block storage volumes.
        </description>
    </extension>
</extensions>
```

Sample Extension Query

- Links to Documentation (in different formats)

```
<extensions xmlns="http://docs.openstack.org/api-specs/v1.0"
            xmlns:atom="http://www.w3.org/2005/Atom"
            >
    <extension name="Public Image Extension"
               namespace="http://docs.rackspacecloud.com/servers/api/ext/pie/v1.0"
               alias="RS-PIE"
               >
        <atom:link rel="describedby" type="application/pdf"
                   href="http://docs.rackspacecloud.com/servers/api/ext/cs-pie-20111111.pdf"/>
        <atom:link rel="describedby" type="application/vnd.sun.wadl+xml"
                   href="http://docs.rackspacecloud.com/servers/api/ext/cs-pie.wadl"/>
        <description>
            Adds the capability to share an image with other users.
        </description>
    </extension>
    <extension name="Cloud Block Storage"
               namespace="http://docs.rackspacecloud.com/servers/api/ext/cbs/v1.0"
               alias="RS-CBS"
               >
        <atom:link rel="describedby" type="application/pdf"
                   href="http://docs.rackspacecloud.com/servers/api/ext/cs-cbs-20111201.pdf"/>
        <atom:link rel="describedby" type="application/vnd.sun.wadl+xml"
                   href="http://docs.rackspacecloud.com/servers/api/ext/cs-cbs.wadl"/>
        <description>
            Allows mounting cloud block storage volumes.
        </description>
    </extension>
</extensions>
```

Sample Extension Query

- Unique Extension IDs

```
<extensions xmlns="http://docs.openstack.org/api-specs/v1.0"
            xmlns:atom="http://www.w3.org/2005/Atom"
            >
    <extension name="Public Image Extension"
               namespace="http://docs.rackspacecloud.com/servers/api/ext/pie/v1.0"
               alias="RAX-PIE"
               >
        <atom:link rel="describedby" type="application/pdf"
                   href="http://docs.rackspacecloud.com/servers/api/ext/cs-pie-20111111.pdf"/>
        <atom:link rel="describedby" type="application/vnd.sun.wadl+xml"
                   href="http://docs.rackspacecloud.com/servers/api/ext/cs-pie.wadl"/>
        <description>
            Adds the capability to share an image with other users.
        </description>
    </extension>
    <extension name="Cloud Block Storage"
               namespace="http://docs.rackspacecloud.com/servers/api/ext/cbs/v1.0"
               alias="RAX-CBS"
               >
        <atom:link rel="describedby" type="application/pdf"
                   href="http://docs.rackspacecloud.com/servers/api/ext/cs-cbs-20111201.pdf"/>
        <atom:link rel="describedby" type="application/vnd.sun.wadl+xml"
                   href="http://docs.rackspacecloud.com/servers/api/ext/cs-cbs.wadl"/>
        <description>
            Allows mounting cloud block storage volumes.
        </description>
    </extension>
</extensions>
```

Sample Extension Query

- Vendor Identifiers

```
<extensions xmlns="http://docs.openstack.org/api-specs/v1.0"
            xmlns:atom="http://www.w3.org/2005/Atom"
            >
    <extension name="Public Image Extension"
               namespace="http://docs.rackspacecloud.com/servers/api/ext/pie/v1.0"
               alias="RAX-PIE"
               >
        <atom:link rel="describedby" type="application/pdf"
                   href="http://docs.rackspacecloud.com/servers/api/ext/cs-pie-20111111.pdf"/>
        <atom:link rel="describedby" type="application/vnd.sun.wadl+xml"
                   href="http://docs.rackspacecloud.com/servers/api/ext/cs-pie.wadl"/>
        <description>
            Adds the capability to share an image with other users.
        </description>
    </extension>
    <extension name="Cloud Block Storage"
               namespace="http://docs.rackspacecloud.com/servers/api/ext/cbs/v1.0"
               alias="RAX-CBS"
               >
        <atom:link rel="describedby" type="application/pdf"
                   href="http://docs.rackspacecloud.com/servers/api/ext/cs-cbs-20111201.pdf"/>
        <atom:link rel="describedby" type="application/vnd.sun.wadl+xml"
                   href="http://docs.rackspacecloud.com/servers/api/ext/cs-cbs.wadl"/>
        <description>
            Allows mounting cloud block storage volumes.
        </description>
    </extension>
</extensions>
```

Vendor Identifiers

- An extension alias always contains a prefix that identifies the vendor.

```
<extension name="Public Image Extension"
               namespace="http://docs.rackspacecloud.com/servers/api/ext/pie/v1.0"
               alias="RAX-PIE">
```

Prefix	Vendor
OS	OpenStack
MLTI	Multi-Vendor
RAX	Rackspace
NASA	Nasa
CTX	Citrix
...	...

Vendor Identifiers

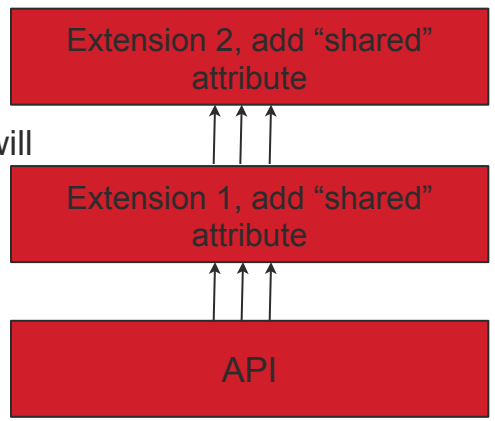
- Namespaces also help ID the vendor

Namespace	Vendor
http://docs.openstack.com/ext/OS/...	OpenStack
http://docs.rackspacecloud.com/...	Rackspace
http://docs.nasa.org/...	Nasa
http://docs.citrix.com/....	Citrix
...	...

Vendor Identifiers

- The idea is to help prevent clashes between extensions.
  - **RAX-PIE**
    - NonXML media types (JSON), headers, parameters
  - **http://docs.rackspacecloud.com/servers/api/ext/pie/v1.0**
    - XML based media types
- Extensions are likely to be implemented by middleware, clashes are likely unless we standardize on an approach

Two extensions add a "shared" attribute one will override the other.



Vendor ID Registry

- OpenStack should maintain a registry of Vendor IDs (prefix and namespaces).
- Anyone should be able to register a Vendor ID.

There shouldn't be a process by which you get approved or anything.

Anyone should be able to register a vendor id.

Data Extensions

- Add additional Data.
  - In XML, attribute may be added to elements so long as they are in the extension namespace
  - In XML, Elements added after last element assuming “Unique Particle Attribution” is not violated
  - In JSON, use alias followed by a colon “:”

```
<image xmlns="http://docs.rackspacecloud.com/servers/api/v1.0"
       xmlns:RAX-PIE="http://docs.rackspacecloud.com/servers/api/ext/pie/v1.0"
       id="1" name="CentOS 5.2"
       serverId="12"
       updated="2010-10-10T12:00:00Z"
       created="2010-08-10T12:00:00Z"
       status="ACTIVE"
       RAX-PIE:shared="true"
       />
```

```
{
    "image" : {
        "id" : 1,
        "name" : "CentOS 5.2",
        "serverId" : 12,
        "updated" : "2010-10-10T12:00:00Z",
        "created" : "2010-08-10T12:00:00Z",
        "status" : "ACTIVE",
        "RAX-PIE:shared" : true
    }
}
```

New Actions

- In XML, actions are defined in the extension namespace
- In JSON, use alias followed by a colon ":" for the action name

```
<mount_volume xmlns="http://docs.rackspacecloud.com/servers/api/ext/cbs/v1.0"
    CBSID="123"/>
```

```
{
    "RAX-CBS:mount_volume" : {
        "CBSID" : "123"
    }
}
```

New Headers and States

- With headers, append name with an X- followed by the alias
  - X-**RAX-CBS**-Header1: Value
  - X-**RAX-CBS**-Header2: Value
- With states, use alias followed by a ":"

```
<image xmlns="http://docs.rackspacecloud.com/servers/api/v1.0"
       xmlns:RS-PIE="http://docs.rackspacecloud.com/servers/api/ext/pie/v1.0"
       id="1" name="CentOS 5.2"
       serverId="12"
       updated="2010-10-10T12:00:00Z"
       created="2010-08-10T12:00:00Z"
       status="RAX-PIE:PrepareShare" progress="80"
       RS-PIE:shared="true"
       />
```

```
{
    "image" : {
        "id" : 1,
        "name" : "CentOS 5.2",
        "serverId" : 12,
        "updated" : "2010-10-10T12:00:00Z",
        "created" : "2010-08-10T12:00:00Z",
        "status" : "RAX-PIE:PrepareShare",
        "progress" : 80,
        "RS-PIE:shared" : true
    }
}
```

New Resources

- Extensions are always defined at /path/to/resource/**alias-ext/newResource**
  All major resources can reference a /alias-ext or optionall /alias, because the vendor controls all resources from the alias.

Promoting Extensions

Specification Governance

- In order to be successful extensions need to be governed
- The Project Tech Lead (PTL) and the Project Policy Board (PPB) are jointly responsible for this.
  They...
  - Promote and approve specification changes (PTL)
  - Decide which extensions are promoted to the core API (PTL)
  - Ensure that each core API provides a minimal set of widely applicable capabilities (PTL)
  - Provide conformance testing (PPB)
  - Ensure inter-project Consistency (PPB)
  - Ensure backward compatibility stability between versions (PPB)

The governance team concerned with consistency between APIs..

..team leads concern with functionality of

Proposal API Coordinator -- that works with the PPB and the PTL.

Governance doesn't necessarily need to be a heavy weight process...

+1 in IRC

PTL Approved Extensions

- The PTL “blesses” an extension by making it an PTL-approved extension.
- PTL-approved extensions use OS as the vendor prefix. That is they become OpenStack extensions.
- An OS extension denotes
  - That the extension is being considered for the next revision of the specification OR
  - That extension is a niche extension that is very useful; it may not make it as a standard feature, but implementors are encouraged to implement it nonetheless, clients can rely on it being there in most cases.

Promotion Path

- Extensions may follow a promotion path
  - Vendor Specific → PTL Approved (OS Extension) → Core Feature
- Some extensions may be developed by multiple vendors; these are known as Multi-Vendor extension, the prefix is MLTI.
  - Multi-Vendor (MLTI) → PTL Approved → Core Feature
- An extension may start as a vendor specific extension and become a multi-vendor extension.
  - Vendor Specific → Multi-Vendor (MLTI) → PTL Approved (OS Extension) → Core Feature

New Features Should Start as Extensions

- This gives us the ability to try things out before a feature enters the standard.
- Allows competing extensions to co-exist
- That means that our API specs are written bottom-up rather than top-down
  - Implementations determine new features
  - The API is not designed in a vacuum
- That said, the PTL can add new features to a new version of the API without introducing them as extensions.

Promotion Path

- Not all extensions should be promoted to core features
  - Extensions may implement niche functionality that doesn't make sense in the core API.
  - Or they may implement functionality that can't make it to core because it would prevent a particular backend from implementing the full set of core features.
- Example: The ability to dynamically change the port number for a load balancer.
  - Most Load Balancers support this, but not all -- can't be in the core
  - We can add that capability as an extension
  - This will be a pretty standard extension that will probably start off as an PTL-approved extension...

Extensions in OpenStack

Status

- Extension Mechanism Built Into
  - Nova (Rackspace)
  - Keystone (Rackspace, HP)
  - Atlas (Rackspace, Citrix)
  - Quantum (Rackspace?, Cisco?)

Compute Extensions

- Slice and Nova Rackspace Extensions
  - Disk Config: Allows automatic/manual management of disk
  - Shared IPs: Allows sharing of IP Addresses in IPv4
  - Affinity ID: Allows new servers to be built next to existing ones
  - Backup Schedule: Schedule automatic backups
  - Region: Users can detect what region a server is in.
- Nova Only Rackspace Extensions
  - Rescue Mode: Place a server in and out or Rescue Mode
  - Interfaces: Users can specify the number of interfaces a server is created with
  - Volumes: Volume support to manage and mount block storage volumes
- Other Nova Extensions (Not supported by Rackspace)
  - Floating IP...
  - Others...

Questions?