Apache Deltacloud & DMTF CIMI

Marios Andreou
Software Engineer, Red Hat
04 February 2012

Agenda

- What is DMTF CIMI
  - Overview/Introduction (collaboration, timeframes, specs/mappings, documents)
  - High-level model overview, examples
  - Launching a machine (template=config+image)
- Deltacloud and CIMI
  - Exposing the CIMI API in a Deltacloud server
  - Example operations
  - The HTML client
  - Future plans

DMTF CIMI

- Cloud Management Working Group – CMWG started work in July 2010
- 34 Actively involved companies and 10+ academic or alliance members
- Weekly meetings and special interest/focus sub-groups
- Democratic process – 1 company 1 vote
- lots of parliamentary judo

DMTF CIMI

- http://dmtf.org/standards/cloud
- DSP0263 CIMI Model and REST Interface
- DSP0264 CIMI Common Information Model
- DSP2027 CIMI Primer
- First WIP docs in Sep 2011 – feedback
- V1 due Q1 2012

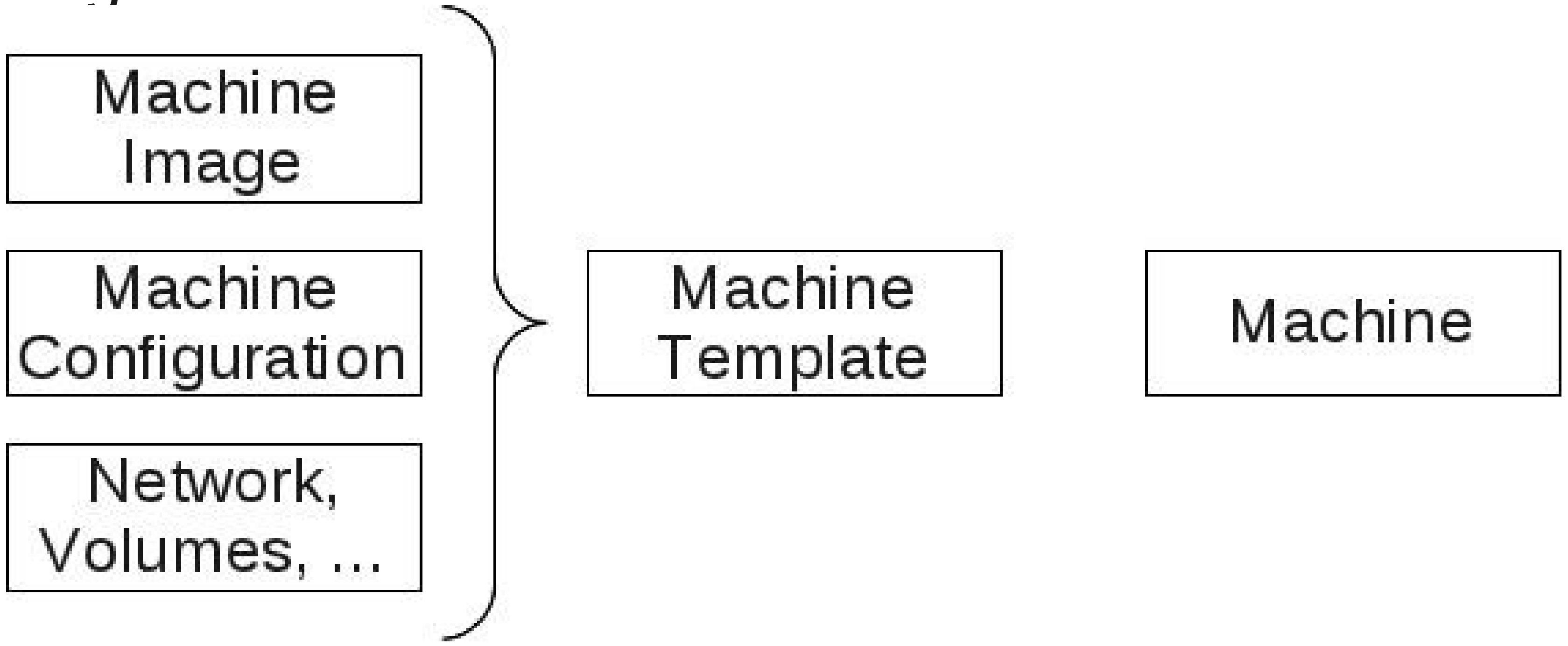
CIMI Model

- Machines, Volumes, Networks, Systems, Monitoring, Extensibility mechanisms
- CIMI Consumer retrieves the cloud entry point – everything else is discovered from here
- REST/HTTP binding, CIM (others anticipated – SOAP?)
- XML and JSON serialization formats
- Everything gets its own MIME type:

```
application/CIMI-Machine+json
```

CIMI Model Entities

- Machine/Network/VolumeTemplate = Configuration + Image



- Machine/Volume/Network/Configuration Collections
- Collection operations – the 'add' URI

CIMI – create Machine

- Retrieve the CEP
- Choose a MachineImage and MachineConfiguration
- ***or*** Choose a pre-defined MachineTemplate
- POST to the MachineCollection 'add' URI “by-reference”:

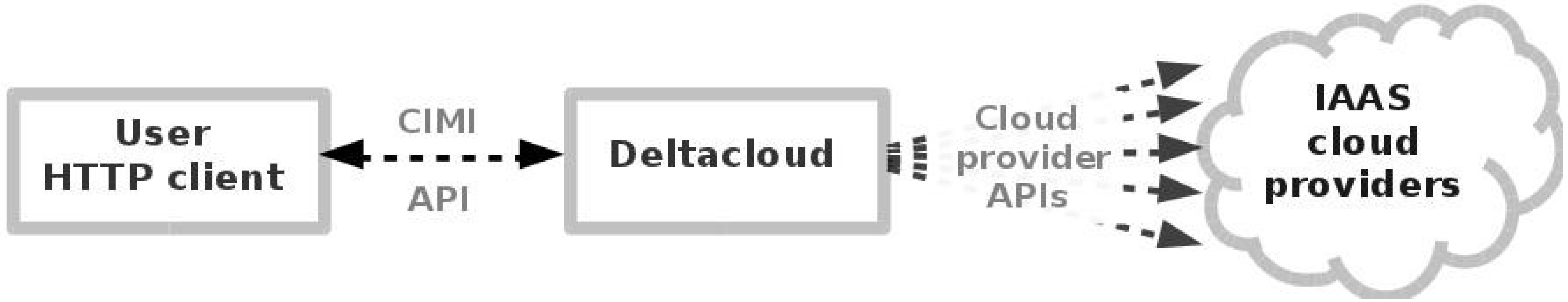
```
POST /machines HTTP/1.1

Content-Type: application/CIMI-MachineCreate+json

X-CIMI-Specification-Version: 1.0

{ "name": "myMachine1",  "description": "My very first
  machine",

"machineTemplate": {    "machineConfig": { "href": "
  http://example.com/configs/small" },

"machineImage": { "href": "
  http://example.com/images/fedora16" }  }  }
```

CIMI Machine

```
GET /machines/843752 HTTP/1.1       ===>  HTTP/1.1 200 OK

Content-Type: application/CIMI-Machine+json

X-CIMI-Specification-Version: 1.0

{ "self": "http://example.com/machines/843752",

  "name": "myMachine1",  "description": "My very first machine",

  "created": "2011/08/15 12:15:00pm",   "status": "STARTED",

  "cpu": "1",  "memory": { "quantity": 4, "units": "gibibyte" },   ...

  "volumes" : [

         { "volume": { "href": "http://example.com/volumes/35782" },
"attachmentPoint": "V" }    ],

  "networkInterfaces": [

    { "vsp": {"href": "..."},  ...   ]

  "operations": [

    { "rel": "edit", "href": "http://example.com/machines/843752" },

    { "rel": "delete", "href": "http://example.com/machines/843752" },

    { "rel": "http://www.dmtf.org/cimi/action/stop",

      "href": "http://example.com/machines/843752" }   ]   }
```

Exposing CIMI via Deltacloud

- Deltacloud server exposes the CIMI API



- /cimi top-level entry-point
- `deltacloud/server/lib/[deltacloud | cimi]`
- `deltacloudd --cimi -i ec2`

  `(==> localhost:3001/cimi)`

Exposing CIMI via Deltacloud

- DSL defining a schema for the CIMI Models

  `scalar, text, href, struct, array`

```
<Machine xmlns="http://www.dmtf.org/cimi">

  <eventLog href="xs:anyURI"/>

  <cpu> xs:string </cpu>

  <memory quantity="xs:integer"
                        units="xs:string"/>

  <volume href="xs:anyURI"
              attachmentPoint="xs:string"/>
```

```
class CIMI::Model::Machine < CIMI::Model::Base
  href :event_log
  text :cpu
  struct :memory do
    scalar :quantity
    scalar :units
   end
   array :volumes do
    scalar :href
    scalar :protocol
    scalar :attachment_point
  end
```

- Serialize/Deserialize from/to json/xml
- Conversion of identifiers from CamelCase (just because)

Exposing CIMI via Deltacloud

- CIMI – specific routes (`/lib/cimi/server.rb`):

```
GET       /cimi/machines
GET       /cimi/machines/:id
POST      /cimi/machines/
POST      /cimi/machines/:id/stop
DELETE    /cimi/machines/:id
```

- CIMI models (`/lib/cimi/model/`):

```
class CIMI::Model::Machine < CIMI::Model::Base

class CIMI::Model::Volume < CIMI::Model::Base

class CIMI::Model::Network < CIMI::Model::Base

class CIMI::Model::System < CIMI::Model::Base
```

Exposing CIMI via Deltacloud

- Making use of the existing Deltacloud drivers:

```
class CIMI::Model::Machine < CIMI::Model::Base

  def self.find(id, context)

     instance =
     context.driver.instances(context.credentials, :id=>id)

     from_instance(instance, context)

  end
```

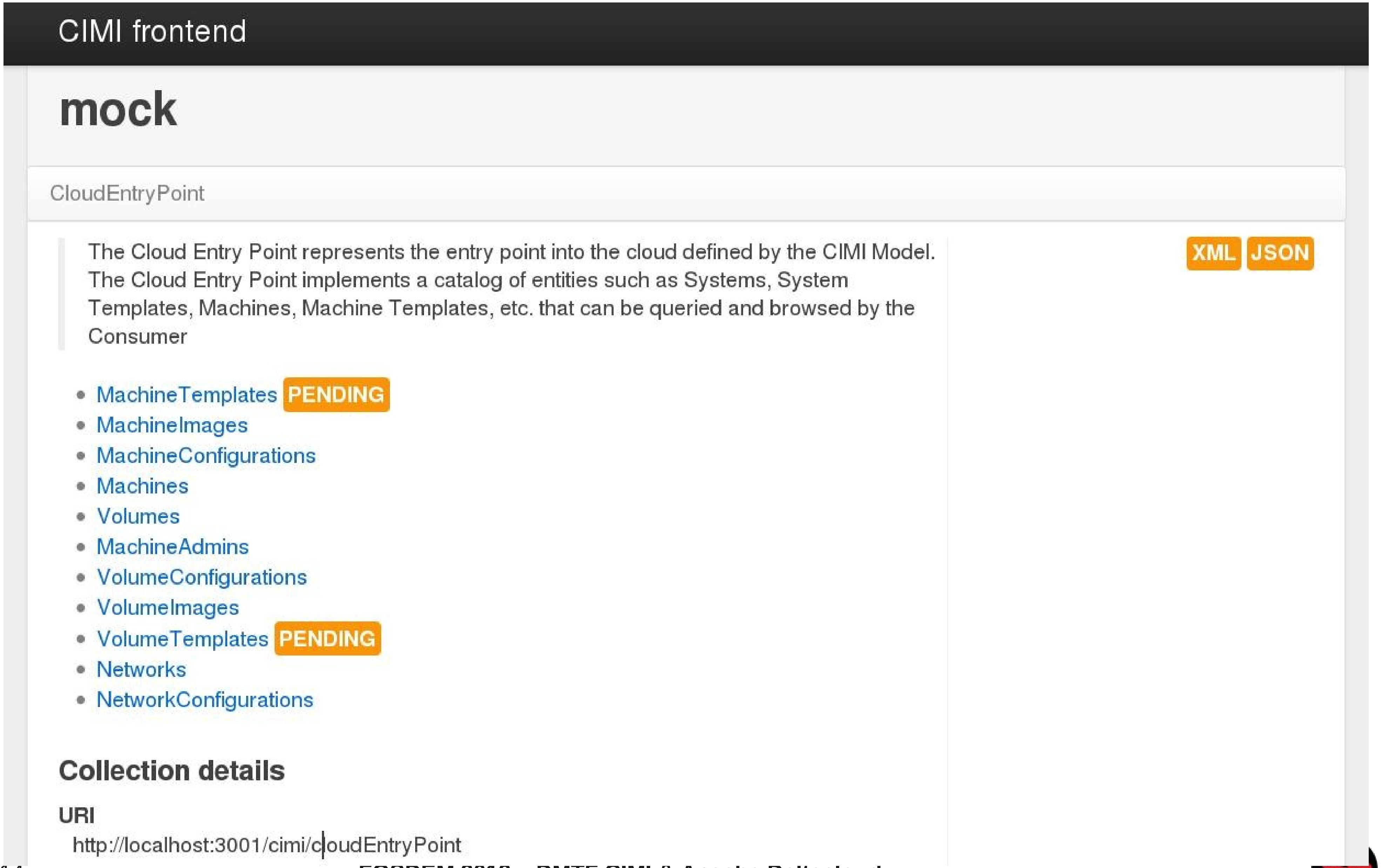
- Conversion from Deltacloud to CIMI objects

```
  def self.from_instance(instance, context)

   self.new(

     :name => instance.id,

     :created => instance.launch_time,

     :uri => context.machine_url(instance.id)  ...
```

CIMI - Deltacloud HTML client

- CIMI Deltacloud – no 'embedded' HTML client
- Stand-alone sinatra client application
- `/deltacloud/clients/cimi`
- Uses rest-client gem to talk to Deltacloud CIMI
- `ruby ./bin/start –u “http://deltacloud:3001/cimi”`
- Haml templates to produce HTML views

CIMI - Deltacloud HTML client



CIMI frontend

mock

CloudEntryPoint

The Cloud Entry Point represents the entry point into the cloud defined by the CIMI Model. The Cloud Entry Point implements a catalog of entities such as Systems, System Templates, Machines, Machine Templates, etc. that can be queried and browsed by the Consumer

- MachineTemplates PENDING
- MachineImages
- MachineConfigurations
- Machines
- Volumes
- MachineAdmins
- VolumeConfigurations
- VolumeImages
- VolumeTemplates PENDING
- Networks
- NetworkConfigurations

Collection details

URI

http://localhost:3001/cimi/cloudEntryPoint

Testing CIMI Deltacloud

- Cucumber scenarios

  `/deltacloud/server/tests/cimi`

```
Scenario: Create a New Machine entity

    When client specifies a Machine Image

   | machineImage | http://example.com/cimi/machine_images/img1 |

    And client specifies a Machine Configuration

   |   machineConfig | http://example.com/cimi/machine_configurations/m1-
                                                    small |

 And client specifies a new Machine using

     | name | sampleMachine1 |

     | description | sampleMachine1Description |

   Then client should be able to create this Machine
```

Future directions

- Finish the implementation ... !
- Deltacloud API vs CIMI API ...
- Conformance test suite...

http://deltacloud.org/contact

marios@redhat.com