OpenNebula/Reservoir Training, January 27-28

Brussels, Belgium

Session 5
Public Cloud Computing

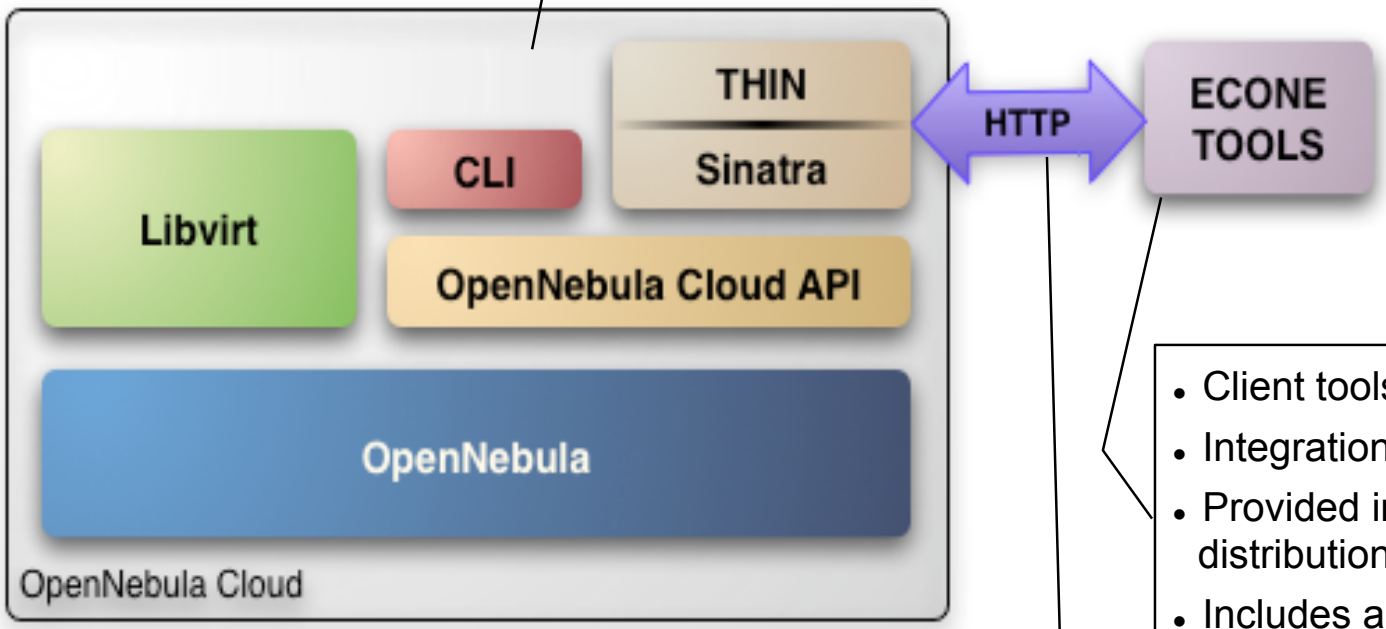
Daniel Molina & Javier Fontán
dmolina/jfontan@opennebula.org

OpenNebula.org

Copyright 2002-2011 © OpenNebula Project Leads (OpenNebula.org). All Rights Reserved.
Creative Commons Attribution Share Alike (CC-BY-SA)

The Public Cloud: Overview

- You can use multiple interfaces for the Cloud
- Transparent to your setup:
  - Hypervisor
  - Storage Model
  - Hybrid configuration



- Client tools uses EC2 libraries
- Integration with EC2 tools
- Provided in the OpenNebula distribution
- Includes a simple S3 replacement

- Supports HTTP and HTTPS protocols
- *EC2 authentication* based on OpenNebula credentials
- Public Cloud users need an OpenNebula account

Creative Commons Attribution Share Alike (CC-BY-SA)

Installing the Public Cloud Components

- OpenNebula distribution supports two Cloud interfaces:
  - EC2 Query API
  - OCCI
- Additional requirements: EC2 development library, web server and web framework

```
fe# gem uninstall rack

fe# gem install rack --version '1.2.0'
fe# gem install sinatra
fe# gem install thin
fe# gem install amazon-ec2 --version '0.9.14'
fe# gem install uuid

Add a "FQDN" for our Public Cloud
fe# vim /etc/hosts
127.0.0.1       localhost
…
192.168.$CN.2   frontend cloud$CN.opennebula.org
```


Creative Commons Attribution Share Alike (CC-BY-SA)

Configuring the Public Cloud

- The EC2 service is configured in `$ONE_LOCATION/etc/econe.conf`
- Hands on... Study the configuration file and adjust it to your cloud

```
# OpenNebula sever contact information
ONE_XMLRPC=http://localhost:2633/RPC2

# Host and port where econe server will run
SERVER=cloud$CN.opennebula.org
PORT=4567

# SSL proxy that serves the API (set if is being used)
#SSL_SERVER=fqdm.of.the.server

# VM types allowed and its template file (inside templates directory)
VM_TYPE=[NAME=m1.small, TEMPLATE=m1.small.erb]
```


Creative Commons Attribution Share Alike (CC-BY-SA)

Configuring the Public Cloud

- You have to define the correspondence between types (simple) and local instantiation of VMs (hard, you should be fine by now)
  - Capacity allocated by this VM type (CPU, MEMORY)
  - Your cloud requirements, e.g. force to use a given kernel (OS) or place public VMs in a given set of cluster nodes (REQUIREMENTS)
  - The network used by Public VMs (NIC)

- VM Types are defined in `econe.conf`. Templates for the VM templates are in `$ONE_LOCATION/etc/ec2query_templates`

- Templates for VM Types are erb files <% Ruby code here %>, you should not need to modify that.

Creative Commons Attribution Share Alike (CC-BY-SA)

Configuring the Public Cloud

- Hands on... Prepare the m1.small type of your cloud to use ttylinux.one as a reference

```
$ more m1.small.erb
NAME   = eco-vm

#Adjust Capacity for this instance type
CPU    = 0.1
MEMORY = 64

OS     = [ kernel = /srv/cloud/one/ttylinux-xen/vmlinuz-xen,
           initrd = /srv/cloud/one/ttylinux-xen/initrd.gz,
           root   = hda1   ]

DISK   = [ IMAGE_ID = <%= erb_vm_info[:img_id] %> ]

NIC    = [ NETWORK =  "One-TD" ]

IMAGE_ID      = <%= erb_vm_info[:ec2_img_id] %>
INSTANCE_TYPE = <%= erb_vm_info[:instance_type ]%>

<% if erb_vm_info[:user_data] %>
CONTEXT = [
       EC2_USER_DATA="<%= erb_vm_info[:user_data] %>",
       TARGET="hdc" ]
<% end %>
```


Creative Commons Attribution Share Alike (CC-BY-SA)

Configuring the Public Cloud

- Hands on...
  - Create a new Public Cloud user

```
fe$ oneuser create ec2-user ec2-pass
fe$ oneuser list
  ID USER            PASSWORD
   0 oneuser         34c629abfcb47856b3d1c0a30798221aefb61605
   1 ec2-user        7030ddf34333388e9a7f0c13a6317ed4d66ac39f
```

  - Start the econe server

```
fe$ econe-server start

fe$ /usr/sbin/lsof -Pi

Check $ONE_LOCATION/var/econe-server.log for errors
```


Creative Commons Attribution Share Alike (CC-BY-SA)

Using the Public Cloud

- The econe-tools are a subset of the functionality provided by the onevm utility, and resembles the ec2-* cli
- Image related commands are:
  - `econe-upload`, place an image in the Cloud repo and returns ID
  - `econe-describe-images`, lists the images
  - `econe-register`, register an image
- Instance related commands are:
  - `econe-run-instances`, starts a VM using an image ID
  - `econe-describe-instances`, lists the VMs
  - `econe-terminate-instances`, shutdowns a VM
- User authentication is based in the OpenNebula credentials
  - `AWSAccessKeyId` is OpenNebula's username
  - `AWSSecretAccessKey` is OpenNebula's password

Creative Commons Attribution Share Alike (CC-BY-SA)

Using the Public Cloud

- Pass your credentials to the econe-tools by (in this order)
  - Command arguments (--access-key <username>, --secret-key <pass>)
  - Environment `EC2_ACCESS_KEY` and `EC2_SECRET_KEY`
  - Environment `ONE_AUTH`
- Point econe-tools to your target cloud
  - Command arguments (--url <http|https>://<fqdn>:<port>) port needed if not the default for the protocol
  - `EC2_URL` environment
- Hands on... upload the ttylinux image, and start it using the public cloud interface.
  - Compare the econe-* (public view) and one* (local view) evolution and information
  - Check the template build by the econe server (onevm show)

Creative Commons Attribution Share Alike (CC-BY-SA)

Using the Public Cloud, uploading an Image

```
fe$ econe-upload -U http://node-x.opennebula.org:4567 --access-key ec2-
user --secret-key ec2-pass /srv/cloud/images/ttylinux/ttylinux.img
Success: ImageId ami-00000003

fe$ export EC2_URL=http://cloud$CN.opennebula.org:4567
fe$ export EC2_ACCESS_KEY=ec2-user
fe$ export EC2_SECRET_KEY=ec2-pass

fe$ econe-describe-images -H
Owner         ImageId       Status          Visibility   Location
-------------------------------------------------------------------
ec2-user      ami-00000003  available       private      23151fac850e5...


This is the local view not accessible to public cloud users
fe$ oneimage list
  ID     USER                 NAME TYPE              REGTIME PUB PER STAT  #VMS
   0  oneuser            ttylinux   OS   Jan 21, 2011 13:59  No  No used     1
   1  oneuser             storage   DB   Jan 21, 2011 13:59  No Yes  rdy     0
   3 ec2-user ec2-71654e30-0872-01   OS   Jan 22, 2011 16:27  No  No  rdy     0

$ oneimage show 3
IMAGE  INFORMATION
ID             : 3
...
```


Creative Commons Attribution Share Alike (CC-BY-SA)

Using the Public Cloud, running an Instance

```
fe$ econe-run-instances ami-00000003
ec2-user      ami-00000004    i-16           m1.small

fe$ econe-describe-instances -H
Owner         Id            ImageId        State         IP              Type
-------------------------------------------------------------------------------
ec2-user      i-10          ami-00000003   running       172.16.10.7     m1.small



This is the local view not accessible to public cloud users

fe$ onevm list
   ID     USER     NAME STAT CPU     MEM         HOSTNAME        TIME
    1  oneuser ttylinux runn  99    63.5M               n04 01 02:41:14
   10 ec2-user   eco-vm runn  99    63.8M               n04 00 01:05:28

fe$ onevm show 14
VIRTUAL MACHINE 14 INFORMATION
ID              : 14
NAME            : eco-vm
STATE           : ACTIVE
...
```


Creative Commons Attribution Share Alike (CC-BY-SA)

Configuring SSL access for the Public Cloud

- SSL security is handle by a proxy that forwards the request to the EC2 Query Service and takes back the answer to the client
- Requirements:
  - A server certificate for the SSL connections
  - An HTTP proxy that understands SSL
  - EC2Query Service configuration to accept petitions from the proxy
- Hands on... Install the proxy (lighttpd) and get the certificates for your cloud

```
fe# yum install lighttpd

fe# cp /automount/share/reservoir/opennebula/certs/server.pem /etc/
lighttpd/server.pem
```


Creative Commons Attribution Share Alike (CC-BY-SA)

Configuring SSL access for the Public Cloud

- Hands on... configure the lighttpd proxy

```
# vim /etc/lighttpd/lighttpd.conf
server.modules              = (
            "mod_access",
            "mod_alias",
            "mod_accesslog",
            "mod_compress",
            "mod_proxy"
...
## bind to port (default: 80)
server.port                 = 443
...
#### proxy module
proxy.server                = ( "" =>
                                ("" =>
                                 (
                                   "host" => "127.0.0.1",
                                   "port" => 4567
                                 )
                                 )
                              )
#### SSL engine
ssl.engine                  = "enable"
ssl.pemfile                 = "/etc/lighttpd/server.pem"
```


Creative Commons Attribution Share Alike (CC-BY-SA)

Configuring SSL access for the Public Cloud

- Hands on... configure the econe server

```
$ vim /srv/cloud/one/etc/econe.conf

#SERVER=node-15.opennebula.org
SERVER=127.0.0.1
PORT=4567

# SSL proxy that serves the API (set if is being used)
SSL_SERVER=cloud$CN.opennebula.org
```

- Hands on... restart services (lighttpd and econe-server) and try your new SSL cloud access (https://node-x.opennebula.org:443)

Creative Commons Attribution Share Alike (CC-BY-SA)