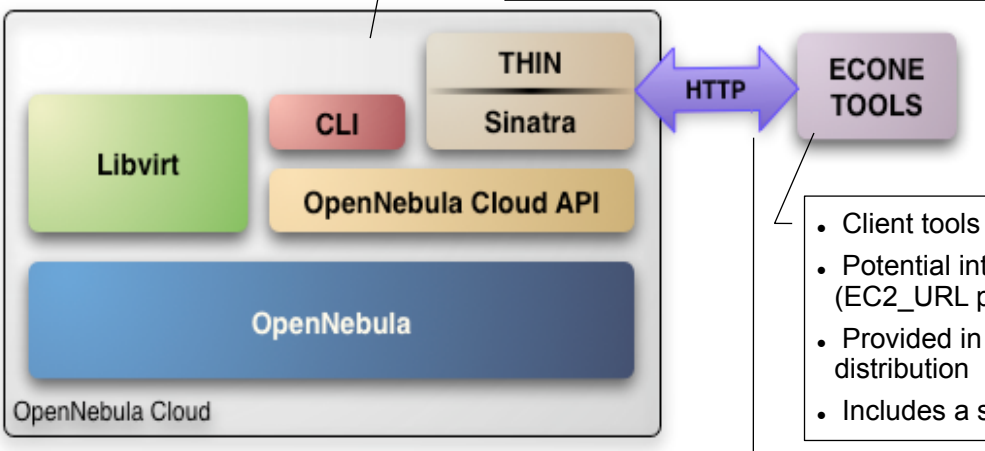
Session 5
Public Cloud Computing

Constatino Vázquez
tinova@fdi.ucm.es

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
- Potential integration with EC2 tools (EC2_URL problems for example)
- 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

Installing the Public Cloud Components

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

```
# gem install rubygems-update ; /var/lib/gems/1.8/bin/update_rubygems
# gem install sinatra
# gem install thin
# gem install amazon-ec2


# gem install sequel
# apt-get install curl libcurl3 libcurl4-gnutls-dev libopenssl-ruby
# gem install curb
# gem install sqlite3-ruby

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

For installing, root & http_proxy is required (sudo –i ; export http_proxy=http://gw:8888)

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

```
$ more econe.conf
# OpenNebula administrator user, the one_auth contents
USER=oneadmin
PASSWORD=onecloud

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

# Host and port where econe server will run keep FQDNs
SERVER=node-y.opennebula.org
PORT=4567

# Configuration for the image repository
#  IMAGE_DIR will store the Cloud images, check space!
DATABASE=/srv/cloud/one/var/econe.db
IMAGE_DIR=/srv/cloud/public_repo/

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

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.

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

CPU    = 0.1
MEMORY = 64

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


DISK = [ source   = <%= erb_vm_info[:img_path] %>,
         clone    = yes,
         target   = hda,
         readonly = no]

#You have to create this network, and it should be owned by oneadmin

NIC    = [ NETWORK = "one-td" ]

IMAGE_ID = <%= erb_vm_info[:img_id] %>
INSTANCE_TYPE = <%= erb_vm_info[:instance_type ]%>
```

Configuring the Public Cloud

- Hands on... start the econe server

```
$ unset EC2_URL
$ econe-server start

$ lsof -i

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


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 not really needed in 1.4
- 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

Using the Public Cloud

- Pass your credentials to the econe-tools by (in this order)
  - Commands arguments (--access-key <username>, --secret-key <pass>)
  - Environemt `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 in 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 onevm (local view) evolution and information
  - Check the template build by the econe server (onevm show)

Using the Public Cloud

```
$ econe-upload -U http://node-x.opennebula.org:4567 --access-key
oneadmin --secret-key onecloud /srv/cloud/images/ttylinux/ttylinux.img
Success: ImageId 1

$ export EC2_URL="http://node-15.opennebula.org:4567"
$ econe-describe-images -H
Owner      ImageId                    Location
------------------------------------------------------------------------
oneadmin   1                          /srv/cloud/public_repo/1

$ econe-run-instances 1
oneadmin    1                                    18         m1.small

$ econe-describe-instances
oneadmin    18    1                                  pending
192.168.169.5   m1.small

This is the local view not accessible to public cloud users
$ onevm list
  ID     USER     NAME STAT CPU     MEM        HOSTNAME        TIME
  19 oneadmin   eco-vm runn   0   65536       84.21.x.y 00 00:01:34

$ onevm show 19
```

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

```
# apt-get install lighttpd
# apt-get install ssl-cert


# /usr/sbin/make-ssl-cert generate-default-snakeoil
# cat /etc/ssl/private/ssl-cert-snakeoil.key /etc/ssl/certs/ssl-cert-
snakeoil.pem > /etc/lighttpd/server.pem
```


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                = 8443
...
#### proxy module
proxy.server               = ( "" =>
                               ("" =>
                                (
                                  "host" => "127.0.0.1",
                                  "port" => 4567
                                )
                                )
                             )
#### SSL engine
ssl.engine                 = "enable"
ssl.pemfile                = "/etc/lighttpd/server.pem"
```

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=node-15.opennebula.org
```

- Hands on... by pass the EC2 library URL checking

```
# sudo vim /var/lib/gems/1.8/gems/amazon-ec2-0.7.9/lib/AWS/EC2.rb
Comment out line 12
```

- Hands on... restart services (lighttpd and econe-server) and try your new SSL cloud access (https://node-x.opennebula.org:8443)