CONSEGÍ 2010

Brasilia-DF, 18-20 August 2010

Design, build and use Private, Hybrid and Public Cloud with OpenNebula

Constantino Vazquez
(tinova@fdi.ucm.es)
Universidad Complutense de Madrid

Workshop Overview

- **Cloud Computing Overview**
- **Planning the Installation**
- **Building your Private Cloud**
  - Installing OpenNebula 1.4
  - Configure OpenNebula 1.4 (storage, hypervisor and network)
  - Administration of an OpenNebula Cloud (hosts, users)
  - Basic usage (networks, VMs)
  - More on usage (VMs, context and scheduling)

Private Cloud

- **Building your HybridCloud**
  - Configuring an Hybrid Cloud with Amazon EC2

Hybrid Cloud

- **Building your Public Cloud**
  - Public Cloud interfaces: The EC2 Query API

Public Cloud

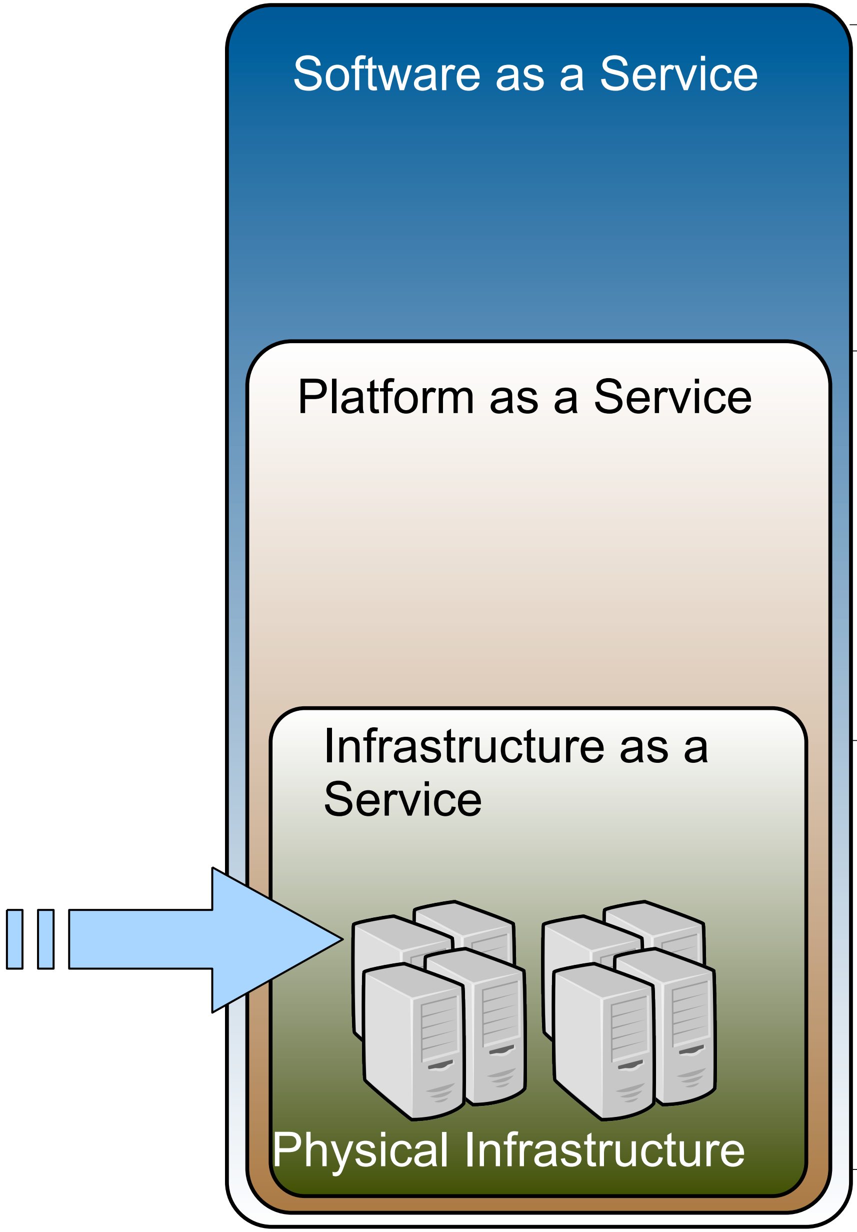
CONSEGÍ 2010

Brasilia-DF, 18-20 August 2010

PART I: Cloud Computing Overview

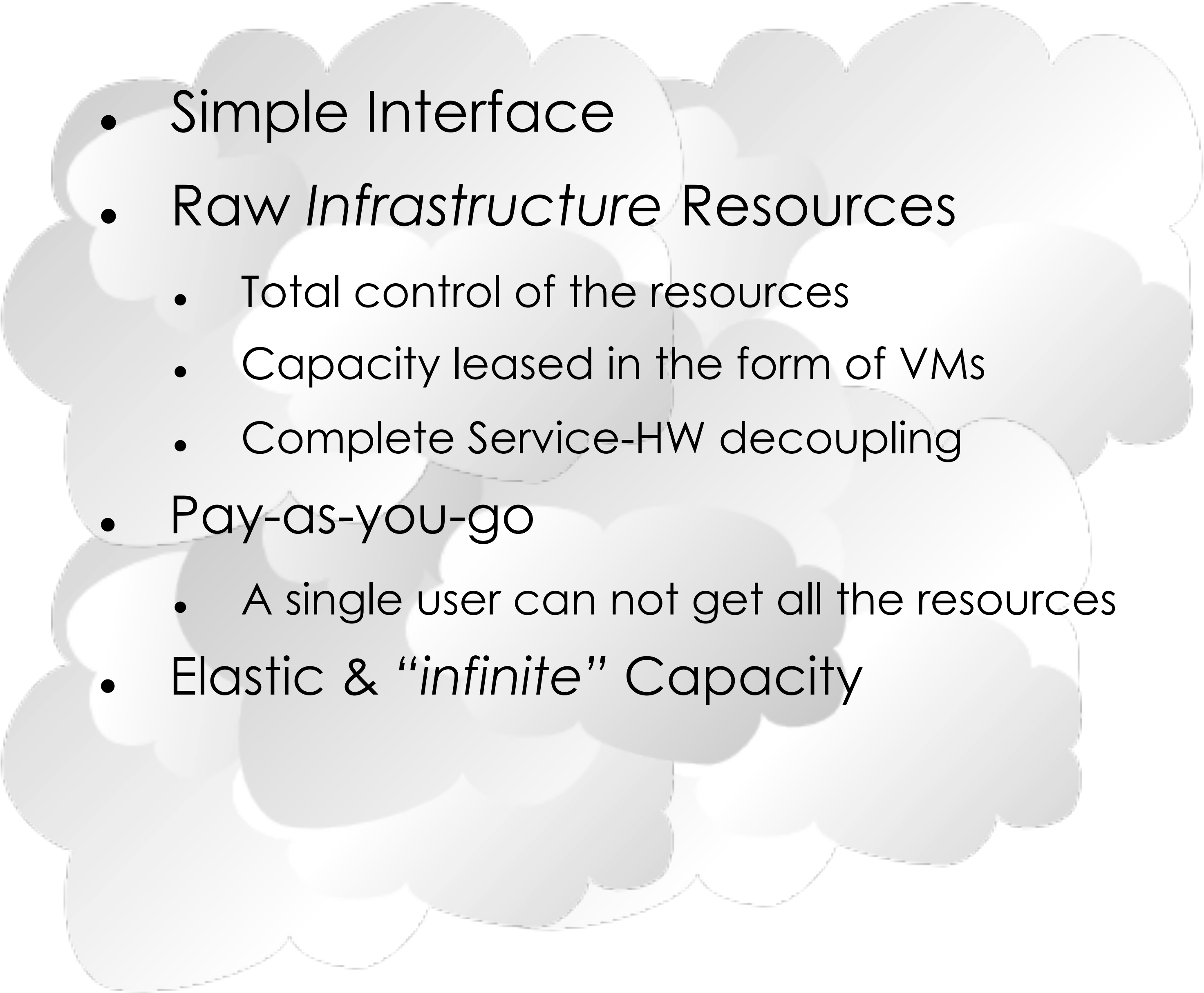
Constantino Vazquez
(tinova@fdi.ucm.es)
Universidad Complutense de Madrid

Cloud Computing in a Nutshell



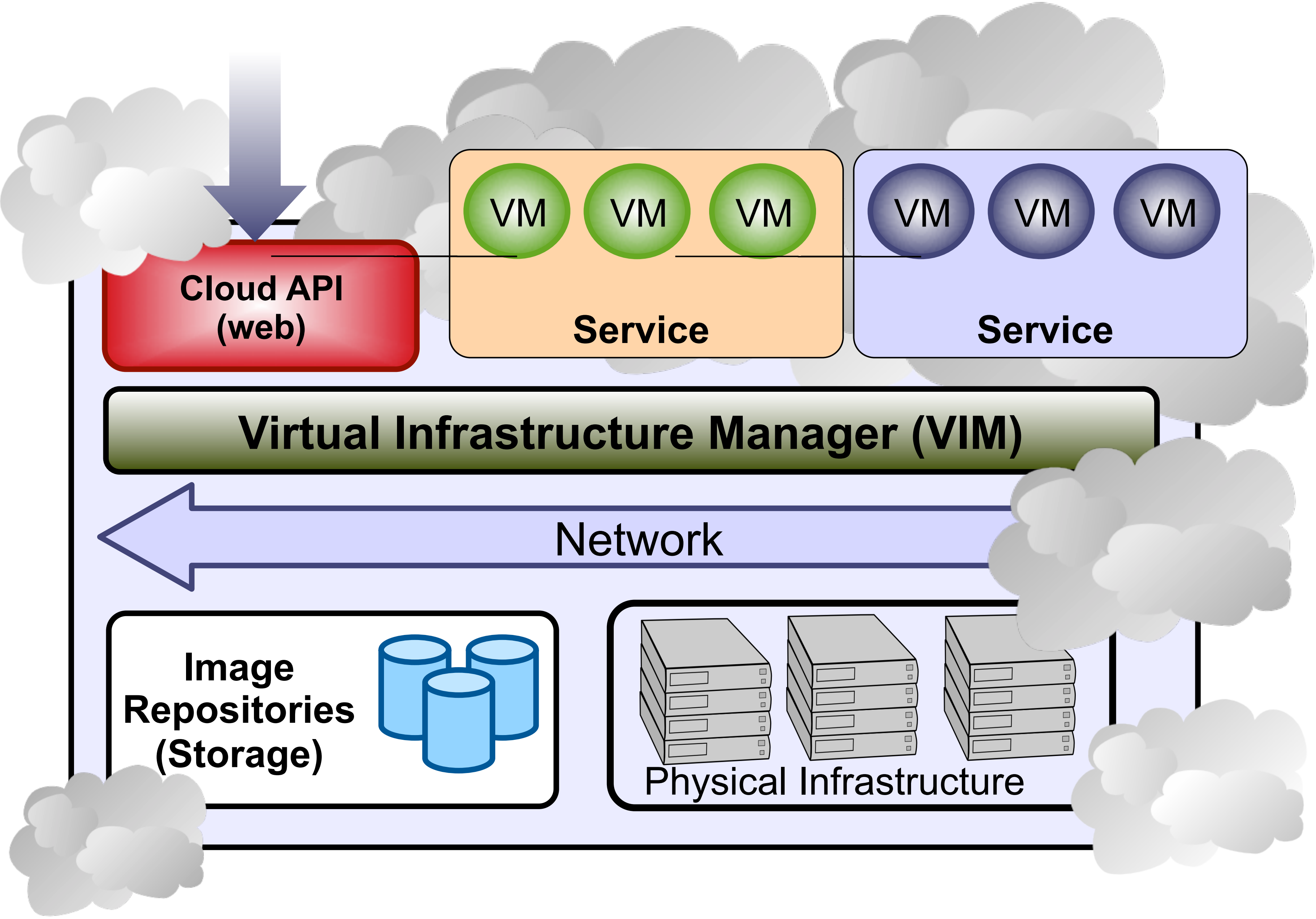
What	Who
On-demand access to any application	**End-user** (does not care about hw or sw)
Platform for building and delivering web applications	**Developer** (no managing of the underlying hw & swlayers)
Delivery of a *raw* computer infrastructure	**System Administrator** (complete management of the computer infrastructure)

The IaaS Clouds a Four Point Check List



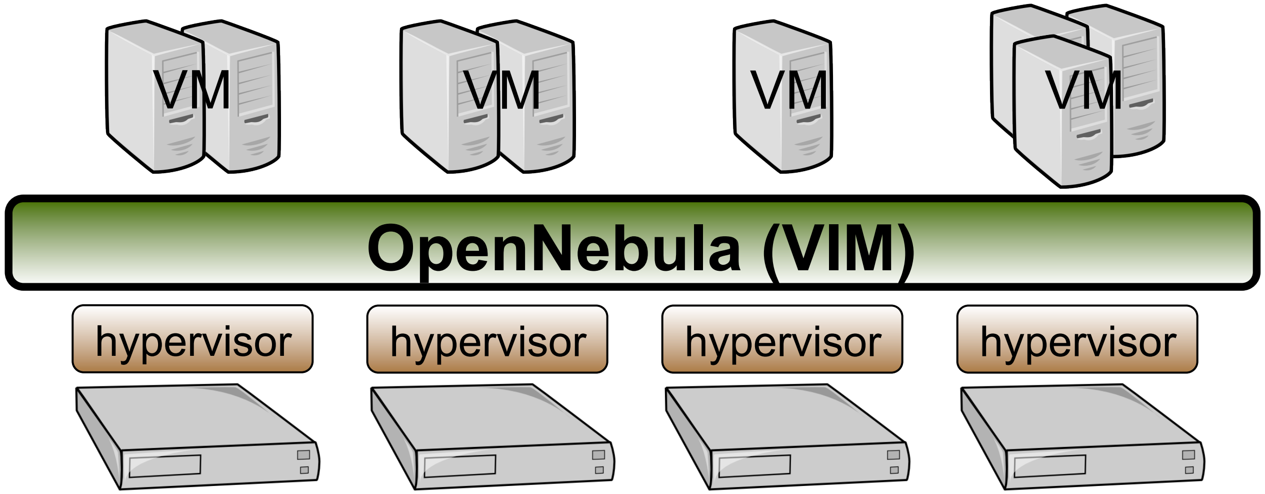
- Simple Interface
- Raw *Infrastructure* Resources
  - Total control of the resources
  - Capacity leased in the form of VMs
  - Complete Service-HW decoupling
- Pay-as-you-go
  - A single user can not get all the resources
- Elastic & “*infinite*” Capacity

The Anatomy of an IaaS Cloud



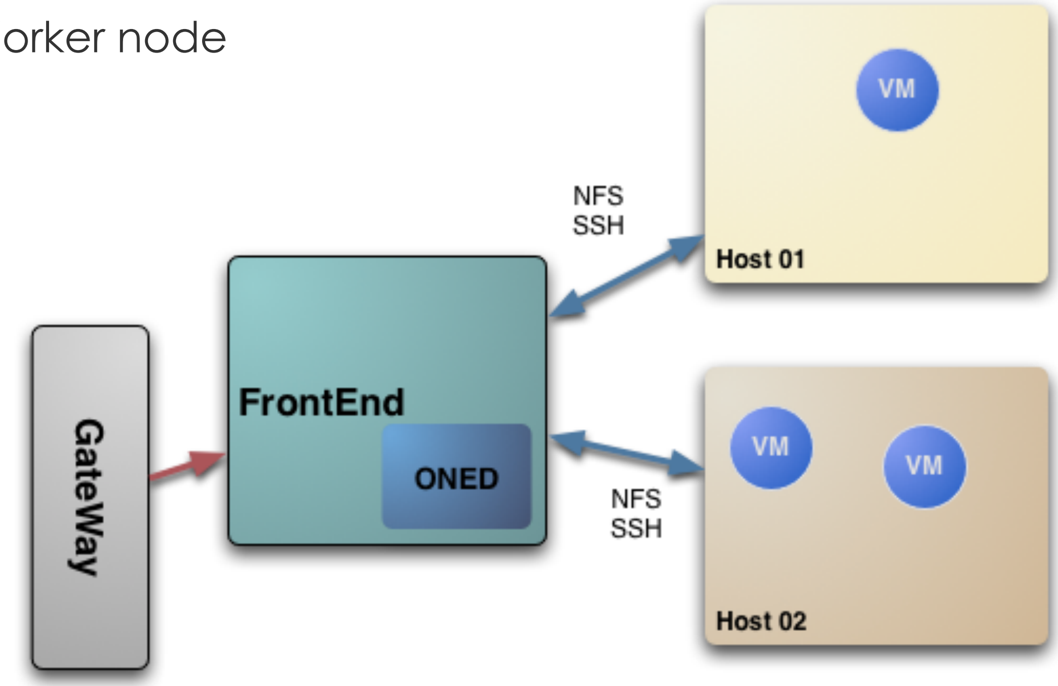
Why a Virtual Infrastructure Manager?

- VMs are great!!...but something more is needed
  - Where did/do I put my VM? (***scheduling & monitoring***)
  - How do I provision a new cluster node? (***clone & context***)
  - What MAC addresses are available? (***networking***)
- Provides a ***uniform view*** of the resource pool
- ***Life-cycle management*** and monitoring of VM
- The VIM ***integrates*** Image, Network and Virtualization



Workshop Testbed

- The workshop **cluster** is composed by three nodes:
  - **FrontEnd**: Ubuntu Server 10.04 OpenNebula will be installed here.
  - **Host 01**: CentOS 5.4 running Xen. Worker node
  - **Host 02**: CentOS 5.4 running Xen. Worker node



- For the hands-on, we will use the OpenNebula ‘dummy’ cloud
  - Please, download OpenNebula v1.4 and untar it

CONSEGÍ 2010

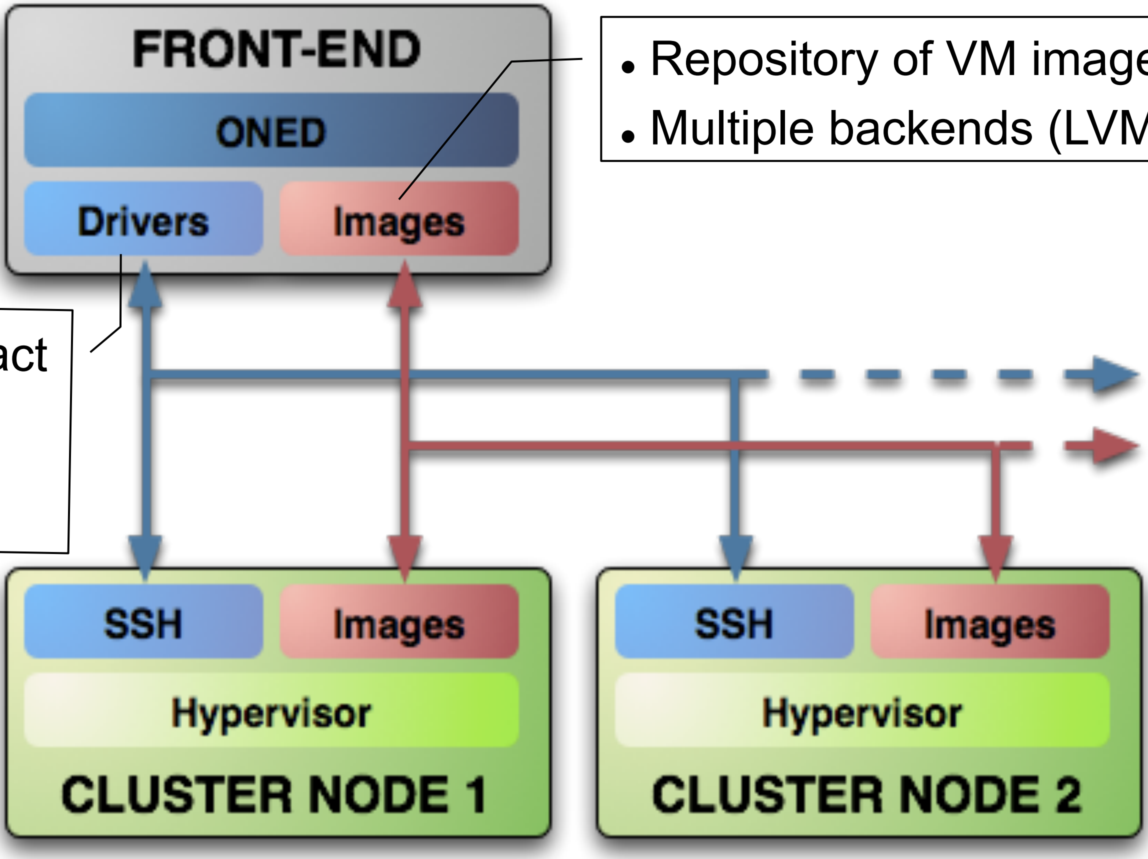
Brasilia-DF, 18-20 August 2010

PART II: Planning the Installation

Constantino Vazquez
(tinova@fdi.ucm.es)
Universidad Complutense de Madrid

Planning the Installation: System Overview

- Executes the OpenNebula Services
- *Usually* acts as a classical cluster front-end



- Repository of VM images
- Multiple backends (LVM, iSCSI..)

- Modular components to interact with the cluster services
- *Types:* storage, monitoring, virtualization and network

- Provides physical resources to VMs
- *Must have* a hypervisor installed

⚠ The same host can be can be a the front-end and a node

Planning the Installation: Working in the Front-End ...

- Choose your installation mode
  - system wide (/usr, /etc...)
  - *self-contained* (under $ONE_LOCATION)

- Install software dependencies.
  - Check the documentation for platform specific notes installation nodes

  http://opennebula.org/documentation:rel1.4:notes

- Dependencies already installed in the Front-End and the Nodes

Planning the Installation: Working in the Front-End ...

- The Users of the private cloud:
  - oneadmin: Account to run the daemons, manage the system and do all the low-level operations (e.g. start VMs, move images...).
  - Users: create and manage their own VMs and networks. *Need to be defined in OpenNebula*
- Installation layout for the workshop
  - OpenNebula code will be placed in /home/oneadmin/SRC
  - We will use the /srv/cloud/one directory to place the OpenNebula software
- NFS sharing between Front-End and Nodes
- Passwordless ssh conections

! The oneadmin account must be created system wide (i.e. front-end and all the nodes) you can use NIS, or a local account with the same ID's in all the hosts. Users do not need a UNIX account in the nodes, nor in the front-end.

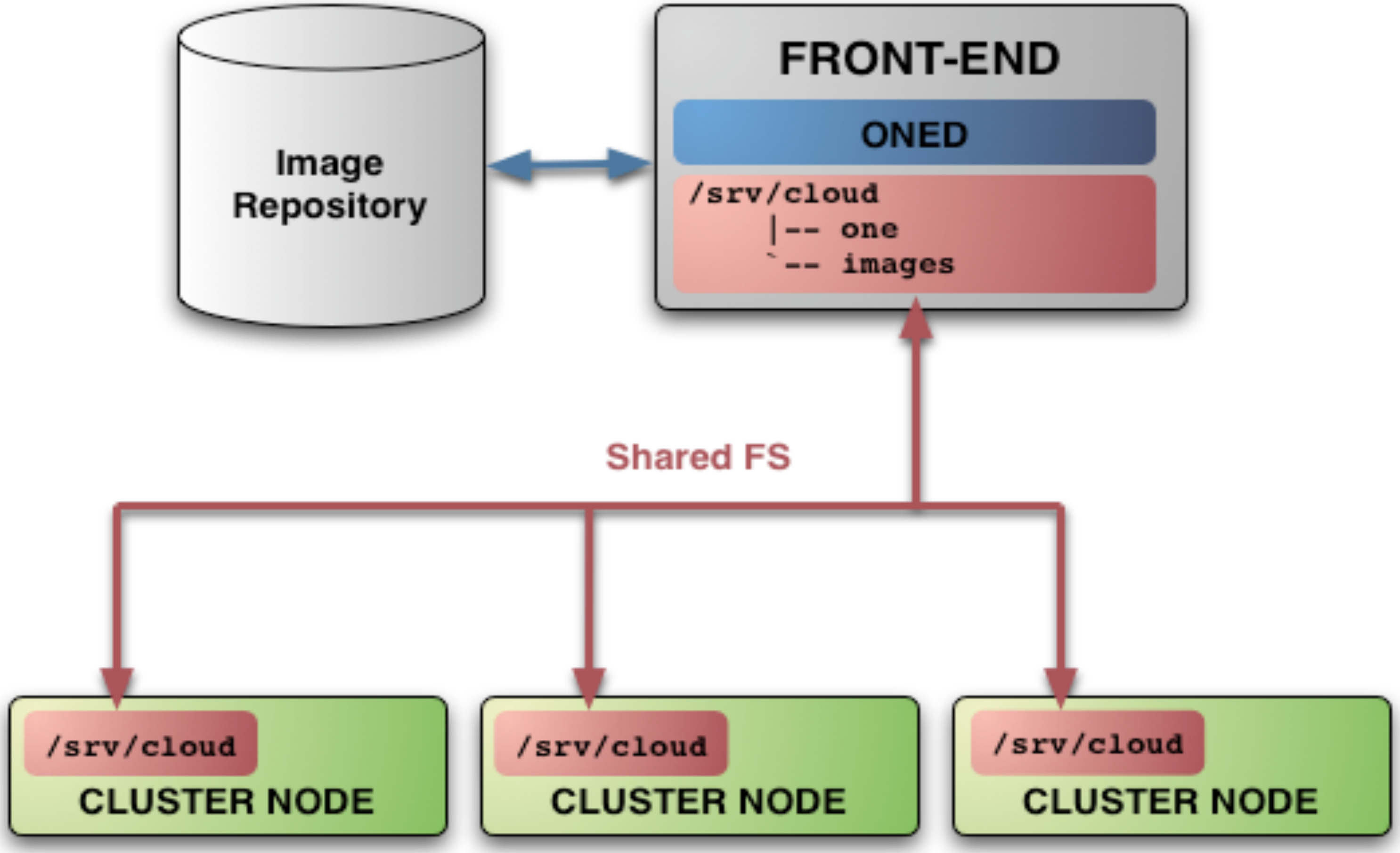
Planning the Installation: Working in the Front-End ...

- Preparing the storage for the private cloud...
  - ***Image Repository:*** Any storage medium for the VM images (usually a high performing SAN)
    - OpenNebula supports multiple back-ends (e.g. LVM for fast cloning)
    - The front-end must have access to the repository
  - ***VM Directory:*** The home of the VM in the cluster node
    - Stores checkpoints, description files and VM disks
    - Actual operations over the VM directory depends on the storage medium
    - Should be shared for live-migrations
    - You can go on without a shared FS and use the SSH back-end
    - Defaults to $ONE_LOCATION/var/$VM_ID

❗ ***Dimensioning the Storage...*** Example: A 64 core cluster will typically run around 80VMs, each VM will require an average of 10GB of disk space. So you will need ~800GB for /srv/cloud/one, you will also want to store 10-15 master images so ~200GB for /srv/cloud/images. A 1TB /srv/cloud will be enough for this example setup.

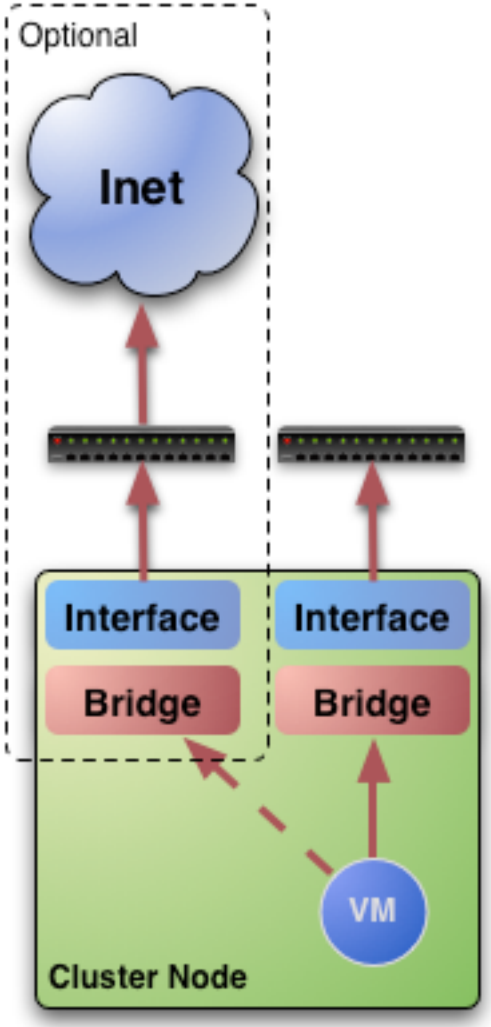
Planning the Installation: Working in the Front-End ...

- In this workshop we will use NFS to share the VM directories
- The Image Repository is /srv/cloud/images



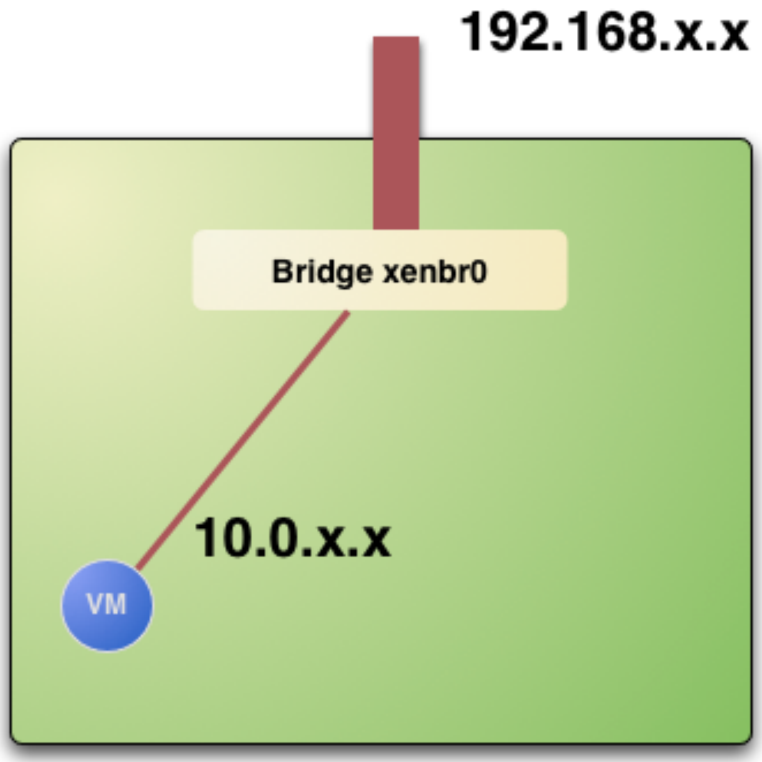
Planning the Installation: Working in the Front-End ...

- Networking for the private cloud
  - OpenNebula management operations uses a ssh connections, it does not require a performing NIC
  - ***Image traffic,*** may require the movement of heavy files (VM images, checkpoints). Dedicated storage links may be a good idea
  - ***VM demands,*** consider the typical requirements of your VMs. Several NICs to support the VM traffic may be a good idea
  - OpenNebula relies on bridge networking for the VMs



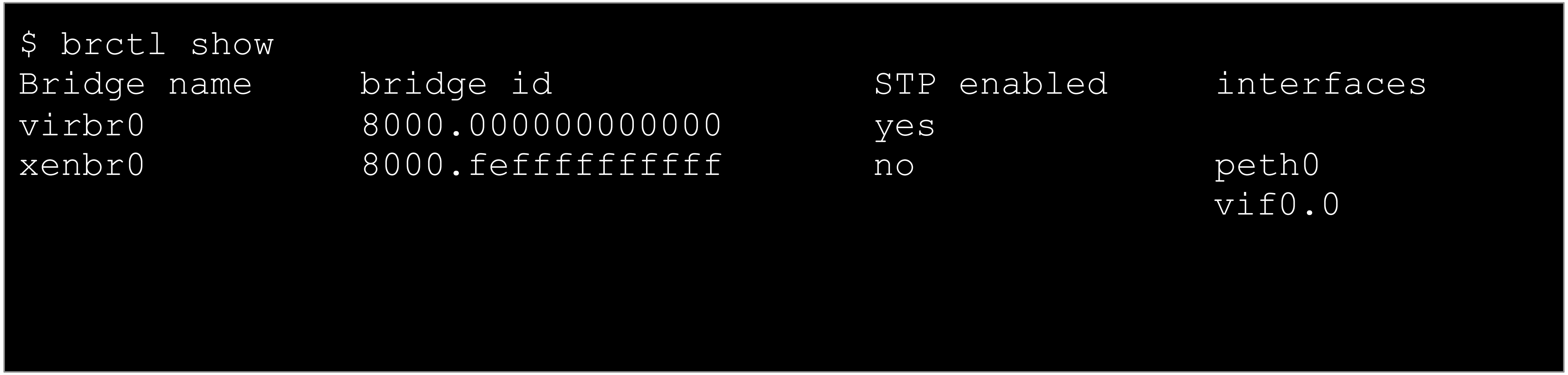
Planning the Installation: The Hypervisor ...

- Installing the Hypervisor
  - OpenNebula supports KVM, Xen and Vmware (*even simultaneously)*. This workshop applies to KVM and Xen
  - Refer to the hypervisor documentation for additional (and better information) on setting up them.
  - In this workshop, we will use XEN.



Planning the Installation: The Hypervisor ...

- The software bridge is essential for having different VMs in the same host with connectivity
- Let’s check the bridge in the hosts



```
$ brctl show
Bridge name     bridge id               STP enabled     interfaces
virbr0          8000.000000000000       yes
xenbr0          8000.feffffffffff       no              peth0
                                                        vif0.0
```

Planning the Installation: The Hypervisor ...

- Test the installation for the oneadmin account

```
$ sudo xm list
Name       ID Mem(MiB) VCPUs State    Time(s)
Domain-0    0      256     1 r-----      8.2
```

- This ensures that oneadmin is capable of running VMs

CONSEGÍ 2010

Brasilia-DF, 18-20 August 2010

PART III: Building a Private Cloud

Constantino Vazquez
(tinova@fdi.ucm.es)
Universidad Complutense de Madrid

Installing OpenNebula 1.4

- Let’s Grab the source code and compile it

```
~/SRC$ scp gw:one-1.4.0.tar.gz .
~/SRC$ tar xzvf one-1.4.0.tar.gz
~/SRC$ cd one-1.4/
~/SRC$ scons
```

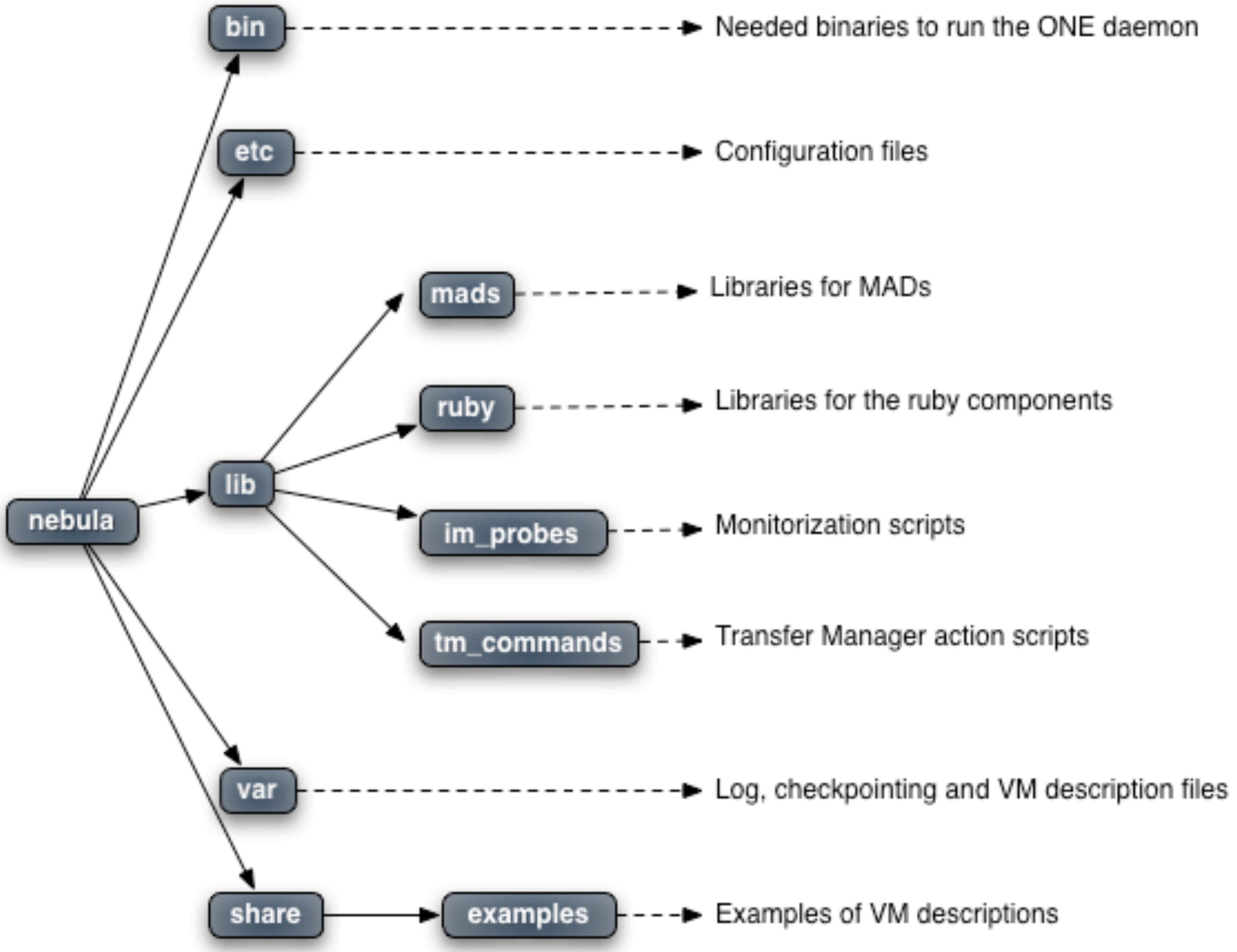
- Install the software in /srv/cloud/one (ONE_LOCATION)

```
$ export ONE_LOCATION=/srv/cloud/one/
$ ./install.sh -d $ONE_LOCATION

Check install.sh -h for other options
```

- Check and explore the installation tree

```
~$ ls -F
bin/  etc/  examples.desktop  include/  lib/  share/  SRC/  var/
```

Installing OpenNebula 1.4

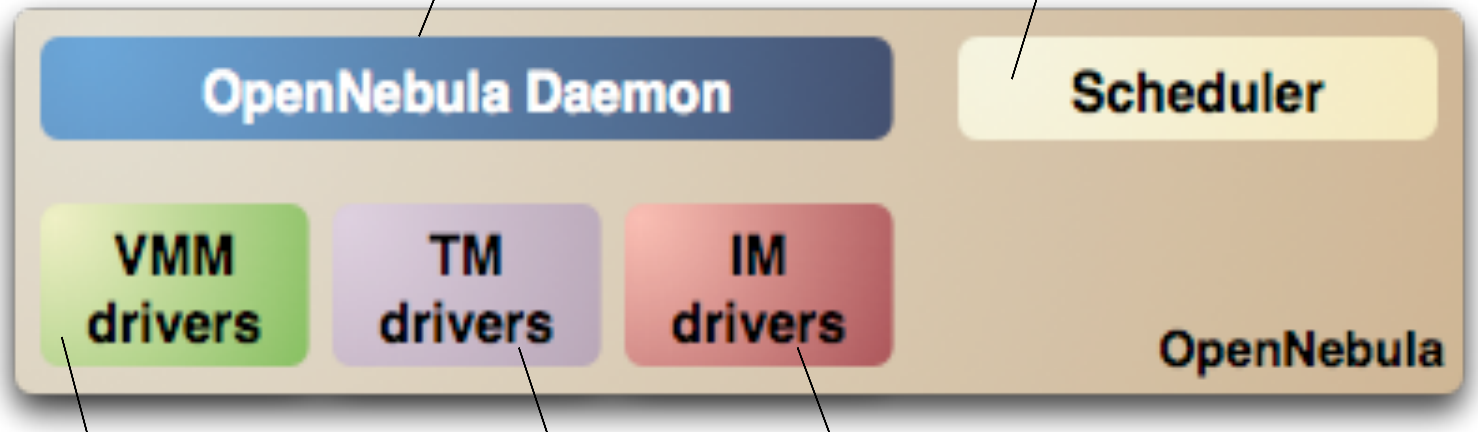


bin
Needed binaries to run the ONE daemon
etc
Configuration files
mads
Libraries for MADs
ruby
Libraries for the ruby components
nebula
lib
im_probes
Monitorization scripts
tm_commands
Transfer Manager action scripts
var
Log, checkpointing and VM description files
share
examples
Examples of VM descriptions

Configuring OpenNebula: The configuration interface

- `$ONE_LOCATION/etc/oned.conf`
- General configuration
- Defines the drivers used in the private cloud

- Match-making scheduler (default)
- Placement policies configured per VM



- `$ONE_LOCATION/etc/im_*/im_*.conf`
- Defines monitoring probes

- `$ONE_LOCATION/etc/vmm_*/vmm_*.conf`
- Defaults values for the hypervisor

- `$ONE_LOCATION/etc/tm_*/tm_*.conf`
- Defines action for generic storage operations

Configuring OpenNebula: The oned.conf file

- General configuration attributes
  - Monitoring intervals, HOST_MONITORING_INTERVAL VM_POLLING_INTERVAL
  - VM_DIR: Path to the VM directory for all the cluster nodes.
  - Network parameters, MAC_PREFIX, NETWORK_SIZE
  - PORT : Port where oned will listen for xml-rpc calls
  - DEBUG_LEVEL

```
HOST_MONITORING_INTERVAL = 60
VM_POLLING_INTERVAL      = 60

#VM_DIR      = /srv/cloud/one/var

MAC_PREFIX   = "00:01"
NETWORK_SIZE = 254

PORT         =  2633
DEBUG_LEVEL  = 3
```

Configuring OpenNebula: The oned.conf file

- Information Drivers, to monitor cluster nodes
  - name: identifies the driver
  - executable: absolute or relative to $ONE_LOCATION/lib/mads
  - arguments: a probe configuration file

```
IM_MAD = [
      name       = "im_xen",
      executable = "one_im_ssh",
      arguments  = "im_xen/im_xen.conf" ]
```

- Transfer Drivers, to interface with the storage
  - name: identifies the driver
  - executable: path to driver executable
  - arguments: storage commands configuration file

```
TM_MAD = [
    name       = "tm_nfs",
    executable = "one_tm",
    arguments  = "tm_nfs/tm_nfs.conf" ]
```

Configuring OpenNebula: The oned.conf file

- Virtualization Drivers, to interface the hypervisors
  - name: identifies the driver
  - executable: absolute or relative to $ONE_LOCATION/lib/mads
  - arguments: (not needed for the distribution drivers)
  - default: default values for the hypervisor
  - type: format of the VM description file to be passed to the driver: xen, kvm or xml

```
VM_MAD = [
    name        = "vmm_xen",
    executable  = "one_vmm_xen",
    default     = "vmm_xen/vmm_xen.conf",
    type        = ”xen" ]
```

- Hooks, custom programs that are executed on specific events, e.g. VM creation.
- Hands on... Check and adjust the values of oned.conf for your cloud

Configuring OpenNebula: Accounts

- Accounts in OpenNebula
  - **`oneadmin`**, has enough privileges to perform any operation on any object. It is created the first time OpenNebula is started using the `ONE_AUTH` data
  - Regular **user accounts** must be created by `oneadmin` and they can only manage their own objects.
- Configuring the oneadmin account
  - Environment variables: `ONE_AUTH`, `ONE_LOCATION` and `ONE_XMLRPC`

```
$ tail .bashrc
export ONE_LOCATION=/srv/cloud/one
export ONE_AUTH=$HOME/.one/one_auth
export PATH=$PATH:$ONE_LOCATION/bin
```

  - Create the password file

```
$ mkdir .one
$ cd .one
$ cat one_auth
oneadmin:onecloud
```

Configuring OpenNebula: Start & Stop

- Use the `one` script

```
$ source .bashrc
$ echo $ONE_AUTH
/srv/cloud/one/.one/one_auth

$one start
oned and scheduler started

$ more $ONE_LOCATION/var/oned.log
Thu Jan 14 18:03:11 2010 [ONE][I]: Init OpenNebula Log system
Thu Jan 14 18:03:11 2010 [ONE][I]: Log Level: 3 [0=ERROR,1=WARNING,
2=INFO,3=DEBUG]
Thu Jan 14 18:03:11 2010 [ONE][I]: ----------------------------------
Thu Jan 14 18:03:11 2010 [ONE][I]:      OpenNebula Configuration File
Thu Jan 14 18:03:11 2010 [ONE][I]: ----------------------------------
```

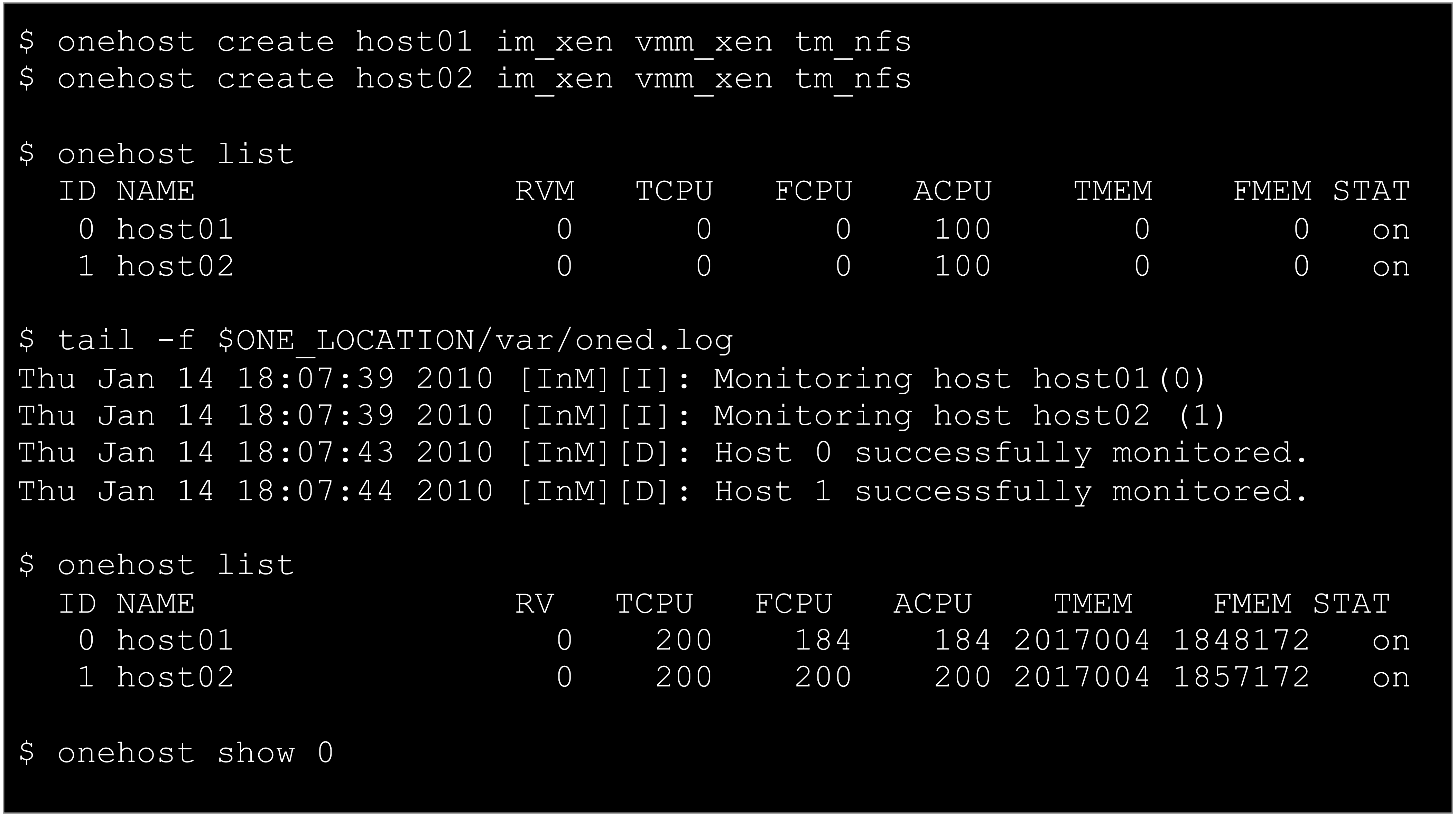
❗ Be sure to configure the oneadmin account (specially, create the ONE_AUTH file) before starting OpenNebula for the first time.

Configuring OpenNebula: Hosts

- Cluster nodes are defined with
  - *Hostname* of the cluster node or IP
  - *Information Driver* to be used to monitor the host
  - *Storage Driver* to clone, delete, move or copy images into the host
  - *Virtualization Driver* to boot, stop, resume VMs in the host
- Cluster nodes are managed with the onehost utility
  - Create & delete hosts
  - List the hosts in the cluster
  - Show detailed information from a host
  - Enable/Disable a host

Configuring OpenNebula: Hosts

- Hands on... configure the hosts of your private cloud



```
$ onehost create host01 im_xen vmm_xen tm_nfs
$ onehost create host02 im_xen vmm_xen tm_nfs

$ onehost list
  ID NAME                RVM   TCPU   FCPU   ACPU    TMEM    FMEM STAT
   0 host01                0      0      0    100       0       0   on
   1 host02                0      0      0    100       0       0   on

$ tail -f $ONE_LOCATION/var/oned.log
Thu Jan 14 18:07:39 2010 [InM][I]: Monitoring host host01(0)
Thu Jan 14 18:07:39 2010 [InM][I]: Monitoring host host02 (1)
Thu Jan 14 18:07:43 2010 [InM][D]: Host 0 successfully monitored.
Thu Jan 14 18:07:44 2010 [InM][D]: Host 1 successfully monitored.

$ onehost list
  ID NAME                RV   TCPU   FCPU   ACPU    TMEM    FMEM STAT
   0 host01                0    200    184    184 2017004 1848172   on
   1 host02                0    200    200    200 2017004 1857172   on

$ onehost show 0
```

- Hands on... Explore and test the onehost command in your cloud

Configuring OpenNebula: Users

- Users are defined within OpenNebula by:
  - *ID* unique identifier for the user
  - *Name* of the user, used for authentication
  - *Password* used for authentication
- Users are managed with the oneuser utility
  - Create & delete users
  - List the users in the cluster
- Hands on... create new users in your private cloud and configure the "*user*" UNIX account

```
$ oneuser create helen mypass
User "Helen" should put helen:mypass in $ONE_AUTH

$ oneuser list
 UID NAME      PASSWORD                                  ENABLE
   0 oneadmin  c24783ba96a35464632a624d9f829136edc0175e    True
   2 helen     34a91f713808846ade4a71577dc7963631ebae14    True

$ oneuser delete helen
```

Configuring OpenNebula: Log Files

- The operations of the OpenNebula daemon and scheduler are logged in:
  - oned: $ONE_LOCATION/var/oned.log, Its verbosity is set by DEBUG_LEVEL in $ONE_LOCATION/etc/oned.conf.
  - Scheduler (mm_sched): All the scheduler information is collected into the $ONE_LOCATION/var/sched.log file.
- VM logs and files are in $ONE_LOCATION/var/<VM_ID>, more in a few slides...
- Drivers can activate ONE_MAD_DEBUG in the associated RC file (or in $ONE_LOCATION/etc/defaultrc)

Using the Private Cloud: Virtual Networks

- A Virtual Network in OpenNebula
  - Defines a separated MAC/IP address space to be used by VMs
  - Each virtual network is associated with a physical network through a bridge
  - Virtual Networks can be isolated (at layer 2 level) with ebtables and hooks
- Virtual Network definition
  - **Name,** of the network
  - **Type**
    - **Fixed**, a set of IP/MAC leases
    - **Ranged,** defines a network range
  - **Bridge**, name of the physical bridge in the physical host where the VM should connect its network interface.
- Virtual Networks are managed with the `onevnet` utility

⚠ Networks created by oneadmin are *public, i.e.* can be used by VMs of any other user

Using the Private Cloud: Virtual Networks

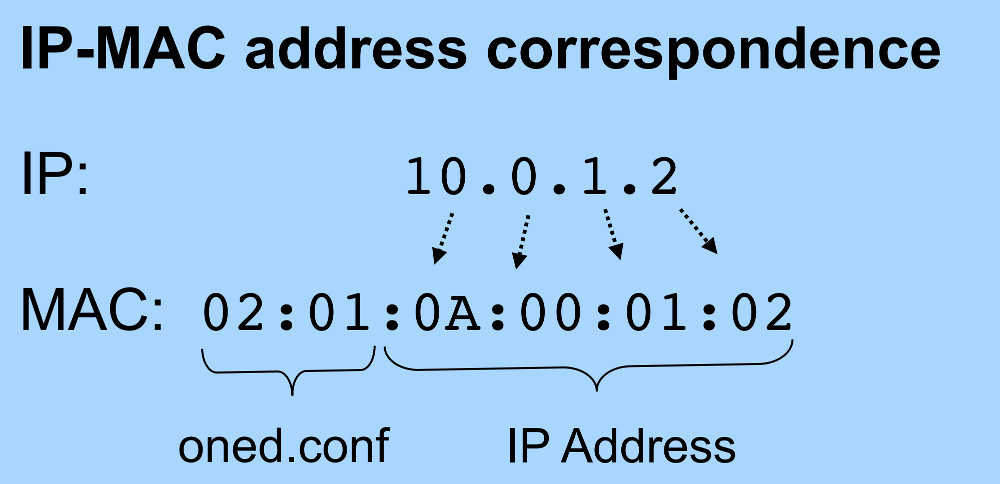
```
$ cat real.net
NAME = ”One-TD"
TYPE = RANGED
BRIDGE = xenbr0
NETWORK_SIZE    = 125
NETWORK_ADDRESS = 192.168.$CN.128

$ cat fake.net
NAME = “One-TD-Invisible"
TYPE = FIXED
BRIDGE = xenbr0
LEASES = [IP=192.168.($CN+100).5]
LEASES = [IP=192.168.($CN+100).10]
LEASES = [IP=192.168.($CN+100).15]
LEASES = [IP=192.168.($CN+100).20]
LEASES = [IP=192.168.($CN+100).25]

$ onevnet -v create real.net
$ onevnet -v create fake.net
```

Using the Private Cloud: Virtual Networks

- Using a Virtual Network with your VMs
  - Define NICs attached to a given virtual network. The VM will get a NIC with a free MAC in the network and attached to the corresponding bridge

```
#A VM with two interfaces each one in a different vlan
NIC=[NETWORK="One-TD"]
NIC=[NETWORK="One-TD-Invisible"]

#Ask for a specific IP/MAC of the Red vlan
NIC=[NETWORK="One-TD", IP=192.168.$CN.140]
```

  - Prepare the VM to use the IP. Sample scripts to set the IP based on the MAC are provided for several Linux distributions.



IP-MAC address correspondence

IP: 10.0.1.2

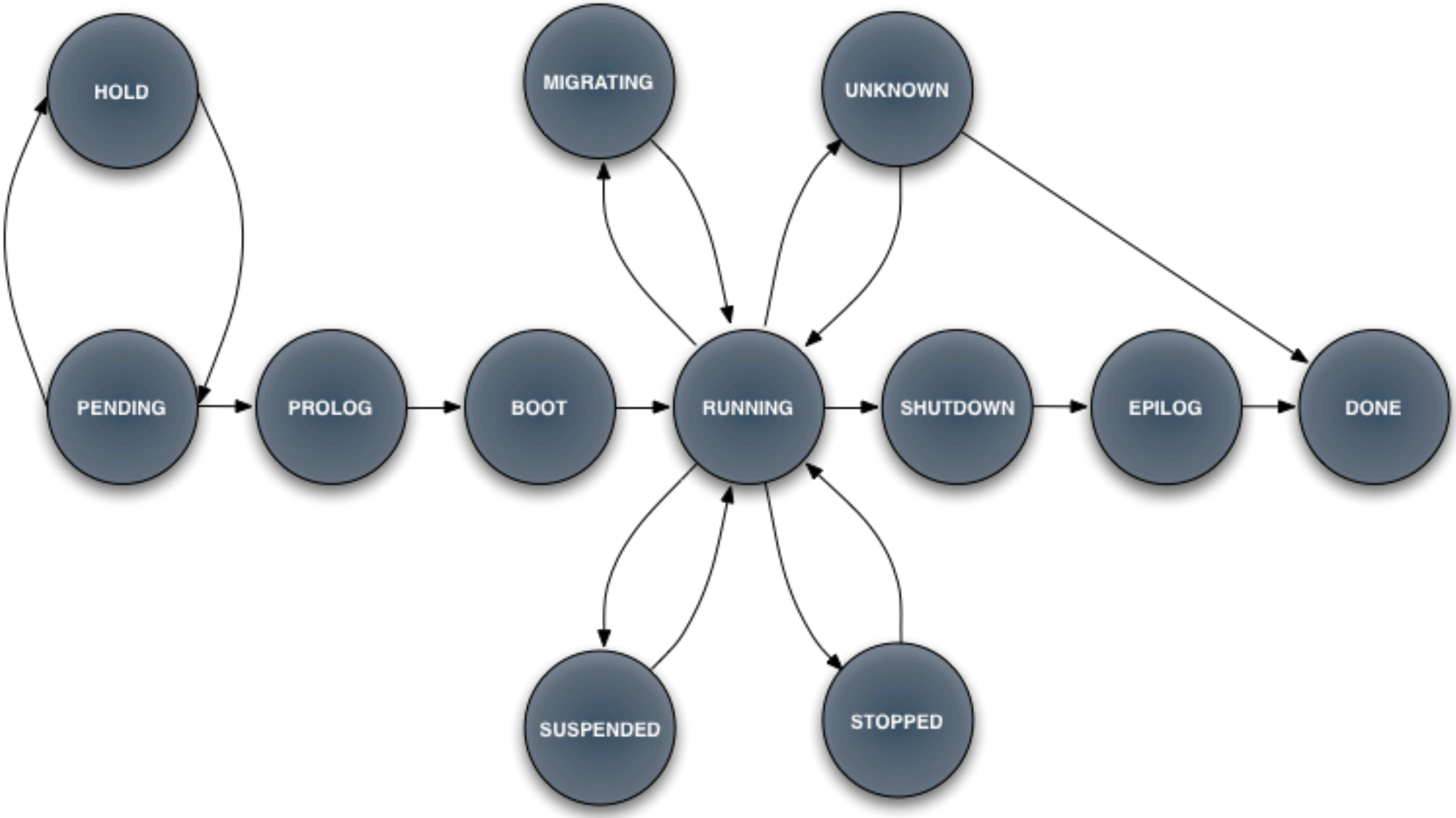
MAC: 02:01:0A:00:01:02

Using the Private Cloud: Virtual Machines

- Preparing a VM to be used with OpenNebula
  - You can use any VM prepared for the target hypervisor
  - **Hint I**: Place the vmcontext.sh script in the boot process to make better use of vlans
  - **Hint II**: Do not pack useless information in the VM images:
    - swap. OpenNebula can create swap partitions on-the-fly in the target host
    - Scratch or volatile storage. OpenNebula can create plain FS on-the-fly in the target host
  - **Hint III:** Install once and deploy many; prepare master images
  - **Hint IV:** Do not put private information (e.g. ssh keys) in the master images, use the `CONTEXT`
  - **Hint V:** Pass arbitrary data to a master image using `CONTEXT`

Using the Private Cloud: Virtual Machines

- Virtual Machine Life-cycle



Using the Private Cloud: Virtual Machines

- A Virtual Machine in OpenNebula
  - A **capacity** in terms memory and CPU
  - A set of **NICs** attached to one or more virtual networks
  - A set of **disk images**, to be “*transferred*” to/from the execution host.
  - A **state file** (optional) or recovery file, with the memory image of a running VM plus some hypervisor specific information.
- Virutal Machines are defined in a VM template
- Each VM has an unique ID in OpenNebula → the VM_ID
- All the files (logs, images, state files...) are stored in `$ONE_LOCATION/var/<VM_ID>`

Using the Private Cloud: Virtual Machines

- Virtual Machine Definition File (*VM templates*)

```
#----------------------------------------
# Name of the VM
#----------------------------------------
NAME = "vm-example" # Optional, Default: one-$VMID

#----------------------------------------
#                Capacity
#----------------------------------------
CPU    = "amount_of_requested_CPU"
MEMORY = "amount_of_requested_MEM"
VCPU   = "number of virtual cpus"

#----------------------------------------
#          OS and boot options
#----------------------------------------
OS = [
  kernel     = "path_to_os_kernel",      # para-virtualization
  initrd     = "path_to_initrd_image",   # para-virtualization
  kernel_cmd = "kernel_command_line",
  root       = "device to be mounted as root"
  bootloader = "path to the boot loader exec”
  boot       = "device to boot from" ]
```

Using the Private Cloud: Virtual Machines

- Virtual Machine Definition File (*VM templates*)

```
#---------------------------------------
#      Features of the hypervisor
#---------------------------------------

FEATURES = [
  pae  = "yes|no",    # Optional, KVM
  acpi = "yes|no" ]   # Optional, KVM

#---------------------------------------
#                VM Disks
#---------------------------------------

DISK = [
  type     = "floppy|disk|cdrom|swap|fs|block",
  source   = "path_to_disk_image_file|physical_dev",
  format   = “type for fs disks”,
  size     = "size_in_GB",
  target   = "device_to_map_disk",
  bus      = "ide|scsi|virtio|xen",
  readonly = "yes|no",
  clone    = "yes|no",
  save     = "yes|no" ]
```

Using the Private Cloud: Virtual Machines

- Virtual Machine Definition File (*VM templates*)

```
#---------------------------------------
#           Network Interfaces
#---------------------------------------

NIC = [
  network = "name_of_the_virtual_network",
  ip      = "ip_address",
  bridge  = "name_of_bridge_to_bind_if",
  target  = "device_name_to_map_if",
  mac     = "HW_address",
  script  = "path_to_script_to_bring_up_if",
  Model   = "NIC model"]

#---------------------------------------
#  I/O Interfaces
#---------------------------------------

INPUT = [
  type = "mouse|tablet",
  bus  = "usb|ps2|xen" ]
```

Using the Private Cloud: Virtual Machines

- Virtual Machine Definition File (*VM templates*)

```
#--------------------------------------
#  I/O Interfaces
#--------------------------------------

GRAPHICS = [
  type   = "vnc|sdl",
  listen = "IP-to-listen-on",
  port   = "port_for_VNC_server",
  passwd = "password_for_VNC_server" ]

#--------------------------------------
#  Raw Hypervisor attributes
#--------------------------------------

RAW = [
  type = "xen|kvm",
  data = "raw_domain_configutarion"]
```

Not all the parameters are supported for each hypervisor. Complete reference and examples for all sections in
`http://www.opennebula.org/doku.php?id=documentation:rel1.4:template`

Using the Private Cloud: Virtual Machines

- Let’s ttylinux VM

```
NAME    = ttylinux
CPU     = 0.1
MEMORY  = 64

DISK    = [
   source   = "/srv/cloud/images/ttylinux/ttylinux.img",
   target   = "hda",
   readonly = "no" ]

NIC       = [ NETWORK = ”One-TD" ]

FEATURES = [ acpi="no" ]

#This may be useful to debug your VMs (can use also console)

GRAPHICS = [
   type = "vnc",
   listen  = "loclahost",
   port = "5902",
   keymap="es"]
```

Using the Private Cloud: Virtual Machines

- Let’s copy the one ttylinux image form the front-end

```
$ cd /srv/one/images
$ scp gw:ttylinux-xen.tar.gz .
$ tar xvzf ttylinux-xen.tar.gz
```

- Virtual Machines are managed with the onevm utility
  - Operations: create, deploy shutdown, livemigrate, stop, cancel, resume, suspend, delete, restart
  - Information: list, show, top, history

```
$ onevm create ttylinux.one

$ onevm list
  ID     USER     NAME STAT CPU     MEM        HOSTNAME        TIME
  1  oneadmin ttylinux pend   0       0                 00 00:00:28

$ onevm top
```

CONSEGÍ 2010

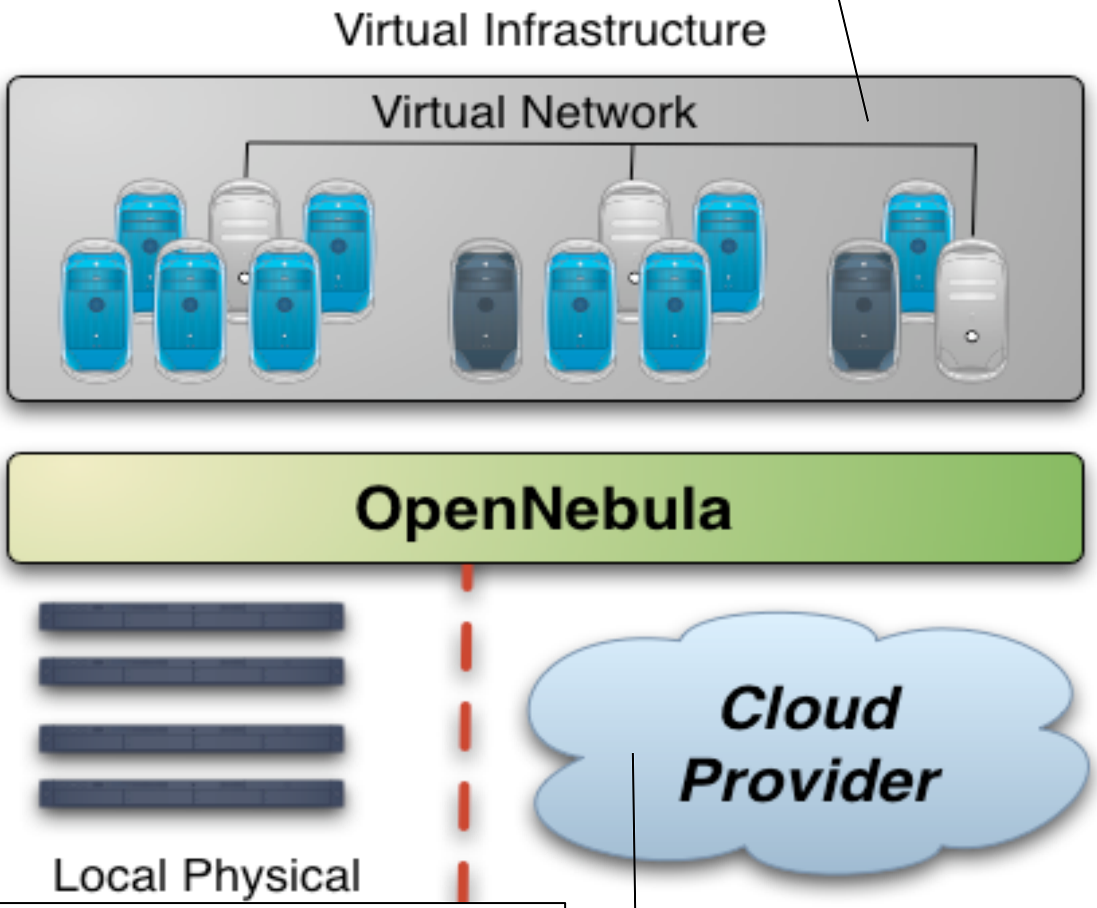
Brasilia-DF, 18-20 August 2010

PART IV: Building your Hybrid Cloud

Constantino Vazquez
(tinova@fdi.ucm.es)
Universidad Complutense de Madrid

Hybrid Cloud Computing: Overview

- VMs can be local or remote
- VM connectivity has to be configured, usually VPNs



- External Clouds are like any other host
- Placement constraints

Installing the Hybrid Cloud Components

- OpenNebula distribution includes drivers to build hybrid clouds with Amazon EC2 and Elastic Hosts
- Let’s try the EC2 tools (`ec2-*`)

```
$ echo $EC2_PRIVATE_KEY

$ echo $EC2_CERT

$ ec2-describe-images
IMAGE ami-da9f7bb3    eggplant/image.manifest.xml     587384515363
      available  private           i386  machine    aki-a71cf9ce    ari-
a51cf9cc
IMAGE ami-a99e7ac0    nginx-apple/image.manifest.xml  587384515363
      available  private           i386  machine    aki-a71cf9ce    ari-
a51cf9cc
```

Configuring the EC2 Hybrid Cloud Driver

- First, we need to add the following drivers to oned.conf

```
IM_MAD = [
  name       = "im_ec2",
  executable = "one_im_ec2",
  arguments  = "im_ec2/im_ec2.conf" ] # No. of instances of each type

VM_MAD = [
  name       = "vmm_ec2",
  executable = "one_vmm_ec2",
  arguments  = "vmm_ec2/vmm_ec2.conf", # Defaults, e.g. keypair
  type       = "xml" ]

TM_MAD = [  #No actual transfers are made by OpenNebula to EC2
    name       = "tm_dummy",
    executable = "one_tm",
    arguments  = "tm_dummy/tm_dummy.conf" ]
```

- Let’s check the values of the driver configurations files

Configuring the EC2 Hybrid Cloud Driver

- Configure the account to be used with Amazon EC2

```
$ vim $ONE_LOCATION/etc/vmm_ec2/vmm_ec2rc
#-------------------------------------------------------------------
# EC2 API TOOLS Configuration.
#-------------------------------------------------------------------
EC2_HOME=/usr
EC2_PRIVATE_KEY="/srv/cloud/one/ec2/pk.pem"
EC2_CERT="/srv/cloud/one/ec2/cert.pem"
```

- Restart the OpenNebula daemon, and check that the new drivers are loaded

```
$ one stop; one start
$ more $ONE_LOCATION/var/oned.log
Fri Jan 15 18:16:46 2010 [VMM][I]: Loading Virtual Machine Manager driv
Fri Jan 15 18:16:46 2010 [VMM][I]:     Loading driver: vmm_kvm (KVM)
Fri Jan 15 18:16:47 2010 [VMM][I]:     Driver vmm_kvm loaded.
Fri Jan 15 18:16:47 2010 [VMM][I]:     Loading driver: vmm_ec2 (XML)
Fri Jan 15 00:16:47 2010 [InM][I]: Loading Information Manager drivers.
Fri Jan 15 00:16:47 2010 [InM][I]:     Loading driver: im_kvm
Fri Jan 15 00:16:47 2010 [InM][I]:     Driver im_kvm loaded
Fri Jan 15 00:16:47 2010 [InM][I]:     Loading driver: im_ec2
```

Configuring the EC2 Hybrid Cloud Driver

- Amazon EC2 cloud is manage by OpenNebula as any other cluster node
  - You can use **several accounts** by adding a driver for each account (use the arguments attribute, `-k` and `-c` options). Then create a host that uses the driver
  - You can use **multiple EC2 zones**, add a driver for each zone (use the arguments attribute, `-u` option), and a host that uses that driver
  - You can limit the use of EC2 instances by modifying the IM file
- Lets create your EC2 hybrid cloud by adding a new host

```
$ onehost create ec2 im_ec2 vmm_ec2 tm_dummy

$ onehost list
  ID NAME                      RVM    TCPU    FCPU    ACPU    TMEM    FMEM STAT
   0 84.21.x.y                   0     200     200     200 2017004 1667080   on
   1 84.21.x.z                   1     200     200     200 2017004 1681676   on
   2 ec2                         0     500     500     500 8912896 8912896   on
```

Using the EC2 Hybrid Cloud

- Virtual Machines can be instantiated locally or in EC2
  - The template must provide a description for both instantiation methods.
  - The EC2 counterpart of your VM (`AMI_ID`) must be available for the driver account
  - The EC2 VM template attribute:

```
EC2 = [
  AMI               = "ami_id for this VM",
  KEYPAIR           = "the keypair to use the instance",
  AUTHORIZED_PORTS  = "ports to access the instance",
  INSTANCETYPE      = "m1.small...",
  ELASTICIP         = "the elastic ip for this instance",
  CLOUD             = "host (EC2 cloud) to use this description with"
]
```

Using the EC2 Hybrid Cloud

- Add an EC2 counterpart to the ttylinux image

```
$vi ttylinux.one
#EC2 template machine, this will be use wen submitting this VM to EC2
EC2 = [ AMI="ami-ccf615a5",
        KEYPAIR="keypair",
        AUTHORIZED_PORTS="22",
        INSTANCETYPE=m1.small]

#Add this if you want to use only EC2 cloud
REQUIREMENTS = "HOSTNAME = \"ec2\""
```

- Create the VM and check progress

```
$ onevm create ttylinux.one
$ onevm list
  ID      USER      NAME  STAT CPU      MEM           HOSTNAME        TIME
  16 oneadmin    one-16 runn    0        0                ec2 00 00:00:35
$ ec2-describe-instances
RESERVATION      r-5eff7536      418314910487    default
INSTANCE         i-bac3f0d2      ami-0572946c                        pending
keypair0         m1.small        2010-01-14T23:32:35+0000           us-
east-1a     aki-a71cf9ce     ari-a51cf9cc            monitoring-
disabled
```

Using the EC2 Hybrid Cloud

- Log in the EC2 instance when running

```
$ onevm show 17
...
VIRTUAL MACHINE TEMPLATE
CPU=0.5
...
EC2=[
  AMI=ami-ccf615a5,
  AUTHORIZED_PORTS=22,
  INSTANCETYPE=m1.small,
  KEYPAIR=keypair ]
IP=ec2-72-44-62-194.compute-1.amazonaws.com
 ...
REQUIREMENTS=HOSTNAME = "ec2"
VMID=17

$ ssh -i keypair.pem root@ec2-72-44-62-194.compute-1.amazonaws.com
Linux ip-10-212-134-128 2.6.21.7-2.fc8xen-ec2-v1.0 #2 SMP Tue Sep 1
10:04:29 EDT 2009 i686
root@ip-10-212-134-128:~#

This costs money!
$ onevm shutdown 17
$ onehost disable ec2
$ onehost list
```

CONSEGÍ 2010

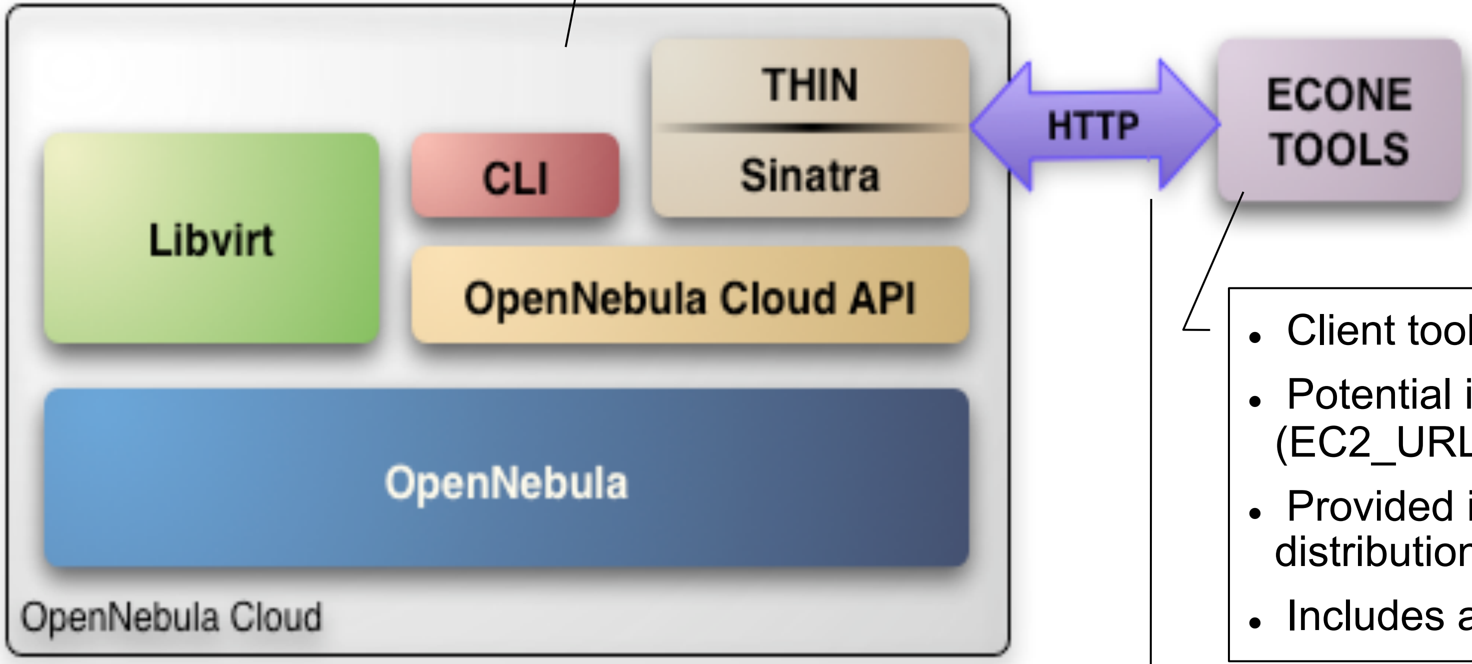
Brasilia-DF, 18-20 August 2010

PART V: Building your Public Cloud

Constantino Vazquez
(tinova@fdi.ucm.es)
Universidad Complutense de Madrid

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

Configuring the Public Cloud

- The EC2 service is configured in `$ONE_LOCATION/etc/econe.conf`

```
$ 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

- Let’s 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

- 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

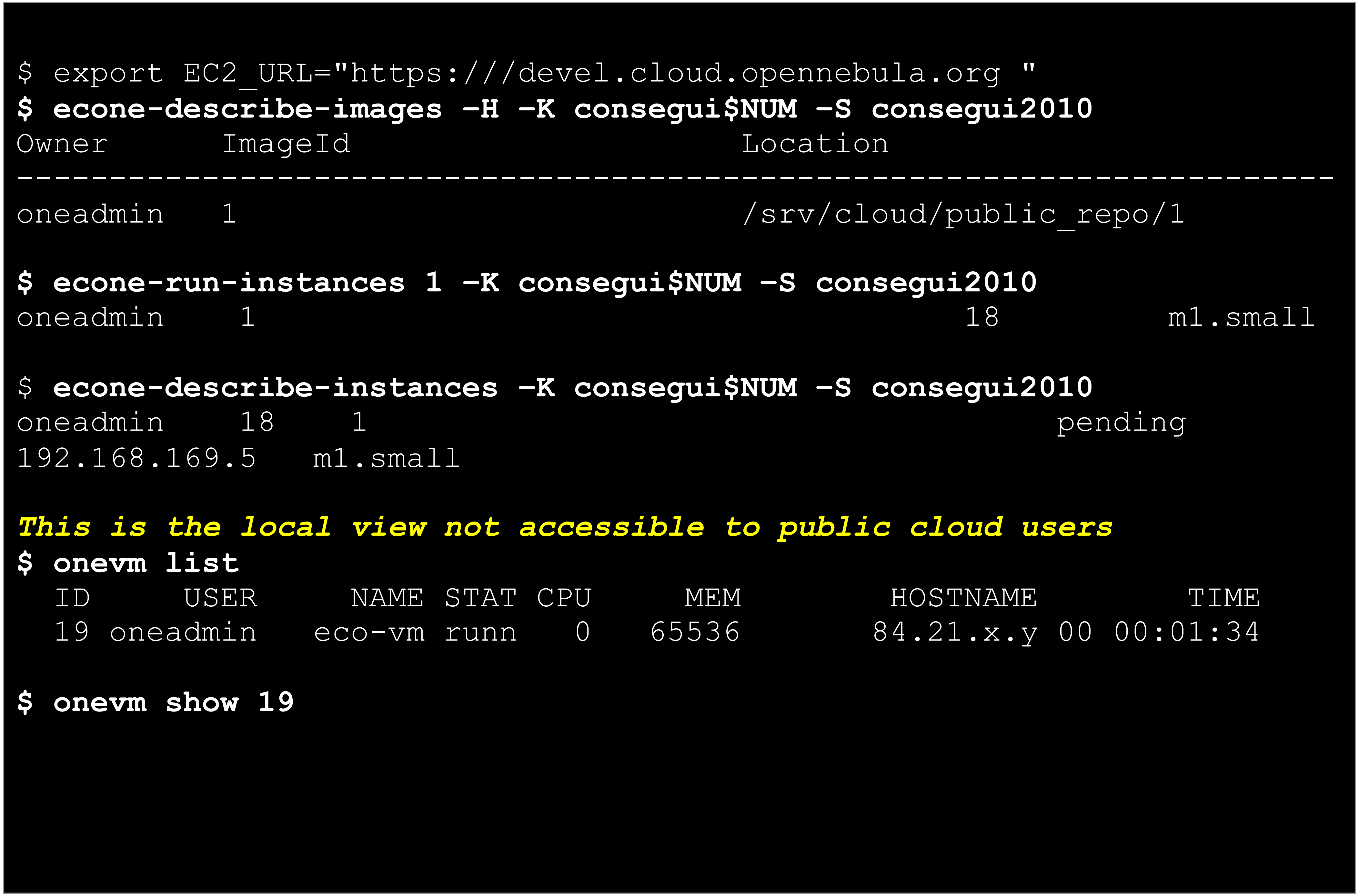
HANDS ON

- Install the clients (./install –c ec2)
- Pass your credentials to the econe-tools by (in this order)
  - Commands arguments (--access-key <username>,
    --secret-key <pass>)

```
U: consegui$NUM    NUM={01-30}
P: consegui2010
```

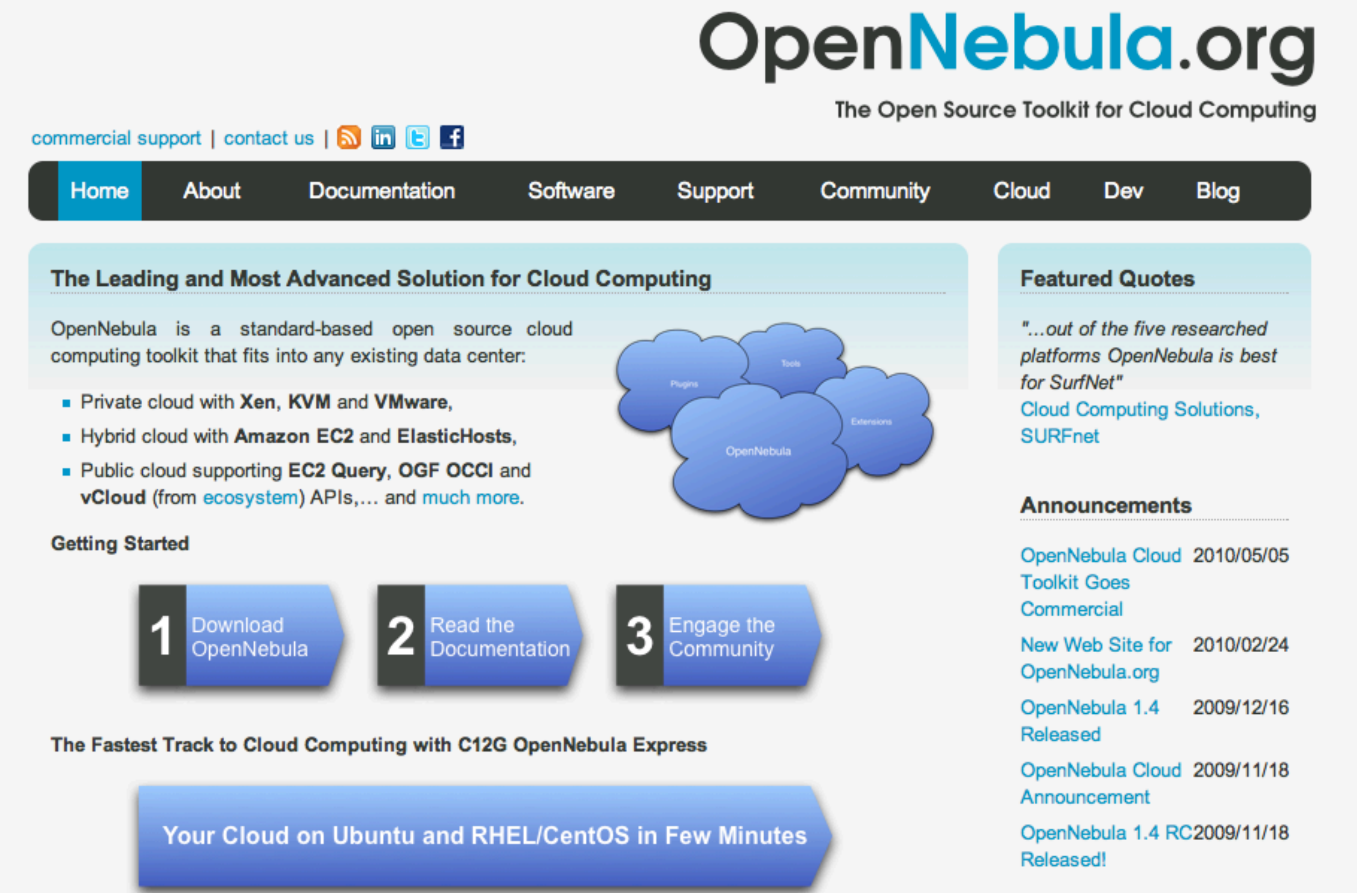
  - Environment `EC2_ACCESS_KEY` and `EC2_SECRET_KEY`
  - Environment `ONE_AUTH`
- Point econe-tools to your target cloud
  - Command arguments (--url <http|https>://<fqdn>:<port>) port needed in not the default for the protocol
  - `EC2_URL` environment

Using the Public Cloud



```
$ export EC2_URL="https:///devel.cloud.opennebula.org "
$ econe-describe-images -H -K consegui$NUM -S consegui2010
Owner      ImageId                     Location
-----------------------------------------------------------------------
oneadmin   1                           /srv/cloud/public_repo/1

$ econe-run-instances 1 -K consegui$NUM -S consegui2010
oneadmin    1                                      18         m1.small

$ econe-describe-instances -K consegui$NUM -S consegui2010
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
```

More info, downloads, mailing lists at



OpenNebula.org

The Open Source Toolkit for Cloud Computing

commercial support | contact us |

The Leading and Most Advanced Solution for Cloud Computing

OpenNebula is a standard-based open source cloud computing toolkit that fits into any existing data center:

- Private cloud with **Xen**, **KVM** and **VMware**,
- Hybrid cloud with **Amazon EC2** and **ElasticHosts**,
- Public cloud supporting **EC2 Query**, **OGF OCCI** and **vCloud** (from ecosystem) APIs,... and much more.

Getting Started

The Fastest Track to Cloud Computing with C12G OpenNebula Express

Featured Quotes

"...out of the five researched platforms OpenNebula is best for SurfNet"
Cloud Computing Solutions, SURFnet

Announcements

OpenNebula Cloud Toolkit Goes Commercial	2010/05/05
New Web Site for OpenNebula.org	2010/02/24
OpenNebula 1.4 Released	2009/12/16
OpenNebula Cloud Announcement	2009/11/18
OpenNebula 1.4 RC Released!	2009/11/18

Time? For Questions

CONSEGÍ 2010

Brasilia-DF, 18-20 August 2010

EXTRAS

Constantino Vazquez
(tinova@fdi.ucm.es)
Universidad Complutense de Madrid

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)