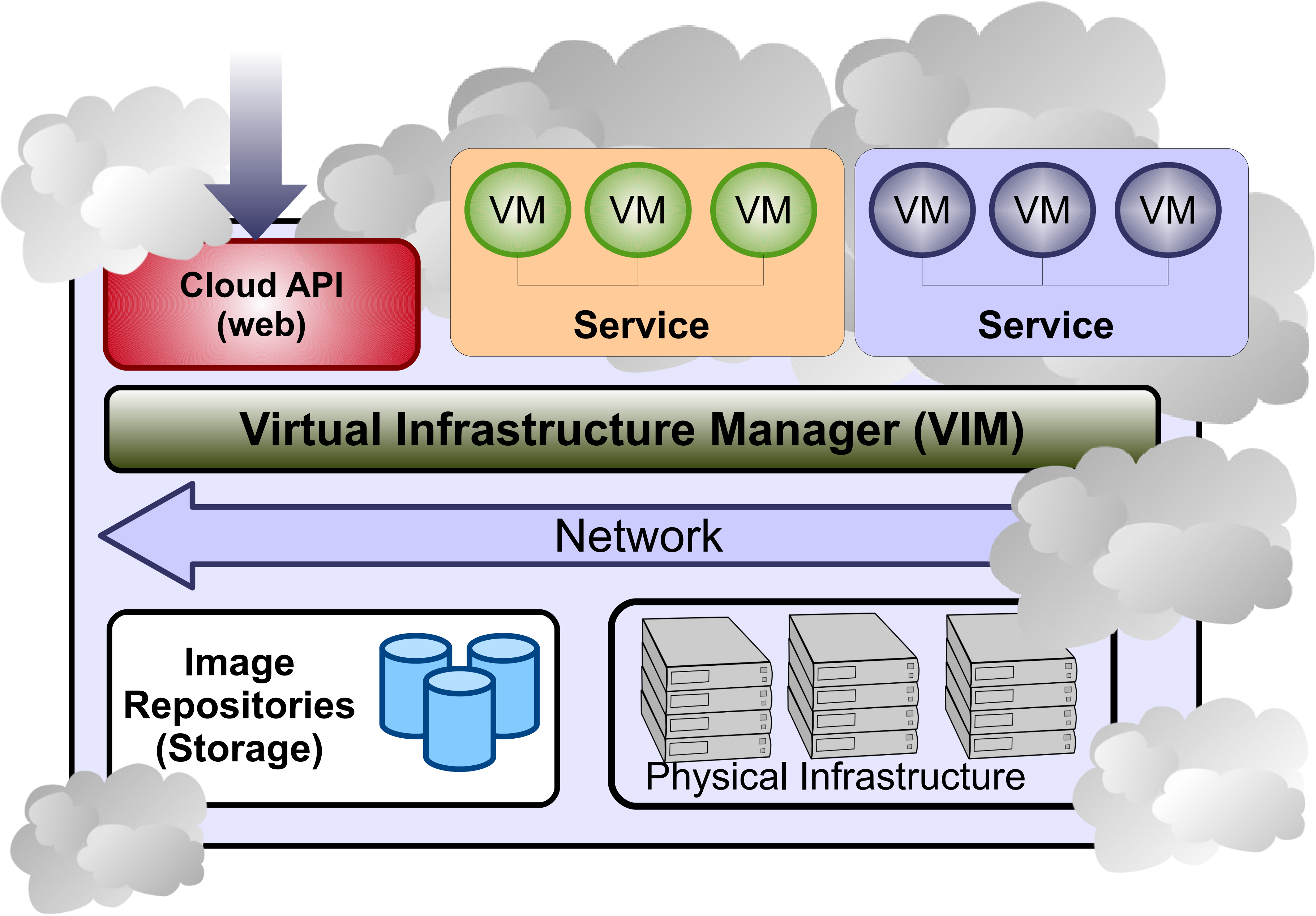
EL / LAK (FOSS) 2010

May 14th, 2010

Deployment of Private, Hybrid & Public Clouds with OpenNebula

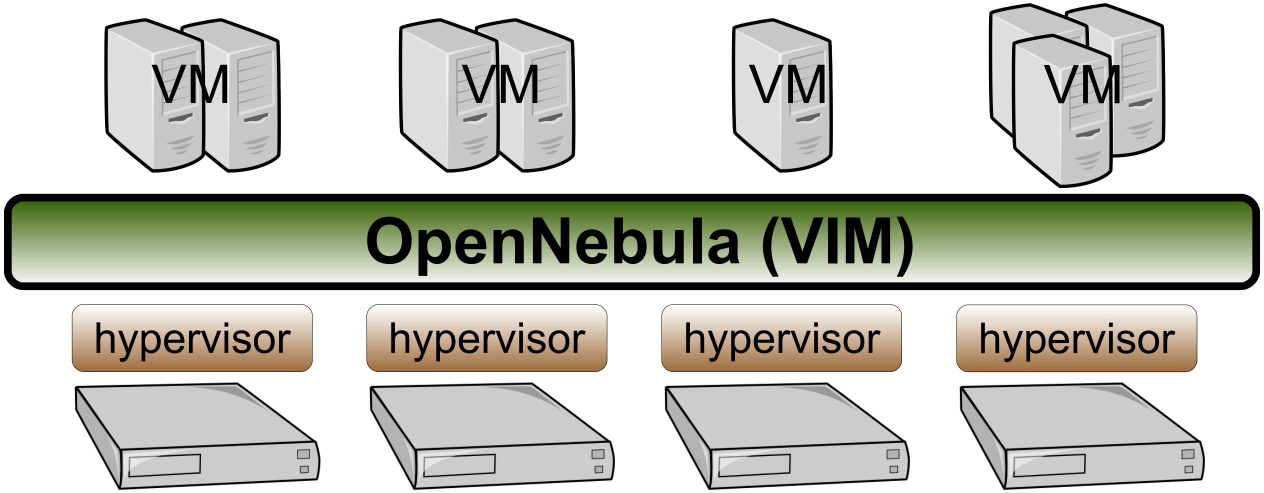
Javier Fontán
University Complutense of Madrid

The Anatomy of an IaaS Cloud



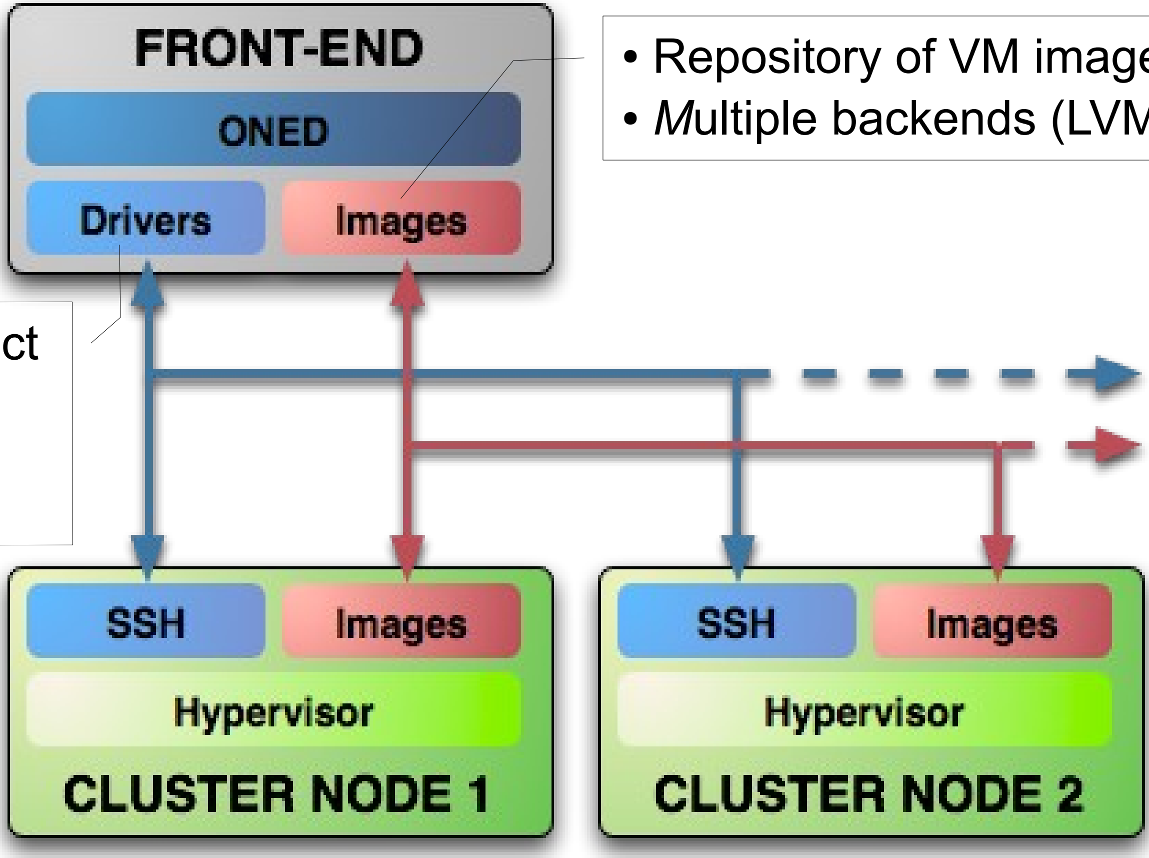
Why a Virtual Infrastructure Manager?

- 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



Overview of an OpenNebula Cloud

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



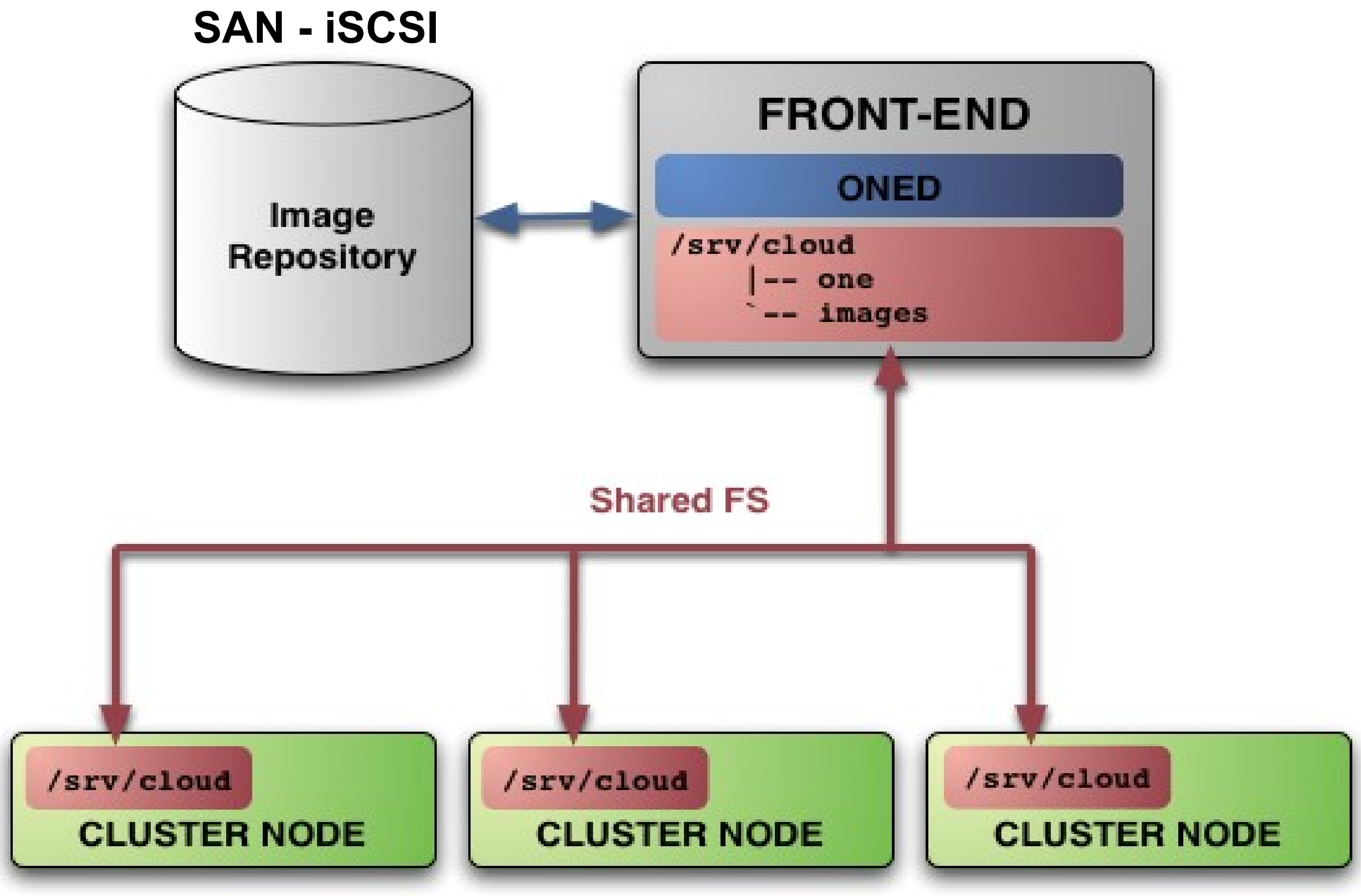
- Repository of VM images
- *M*ultiple backends (LVM, iSCSI..)

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

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

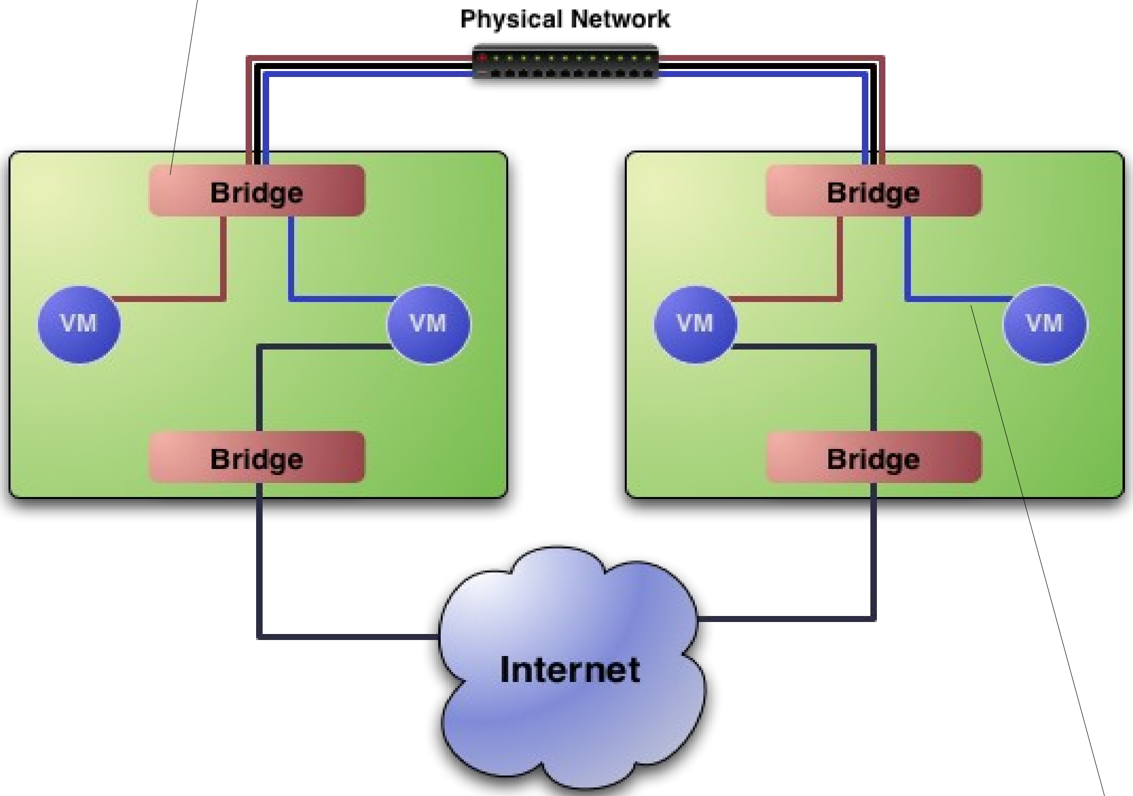
The Storage Subsystem

- Multiple storage backends out of the box: NFS, SSH, LVM
- Easily extended through plugins: parallel-scp, bittorrent, image proxys



The Network Subsystem

- OpenNebula management operations uses a ssh connections
- OpenNebula uses bridge networking
- NAT, firewalling and other services are configured with hooks



- Networks are isolated at layer 2 (IEEE 802.1Q, ebtables)
- You can put any TCP/IP service as part of the VMs (e.g. DHCP, nagios...)

EL / LAK (FOSS) 2010

May 14th, 2010

PART II: Using your Private Cloud

Javier Fontán
University Complutense of Madrid

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.

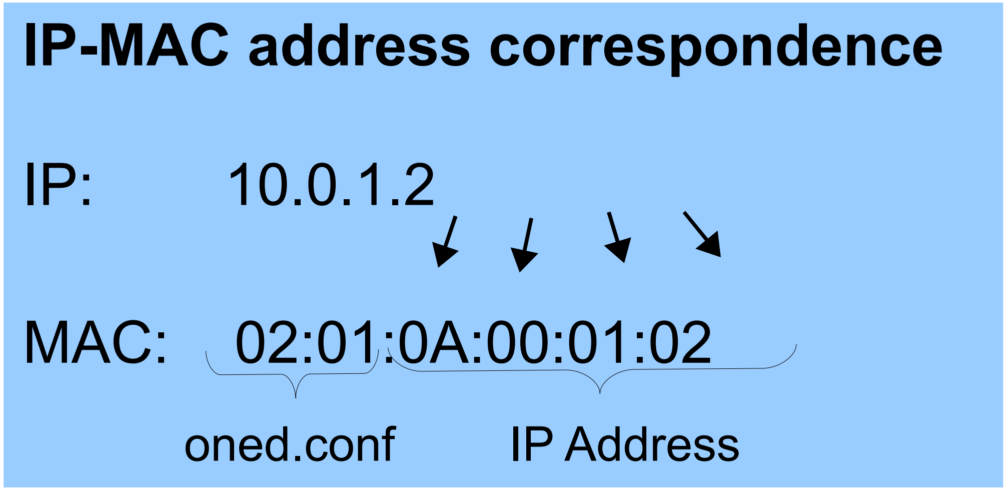
Using the Private Cloud: Virtual Networks

- 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="Blue LAN"]
NIC=[NETWORK="Red LAN"]

#Ask for a specific IP/MAC of the Red vlan
NIC=[NETWORK="Red LAN", IP=192.168.0.3]
```

  - 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

oned.conf

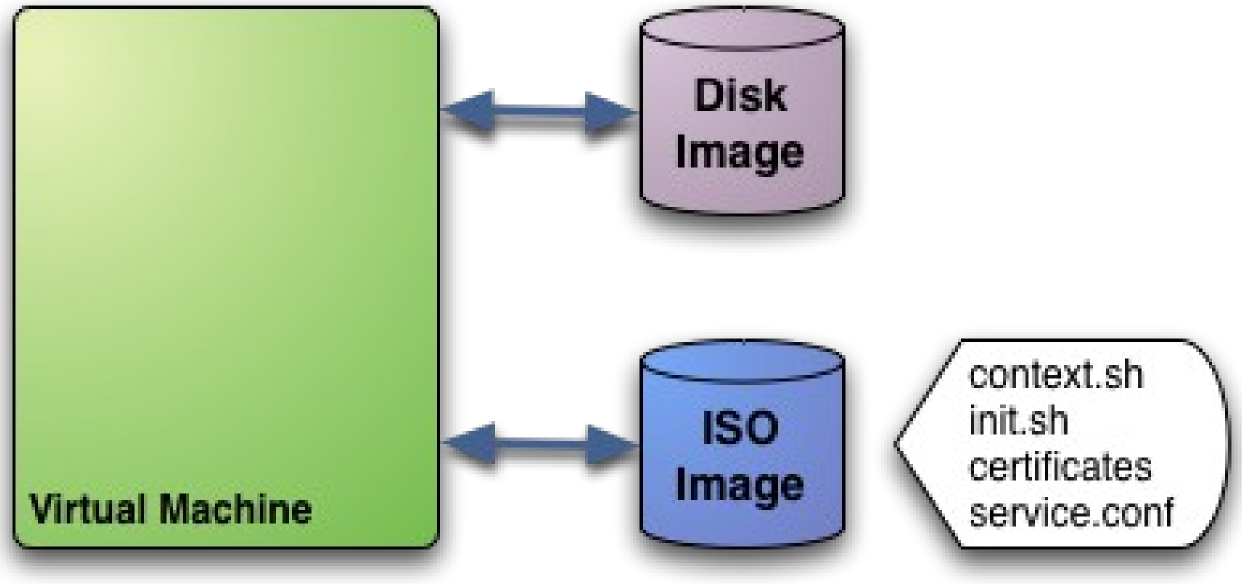
IP Address

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 "*transfered*" 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

- Context contains data to be passed to the VM at boot time



Boot process of the VM:

- mount iso
- Source context.sh
- In this example it will execute init.sh so you can try anything

Using the Private Cloud: Virtual Machines

- Tunning the placement of VMs with the Match-making scheduler
  - First those hosts that do not meet the VM requirements are filtered out (`REQUIREMENTS`)
  - `RANK` is evaluated for the remaining hosts
  - That with the highest `RANK` is used for the VM
- Placement policies are specified per VM

```
#---------------------------------------
#           Scheduler
#---------------------------------------
# Use Host Monitor attributes
REQUIREMENTS = "Bool_expression_for_reqs"
RANK         = "Arith_expression_to_rank_hosts"
```

- Hands on... try a simple VM pinning

```
REQUIREMENTS = "HOSTNAME=\”...\”"
```

- Hands on... try a simple load-aware policy

```
RANK = FREECPU
```

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`

PART III: Using External Cloud Providers (Hybrid Cloud)

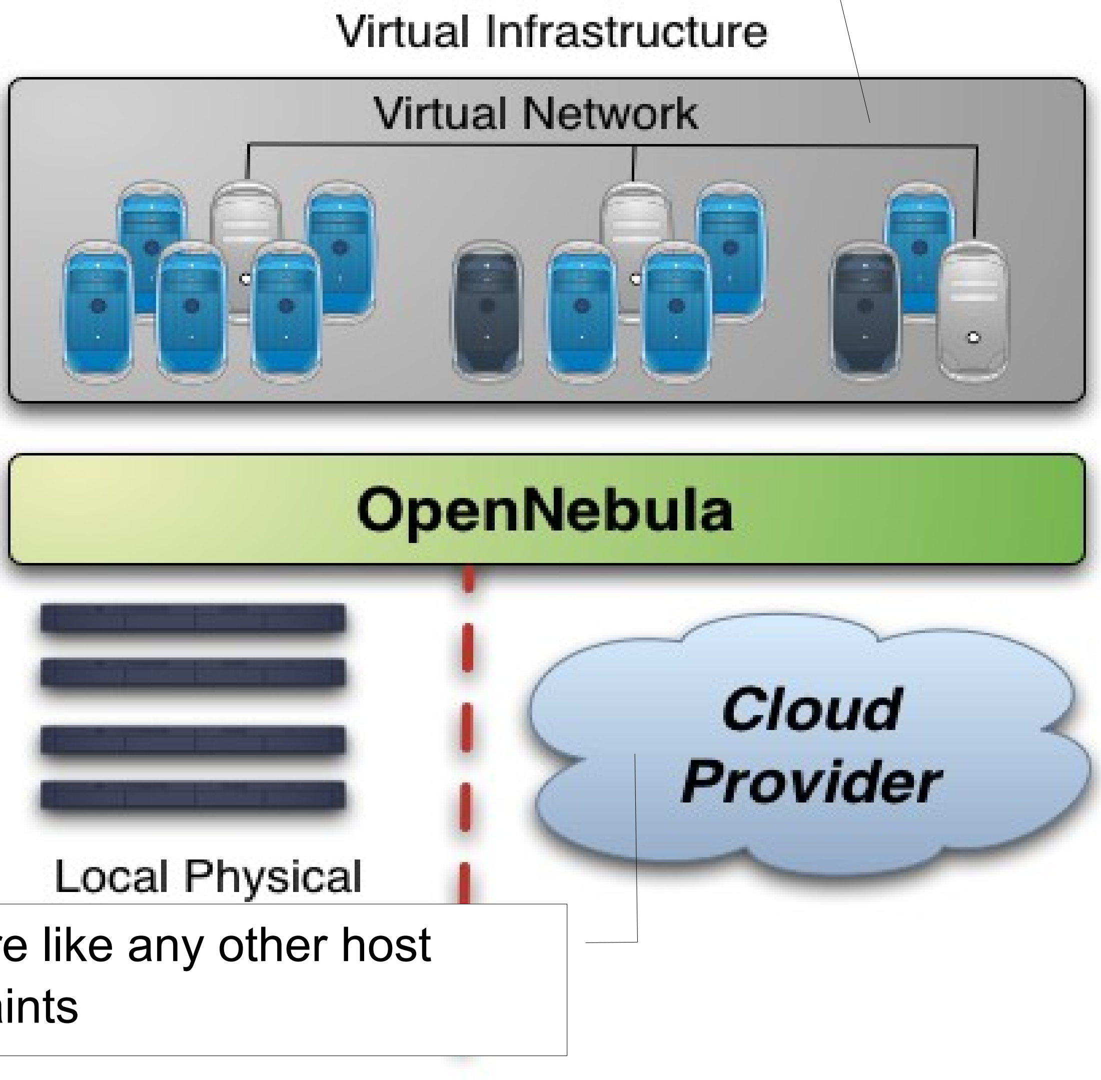
EL / LAK (FOSS) 2010

May 14th, 2010

Javier Fontán
University Complutense of 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

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

```
$ 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"
]
```

EL / LAK (FOSS) 2010

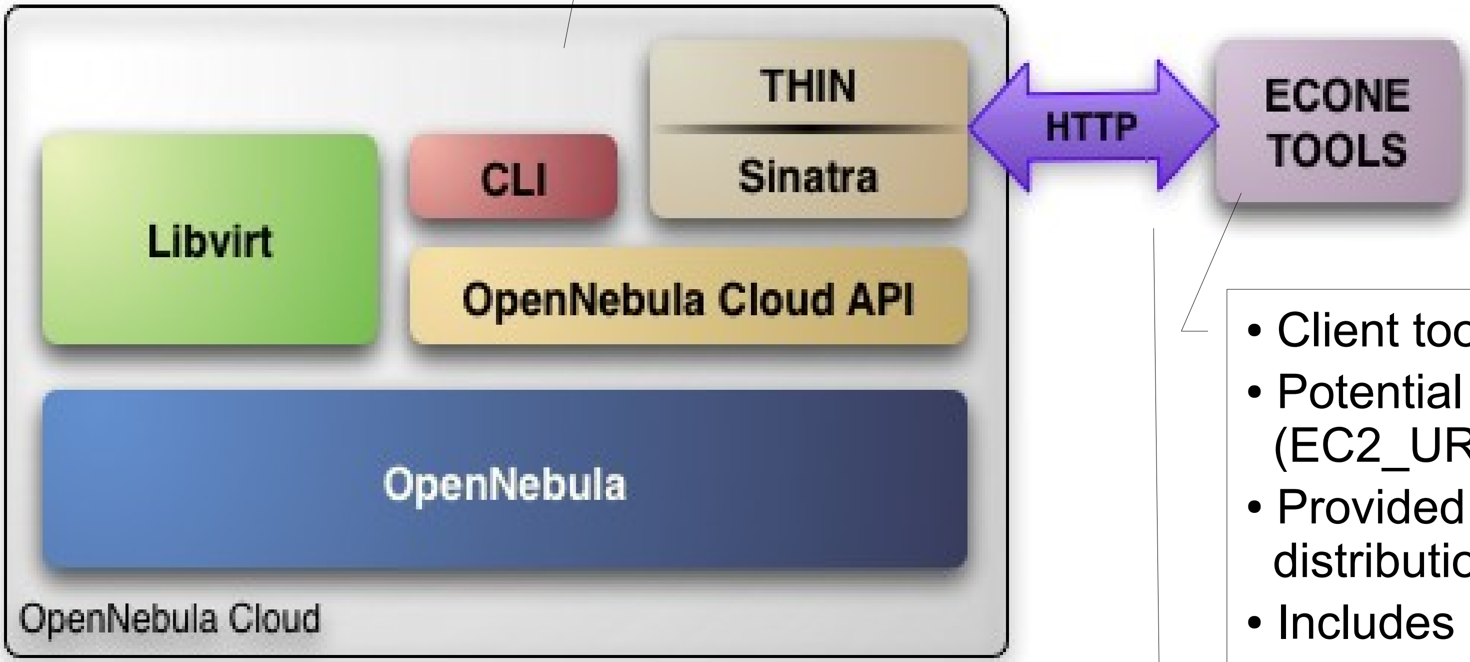
May 14th, 2010

PART IV: Share your Cloud! (Cloud Interfaces)

Javier Fontán
University Complutense of 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

- 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.

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

EL / LAK (FOSS) 2010

May 14th, 2010

PART V: Customizing your Cloud

Javier Fontán
University Complutense of Madrid

Customizing and Extending your Cloud

- You can customize your cloud by:
  - Tunning or adapting the transfer operations to your **storage back-end**
  - Adding new **monitorization** probes to improve the VM placement
  - Adjusting VM operations to your hypervisor installation
  - Trigger **custom actions** on specific VM events (e.g. “on VM creation update the accounting DB” or “on VM shutdown send an email”)
- You can extend your cloud by:
  - Developing new drivers for other hypervisors
  - Developing new drivers for other storage back-ends
  - Developing Cloud applications using the OpenNebula API or the Cloud APIs

❗ OpenNebula is very scripting friendly, drivers can be written in any language. You can modify the current ones or use them as templates for new ones.