Open Nebula Tutorial

Giacomo Righetti, Massimo Coppola
University of Pisa & ISTI - CNR
Presentation adapted for the SPD 2010-11 course, 17/05/2011

Outline

- What is Open Nebula?
- Some features
- The OpenNebula Ecosystem
- How to...?
- Top level interfaces
- Architecture
- Some hints on how to modify the system

Section I

» Introduction

What is Open Nebula?

- Fully open-source toolkit to build any type of IaaS cloud
  - **private**, **public** and **hybrid**
- Apache License, v.2.0
- Full description of features
  - http://www.opennebula.org/documentation:features
  - http://opennebula.org/_media/documentation:opennebula_2.0_features_rev20101026.pdf
- C++
- Ruby & Java bindings

Why Open Nebula?

Capabilities for Cloud Management
open-source toolkit to administer the complexity of large-scale distributed infrastructures

Capabilities for Integration
Open, flexible and extensible architecture, interfaces and components to fit existing data centers

Capabilities for Production Environments
Scalability and performance tested on very large-scale infrastructures consisting of thousands of cores, with the security and fault tolerance levels required in production

Leverage the Cloud Software Ecosystems
A SW ecosystem is being built around OpenNebula and the most common cloud interfaces, Amazon AWS, OGC OCCI and VMware vCloud

Fully Open Source Cloud Software
OpenNebula is NOT a feature or performance limited edition of an Enterprise version. OpenNebula is truly open, and not open core

Features (1)

- Private cloud
  - **Xen**, **KVM** and **VMware**
- Hybrid cloud (**cloudbursting**)
  - **Amazon EC2**
  - other providers through **Deltacloud**
- Public cloud
  - **EC2 Query**, **OGF OCCI** and **vCloud** APIs

Comparison with similar technologies

	Platform ISF	VMware Vsphere	Eucalyptus	Nimbus	OpenNebula
Virtualization Management	VMware, Xen	VMware	Xen, KVM, WMware	Xen	Xen, KVM, VMware
Virtual Network Management	Yes	Yes	No	Yes	Yes
Image Management	Yes	Yes	Yes	Yes	Yes
Service Contextualization	No	No	No	Yes	Yes
Scheduling	Yes	Yes	No	No	Yes
Administration Interface	Yes	Yes	No	No	Yes
Hybrid Cloud Computing	No	No	No	No	Yes
Cloud Interfaces	No	vCloud	EC2	WSRF, EC2	EC2 Query, OGF OCCI
Flexibility and Extensibility	Yes	No	Yes	Yes	Yes
Open Source	No	No	GPL	Apache	Apache

7

Features (2)

- Cloudbursting
  - Local infrastructure can be supplemented with computing capacity from an external Cloud to meet peak demands
- Federation
  - Federate different cloud instances to build a hierarchy of independent virtualization clusters; enabling higher levels of scalability

Federation

"This is a different version than the federation concept of Contrail"

8

The Contrail's Federation Concept

- ... let's define it!
- Proposal:
  - Community cloud plus public clouds
- Community Clouds are usually defined as:
  - "A community cloud is controlled and used by a group of organizations that have shared interests, such as specific security requirements or a common mission. The members of the community share access to the data and applications in the cloud"
    - National Institute of Standards and Technology (NIST)
  - Lacks of economical model! Contrail adds it explicitly, among other things

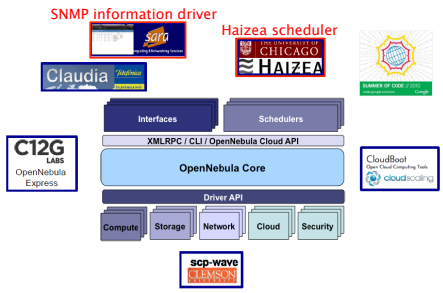
OpenNebula as an Open-Source Software

- Distributed under Apache Open Source License
  - not just "Open Core" as Eucalyptus
- Core part
  - described later
- Ecosystem part
  - The OpenNebula Ecosystem Catalog
    - http://www.opennebula.org/software:ecosystem
      - Management console (web interface that maybe can be exploited)
      - Deltacloud Adaptor for Hybrid Cloud Computing
      - Python bindings
      - ...

The Open Nebula Ecosystem

- **Tools** providing added value on top of OpenNebula clouds by leveraging its APIs
  - service or VM image managers to user-friendly dashboards
- **Extensions** adding new functionality to OpenNebula cloud instances
  - support for enforcement of energy policies, advance reservation of capacity to new cloud interfaces
- **Plugins** enlarging the networking, storage and virtualization technologies and services that OpenNebula can interface with
  - support for new hypervisors, networking devices or storage organizations to access to cloud commercial offerings for hybrid cloud computing

Examples of Components in the Ecosystem



What is all about? Open Nebula Entities

User | Image | Host | Virtual Machine | Virtual Network

Can be managed from different layers

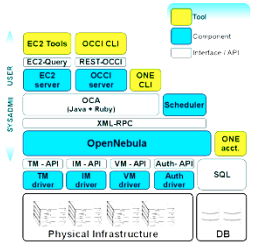
we still need to identify precisely how and to what extent the different layers interface with them

Section II

» Top level interfaces

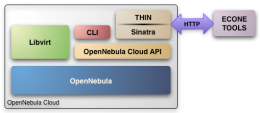
How to interact with OpenNebula

- EC2 Query API
- OCCI OGF
- ONE CLI
- XML RPC
- OCA "Open Nebula Cloud API"
- Low Level Drivers



EC2 Query API

- EC2 is the de-facto standard Cloud API from Amazon
- HTTP/HTTPS
  - GET/POST + ActionOperation
- implements a subset of the EC2 Query interface
  - enabling the creation of public clouds managed by OpenNebula

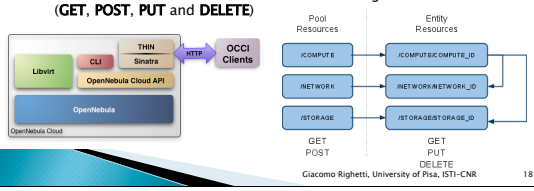


EC2 Query API

- The subset implemented is:
  - **upload image**
    - uploads an image to the repository manager
  - **register image**
    - registers an image (previously uploaded in the repository manager) in order to be launched
  - **describe images**
    - lists all registered images belonging to one particular user
  - **run instances**
    - runs an instance of a particular image
  - **describe instances**
    - outputs a list of launched images belonging to one particular user
  - **terminate instances**
    - shutdown a virtual machine (or cancel, depending on its state)

Open Nebula OCCI Design

- OCCI is the standard Cloud API being developed by the OGF consoltium
- OpenNebula OCCI RESTful webservice
  - Launches and manages images, virtual networks and virtual machines
  - Uses the latest draft of the OGF OCCI API specification
  - Interactions with the resources are done through HTTP verbs (**GET**, **POST**, **PUT** and **DELETE**)



Open Nebula OCCI Design – Pool Resources

- The "COMPUTE" pool
  - HTTP methods: GET, POST

```
<COMPUTES>
  <COMPUTE href="http://www.occi.org/compute/234">
  <COMPUTE href="http://www.occi.org/compute/432">
  <COMPUTE href="http://www.occi.org/compute/123">
</COMPUTES>
```

- The "STORAGE" and "NETWORK" pool
  - HTTP methods: GET, POST
    - Similar structure

Open Nebula OCCI Design – Entity Resources

- The "STORAGE" object
  - HTTP methods: GET, DELETE

```
<STORAGE>
  <ID>123</ID>
  <NAME>Ubuntu 9.04 LAMP</NAME>
  <TYPE>OS</TYPE>
        <SIZE>2048</SIZE>
  <URL>file:///images/ubuntu/jaunty.img</URL>
</STORAGE>
```

- The "NETWORK" object
  - HTTP methods: GET, DELETE

```
<NETWORK>
  <ID>123</ID>
  <NAME>Blue Network</NAME>
  <ADDRESS>192.168.0.1</ADDRESS>
  <SIZE>200</SIZE>
</NETWORK>
```


Open Nebula OCCI Design – Entity Resources

- The "COMPUTE" object
  - HTTP methods: GET, PUT, DELETE

```
<COMPUTE>
  <ID>123AF</ID>
  <NAME>Web Server</NAME>
  <TYPE>small</TYPE>
  <STATE>running</STATE>
   <DISKS>
     <DISK image=http://www.occi.org/storage/234 dev=sda1/>
     <SWAP size=1024 dev=sda2/>
     <FS size=1024 format=ext3 dev=sda3/>
   </DISKS>
   <NICS>
     <NIC network=http://www.occi.org/network/123 ip="19.12.1.1"/>
     <NIC network=0/>
   </NICS>
</COMPUTE>
```


Open Nebula OCCI Design – implementation choices

- OCCI specification (1.1 in public comments phase)
  - http://occi-wg.org/2011/01/31/occi-1-1-document-series-in-public-comment/
- Assumptions:
  - Representation Format
    - XML
    - Resource attributes set by OpenNebula needs
  - Specification not clear about *linking resources*
    - XML nesting
  - Specification of local devices
    - OpenNebula uses unix devices with "dev" attributes
      - e.g. : <DISK image="ab5c9770-7ade-012c-f1d5-00254bd6f386" dev="sda1"/>
  - Management verbs not well defined (for stop, resume, etc)
    - Update representation through PUT chosen
      - More RESTful
      - Sometimes can be misleading
  - Storage POST not well defined
    - Upload image through HTTP multipart

Open Nebula OCCI Design – CLI

- Managing compute resources
  - occi-compute {create, list, show, update, delete}
- Managing network resources
  - occi-network {create, list, show, delete}
- Managing storage resources
  - occi-storage {create, list, show, delete}

Command Line Interface 2.0

OpenNebula provides six commands to interact with the system:

- onevm
  - submit, control and monitor virtual machines
- oneauth
  - tools to manage authentication and authorization
- onehost
- onevnet
- oneuser
- oneimage
- onecluster
  - cluster = one or more host

Command Line Interface 2.0 – oneauth

- Seems a bit limited:
  - quota set <id> <cpu> <memory>
    - sets quota for a user
  - login <username> [<expire time in seconds>]
  - create
    - creates a user (with an optional parameter to set public key)
  - key
    - gets public key

XML-RPC

- Low level API,
  - used to craft specialized libraries
  - VM
  - Networks
  - Images
  - Users
  - Hosts
  - Clusters
- Refer here for further informations:
  - http://www.opennebula.org/documentation:rel2.2:api

Open Nebula Cloud API

- Provides wrapper methods for XML-RPC plus helper methods
- Bindings for:
  - Java
  - Ruby
    - Object oriented
    - Interpreted
    - Dynamic typing

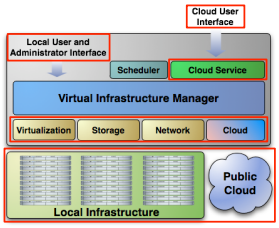
Some comments

- Not well documented
  - OCCI vs Open Nebula Cloud API
    - How to choose between the two?
    - Overlapping functionalities?

Low Level Drivers

- Perform the actual interaction with the cloud infrastructure
- Storage
- Virtualization
- Monitoring
- Authorization
- Very poor documented!
  - «*Use the driver interfaces if…* you need OpenNebula to interface any specific storage, virtualization, monitoring or authorization system already deployed in your datacenter or to tune the behavior of the standard OpenNebula drivers»

How Does It Work?



OpenNebula orchestrates **virtualization**, **storage**, **network**, **monitoring**, and **security** technologies

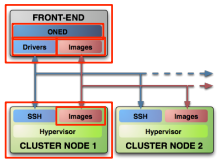
enabling the dynamic placement of multi-tier services on distributed infrastructures

combining both data center resources and remote cloud resources

according to allocation policies

Deployment

- OpenNebula assumes that your physical infrastructure adopts a classical cluster-like architecture with a front-end, and a set of cluster nodes where Virtual Machines will be executed.
- There is at least one physical network joining all the cluster nodes with the front-end



Front-end
executes the OpenNebula and cluster services

Nodes
hypervisor-enabled hosts that provide the resources needed by the Virtual Machines

Image repository
any storage medium that holds the base images of the VMs

OpenNebula daemon
is the core service of the system. It manages the life-cycle of the VMs and orchestrates the cluster subsystems (network, storage and hypervisors)

Drivers
programs used by the core to interface with an specific cluster subsystem, e.g. a given hypervisor or storage file system

Users

- A User in OpenNebula is an entity that owns resources
- From OpenNebula 2.0 onwards - Auth module
  - Pluggable architecture
  - Drivers for ssh and ldap
- Two account types in the OpenNebula system:
  - **oneadmin**
    - created **the first time** OpenNebula is started using the ONE_AUTH data.
    - has enough privileges to perform any operation on any object (virtual machine, network, host or user)
  - **regular user**
    - accounts must be created by *oneadmin* and they **can only manage their own objects** (virtual machines and networks)

Hosts

- The physical nodes have to be added to the system as OpenNebula hosts
- Physical host monitoring is performed by *probes*
  - scripts designed to extract pieces of information from the host operating system
- The following informations need to be gathered for each node:
  - *Hostname* of the cluster node or IP
  - *Information Driver* to be used to monitor the host (e.g. im_kvm)
  - *Storage Driver* to clone, delete, move or copy images into the host (e.g. tm_nfs)
  - *Virtualization Driver* to boot, stop, resume or migrate VMs in the host (e.g. vmm_kvm)
- By default, all hosts belong to the "default" cluster
  - the administrator can logically group hosts by any attribute like the physical location (e.g. CLUSTER = production), or a given feature (e.g. CLUSTER = intel)
- For a complete reference see
  - http://opennebula.org/documentation:rel2.0:cluster_guide

Images

- An Image in OpenNebula is a VM image contained in a Repository
- Has metadata on how to use the VM image
- Image definition
  - **Name**, of the image
  - **Source**, of the file
- Can be **persistent or public**
- Images are managed with the *oneimage* command

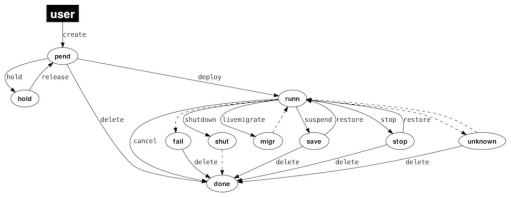
Virtual Networks

- 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* command

Virtual Machines

- A Virtual Machine in OpenNebula
  - **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
- Virtual Machines are defined in a **VM template** (text file)
- All the files (logs, images, state files...) are stored in $ONE_LOCATION/var/<VM_ID>
- Virtual Machines are managed with the *onevm* command

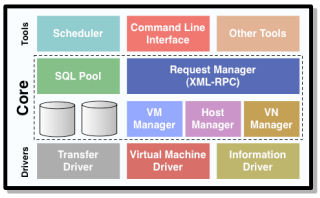
Virtual Machine States



Section III

» Architecture

Architecture



1. *Tools*
   Management tools developed using the interfaces provided by the OpenNebula Core
2. *Core*
   The main virtual machine, storage, virtual network and host management components
3. *Drivers*
   To plug-in different virtualization, storage and monitoring technologies and Cloud services into the core

Tools – Scheduler (1)

- The Scheduler module is in charge of the assignment between pending Virtual Machines and cluster nodes
  - It is a separate process that can be started independently of `oned`
- OpenNebula comes with a match making scheduler (`mm_sched`) that implements the *Rank Scheduling Policy*
  - prioritize those resources more suitable for the VM. You can configure several resource and load aware policies by simply specifying specific RANK expressions in the Virtual Machine definition files
- Users can require their virtual machines to be deployed in a host that meets certain *constraints*
  - defined using any attribute reported by `onehost` like the architecture (ARCH), or the cluster the host is assigned to
  - http://opennebula.org/documentation:rel2.0:template#placement_section

Tools – Scheduler (2)

- You can use OpenNebula without the scheduling process to operate it in a VM management mode
  - Start or migration of VMs in this case is explicitly performed using the `onevm` command
- The Haizea lease manager can also be used as a scheduling module in OpenNebula
  - support advance reservation of resources and queuing of best effort requests

The OpenNebula Core (1)

- The core consists of:
  - a set of components to control and monitor
    - virtual machines
    - virtual networks
    - storage and hosts
- The core performs its actions (e.g. monitor a host, or cancel a VM) by invoking a suitable driver

The OpenNebula Core (2)

- The main functional components of OpenNebula core are:
  - **RequestManager**
    - Handle client requests
  - **Virtual Machine Manager**
    - to manage and monitor of VMs
  - **Transfer Manager**
    - to manage VM images
  - **Virtual Network Manager**
    - to manage virtual networks
  - **Host Manager**
    - to manage and monitor physical resources
  - **Database**
    - persistent storage for ONE data structures

RequestManager

- Exposes a XML-RPC interface
  - depending on the invoked method a given component is called internally
- The XML-RPC decouples most of the functionality in the OpenNebula core, from external components
  - i.e. the Scheduler

Virtual Machine Manager

- Manages and monitors the VMs
- The operations of the VM Manager are abstracted from the underlying hypervisor by the use of pluggable drivers

Transfer Manager

- The Transfer Manager is in charge of all the files transfers needed for the correct deployment of virtual machines
  - From image repository to cluster node
    - and viceversa
  - Between cluster nodes for cold migrations
    - Checkpoint files
  - To the cluster frontend when the virtual machine is stopped

Virtual Network Manager

- The Virtual Network Manager is responsible for the handling of IP and MAC addresses
  - It allows the creation of virtual networks
    - keeping track of leases (a set formed by one IP and one MAC valid on a particular network) and their association with virtual machines and the physical bridges the VM are using

Host Manager

- Manages and monitors the physical hosts
- Monitor and management actions are performed also through a suitable driver
- The host monitoring infrastructure is flexible and can be extended to include any host attribute

InformationManager

- It is in charge of monitoring the cluster nodes
- It comes with various sensors, each one responsible of a different aspects of the computer to be monitored
  - CPU, memory, hostname, …
- There are sensors prepared to gather information from different hypervisors
  - KVM, XEN
- But the model used is not publish/subscribe!
  - The core module explicitly polls the InformationDriver (probes)
  - There is the need to investigate how to integrate other monitoring system
    - E.g. Ganglia

Other managers

- ActionManager
  - checks if an action can be performed on the system itself
- AuthManager
- HookManager
  - enables the triggering of custom scripts tied to a change in state in a particular resource
- MadManager
  - general functionalities for driver management
- LifeCycleManager
- DispatchManager
- Overlapping functionalities
  - LifeCycleManager vs VirtualMachineManager
    - Both control VMs

Database

- A persistent generic pool based on a SQLite3 backend is the core component of the OpenNebula internal data structures
  - scalability and reliability (in case of failure the state of OpenNebula is automatically recovered) needed in the management VMs
- This information can be accessed through the SQLite3 interface to develop custom accounting applications

Accounting

- Physical resources
  - User / VM
  - Aggregated information
  - Considering migration
  - Running time, Transfer time, Host type
- Virtual resources
  - User / VM
  - Does not consider migration
- Virtualized services
  - service specific tools and metrics
    - e.g. number of connections to an apache web server

Drivers

- Separate processes that communicate with the OpenNebula core using an internal ASCII protocol
  - The actual protocol is found here:
    - http://www.opennebula.org/documentation:archives:rel1.0:architecture
- Before loading a driver, two *run commands* (RC) files are sourced to optionally obtain environmental variables

Some considerations

- How easy is to add a Manager?
- How easy is to support a new hypervisor?
- How the authentication/authorization module can be unplugged to use an external driver that takes care of these duties?
- How to support another storage system?
  - Different DB, different storage strategies
- Source code needs to be studied

How to change the Authentication/Authorization module (1)

- Configuration file for auth module is located at $ONE_LOCATION/etc/auth/auth.conf
- Default configuration:

```
:database: sqlite://auth.db
:authentication: simple
:quota:
  :enabled: false
  :defaults:
    :cpu: 10.0
    :memory: 1048576
```

- To load auth module and enable its use uncomment in the OpenNebula configuration file ($ONE_LOCATION/etc/oned.conf)
  - AUTH_MAD = [executable = "one_auth_mad"]
- But... how to provide a new Authorization module?
  - http://opennebula.org/documentation:rel2.0:auth

How to change the Authentication/Authorization module (2)

- Authorization systems are classes with a method called `auth` defined
  - called each time OpenNebula needs to authenticate a user
- This authorization classes reside in $ONE_LOCATION/lib/ruby
  - simple_auth.rb
  - ssh_auth.rb
- To create a new authorization system you will need to create a class similar to those
  - Its name must conform <name>Auth
  - name should be capitalized so then it will be easily selected in the configuration file
- C++ code at certain point searches for a driver and calls ruby scripts
  - E.g. authentication

Tweaking Permissions

- For an authorization message to be successful the user needs to have permissions to perform all the actions
- Permission policies are defined in $ONE_LOCATION/lib/ruby/simple_permissions.rb
- The method `auth_object` is called every time an authorization is needed:
  - def auth_object(uid, object, id, action, owner, pub)
- For more information see here
  - http://opennebula.org/documentation:rel2.0:auth

Section IV

» How to ... ?

Build a private cloud ? (1)

- Provide local users with a flexible and agile private infrastructure to run virtualized service workloads within the administrative domain
- To create a private cloud it is necessary to:
  - Add physical **Hosts**
  - Create **Users**
  - Define VM **Images**
  - Define **Virtual Networks**
  - Create **Virtual Machine Templates**
  - Launch **Virtual Machines**

Build a private cloud ? (2)

- `onehost list`
  - check the hosts in the physical cluster
- `oneimage register imageName`
  - it is necessary to build an image template to register the image file
- `oneimage list`
- `onevm create myfirstVM.template`
  - It is necessary to define a virtual machine template
- `onevm livemigrate 0 1`
  - Performs a migration moving the VM with VID=0 to *host02* (HID=1)

Image template

- Example of an OS image:
  - NAME = "Ubuntu Desktop"
    PATH = /home/cloud/images/ubuntu-desktop/disk.0
    PUBLIC = YES
    DESCRIPTION = "Ubuntu 10.04 desktop for students."
- For a complete reference see
  - http://opennebula.org/documentation:rel2.0:img_template

Virtual machine template

- Example of a VM image:
  - CPU = 1
    MEMORY = 2056
    DISK = [image = "Ubuntu Desktop"]
    DISK = [type = swap, size = 1024]
    NIC = [NETWORK = "Public network"]
- For a complete reference see
  - http://opennebula.org/documentation:rel2.0:template

Build a hybrid cloud ? (1)

- A Hybrid Cloud is an **extension of a Private Cloud to combine local resources with resources from remote Cloud providers**
- The remote provider could be a commercial Cloud service (Amazon EC2, ElasticHosts), or a partner infrastructure running a different OpenNebula instance. Such support for cloudbursting enables highly scalable hosting environments
- The federation is not performed at service or application level but at infrastructure level

Build a hybrid cloud ? (2)

- Amazon EC2 cloud appears as a new host
  - EC2 = [AMI="ami-acc723c5", AUTHORIZED_PORTS="22"]
- launch the Virtual Machine within Amazon EC2:
  - `onevm submit myEC2Machine.one`
- No modification in the operation of OpenNebula to integrate Cloud services. A Cloud service is managed as any other OpenNebula host that may provide "infinite" capacity for the execution of VMs

Build a public cloud ?

- A Public Cloud is an **extension of a Private Cloud to expose RESTful Cloud interfaces**
- **No modification in the operation of OpenNebula to expose Cloud interfaces**
  - users can interface the infrastructure using any Private or Public Cloud interface

Questions?

Thank you for your attention!

References

- Cloud computing Use Cases Whitepaper 4.0
- Constantino Vàzquez Blanco, "Using the OGF OCCI Interface on OpenNebula/RESERVOIR"
- Borja Sotomayor, "The OpenNebula Cloud Toolkit: Experiences and Outlook"
- Constantino Vàzquez Blanco, Borja Sotomayor, "OpenNebula Tutorial"
- Peter Sempolinski, Douglas Thain, A comparison and critique of Eucalyptus, OpenNebula and Nimbus
- http://opennebula.org/documentation:documentation

Backup slides

Setup – Open Nebula Express

- OpenNebula Express is an installer contributed by C12G that eases the installation and deployment of OpenNebula clouds
  - a fully operational cloud from a cluster with a clean install of the operating system
- Supported Linux distributions on the worker nodes:
  - Ubuntu 10.04, CentOS 5.5
- For the complete installation & configuration guide refer to:
  - http://opennebula.org/software:addons:express
  - http://opennebula.org/documentation:rel2.0:cg

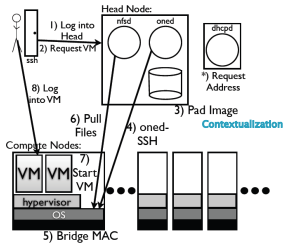
General trends in Eucalyptus, Open Nebula, Nimbus

	Eucalyptus	OpenNebula	Nimbus
Philosophy	Mimic Amazon EC2	Private, highly customizable cloud	Cloud resources tailored to scientific researchers
Customizability	Some for admin, less for user	Basically everything	Many parts except for image storage and globus credentials
DHCP	On cluster controller	Variable	On individual compute node
Internal Security	Tight. Root required for many things.	Looser, but can be made more tight if needed.	Fairly tight, unless deploying a fully private cloud.
User Security	Users are given custom credentials via a web interface	User logs into head (unless optional front-end used)	Users x509 credential is registered with cloud
An Ideal Setting	Large group of machines for bunch of semi-trusted users	Smaller group of machines for highly trusted users	Deploy for less to semi-trusted users familiar with x509
Network Issues	dhcpd on cluster controller	Admin must set manually but has many options	dhcpd on every node and Nimbus assigns MAC

Open Nebula

- Greater level of centralization and customizability
  - Great control over *entities* parameters
  - "Centralization" leads to a system that is easy to administer
- Pure private cloud
  - Log into head node to access functions
  - Shared File System, NFS
    - The compute nodes do not need a large amount of hard disk resources
    - can be a bottleneck for resources
    - No encryption, packet sniffing problems
  - SCP
- ON is best suited to small to medium sized set of reasonably trusted users, and for users who know what they are doing from a technical perspective

Constructing a VM



*The steps for constructing a VM in a configuration of OpenNebula: 1) A user uses ssh to login to the head node. 2) The user uses the onevm command to request a VM. 3) The VM template disk image is copied and a copy is padded to the correct size and configuration within the NFS directory on the head node. 4) The oned process on the head node uses ssh to log into a compute node. 5) The compute node sets up network bridging to provide a virtual NIC with a virtual MAC. 6) Files needed by the VMM on the compute node will be pulled to the compute node via the NFS. 7) VM is spawned on the VMM. 8) The user can now SSH directly into the VM. *) Typically, the dhcp server is handled separately from the OpenNebula configuration.*

Cloud Deployment Models (1)

- **Public Cloud**
  - characterized as being available to clients from a third party service provider via the Internet. The term "public" does not always mean free, even though it can be free or fairly inexpensive to use. A public cloud does not mean that a user's data is publically visible; public cloud vendors typically provide an access control mechanism for their users. Public clouds provide an elastic, cost effective means to deploy solutions.
- **Private Cloud**
  - offers many of the benefits of a public cloud computing environment, such as being elastic and service based. The difference between a private cloud and a public cloud is that in a private cloud-based service, data and processes are managed within the organization without the restrictions of network bandwidth, security exposures and legal requirements that using public cloud services might entail. In addition, private cloud services offer the provider and the usergreater control of the cloud infrastructure, improving security and resiliency because user access and the networks used are restricted and designated

Cloud Deployment Models (2)

- **Hybrid Cloud**
  - combination of a public and private cloud that interoperates. In this model users typically outsource non-business critical information and processing to the public cloud, while keeping business-critical services and data in their control