# OpenNebula
## Build Your Cloud

**Kyoungho An**
kyoungho.an@gmail.com
http://www.dre.vanderbilt.edu/~kyoungho

**Institute for Software Integrated Systems (ISIS)**
**Department of Electrical Engineering and Computer Science**
**Vanderbilt University**

**November 2, 2012**

# Presentation Roadmap

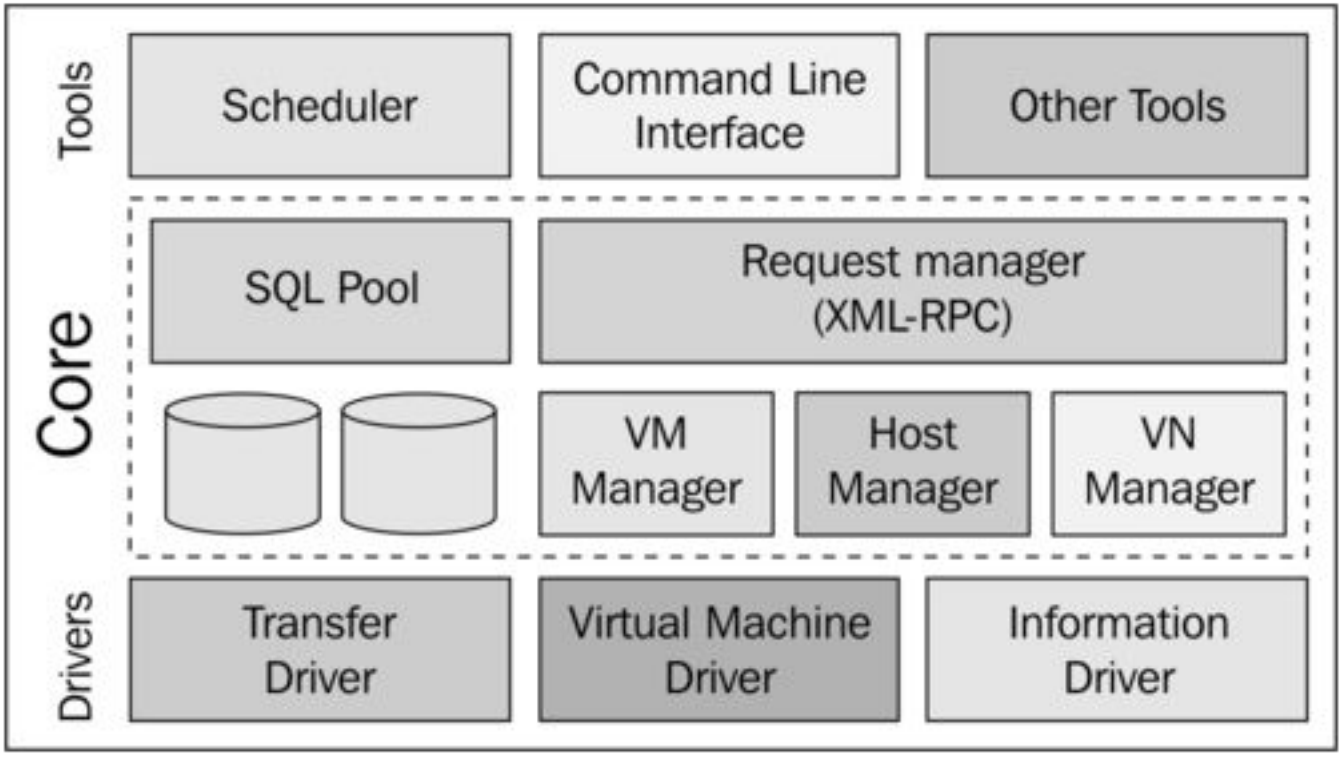
- Overview of OpenNebula
  - The Core Features
  - Supported Hypervisors
- Build Up Your Cloud
  - Required Configurations
  - OpenNebula Installation
  - Frontend Configuration
  - Host Configuration
  - Xen Installation
  - Shared Storage
- Operate Your Cloud
  - Managing Virtual Networks
  - Managing Disk Images
  - Managing Virtual Machines

# OpenNebula?

- Cloud Computing Service Models
  - Software as a Service (SaaS)
  - Platform as a Service (PaaS)
  - Infrastructure as a Service (IaaS)
- OpenNebula
  - Support standard interfaces (OCCI, EC2 APIs)
  - Support various hypervisors (KVM, Xen, VMware)
  - Core of OpenNebula is written in C++

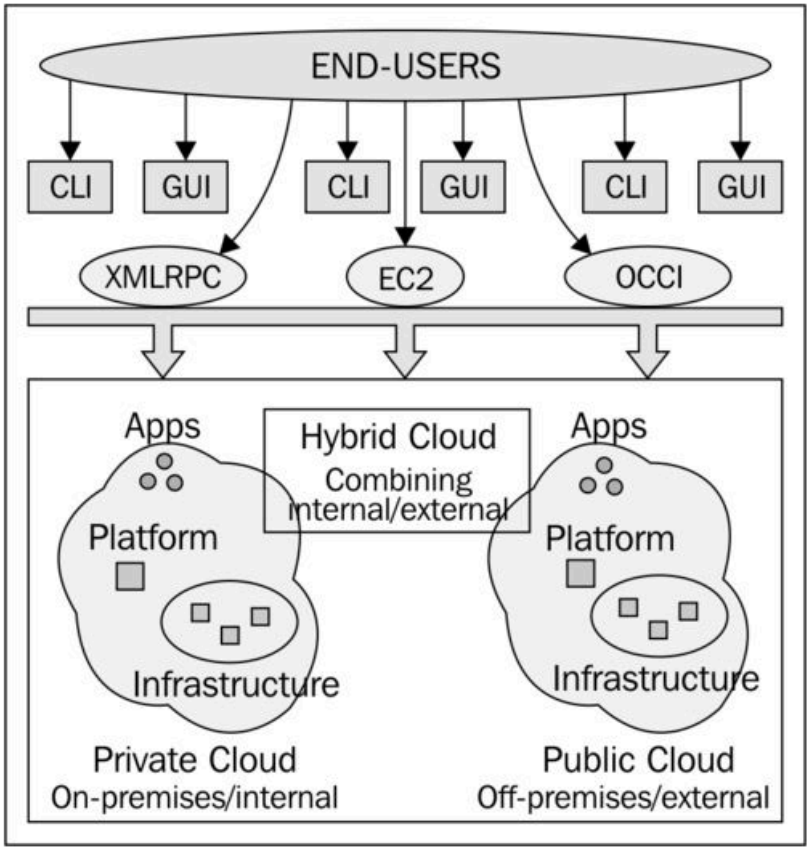
# OpenNebula?

- OpenNebula Components



# The Core Features

- Cloud Deployment Models
  - Private Cloud
  - Public Cloud
  - Hybrid Cloud

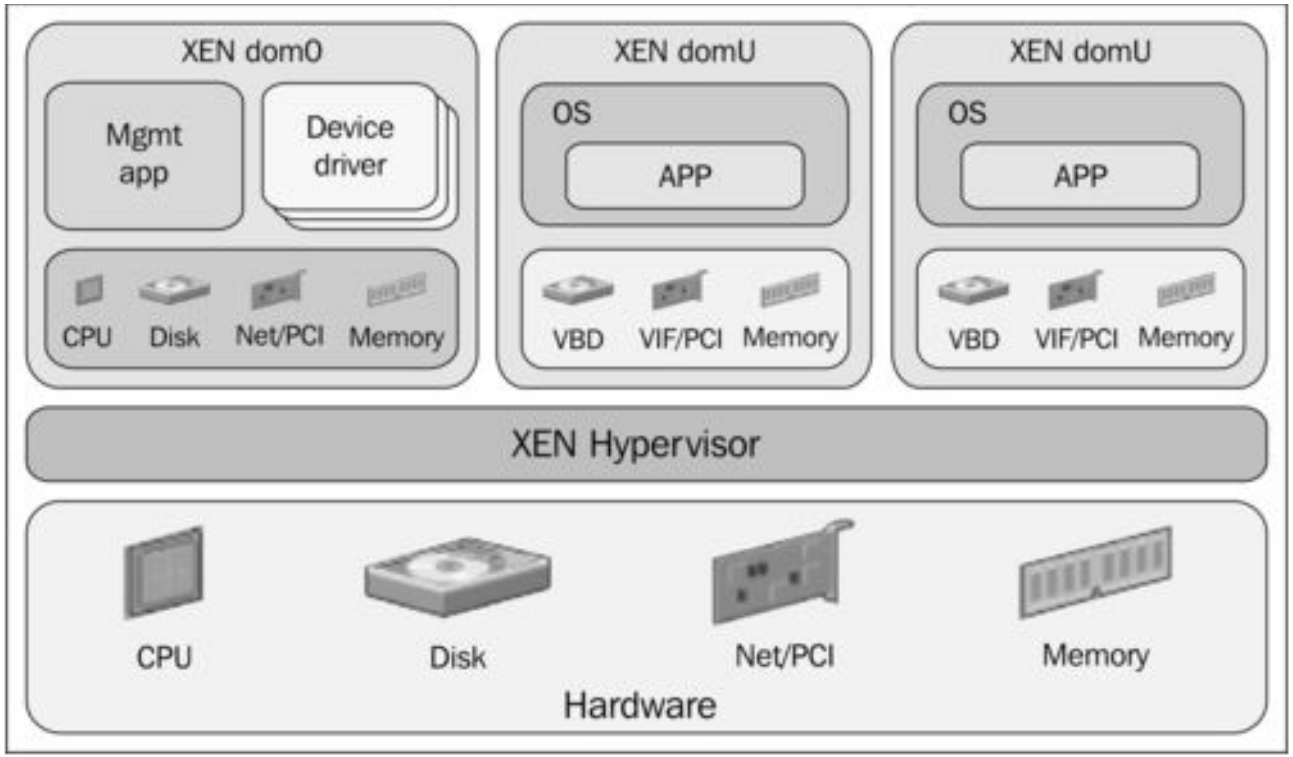


# The Core Features

- User management
- VM Image management
- Virtual Network management
- Service management
- Infrastructure management
- Storage management
- Information management
- Scheduling
- User interface
- Operations center

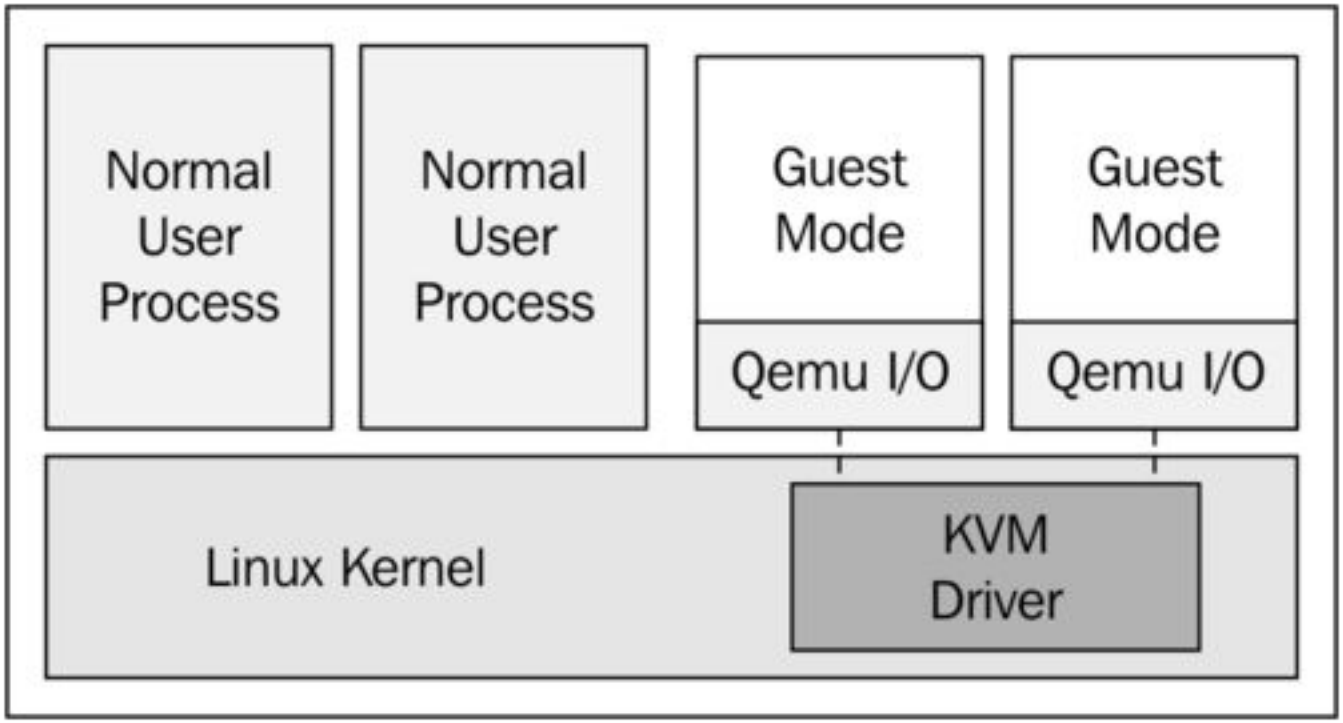
# Supported Hypervsiors - Xen

- Xen components
  - Hypervisor
  - Dom0
  - DomU



# Supported Hypervsiors - KVM

- Kernel-based Virtual Machine (KVM)
- Quick Emulator (QEMU)

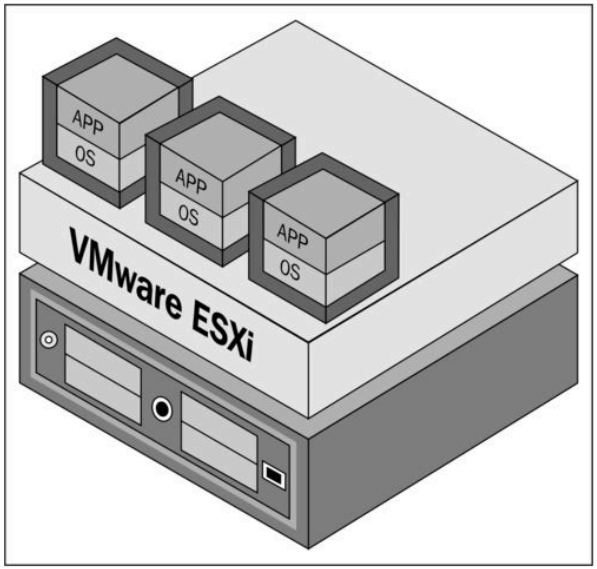


# Supported Hypervsiors - Libvirt

- The abstraction layer in a KVM/Xen Virtualization stack
  - VM management
  - Remote machine support
  - Storage management
  - Network interfaces management

# Supported Hypervsiors - VMware

- VMware ESXi
  - Runs on its own kernel
- VMware ESX
- VMware Server
- Offers better performance and integration with Windows

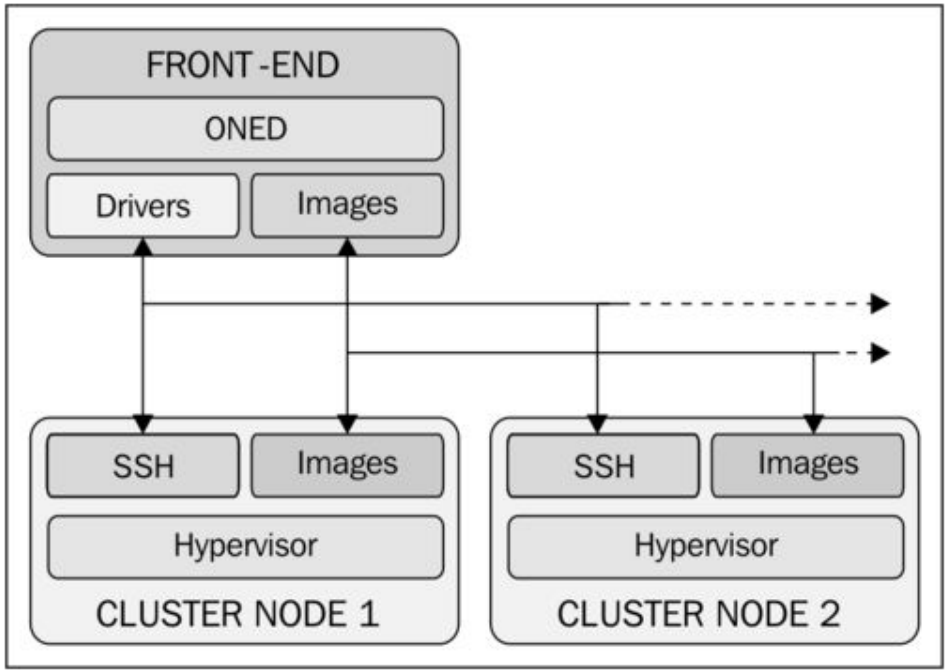


# Presentation Roadmap

- Overview of OpenNebula
  - The Core Features
  - Supported Hypervisors
- Build Up Your Cloud
  - Required Configurations
  - OpenNebula Installation
  - Frontend Configuration
  - Host Configuration
  - Xen Installation
  - Shared Storage
- Operate Your Cloud
  - Managing Virtual Networks
  - Managing Disk Images
  - Managing Virtual Machines

# Required Configurations

- OpenNebula requires a cluster-like setup with a single frontend and a bunch of cluster nodes
- OpenNebula components
  - Frontend
  - Hosts
  - Image Repository
  - Physical network



# Frontend Software Requirements

- The machine that runs OpenNebula is called frontend
- Frontend software components
  - Management daemon (oned)
  - Virtual Machine scheduler (mm_sched)
  - The monitoring and accounting daemon (onecctd)
  - The web interface server (Sunstone)
  - The cloud API servers (EC2-query and/or OCCI)
- The components communicate through XML-RPC
- The components can be installed on different machines for security or performance reasons

# Host Software Requirements

- The hosts are physical machines that will run the VMs
- Hosts are managed directly by the OpenNebula daemons using SSH for communication
- There is no need to install OpenNebula packages on the hosts
- Host software components
  - An SSH server running with public key authentication enabled
  - A hypervisor such as Xen/KVM
  - Ruby 1.8.7

# OpenNebula Installation

- OpenNebula installation through sources
- To download the OpenNebula sources
  - http://downloads.opennebula.org
- For Ubuntu, you need to install the following libraries
  $ sudo apt-get install g++ libxmlrpc-c3-dev scons libsqlite3-dev libmysqlclient-dev libxml2-dev libssl-dev ruby

# OpenNebula Installation

- Build OpenNebula sources
  $ scons [ option=value ]

The available optional build options are:

| Option | Value |
|---|---|
| sqlite_db | Path to SQLite install (if you needed, for whatever reason, to install this library in a non-standard system folder) |
| sqlite | **No**, if you do not want to build SQLite support |
| mysql | **Yes**, if you want to build MySQL support |
| xmlrpc | Path to xmlrpc install (if you needed, for whatever reason, to install this library in a non-standard system folder) |
| parsers | **Yes**, if you want to rebuild flex/bison files |

# OpenNebula Installation

- Install OpenNebula on your Ubuntu system
  $ ./install.sh [ options]

The available `install.sh` options are as follows:

| Option | Value or function |
|---|---|
| **-u** | User that will run OpenNebula; defaults to user executing `install.sh`. |
| **-g** | Group of the user that will run OpenNebula; defaults to user executing `install.sh`. |
| **-k** | Keep configuration files of existing OpenNebula installation; useful when upgrading. This flag should not be set when installing OpenNebula for the first time. |
| **-d** | The target installation directory. If defined, it will specify the path for a self-contained install. If it is not defined, the installation will be performed system-wide. |
| **-c** | Only for installation of client utilities: OpenNebula CLI, OCCI and EC2 client files. |
| **-r** | Remove OpenNebula; only useful if `-d` was not specified, otherwise `rm -rf $ONE_LOCATION` would do the job. |
| **-h** | This prints installer help. |



# OpenNebula Installation

- Basic OpenNebula configuration
  - SSH public-key authentication
    $ ssh-keygen
  - One daemon per oneadmin user
    $ mkdir –p ~/.one
    $ echo “oneadmin:password” > ~/.one/one_auth
    $ chmod 600 ~/.one/one_auth
  - OpenNebula environment variables
    $ echo export ONE_LOCATION=$HOME/one >> ~/.profile
    $ echo –e PATH=$ONE_LOCATION/bin:$PATH >> ~/.profile
  - First start of oned
    $ one start
    $ tail –f /var/log/one/oned.log

# Frontend Configuration

- The configuration file for the one daemon is called oned.conf
- It is placed inside the /etc/one or $ONE_LOCATION/etc

A suggested working configuration example is as follows:

```
HOST_MONITORING_INTERVAL = 20
HOST_PER_INTERVAL = 15
VM_POLLING_INTERVAL = 10
VM_PER_INTERVAL = 5
VM_DIR=/srv/nfs/images
SCRIPTS_REMOTE_DIR=/tmp/one
PORT=2633
DB = [ backend = "sqlite" ]
VNC_BASE_PORT = 5900
DEBUG_LEVEL=3
```

# Frontend Configuration

- Virtual network configuration
  - NETWORK_SIZE
    - A, B, C
    - /8, /16, /24
  - MAC_PREFIX

A working example is as follows:

```
NETWORK_SIZE = 254
MAC_PREFIX = "02:00"
```

# Frontend Configuration

- Image Repository configuration
  - DEFAULT_IMAGE_TYPE
    - OS, CDROM, DATABLOCK
  - DEFAULT_DEVICE_PREFIX
    - hd: IDE interface
    - sd: SCSI interface
    - xvd: Xen disk
    - vd: Virtio KVM disk

A working example is as follows:

```
DEFAULT_IMAGE_TYPE = "OS"
DEFAULT_DEVICE_PREFIX = "hd"
```

# Frontend Configuration

- Information Manager driver configuration
  - The Information Manager drivers are used to gather information from the hosts, and they depend on the hypervisor
  - name: The name of the driver
  - executable: The path to the driver executable script
  - argument: Passed to the executable
    - -r: the number of retried when a monitored host does not repond
    - -t: the number of threads

```
IM_MAD = [
name = "im_kvm",
executable = "one_im_ssh",
arguments = "-r 0 -t 15 kvm" ]
```

# Frontend Configuration

- Virtualization Manager driver configuration
  - Virtualization drivers are used to create, manage, and monitor VMs on the hosts
  - name: The name of the driver
  - executable: The path to the driver executable script
  - argument: Passed to the executable
  - type: The driver type
  - default: The default values and configuration parameters for the driver

```
VM_MAD = [
name = "vmm_xen", executable = "one_vmm_exec", arguments = "-t 15
-r 0 xen", default = "vmm_exec/vmm_exec_xen.conf", type = "xen" ]
```

# Frontend Configuration

- Transfer Manager driver configuration
  - Transfer Manager drivers are used to transfer, create, remove, and clone VM images
  - name: The name of the driver
  - executable: The path to the transfer driver executable script
  - argument: Passed to the executable

```
# SHARED Transfer Manager Driver Configuration

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

# Host Configuration

- The oneadmin account and passwordless login
  root@host1 $ sudo adduser oneadmin
  oneadmin@front-end $ ssh-copy-id oneadmin@host1
  oneadmin$front-end $ ssh oneadmin$host1
- Configuring sudo
  - To give administrative privileges to the oneadmin account on the hosts, add it to the sudo group
    $ sudo adduser oneadmin sudo
- Configuring network bridges

```
iface eth0 inet manual
auto lan0
iface lan0 inet static
bridge_ports eth0
bridge_stp off
bridge_fd 0
address 192.168.66.97
netmask 255.255.255.0
gateway 192.168.66.1
dns-nameservers 192.168.66.1
```

# Xen Installtion

- Installing on Debian Squeeze through standard repositories
  $ apt-get install sudo openssh-server ruby xen-hypervisor-4.0-amd64 linux-image-xen-amd64 xen-qemu-dm-4.0
  - Xen-hypervisor-4.0-amd64: This is the core of Xen. It is the kernel that will execyte Dom0 and DomU instances, and boot up by GRUB before anything else. It controls the CPU and memory sharing between running instances.
  - Linux-image-xen-amd64: This is a Linux kernel with support for Dom0 (the instance used for managing the entire Xen system) and DomU (the kernel for virtual machines)
  - Xen-qemu-dm-4.0: This is a QEMU patch for specific Xen support. With this you can run a fully virtual machine using CPU virtualization support.

# Xen Installtion

- Installing on Debian Squeeze through standard repositories
  $ apt-get install sudo openssh-server ruby xen-hypervisor-4.0-amd64 linux-image-xen-amd64 xen-qemu-dm-4.0
  - Xen-hypervisor-4.0-amd64: This is the core of Xen. It is the kernel that will execyte Dom0 and DomU instances, and boot up by GRUB before anything else. It controls the CPU and memory sharing between running instances.
  - Linux-image-xen-amd64: This is a Linux kernel with support for Dom0 (the instance used for managing the entire Xen system) and DomU (the kernel for virtual machines)
  - Xen-qemu-dm-4.0: This is a QEMU patch for specific Xen support. With this you can run a fully virtual machine using CPU virtualization support.

# Xen Installtion

- Installing on Debian Squeeze through standard repositories
  $ apt-get install sudo openssh-server ruby xen-hypervisor-4.0-amd64 linux-image-xen-amd64 xen-qemu-dm-4.0
  - Xen-hypervisor-4.0-amd64: This is the core of Xen. It is the kernel that will execyte Dom0 and DomU instances, and boot up by GRUB before anything else. It controls the CPU and memory sharing between running instances.
  - Linux-image-xen-amd64: This is a Linux kernel with support for Dom0 (the instance used for managing the entire Xen system) and DomU (the kernel for virtual machines)
  - Xen-qemu-dm-4.0: This is a QEMU patch for specific Xen support. With this you can run a fully virtual machine using CPU virtualization support.

# Shared Storage

- Install the NFS server on the frontend
  $ sudo apt-get install nfs-kernel-server
- Configure /etc/exports
  /var/lib/one 192.168.66.0/24 (rw, async, no_subtree_check, no_roo_squash)
- Reload the NFS service when finishing configuring exports
  $ sudo /etc/init.d/nfs-kernel-server reload
- Install the NFS client on the hosts
  $ sudo apt-get install nfs-common
- Mount the file system
  $ sudo mkdir –p /var/lib/lon
  $ sudo mount 192.168.66.10:/var/lib/one /var/lib/one

# Presentation Roadmap

- Overview of OpenNebula
  - The Core Features
  - Supported Hypervisors
- Build Up Your Cloud
  - Required Configurations
  - OpenNebula Installation
  - Frontend Configuration
  - Host Configuration
  - Xen Installation
  - Shared Storage
- Operate Your Cloud
  - Managing Virtual Networks
  - Managing Disk Images
  - Managing Virtual Machines

# Questions?

# Referneces


- OpenNebula 3 Cloud Computing by Giovanni Toraldo