

---

**OpenNebula.org**

**OpenNebula 4.6 Design and Installation  
Guide**

*Release 4.6*

**OpenNebula Project**

June 12, 2014



## CONTENTS

<b>1</b>	<b>Building your Cloud</b>	<b>1</b>
1.1	An Overview of OpenNebula . . . . .	1
1.2	Understanding OpenNebula . . . . .	6
1.3	Planning the Installation . . . . .	11
1.4	Installing the Software . . . . .	16
1.5	Glossary . . . . .	24
<b>2</b>	<b>Quick Starts</b>	<b>27</b>
2.1	Quickstart: OpenNebula on CentOS 6 and KVM . . . . .	27
2.2	Quickstart: OpenNebula on CentOS 6 and Xen . . . . .	33
2.3	Quickstart: OpenNebula on CentOS 6 and ESX 5.x . . . . .	39
2.4	Quickstart: OpenNebula on Ubuntu 14.04 and KVM . . . . .	53
2.5	Quickstart: Create Your First vDC . . . . .	58



## BUILDING YOUR CLOUD

### 1.1 An Overview of OpenNebula

OpenNebula is the **open-source industry standard for data center virtualization**, offering a **simple but feature-rich and flexible solution** to build and manage enterprise clouds and virtualized data centers. OpenNebula is designed to be simple. Simple to install, update and operate by the admins, and simple to use by end users. Being focused on simplicity, we integrate with existing technologies whenever possible. You'll see that OpenNebula works with MySQL, Ceph, LVM, GlusterFS, Open vSwitch, LDAP... This allows us to deliver a light, flexible and robust cloud manager. This introductory guide gives an overview of OpenNebula and summarizes its main benefits for the different stakeholders involved in a cloud computing infrastructure.

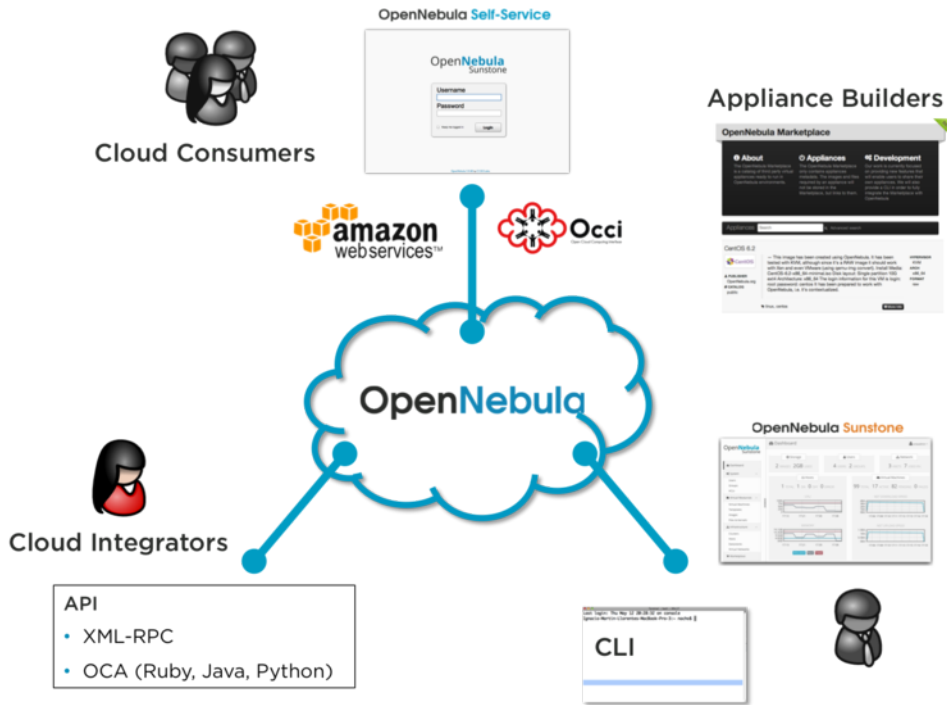
#### 1.1.1 What Are the Key Features Provided by OpenNebula?

You can refer to our a summarized table of [Key Features](#) or to the *Detailed Features and Functionality Guide* included in the documentation of each version.

#### 1.1.2 What Are the Interfaces Provided by OpenNebula?

OpenNebula provides many different interfaces that can be used to interact with the functionality offered to manage physical and virtual resources. There are four main different perspectives to interact with OpenNebula:

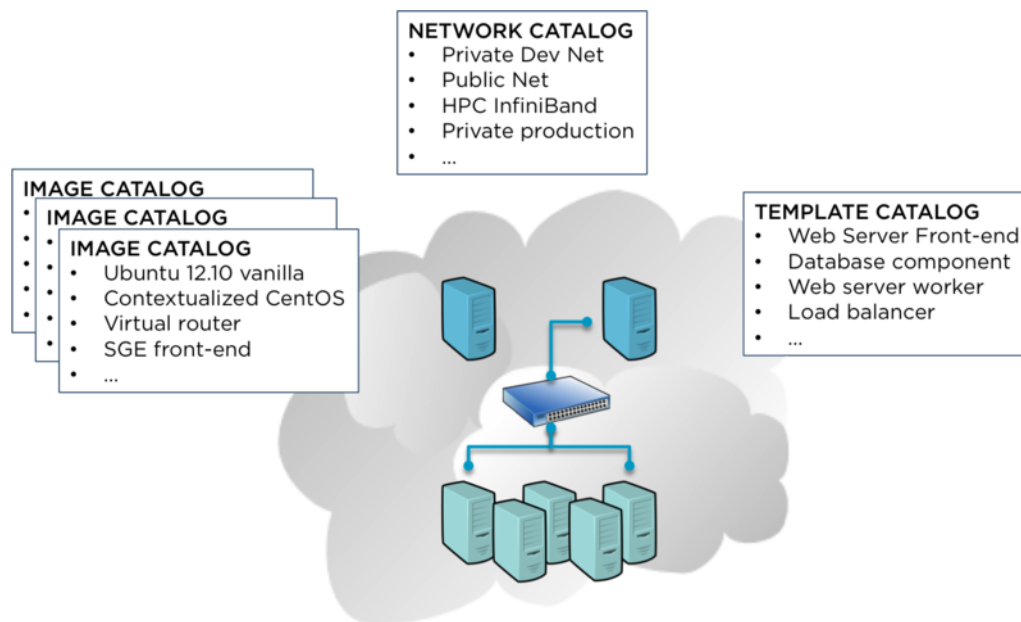
- Cloud interfaces for **Cloud Consumers**, like the *OCCI* and *EC2 Query and EBS* interfaces, and a simple *Sunstone cloud user view* that can be used as a self-service portal.
- Administration interfaces for **Cloud Advanced Users and Operators**, like a Unix-like *command line interface* and the powerful *Sunstone GUI*.
- Extensible low-level APIs for **Cloud Integrators** in *Ruby*, *JAVA* and *XMLRPC API*
- A *Marketplace* for **Appliance Builders** with a catalog of virtual appliances ready to run in OpenNebula environments.



### 1.1.3 What Does OpenNebula Offer to Cloud Consumers?

OpenNebula provides a powerful, scalable and secure multi-tenant cloud platform for fast delivery and elasticity of virtual resources. Multi-tier applications can be deployed and consumed as pre-configured virtual appliances from catalogs.

- **Image Catalogs:** OpenNebula allows to store *disk images in catalogs* (termed datastores), that can be then used to define VMs or shared with other users. The images can be OS installations, persistent data sets or empty data blocks that are created within the datastore.
- **Network Catalogs:** *Virtual networks* can be also be organised in network catalogs, and provide means to interconnect virtual machines. This kind of resources can be defined as fixed or ranged networks, and can be used to achieve full isolation between virtual networks.
- **VM Template Catalog:** The *template catalog* system allows to register *virtual machine* definitions in the system, to be instantiated later as virtual machine instances.
- **Virtual Resource Control and Monitoring:** Once a template is instantiated to a virtual machine, there are a number of operations that can be performed to control lifecycle of the *virtual machine instances*, such as migration (live and cold), stop, resume, cancel, poweroff, etc.
- **Multi-tier Cloud Application Control and Monitoring:** OpenNebula allows to *define, execute and manage multi-tiered elastic applications*, or services composed of interconnected Virtual Machines with deployment dependencies between them and *auto-scaling rules*.

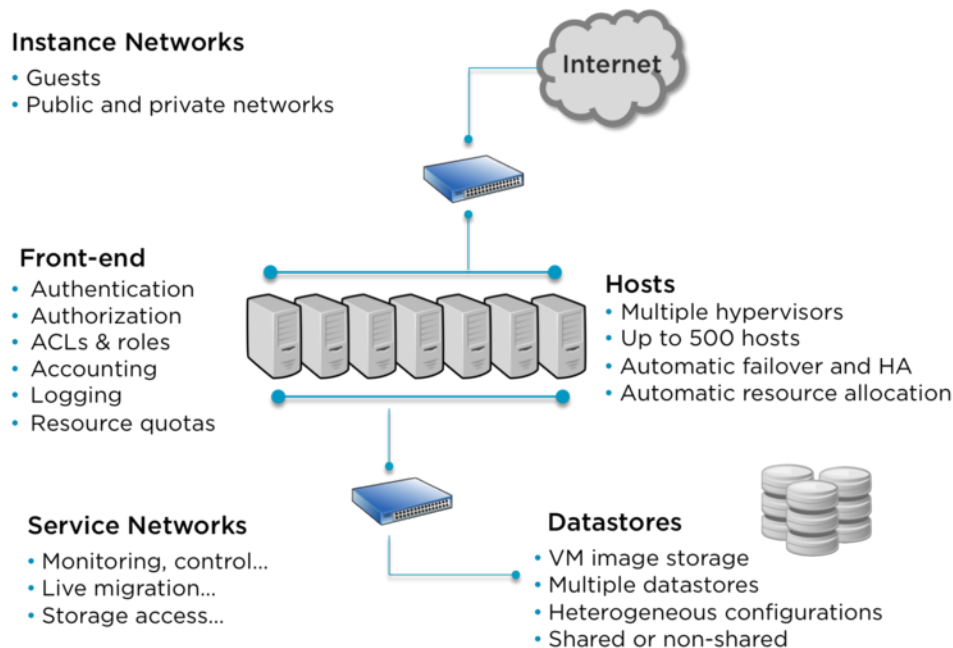


### 1.1.4 What Does OpenNebula Offer to Cloud Operators?

OpenNebula is composed of the following subsystems:

- **Users and Groups:** OpenNebula features advanced multi-tenancy with powerful *users and groups management*, *fine-grained ACLs* for resource allocation, and *resource quota management* to track and limit computing, storage and networking utilization.
- **Virtualization:** Various hypervisors are supported in the *virtualization manager*, with the ability to control the complete lifecycle of Virtual Machines and multiple hypervisors in the same cloud infrastructure.
- **Hosts:** The *host manager* provides complete functionality for the management of the physical hosts in the cloud.
- **Monitoring:** Virtual resources as well as *hosts* are periodically monitored for key performance indicators. The information can then be used by a powerful and flexible *scheduler* for the definition of workload and resource-aware allocation policies. You can also *gain insight application status and performance*.
- **Accounting:** A Configurable *accounting system* to visualize and report resource usage data, to allow their integration with chargeback and billing platforms, or to guarantee fair share of resources among users.
- **Networking:** An easily adaptable and customizable *network subsystem* is present in OpenNebula in order to better integrate with the specific network requirements of existing data centers and to allow full isolation between virtual machines that composes a virtualised service.
- **Storage:** The support for multiple datastores in the *storage subsystem* provides extreme flexibility in planning the storage backend and important performance benefits.
- **Security:** This feature is spread across several subsystems: *authentication and authorization mechanisms* allowing for various possible mechanisms to identify and authorize users, a powerful *Access Control List* mechanism allowing different role management with fine grain permission granting over any resource managed by OpenNebula, support for isolation at different levels...
- **High Availability:** Support for *HA architectures* and *configurable behavior in the event of host or VM failure* to provide easy to use and cost-effective failover solutions.
- **Clusters:** *Clusters* are pools of hosts that share datastores and virtual networks. Clusters are used for load balancing, high availability, and high performance computing.

- **Multiple Zones:** The *Data Center Federation* functionality allows for the centralized management of multiple instances of OpenNebula for scalability, isolation and multiple-site support.
- **VDCs.** An OpenNebula instance (or Zone) can be further compartmentalized in *Virtual Data Centers (VDCs)*, which offer a fully-isolated virtual infrastructure environments where a group of users, under the control of the VDC administrator, can create and manage compute, storage and networking capacity.
- **Cloud Bursting:** OpenNebula gives support to build a *hybrid cloud*, an extension of a private cloud to combine local resources with resources from remote cloud providers. A whole public cloud provider can be encapsulated as a local resource to be able to use extra computational capacity to satisfy peak demands.
- **App Market:** OpenNebula allows the deployment of a [private centralized catalog of cloud applications](#) to share and distribute virtual appliances across OpenNebula instances



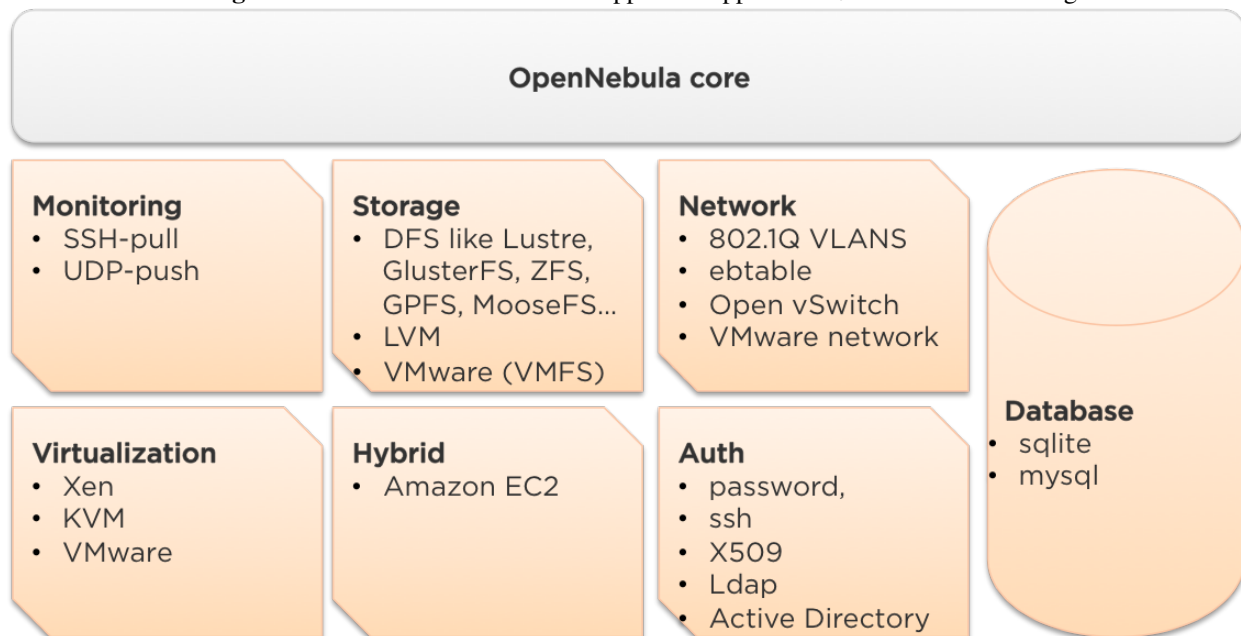
### 1.1.5 What Does OpenNebula Offer to Cloud Builders?

OpenNebula offers broad support for commodity and enterprise-grade hypervisor, monitoring, storage, networking and user management services:

- **User Management:** OpenNebula can validate users using its own internal user database based on *passwords*, or external mechanisms, like *ssh*, *x509*, *ldap* or *Active Directory*
- **Virtualization:** Several hypervisor technologies are fully supported, like *Xen*, *KVM* and *VMware*.
- **Monitoring:** OpenNebula provides its own *customizable and highly scalable monitoring system* and also can be integrated with external data center monitoring tools.
- **Networking:** Virtual networks can be backed up by *802.1Q VLANs*, *etables*, *Open vSwitch* or *VMware networking*.
- **Storage:** Multiple backends are supported like the regular (shared or not) *filesystem datastore* supporting popular distributed file systems like *NFS*, *Lustre*, *GlusterFS*, *ZFS*, *GPFS*, *MooseFS*...; the *VMware datastore* (both regular filesystem or VMFS based) specialized for the VMware hypervisor that handle the *vmdk* format; the *LVM datastore* to store disk images in a block device form; and *Ceph* for distributed block device.
- **Databases:** Aside from the original *sqlite* backend, *mysql* is also supported.



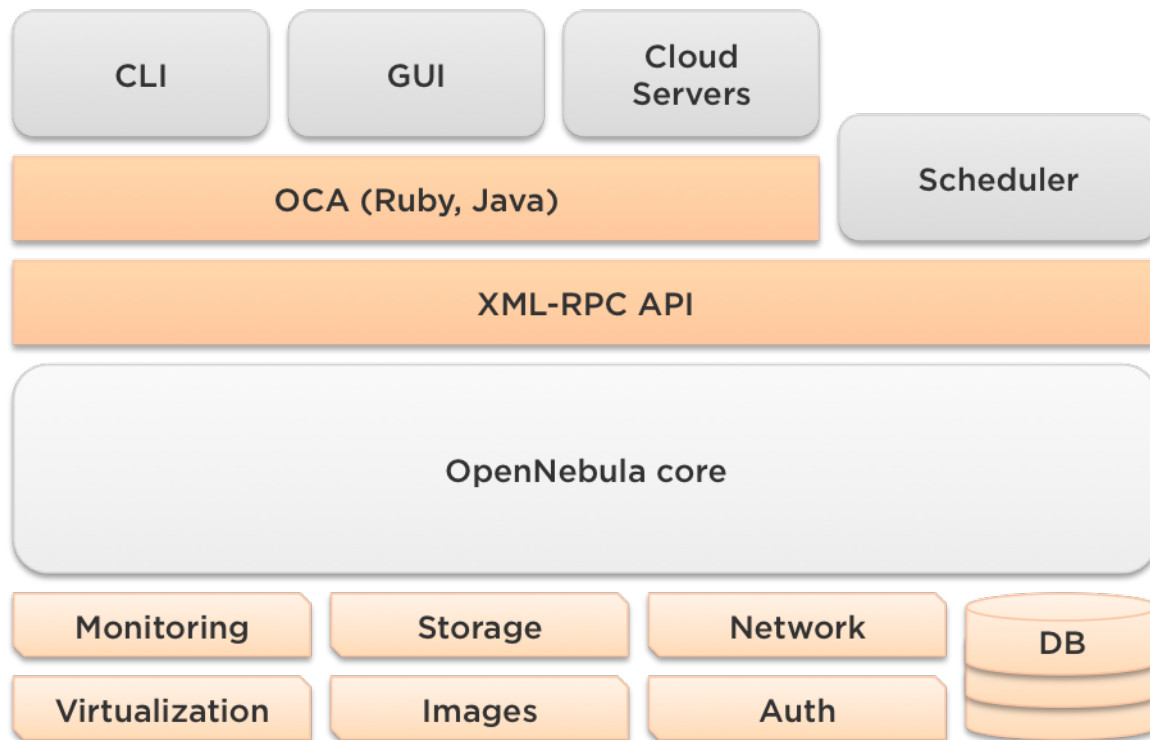
- **Cloud Bursting:** Out of the box connectors are shipped to support *Amazon EC2* cloudbursting.



### 1.1.6 What Does OpenNebula Offer to Cloud Integrators?

OpenNebula is fully platform independent and offers many tools for cloud integrators:

- **Modular and extensible architecture** with *customizable plug-ins* for integration with any third-party data center service
- **API for integration** with higher level tools such as billing, self-service portals... that offers all the rich functionality of the OpenNebula core, with bindings for *ruby* and *java*.
- **Sunstone Server custom routes** to extend the *sunstone server*.
- **OneFlow API** to create, control and monitor *multi-tier applications or services composed of interconnected Virtual Machines*.
- **Hook Manager** to *trigger administration scripts upon VM state change*.



## 1.2 Understanding OpenNebula

This guide is meant for the cloud architect and administrator, to help him to understand the way OpenNebula categorizes the infrastructure resources, and how they are consumed by the users.

In a tiny installation with a few hosts, you can use OpenNebula with the two default groups for the administrator and the users, without giving much thought to the infrastructure partitioning and user organization. But for medium and big deployments you will probably want to provide some level of isolation and structure.

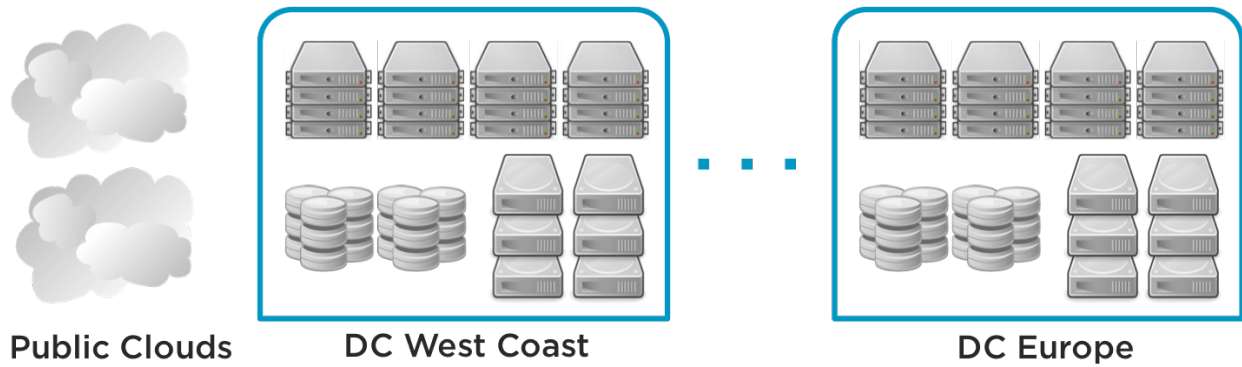
Although OpenNebula has been designed and developed to be easy to adapt to each individual company use case and processes, and perform fine-tuning of multiple aspects, OpenNebula brings a pre-defined model for cloud provisioning and consumption.

The OpenNebula model is a result of our collaboration with our user community during the last years.

### 1.2.1 The Infrastructure Perspective

Common large IT shops have multiple Data Centers (DCs), each one of them consisting of several physical Clusters of infrastructure resources (hosts, networks and storage). These Clusters could present different architectures and software/hardware execution environments to fulfill the needs of different workload profiles. Moreover, many organizations have access to external public clouds to build hybrid cloud scenarios where the private capacity of the Data Centers is supplemented with resources from external clouds to address peaks of demand. Sysadmins need a single comprehensive framework to dynamically allocate all these available resources to the multiple groups of users.

For example, you could have two Data Centers in different geographic locations, Europe and USA West Coast, and an agreement for cloudbursting with a public cloud provider, such as Amazon. Each Data Center runs its own full OpenNebula deployment. Multiple OpenNebula installations can be configured as a federation, and in this case they will share the same user accounts, groups, and permissions across Data Centers.



## 1.2.2 The Organizational Perspective

Users are organized into Groups (also called Projects, Domains, Tenants...). A Group is an authorization boundary that can be seen as a business unit if you are considering it as private cloud or as a complete new company if it is public cloud.

A Group is simply a boundary, you need to populate resources into the Group which can then be consumed by the users of that Group. A vDC (Virtual Data Center) is a Group plus Resource Providers assigned. A Resource Provider is a Cluster of infrastructure resources (physical hosts, networks, storage and external clouds) from one of the Data Centers.

Different authorization scenarios can be enabled with the powerful and configurable ACL system provided, from the definition of vDC Admins to the privileges of the users that can deploy virtual machines. Each vDC can execute different types of workload profiles with different performance and security requirements.

The following are common enterprise use cases in large cloud computing deployments:

- **On-premise Private Clouds** Serving Multiple Projects, Departments, Units or Organizations. On-premise private clouds in large organizations require powerful and flexible mechanisms to manage the access privileges to the virtual and physical infrastructure and to dynamically allocate the available resources. In these scenarios, the Cloud Administrator would define a vDC for each Department, dynamically allocating resources according to their needs, and delegating the internal administration of the vDC to the Department IT Administrator.
- **Cloud Providers** Offering Virtual Private Cloud Computing. Cloud providers providing customers with a fully-configurable and isolated environment where they have full control and capacity to administer its users and resources. This combines a public cloud with the control usually seen in a personal private cloud system.



For example, you can think Web Development, Human Resources, and Big Data Analysis as business units represented by vDCs in a private OpenNebula cloud.

- **BLUE:** Allocation of (ClusterA-DC\_West\_Coast + Cloudbursting) to Web Development
- **RED:** Allocation of (ClusterB-DC\_West\_Coast + ClusterA-DC\_Europe + Cloudbursting) to Human Resources
- **GREEN:** Allocation of (ClusterC-DC\_West\_Coast + ClusterB-DC\_Europe) to Big Data Analysis

Web Development



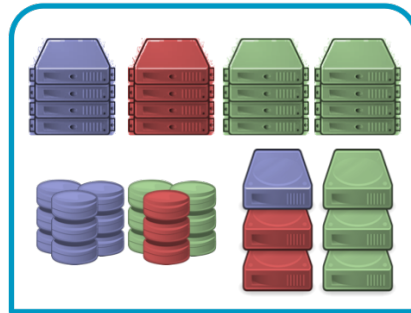
Human Resources



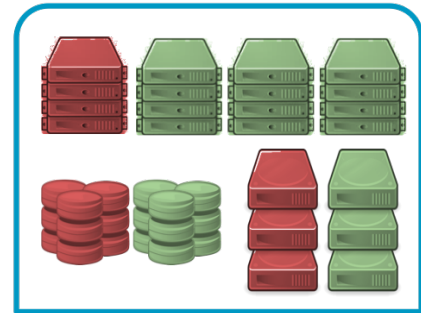
Big Data Analysis



Public Clouds



DC West Coast



DC Europe

### 1.2.3 A Cloud Provisioning Model Based on vDCs

A vDC is a fully-isolated virtual infrastructure environment where a Group of users, optionally under the control of the vDC admin, can create and manage compute and storage capacity. The users in the vDC, including the vDC administrator, would only see the virtual resources and not the underlying physical infrastructure. The physical resources allocated by the cloud administrator to the vDC can be completely dedicated to the vDC, providing isolation at the physical level too.

The privileges of the vDC users and the administrator regarding the operations over the virtual resources created by other users can be configured. In a typical scenario the vDC administrator can upload and create images and virtual machine templates, while the users can only instantiate virtual machine templates to create their machines. The administrators of the vDC have full control over other users' resources and can also create new users in the vDC.



Create



VMs



Templates



dani



Log out



OpenNebula

## Virtual Machines

Search



<p>CentOS 6.4 Server-81</p> <p>x8 - 8GB</p> <p>Ubuntu 12.04</p> <p>192.168.1.7</p> <p>DEPLOYING (1/3) 3m ago</p>	<p>CentOS 6.4 Server-80</p> <p>x1 - 512MB</p> <p>Ubuntu 12.04</p> <p>192.168.1.6</p> <p>RUNNING 3m ago</p>	<p>CentOS 6.4 + Nginx</p> <p>x1 - 512MB</p> <p>Ubuntu 12.04</p> <p>192.168.1.5</p> <p>RUNNING 3m ago</p>
<p>Ubuntu Server 12.04-78</p> <p>x1 - 512MB</p> <p>Ubuntu 12.04</p> <p>192.168.1.4</p> <p>RUNNING 3m ago</p>	<p>Ubuntu Server 12.04-77</p> <p>x1 - 512MB</p> <p>Ubuntu 12.04</p> <p>192.168.1.3</p> <p>OFF 3m ago</p>	<p>Development VM</p> <p>x1 - 512MB</p> <p>Ubuntu 12.04</p> <p>192.168.1.2</p> <p>RUNNING 4m ago</p>

« 1 » 6

[Create Virtual Machine](#)

OpenNebula 4.5.80 by C12G Labs.

Users can then access their vDC through any of the existing OpenNebula interfaces, such as the CLI, Sunstone Cloud View, OCA, or the OCCI and AWS APIs. vDC administrators can manage their vDCs through the CLI or the vDC admin view in Sunstone. Cloud Administrators can manage the vDCs through the CLI or Sunstone.

The Cloud provisioning model based on vDCs enables an integrated, comprehensive framework to dynamically provision the infrastructure resources in large multi-datacenter environments to different customers, business units or groups. This brings several benefits:

- Partitioning of cloud physical resources between Groups of users
- Complete isolation of users, organizations or workloads
- Allocation of Clusters with different levels of security, performance or high availability
- Containers for the execution of software-defined data centers
- Way of hiding physical resources from Group members
- Simple federation, scalability and cloudbursting of private cloud infrastructures beyond a single cloud instance and data center

### 1.2.4 Cloud Usage Models

OpenNebula has three pre-defined user roles to implement two typical enterprise cloud scenarios: infrastructure management and infrastructure provisioning.

In both scenarios, the Cloud Administrator manages the physical infrastructure, creates users and vDC, and prepares base templates and images for other users.

Role	Capabilities
<b>Cloud Admin.</b>	<ul style="list-style-type: none"> <li>• Operates the Cloud infrastructure (i.e. computing nodes, networking fabric, storage servers)</li> <li>• Creates and manages OpenNebula infrastructure resources: Hosts, Virtual Networks, Datastores</li> <li>• Creates and manages <i>Application Flows</i></li> <li>• Creates new groups for vDCs</li> <li>• Assigns resource providers to a vDC and sets quota limits</li> <li>• Defines base instance types to be used by the vDCs. These types define the capacity of the VMs (memory, cpu and additional storage) and connectivity.</li> <li>• Prepare VM images to be used by the vDCs</li> <li>• Monitor the status and health of the cloud</li> <li>• Generate activity reports</li> </ul>

### Infrastructure Management

In this usage model, users are familiar with virtualization concepts. Except for the infrastructure resources, the web interface offers the same operations available to the Cloud Admin.

End users can use the templates and images pre-defined by the cloud administrator, but are also allowed to create their own. They are also able to manage the life-cycle of their resources, including advanced features that may harm the VM guests, like hot-plugging of new disks, resize of Virtual Machines, modify boot parameters, etc.

Role	Capabilities
<b>User</b>	<ul style="list-style-type: none"> <li>• Instantiates VMs using their own templates</li> <li>• Creates new Images</li> <li>• Manages their VMs, including advanced life-cycle features</li> <li>• Creates and manages <i>Application Flows</i></li> <li>• Check their usage and quotas</li> <li>• Upload SSH keys to access the VMs</li> </ul>

### Infrastructure Provisioning

In a infrastructure provisioning model, the end users access a simplified web interface that allows them to launch Virtual Machines from pre-defined Templates and Images. They can access their VMs, and perform basic operations like shutdown. The changes made to a VM disk can be saved back, but new Images cannot be created from scratch.

Optionally, each vDC can define one or more users as vDC Admins. These admins can create new users inside the vDC, and also manage the resources of the rest of the users. A vDC Admin may, for example, shutdown a VM from other user to free group quota usage.

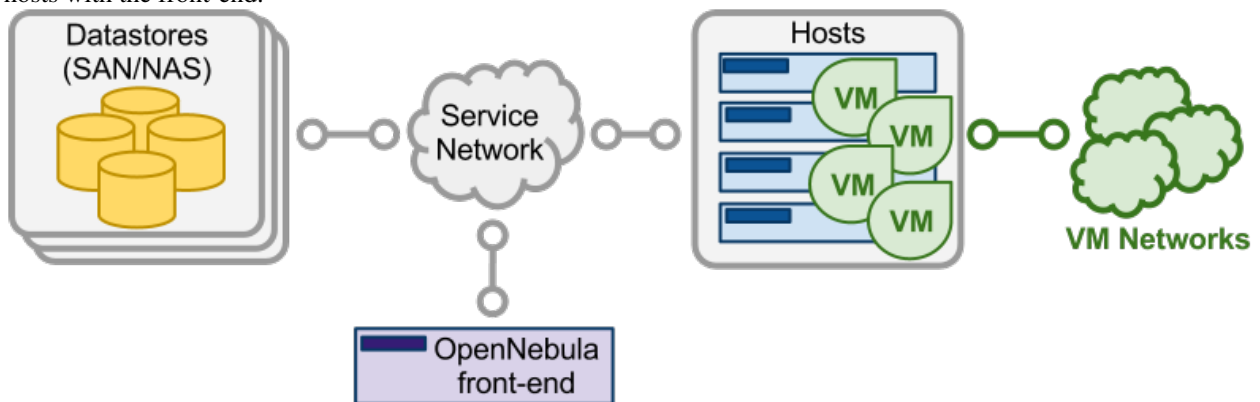
Role	Capabilities
<b>vDC Admin.</b>	<ul style="list-style-type: none"> <li>• Creates new users in the vDC</li> <li>• Operates on vDC virtual machines and disk images</li> <li>• Creates and registers disk images to be used by the vDC users</li> <li>• Checks vDC usage and quotas</li> </ul>
<b>vDC User</b>	<ul style="list-style-type: none"> <li>• Instantiates VMs using the templates defined by the Cloud Admins and the images defined by the Cloud Admins or vDC Admins.</li> <li>• Instantiates VMs using their own Images saved from a previous running VM</li> <li>• Manages their VMs, including                             <ul style="list-style-type: none"> <li>– reboot</li> <li>– power off/on (short-term switching-off)</li> <li>– shutdown</li> <li>– make a VM image snapshot</li> <li>– obtain basic monitor information and status (including IP addresses)</li> </ul> </li> <li>• Delete any previous disk snapshot</li> <li>• Check user usage and quotas</li> <li>• Upload SSH keys to access the VMs</li> </ul>

### 1.3 Planning the Installation

In order to get the most out of a OpenNebula Cloud, we recommend that you create a plan with the features, performance, scalability, and high availability characteristics you want in your deployment. This guide provides information to plan an OpenNebula installation, so you can easily architect your deployment and understand the technologies involved in the management of virtualized resources and their relationship.

#### 1.3.1 Architectural Overview

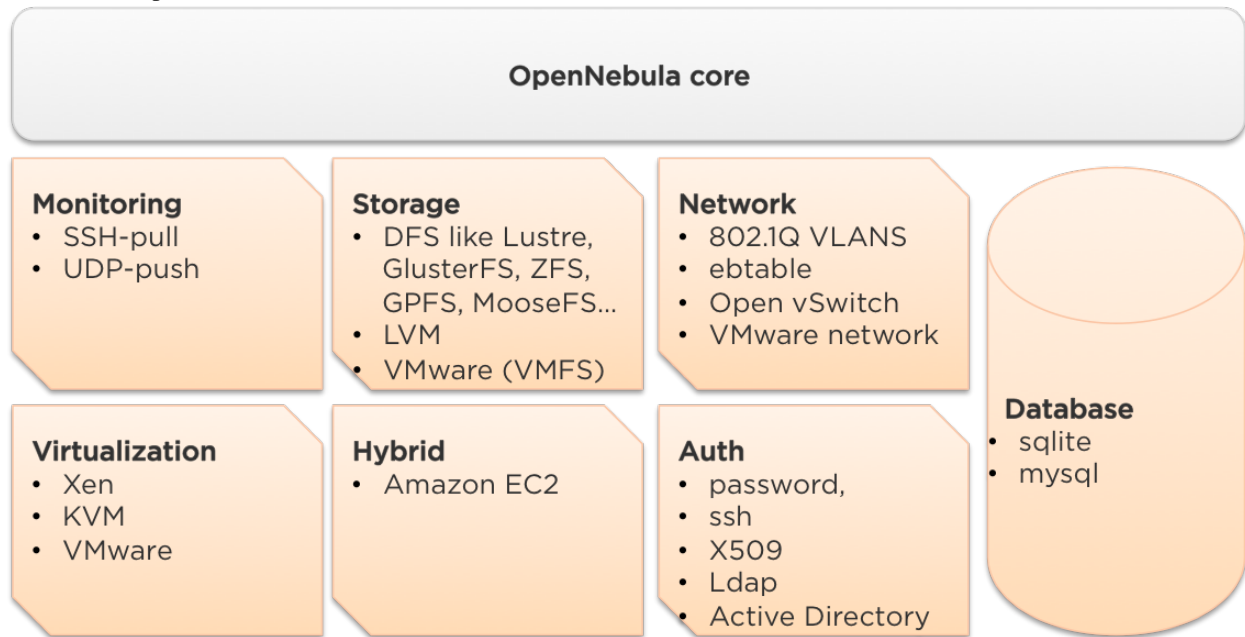
OpenNebula assumes that your physical infrastructure adopts a classical cluster-like architecture with a front-end, and a set of hosts where Virtual Machines (VM) will be executed. There is at least one physical network joining all the hosts with the front-end.



The basic components of an OpenNebula system are:

- **Front-end** that executes the OpenNebula services.
- Hypervisor-enabled **hosts** that provide the resources needed by the VMs.
- **Datastores** that hold the base images of the VMs.
- Physical **networks** used to support basic services such as interconnection of the storage servers and OpenNebula control operations, and VLANs for the VMs.

OpenNebula presents a highly modular architecture that offers broad support for commodity and enterprise-grade hypervisor, monitoring, storage, networking and user management services. This guide briefly describes the different choices that you can make for the management of the different subsystems. If your specific services are not supported we recommend to check the drivers available in the [Add-on Catalog](#). We also provide information and support about how to develop new drivers.



### 1.3.2 Front-End

The machine that holds the OpenNebula installation is called the front-end. This machine needs network connectivity to each host, and possibly access to the storage Datastores (either by direct mount or network). The base installation of OpenNebula takes less than 50MB.

OpenNebula services include:

- Management daemon (`oned`) and scheduler (`mm_sched`)
- Web interface server (`sunstone-server`)

**Warning:** Note that these components communicate through *XML-RPC* and may be installed in different machines for security or performance reasons

There are several certified platforms to act as front-end for each version of OpenNebula. Refer to the *platform notes* and chose the one that better fits your needs.

OpenNebula's default database uses **sqlite**. If you are planning a production or medium to large scale deployment, you should consider using *MySQL*.



If you are interested in setting up a high available cluster for OpenNebula, check the *High OpenNebula Availability Guide*.

The maximum number of servers (virtualization hosts) that can be managed by a single OpenNebula instance (zone) strongly depends on the performance and scalability of the underlying platform infrastructure, mainly the storage subsystem. We do not recommend more than 500 servers within each zone, but there are users with 1,000 servers in each zone. You may find interesting the following guide about *how to tune OpenNebula for large deployments*.

### 1.3.3 Monitoring

The monitoring subsystem gathers information relative to the hosts and the virtual machines, such as the host status, basic performance indicators, as well as VM status and capacity consumption. This information is collected by executing a set of static probes provided by OpenNebula. The output of these probes is sent to OpenNebula in two different ways:

- **UDP-push Model:** Each host periodically sends monitoring data via UDP to the frontend which collects it and processes it in a dedicated module. This model is highly scalable and its limit (in terms of number of VMs monitored per second) is bounded to the performance of the server running oned and the database server. Please read the *UDP-push guide* for more information.
- **Pull Model:** OpenNebula periodically actively queries each host and executes the probes via `ssh`. This mode is limited by the number of active connections that can be made concurrently, as hosts are queried sequentially. Please read the *KVM and Xen SSH-pull guide* or the *ESX-pull guide* for more information.

**Warning: Default:** UDP-push Model is the default IM for KVM and Xen in OpenNebula  $\geq$  4.4.

Please check the *the Monitoring Guide* for more details.

### 1.3.4 Virtualization Hosts

The hosts are the physical machines that will run the VMs. There are several certified platforms to act as nodes for each version of OpenNebula. Refer to the *platform notes* and chose the one that better fits your needs. The Virtualization Subsystem is the component in charge of talking with the hypervisor installed in the hosts and taking the actions needed for each step in the VM lifecycle.

OpenNebula natively supports three hypervisors:

- *Xen*
- *KVM*
- *VMware*

**Warning: Default:** OpenNebula is configured to interact with hosts running KVM.

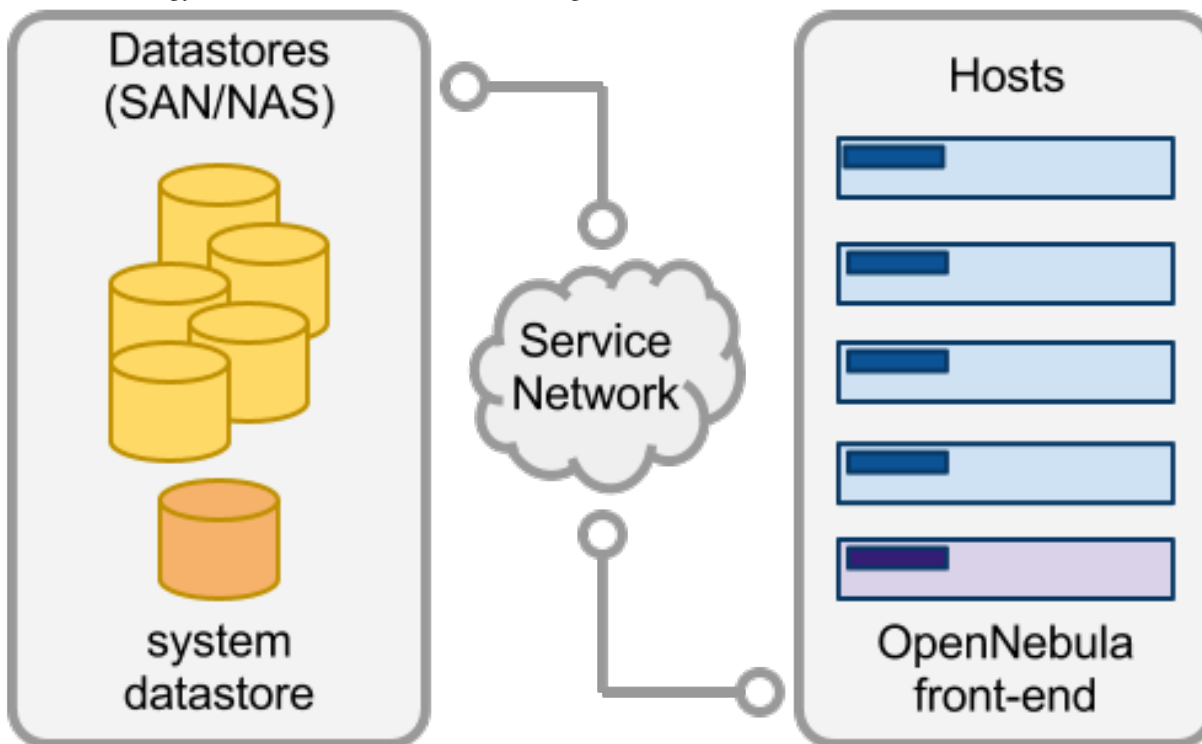
Please check the *Virtualization Guide* for more details of the supported virtualization technologies.

If you are interested in failover protection against hardware and operating system outages within your virtualized IT environment, check the *Virtual Machines High Availability Guide*.

### 1.3.5 Storage

OpenNebula uses Datastores to handle the VM disk Images. A Datastore is any storage medium used to store disk images for VMs, previous versions of OpenNebula refer to this concept as Image Repository. Typically, a datastore

will be backed by SAN/NAS servers. In general, each Datastore has to be accessible through the front-end using any suitable technology NAS, SAN or direct attached storage.



When a VM is deployed the Images are *transferred* from the Datastore to the hosts. Depending on the actual storage technology used it can mean a real transfer, a symbolic link or setting up an LVM volume.

OpenNebula is shipped with 3 different datastore classes:

- *System Datastores* to hold images for running VMs, depending on the storage technology used these temporal images can be complete copies of the original image, qcow deltas or simple filesystem links.
- **Image Datastores** store the disk images repository. Disk images are moved, or cloned to/from the System datastore when the VMs are deployed or shutdown; or when disks are attached or snapshotted.
- *File Datastore* is a special datastore used to store plain files and not disk images. The plain files can be used as kernels, ramdisks or context files.

Image datastores can be of different type depending on the underlying storage technology:

- *File-system*, to store disk images in a file form. The files are stored in a directory mounted from a SAN/NAS server.
- *vmfs*, a datastore specialized in VMFS format to be used with VMware hypervisors. Cannot be mounted in the OpenNebula front-end since VMFS is not \*nix compatible.
- *LVM*, The LVM datastore driver provides OpenNebula with the possibility of using LVM volumes instead of plain files to hold the Virtual Images. This reduces the overhead of having a file-system in place and thus increases performance..
- *Ceph*, to store disk images using Ceph block devices.

**Warning: Default:** The system and images datastores are configured to use a shared filesystem.

Please check the *Storage Guide* for more details.

### 1.3.6 Networking

OpenNebula provides an easily adaptable and customizable network subsystem in order to better integrate with the specific network requirements of existing datacenters. At least two different physical networks are needed:

- A **service network** is needed by the OpenNebula front-end daemons to access the hosts in order to manage and monitor the hypervisors, and move image files. It is highly recommended to install a dedicated network for this purpose.
- A **instance network** is needed to offer network connectivity to the VMs across the different hosts. To make an effective use of your VM deployments you'll probably need to make one or more physical networks accessible to them.

The OpenNebula administrator may associate one of the following drivers to each Host:

- **dummy**: Default driver that doesn't perform any network operation. Firewalling rules are also ignored.
- *fw*: Firewall rules are applied, but networking isolation is ignored.
- *802.1Q*: restrict network access through VLAN tagging, which also requires support from the hardware switches.
- *eatables*: restrict network access through Eatables rules. No special hardware configuration required.
- *ovswitch*: restrict network access with [Open vSwitch Virtual Switch](#).
- *VMware*: uses the VMware networking infrastructure to provide an isolated and 802.1Q compatible network for VMs launched with the VMware hypervisor.

**Warning: Default:** The default configuration connects the virtual machine network interface to a bridge in the physical host.

Please check the *Networking Guide* to find out more information of the networking technologies supported by OpenNebula.

### 1.3.7 Authentication

You can choose from the following authentication models to access OpenNebula:

- *Built-in User/Password*
- *SSH Authentication*
- *X509 Authentication*
- *LDAP Authentication*

**Warning: Default:** OpenNebula comes by default with an internal built-in user/password authentication.

Please check the *External Auth guide* to find out more information of the auth technologies supported by OpenNebula.

### 1.3.8 Advanced Components

Once you have an OpenNebula cloud up and running, you can install the following advanced components:

- *Application Flow and Auto-scaling*: OneFlow allows users and administrators to define, execute and manage multi-tiered applications, or services composed of interconnected Virtual Machines with deployment dependencies between them. Each group of Virtual Machines is deployed and managed as a single entity, and is completely integrated with the advanced OpenNebula user and group management.

- *Cloud Bursting*: Cloud bursting is a model in which the local resources of a Private Cloud are combined with resources from remote Cloud providers. Such support for cloud bursting enables highly scalable hosting environments.
- *Public Cloud*: Cloud interfaces can be added to your Private Cloud if you want to provide partners or external users with access to your infrastructure, or to sell your overcapacity. The following interfaces provide a simple and remote management of cloud (virtual) resources at a high abstraction level: *Amazon EC2 and EBS APIs* or *OGF OCCl*.
- *Application Insight*: OneGate allows Virtual Machine guests to push monitoring information to OpenNebula. Users and administrators can use it to gather metrics, detect problems in their applications, and trigger OneFlow auto-scaling rules.

## 1.4 Installing the Software

This page shows you how to install OpenNebula from the binary packages.

### 1.4.1 Step 1. Front-end Installation

Using the packages provided in our site is the recommended method, to ensure the installation of the latest version and to avoid possible packages divergences of different distributions. There are two alternatives here, to install OpenNebula you can add **our package repositories** to your system, or visit the [software menu](#) to **download the latest package** for your Linux distribution.

Do not forget that we offer Quickstart guides for:

- *OpenNebula on CentOS and KVM*
- *OpenNebula on CentOS and Xen*
- *OpenNebula on CentOS and VMware*
- *OpenNebula on Ubuntu and KVM*

If there are no packages for your distribution, head to the *Building from Source Code guide*.

### 1.1. Installing on CentOS/RHEL

Before installing:

- Activate the [EPEL](#) repo.

There are packages for the front-end, distributed in the various components that conform OpenNebula, and packages for the virtualization host.

To install a CentOS/RHEL OpenNebula front-end with packages from **our repository**, execute the following as root:

```
# cat << EOT > /etc/yum.repos.d/opennebula.repo
[opennebula]
name=opennebula
baseurl=http://downloads.opennebula.org/repo/CentOS/6/stable/\$basearch
enabled=1
gpgcheck=0
EOT
# yum install opennebula-server opennebula-sunstone opennebula-ruby
```

## CentOS/RHEL Package Description

These are the packages available for this distribution:

- **opennebula**: Command Line Interface
- **opennebula-server**: Main OpenNebula daemon, scheduler, etc
- **opennebula-sunstone**: OpenNebula Sunstone, EC2, OCCI
- **opennebula-ruby**: Ruby Bindings
- **opennebula-java**: Java Bindings
- **opennebula-gate**: Gate server that enables communication between VMs and OpenNebula
- **opennebula-flow**: Manages services and elasticity
- **opennebula-node-kvm**: Meta-package that installs the oneadmin user, libvirt and kvm
- **opennebula-common**: Common files for OpenNebula packages

## 1.2. Installing on openSUSE

Before installing:

- Activate the PackMan repo.

```
# zypper ar -f -n packman http://packman.inode.at/suse/openSUSE_12.3 packman
```

To install an openSUSE OpenNebula front-end with packages **from our repository**, execute the following as root:

```
# zypper addrepo --no-gpgcheck --refresh -t YUM http://downloads.opennebula.org/repo/openSUSE/12.3/s
# zypper refresh
# zypper install opennebula opennebula-sunstone
```

To install an openSUSE OpenNebula front-end with packages from **our repository**, execute the following as root:

```
# tar xvzf openSUSE-12.3-<OpenNebula version>.tar.gz
# zypper install opennebula opennebula-sunstone
```

After installation you need to manually create `/var/lib/one/.one/one_auth` with the following contents:

```
oneadmin:<password>
```

## openSUSE Package Description

These are the packages available for this distribution:

- **opennebula**: main OpenNebula binaries
- **opennebula-devel**: Examples, manpages and `install_gems` (depends on **opennebula**)
- **opennebula-sunstone**: OpenNebula Sunstone (depends on **opennebula**)

## 1.3. Installing on Debian/Ubuntu

The JSON ruby library packaged with Debian 6 is not compatible with OpenNebula. To make it work a new gem should be installed and the old one disabled. You can do so executing these commands:

```
$ sudo gem install json
$ sudo mv /usr/lib/ruby/1.8/json.rb /usr/lib/ruby/1.8/json.rb.no
```

To install OpenNebula on a Debian/Ubuntu front-end from packages from **our repositories** execute as root:

```
# wget http://downloads.opennebula.org/repo/Debian/repo.key
# apt-key add repo.key
```

### Debian

```
# echo "deb http://downloads.opennebula.org/repo/Debian/7 stable opennebula" > /etc/apt/sources.list
```

### Ubuntu 12.04

```
# echo "deb http://downloads.opennebula.org/repo/Ubuntu/12.04 stable opennebula" > /etc/apt/sources.list
```

### Ubuntu 14.04

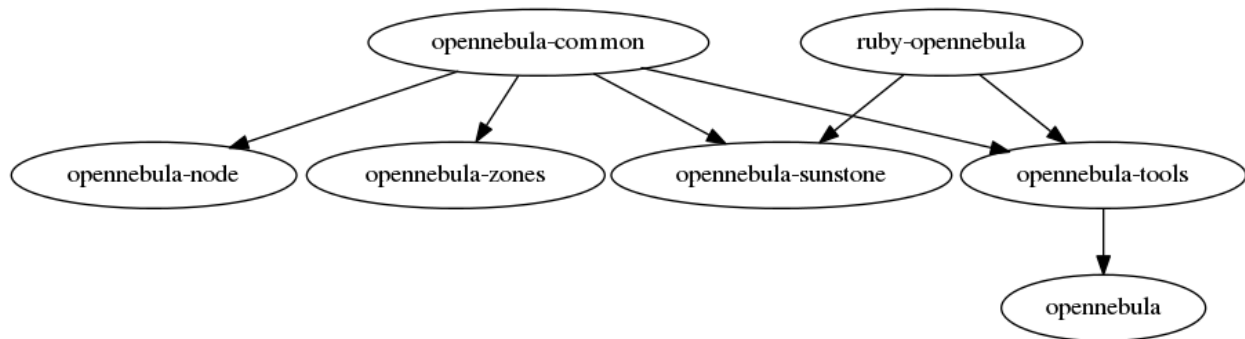
```
# echo "deb http://downloads.opennebula.org/repo/Ubuntu/14.04 stable opennebula" > /etc/apt/sources.list
```

To install the packages on a Debian/Ubuntu front-end:

```
# apt-get update
# apt-get install opennebula opennebula-sunstone
```

### Debian/Ubuntu Package Description

These are the packages available for these distributions:



- **opennebula-common**: provides the user and common files
- **ruby-opennebula**: Ruby API
- **libopennebula-java**: Java API
- **libopennebula-java-doc**: Java API Documentation
- **opennebula-node**: prepares a node as an opennebula-node
- **opennebula-sunstone**: OpenNebula Sunstone Web Interface
- **opennebula-tools**: Command Line interface
- **opennebula-gate**: Gate server that enables communication between VMs and OpenNebula
- **opennebula-flow**: Manages services and elasticity
- **opennebula**: OpenNebula Daemon

### 1.4.2 Step 2. Ruby Runtime Installation

Some OpenNebula components need ruby libraries. OpenNebula provides a script that installs the required gems as well as some development libraries packages needed.

As root execute:

```
# /usr/share/one/install_gems
```

The previous script is prepared to detect common linux distributions and install the required libraries. If it fails to find the packages needed in your system, manually install these packages:

- sqlite3 development library
- mysql client development library
- curl development library
- libxml2 and libxslt development libraries
- ruby development library
- gcc and g++
- make

If you want to install only a set of gems for an specific component read *Building from Source Code* where it is explained in more depth.

### 1.4.3 Step 3. Starting OpenNebula

Log in as the `oneadmin` user follow these steps:

- If you installed from packages, you should have the `one/one_auth` file created with a randomly-generated password. Otherwise, set `oneadmin`'s OpenNebula credentials (username and password) adding the following to `~/one/one_auth` (change password for the desired password):

```
$ mkdir ~/.one
$ echo "oneadmin:password" > ~/.one/one_auth
$ chmod 600 ~/.one/one_auth
```

**Warning:** This will set the `oneadmin` password on the first boot. From that point, you must use the `'oneuser passwd'` command to change `oneadmin`'s password.

- You are ready to start the OpenNebula daemons:

```
$ one start
```

**Warning:** Remember to always start OpenNebula as `oneadmin`!

### 1.4.4 Step 4. Verifying the Installation

After OpenNebula is started for the first time, you should check that the commands can connect to the OpenNebula daemon. In the front-end, run as `oneadmin` the command `onevm`:

```
$ onevm list
  ID USER      GROUP      NAME          STAT UCPU    UMEM HOST          TIME
```

If instead of an empty list of VMs you get an error message, then the OpenNebula daemon could not be started properly:

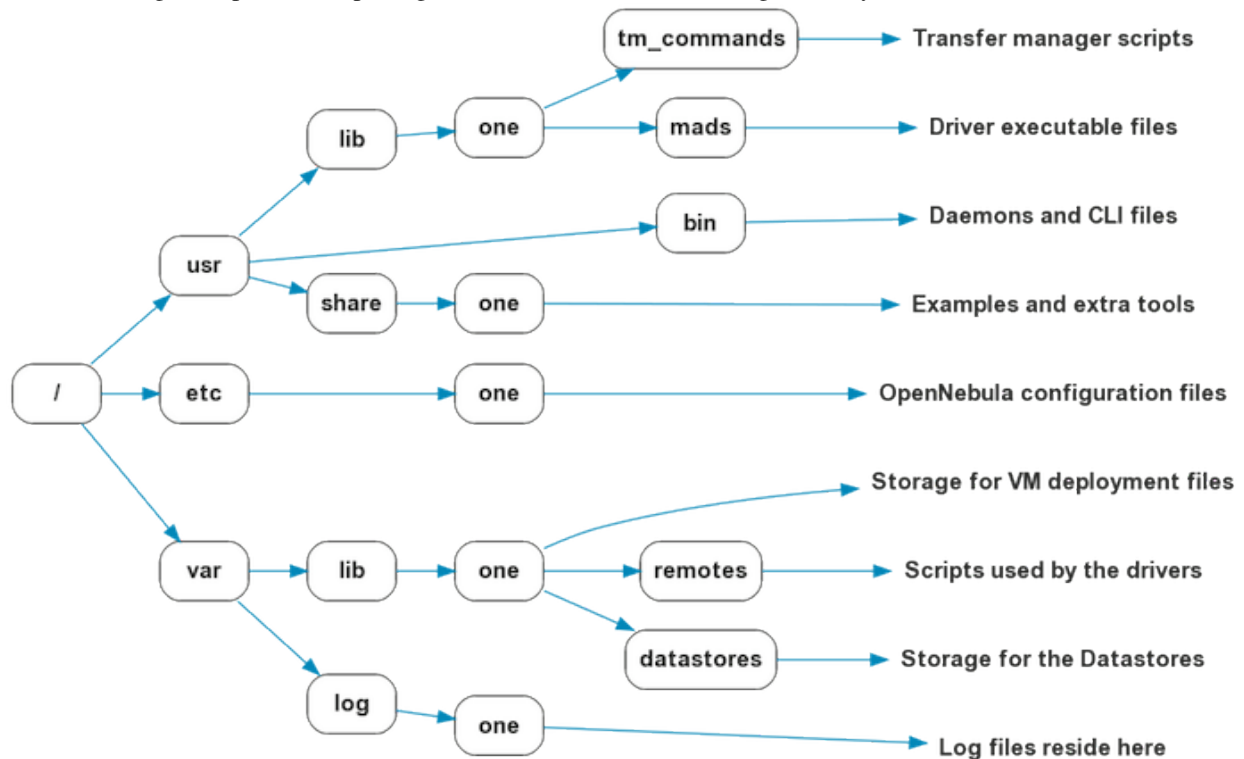
```
$ onevm list
Connection refused - connect(2)
```

The OpenNebula logs are located in `/var/log/one`, you should have at least the files `oned.log` and `sched.log`, the core and scheduler logs. Check `oned.log` for any error messages, marked with `[E]`.

**Warning:** The first time OpenNebula is started, it performs some SQL queries to check if the DB exists and if it needs a bootstrap. You will have two error messages in your log similar to these ones, and can be ignored:

```
[ONE][I]: Checking database version.
[ONE][E]: (..) error: no such table: db_versioning
[ONE][E]: (..) error: no such table: user_pool
[ONE][I]: Bootstrapping OpenNebula database.
```

After installing the OpenNebula packages in the front-end the following directory structure will be used



## 1.4.5 Step 5. Node Installation

### 5.1. Installing on CentOS/RHEL

When the front-end is installed and verified, it is time to install the packages for the nodes if you are using KVM. To install a CentOS/RHEL OpenNebula front-end with packages from our repository, add the repo using the snippet from the previous section and execute the following as root:

```
# sudo yum install opennebula-node-kvm
```

If you are using Xen you can prepare the node with `opennebula-common`:

```
# sudo yum install opennebula-common
```

For further configuration and/or installation of other hypervisors, check their specific guides: *Xen*, *KVM* and *VMware*.



## 5.2. Installing on openSUSE

When the front-end is installed, it is time to install the virtualization nodes. Depending on the chosen hypervisor, check their specific guides: *Xen*, *KVM* and *VMware*.

## 5.3. Installing on Debian/Ubuntu

When the front-end is installed, it is time to install the packages for the nodes if you are using KVM. To install a Debian/Ubuntu OpenNebula front-end with packages from our repository, add the repo as described in the previous section and then install the node package.

```
$ sudo apt-get install opennebula-node
```

For further configuration and/or installation of other hypervisors, check their specific guides: *Xen*, *KVM* and *VMware*.

### 1.4.6 Step 6. Manual Configuration of Unix Accounts

**Warning:** This step can be skipped if you have installed the node/common package for CentOS or Ubuntu, as it has already been taken care of.

The OpenNebula package installation creates a new user and group named `oneadmin` in the front-end. This account will be used to run the OpenNebula services and to do regular administration and maintenance tasks. That means that you eventually need to login as that user or to use the `sudo -u oneadmin` method.

The hosts need also this user created and configured. Make sure you change the `uid` and `gid` by the ones you have in the front-end.

- Get the user and group id of `oneadmin`. This id will be used later to create users in the hosts with the same id. In the **front-end**, execute as `oneadmin`:

```
$ id oneadmin
uid=1001(oneadmin) gid=1001(oneadmin) groups=1001(oneadmin)
```

In this case the user id will be 1001 and group also 1001.

Then log as root **in your hosts** and follow these steps:

- Create the `oneadmin` group. Make sure that its id is the same as in the frontend. In this example 1001:

```
# groupadd --gid 1001 oneadmin
```

- Create the `oneadmin` account, we will use the OpenNebula `var` directory as the home directory for this user.

```
# useradd --uid 1001 -g oneadmin -d /var/lib/one oneadmin
```

**Warning:** You can use any other method to make a common `oneadmin` group and account in the nodes, for example NIS.

### 1.4.7 Step 7. Manual Configuration of Secure Shell Access

You need to create `ssh` keys for the `oneadmin` user and configure the host machines so it can connect to them using `ssh` without need for a password.

Follow these steps in the **front-end**:

- Generate `oneadmin` ssh keys:

```
$ ssh-keygen
```

When prompted for password press enter so the private key is not encrypted.

- Append the public key to `~/.ssh/authorized_keys` to let `oneadmin` user log without the need to type a password.

```
$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

- Many distributions (RHEL/CentOS for example) have permission requirements for the public key authentication to work:

```
$ chmod 700 ~/.ssh/  
$ chmod 600 ~/.ssh/id_dsa.pub  
$ chmod 600 ~/.ssh/id_dsa  
$ chmod 600 ~/.ssh/authorized_keys
```

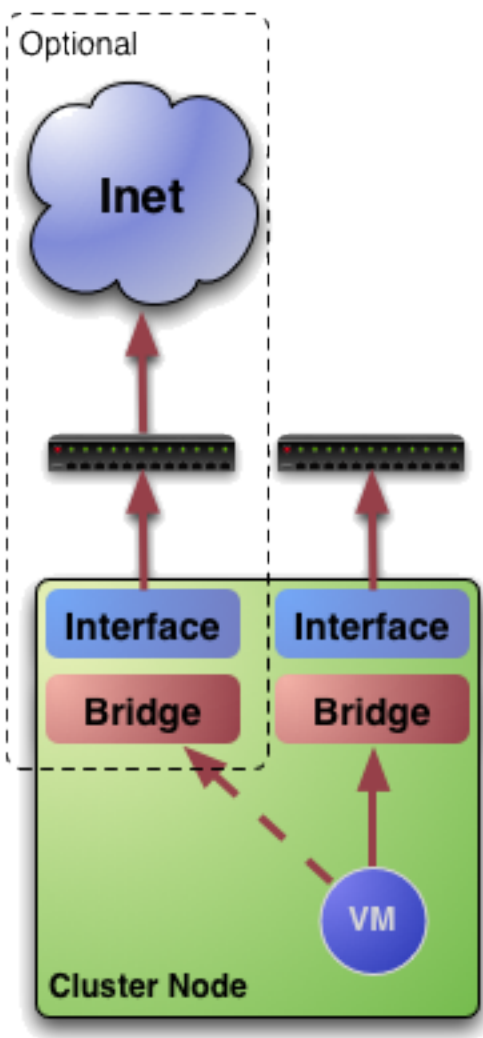
- Tell ssh client to not ask before adding hosts to `known_hosts` file. Also it is a good idea to reduced the connection timeout in case of network problems. This is configured into `~/.ssh/config`, see `man ssh_config` for a complete reference.:

```
$ cat ~/.ssh/config  
ConnectTimeout 5  
Host *  
    StrictHostKeyChecking no
```

- Check that the `sshd` daemon is running in the hosts. Also remove any `Banner` option from the `sshd_config` file in the hosts.
- Finally, Copy the front-end `/var/lib/one/.ssh` directory to each one of the hosts in the same path.

To test your configuration just verify that `oneadmin` can log in the hosts without being prompt for a password.

### 1.4.8 Step 8. Networking Configuration



A network connection is needed by the OpenNebula front-end daemons to access the hosts to manage and monitor the hypervisors; and move image files. It is highly recommended to install a dedicated network for this purpose.

There are various network models (please check the *Networking guide* to find out the networking technologies supported by OpenNebula), but they all have something in common. They rely on network bridges with the same name in all the hosts to connect Virtual Machines to the physical network interfaces.

The simplest network model corresponds to the dummy drivers, where only the network bridges are needed.

For example, a typical host with two physical networks, one for public IP addresses (attached to eth0 NIC) and the other for private virtual LANs (NIC eth1) should have two bridges:

```
$ brctl show
bridge name bridge id          STP enabled interfaces
br0          8000.001e682f02ac no          eth0
br1          8000.001e682f02ad no          eth1
```

## 1.4.9 Step 9. Storage Configuration

OpenNebula uses Datastores to manage VM disk Images. There are two configuration steps needed to perform a basic set up:

- First, you need to configure the **system datastore** to hold images for the running VMs, check the *the System Datastore Guide*, for more details.
- Then you have to setup one or more datastore for the disk images of the VMs, you can find more information on setting up *Filesystem Datastores here*.

The suggested configuration is to use a shared FS, which enables most of OpenNebula VM controlling features. OpenNebula **can work without a Shared FS**, but this will force the deployment to always clone the images and you will only be able to do *cold* migrations.

The simplest way to achieve a shared FS backend for OpenNebula datastores is to export via NFS from the OpenNebula front-end both the `system (/var/lib/one/datastores/0)` and the `images (/var/lib/one/datastores/1)` datastores. They need to be mounted by all the virtualization nodes to be added into the OpenNebula cloud.

## 1.4.10 Step 10. Adding a Node to the OpenNebula Cloud

To add a node to the cloud, there are four needed parameters: name/IP of the host, virtualization, network and information driver. Using the recommended configuration above, and assuming a KVM hypervisor, you can add your host `node01` to OpenNebula in the following fashion (as `oneadmin`, in the front-end):

```
$ onehost create node01 -i kvm -v kvm -n dummy
```

To learn more about the host subsystem, read *this guide*.

## 1.4.11 Step 11. Next steps

Now that you have a fully functional cloud, it is time to start learning how to use it. A good starting point is this *overview of the virtual resource management*.

# 1.5 Glossary

## 1.5.1 OpenNebula Components

- **Front-end:** Machine running the OpenNebula services.
- **Host:** Physical machine running a supported hypervisor. See the *Host subsystem*.
- **Cluster:** Pool of hosts that share datastores and virtual networks. Clusters are used for load balancing, high availability, and high performance computing.
- **Image Repository:** Storage for registered Images. Learn more about the *Storage subsystem*.
- **Sunstone:** OpenNebula web interface. Learn more about *Sunstone*
- **OCCI Service:** Server that enables the management of OpenNebula with OCCI interface. Learn more about *OCCI Service*
- **Self-Service** OpenNebula web interfaced towards the end user. It is implemented by configuring a user view of the Sunstone Portal.

- **EC2 Service:** Server that enables the management of OpenNebula with EC2 interface. Learn more about *EC2 Service*
- **OCA:** OpenNebula Cloud API. It is a set of libraries that ease the communication with the XML-RPC management interface. Learn more about *ruby* and *java* APIs.

### 1.5.2 OpenNebula Resources

- **Template:** Virtual Machine definition. These definitions are managed with the *onetemplate command*.
- **Image:** Virtual Machine disk image, created and managed with the *oneimage command*.
- **Virtual Machine:** Instantiated Template. A Virtual Machine represents one life-cycle, and several Virtual Machines can be created from a single Template. Check out the *VM management guide*.
- **Virtual Network:** A group of IP leases that VMs can use to automatically obtain IP addresses. See the *Networking subsystem*.
- **VDC:** Virtual Data Center, fully-isolated virtual infrastructure environments where a group of users, under the control of the VDC administrator.
- **Zone:** A group of interconnected physical hosts with hypervisors controlled by OpenNebula.

### 1.5.3 OpenNebula Management

- **ACL:** Access Control List. Check *the managing ACL rules guide*.
- **oneadmin:** Special administrative account. See the *Users and Groups guide*.
- **Federation:** Several OpenNebula instances can be *configured as zones*.



## QUICK STARTS

### 2.1 Quickstart: OpenNebula on CentOS 6 and KVM

The purpose of this guide is to provide users with step by step guide to install OpenNebula using CentOS 6 as the operating system and KVM as the hypervisor.

After following this guide, users will have a working OpenNebula with graphical interface (Sunstone), at least one hypervisor (host) and a running virtual machines. This is useful at the time of setting up pilot clouds, to quickly test new features and as base deployment to build a large infrastructure.

Throughout the installation there are two separate roles: **Frontend** and **Nodes**. The Frontend server will execute the OpenNebula services, and the Nodes will be used to execute virtual machines. Please not that **it is possible** to follow this guide with just one host combining both the Frontend and Nodes roles in a single server. However it is recommended execute virtual machines in hosts with virtualization extensions. To test if your host supports virtualization extensions, please run:

```
grep -E 'svm|vmx' /proc/cpuinfo
```

If you don't get any output you probably don't have virtualization extensions supported/enabled in your server.

#### 2.1.1 Package Layout

- `opennebula-server`: OpenNebula Daemons
- `opennebula`: OpenNebula CLI commands
- `opennebula-sunstone`: OpenNebula's web GUI
- `opennebula-java`: OpenNebula Java API
- `opennebula-node-kvm`: Installs dependencies required by OpenNebula in the nodes
- `opennebula-gate`: Send information from Virtual Machines to OpenNebula
- `opennebula-flow`: Manage OpenNebula Services
- `opennebula-context`: Package for OpenNebula Guests

Additionally `opennebula-common` and `opennebula-ruby` exist but they're intended to be used as dependencies. `opennebula-occi`, which is RESTful service to manage the cloud, is included in the `opennebula-sunstone` package.

**Warning:** In order to avoid problems, we recommend to disable SELinux in all the nodes, **Frontend** and **Nodes**:

```
# vi /etc/sysconfig/selinux
...
SELINUX=disabled
...

# setenforce 0
# getenforce
Permissive
```

### 2.1.2 Step 1. Installation in the Frontend

**Warning:** Commands prefixed by # are meant to be run as `root`. Commands prefixed by \$ must be run as `oneadmin`.

#### 1.1. Install the repo

Enable the EPEL repo:

```
# yum install http://dl.fedoraproject.org/pub/epel/6/i386/epel-release-6-8.noarch.rpm
```

Add the OpenNebula repository:

```
# cat << EOT > /etc/yum.repos.d/opennebula.repo
[opennebula]
name=opennebula
baseurl=http://downloads.opennebula.org/repo/CentOS/6/stable/x86_64
enabled=1
gpgcheck=0
EOT
```

#### 1.2. Install the required packages

A complete install of OpenNebula will have at least both `opennebula-server` and `opennebula-sunstone` package:

```
# yum install opennebula-server opennebula-sunstone
```

#### 1.3. Configure and Start the services

There are two main processes that must be started, the main OpenNebula daemon: `oned`, and the graphical user interface: `sunstone`.

Sunstone listens only in the loopback interface by default for security reasons. To change it edit `/etc/one/sunstone-server.conf` and change `:host: 127.0.0.1` to `:host: 0.0.0.0`.

Now we can start the services:

```
# service opennebula start
# service opennebula-sunstone start
```



## 1.4. Configure NFS

**Warning:** Skip this section if you are using a single server for both the frontend and worker node roles.

Export `/var/lib/one/` from the frontend to the worker nodes. To do so add the following to the `/etc/exports` file in the frontend:

```
/var/lib/one/ *(rw, sync, no_subtree_check, root_squash)
```

Refresh the NFS exports by doing:

```
# service rpcbind restart
# service nfs restart
```

## 1.5. Configure SSH Public Key

OpenNebula will need to SSH passwordlessly from any node (including the frontend) to any other node.

Add the following snippet to `~/.ssh/config` as `oneadmin` so it doesn't prompt to add the keys to the `known_hosts` file:

```
# su - oneadmin
$ cat << EOT > ~/.ssh/config
Host *
    StrictHostKeyChecking no
    UserKnownHostsFile /dev/null
EOT
$ chmod 600 ~/.ssh/config
```

## 2.1.3 Step 2. Installation in the Nodes

### 2.1. Install the repo

Add the OpenNebula repository:

```
# cat << EOT > /etc/yum.repos.d/opennebula.repo
[opennebula]
name=opennebula
baseurl=http://downloads.opennebula.org/repo/CentOS/6/stable/x86_64
enabled=1
gpgcheck=0
EOT
```

### 2.2. Install the required packages

```
# yum install opennebula-node-kvm
```

Start the required services:

```
# service messagebus start
# service libvirtd start
```

### 2.3. Configure the Network

**Warning:** Backup all the files that are modified in this section before making changes to them.

You will need to have your main interface, typically `eth0`, connected to a bridge. The name of the bridge should be the same in all nodes.

To do so, substitute `/etc/sysconfig/network-scripts/ifcfg-eth0` with:

```
DEVICE=eth0
BOOTPROTO=none
NM_CONTROLLED=no
ONBOOT=yes
TYPE=Ethernet
BRIDGE=br0
```

And add a new `/etc/sysconfig/network-scripts/ifcfg-br0` file.

If you were using DHCP for your `eth0` interface, use this template:

```
DEVICE=br0
TYPE=Bridge
ONBOOT=yes
BOOTPROTO=dhcp
NM_CONTROLLED=no
```

If you were using a static IP address use this other template:

```
DEVICE=br0
TYPE=Bridge
IPADDR=<YOUR_IPADDRESS>
NETMASK=<YOUR_NETMASK>
ONBOOT=yes
BOOTPROTO=static
NM_CONTROLLED=no
```

After these changes, restart the network:

```
# service network restart
```

### 2.4. Configure NFS

**Warning:** Skip this section if you are using a single server for both the frontend and worker node roles.

Mount the datastores export. Add the following to your `/etc/fstab`:

```
192.168.1.1:/var/lib/one/ /var/lib/one/ nfs soft,intr,rsize=8192,wsize=8192,noauto
```

**Warning:** Replace `192.168.1.1` with the IP of the frontend.

Mount the NFS share:

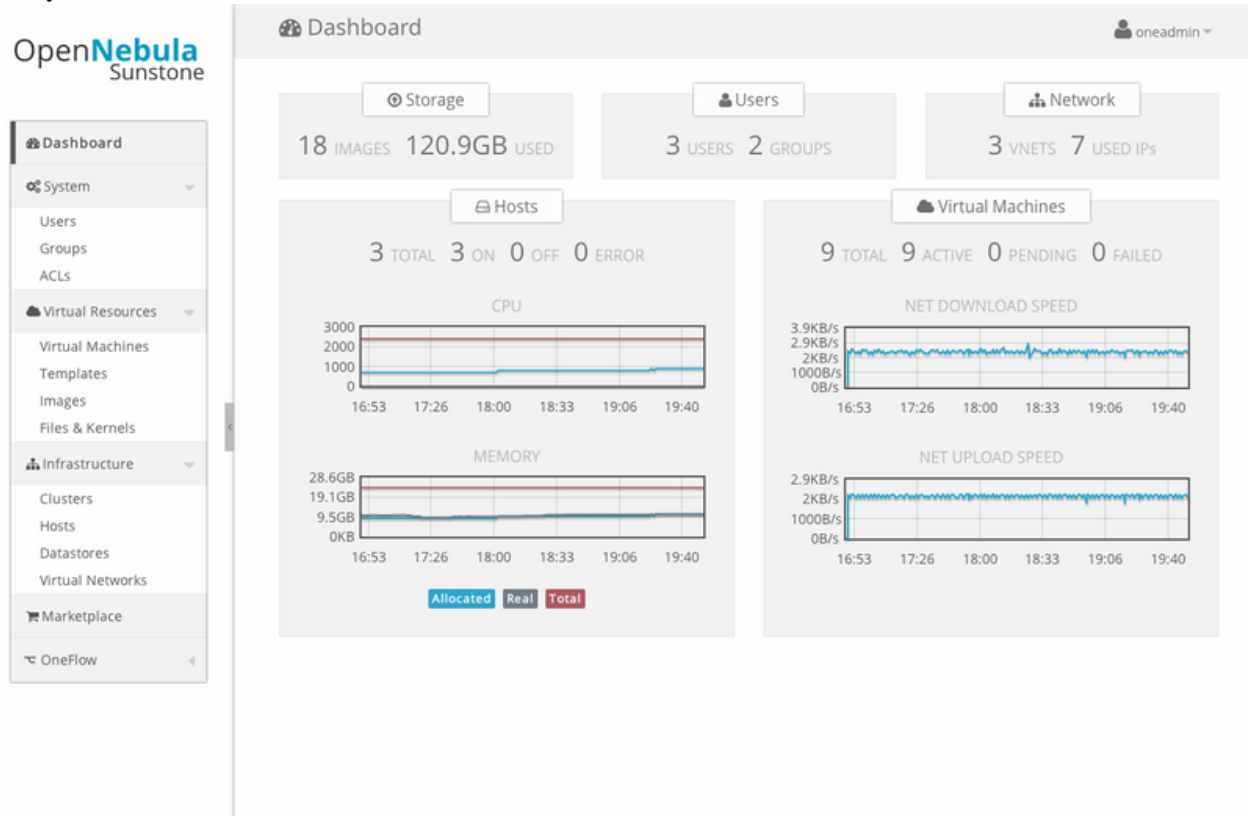
```
# mount /var/lib/one/
```

If the above command fails or hangs, it could be a firewall issue.

## 2.1.4 Step 3. Basic Usage

**Warning:** All the operations in this section can be done using Sunstone instead of the command line. Point your browser to: `http://frontend:9869`.

The default password for the `oneadmin` user can be found in `~/ .one/one_auth` which is randomly generated on every installation.



To interact with OpenNebula, you have to do it from the `oneadmin` account in the frontend. We will assume all the following commands are performed from that account. To login as `oneadmin` execute `su - oneadmin`.

### 3.1. Adding a Host

To start running VMs, you should first register a worker node for OpenNebula.

Issue this command for each one of your nodes. Replace `localhost` with your node's hostname.

```
$ onehost create localhost -i kvm -v kvm -n dummy
```

Run `onehost list` until it's set to on. If it fails you probably have something wrong in your ssh configuration. Take a look at `/var/log/one/oned.log`.

### 3.2. Adding virtual resources

Once it's working you need to create a network, an image and a virtual machine template.

To create networks, we need to create first a network template file `mynetwork.one` that contains:

```
NAME = "private"
TYPE = FIXED

BRIDGE = br0

LEASES = [ IP=192.168.0.100 ]
LEASES = [ IP=192.168.0.101 ]
LEASES = [ IP=192.168.0.102 ]
```

**Warning:** Replace the leases with free IPs in your host's network. You can add any number of leases.

Now we can move ahead and create the resources in OpenNebula:

```
$ onevnet create mynetwork.one

$ oneimage create --name "CentOS-6.5_x86_64" \
  --path "http://appliances.cl2g.com/CentOS-6.5/centos6.5.qcow2.gz" \
  --driver qcow2 \
  --datastore default

$ onetemplate create --name "CentOS-6.5" --cpu 1 --vcpu 1 --memory 512 \
  --arch x86_64 --disk "CentOS-6.5_x86_64" --nic "private" --vnc \
  --ssh
```

You will need to wait until the image is ready to be used. Monitor its state by running `oneimage list`.

In order to dynamically add ssh keys to Virtual Machines we must add our ssh key to the user template, by editing the user template:

```
$ EDITOR=vi oneuser update oneadmin
```

Add a new line like the following to the template:

```
SSH_PUBLIC_KEY="ssh-dss AAAAB3NzaC1kc3MAAACBANBWTQmm4Gt..."
```

Substitute the value above with the output of `cat ~/.ssh/id_dsa.pub`.

### 3.3. Running a Virtual Machine

To run a Virtual Machine, you will need to instantiate a template:

```
$ onetemplate instantiate "CentOS-6.5" --name "My Scratch VM"
```

Execute `onevm list` and watch the virtual machine going from PENDING to PROLOG to RUNNING. If the vm fails, check the reason in the log: `/var/log/one/<VM_ID>/vm.log`.

#### 2.1.5 Further information

- *Planning the Installation*
- *Installing the Software*
- *FAQs. Good for troubleshooting*
- *Main Documentation*

## 2.2 Quickstart: OpenNebula on CentOS 6 and Xen

The purpose of this guide is to provide users with step by step guide to install OpenNebula using CentOS 6 as the operating system and Xen as the hypervisor.

After following this guide, users will have a working OpenNebula with graphical interface (Sunstone), at least one hypervisor (host) and a running virtual machines. This is useful at the time of setting up pilot clouds, to quickly test new features and as base deployment to build a large infrastructure.

Throughout the installation there are two separate roles: **Frontend** and **Nodes**. The Frontend server will execute the OpenNebula services, and the Nodes will be used to execute virtual machines. Please not that **it is possible** to follow this guide with just one host combining both the Frontend and Nodes roles in a single server. However it is recommended execute virtual machines in hosts with virtualization extensions. To test if your host supports virtualization extensions, please run:

```
grep -E 'svm|vmx' /proc/cpuinfo
```

If you don't get any output you probably don't have virtualization extensions supported/enabled in your server.

**Warning:** In order to avoid problems, we recommend to disable SELinux in all the nodes, **Frontend** and **Nodes**:

```
# vi /etc/sysconfig/selinux
...
SELINUX=disabled
...

# setenforce 0
# getenforce
Permissive
```

### 2.2.1 Package Layout

- `opennebula-server`: OpenNebula Daemons
- `opennebula`: OpenNebula CLI commands
- `opennebula-sunstone`: OpenNebula's web GUI
- `opennebula-java`: OpenNebula Java API
- `opennebula-node-kvm`: Installs dependencies required by OpenNebula in the nodes
- `opennebula-gate`: Send information from Virtual Machines to OpenNebula
- `opennebula-flow`: Manage OpenNebula Services
- `opennebula-context`: Package for OpenNebula Guests

Additionally `opennebula-common` and `opennebula-ruby` exist but they're intended to be used as dependencies. `opennebula-occi`, which is RESTful service to manage the cloud, is included in the `opennebula-sunstone` package.

### 2.2.2 Step 1. Installation in the Frontend

**Warning:** Commands prefixed by # are meant to be run as `root`. Commands prefixed by \$ must be run as `oneadmin`.

### 1.1. Install the repo

Enable the EPEL repo:

```
# yum install http://dl.fedoraproject.org/pub/epel/6/i386/epel-release-6-8.noarch.rpm
```

Add the OpenNebula repository:

```
# cat << EOT > /etc/yum.repos.d/opennebula.repo
[opennebula]
name=opennebula
baseurl=http://downloads.opennebula.org/repo/CentOS/6/stable/x86_64
enabled=1
gpgcheck=0
EOT
```

### 1.2. Install the required packages

A complete install of OpenNebula will have at least both `opennebula-server` and `opennebula-sunstone` package:

```
# yum install opennebula-server opennebula-sunstone
```

### 1.3. Configure and Start the services

There are two main processes that must be started, the main OpenNebula daemon: `oned`, and the graphical user interface: `sunstone`.

Sunstone listens only in the loopback interface by default for security reasons. To change it edit `/etc/one/sunstone-server.conf` and change `:host: 127.0.0.1` to `:host: 0.0.0.0`.

Now we can start the services:

```
# service opennebula start
# service opennebula-sunstone start
```

### 1.4. Configure NFS

**Warning:** Skip this section if you are using a single server for both the frontend and worker node roles.

Export `/var/lib/one/` from the frontend to the worker nodes. To do so add the following to the `/etc/exports` file in the frontend:

```
/var/lib/one/ *(rw, sync, no_subtree_check, root_squash)
```

Refresh the NFS exports by doing:

```
# service rpcbind restart
# service nfs restart
```

## 1.5. Configure SSH Public Key

OpenNebula will need to SSH passwordlessly from any node (including the frontend) to any other node.

Add the following snippet to `~/.ssh/config` as `oneadmin` so it doesn't prompt to add the keys to the `known_hosts` file:

```
# su - oneadmin
$ cat << EOT > ~/.ssh/config
Host *
    StrictHostKeyChecking no
    UserKnownHostsFile /dev/null
EOT
$ chmod 600 ~/.ssh/config
```

## 2.2.3 Step 2. Installation in the Nodes

**Warning:** The process to install Xen might change in the future. Please refer to the CentOS documentation on [Xen4 CentOS6 QuickStart](#) if any of the following steps do not work.

### 2.1. Install the repo

Add the CentOS Xen repo:

```
# yum install centos-release-xen
```

Add the OpenNebula repository:

```
# cat << EOT > /etc/yum.repos.d/opennebula.repo
[opennebula]
name=opennebula
baseurl=http://downloads.opennebula.org/repo/CentOS/6/stable/x86_64
enabled=1
gpgcheck=0
EOT
```

### 2.2. Install the required packages

```
# yum install opennebula-common xen nfs-utils ruby
```

Enable the Xen kernel by doing:

```
# /usr/bin/grub-bootxen.sh
```

Disable `xend` since it is a deprecated interface:

```
# chkconfig xend off
```

Now you must **reboot** the system in order to start with a Xen kernel.

## 2.3. Configure the Network

**Warning:** Backup all the files that are modified in this section before making changes to them.

You will need to have your main interface, typically `eth0`, connected to a bridge. The name of the bridge should be the same in all nodes.

To do so, substitute `/etc/sysconfig/network-scripts/ifcfg-eth0` with:

```
DEVICE=eth0
BOOTPROTO=none
NM_CONTROLLED=no
ONBOOT=yes
TYPE=Ethernet
BRIDGE=br0
```

And add a new `/etc/sysconfig/network-scripts/ifcfg-br0` file.

If you were using DHCP for your `eth0` interface, use this template:

```
DEVICE=br0
TYPE=Bridge
ONBOOT=yes
BOOTPROTO=dhcp
NM_CONTROLLED=no
```

If you were using a static IP address use this other template:

```
DEVICE=br0
TYPE=Bridge
IPADDR=<YOUR_IPADDRESS>
NETMASK=<YOUR_NETMASK>
ONBOOT=yes
BOOTPROTO=static
NM_CONTROLLED=no
```

After these changes, restart the network:

```
# service network restart
```

## 2.4. Configure NFS

**Warning:** Skip this section if you are using a single server for both the frontend and worker node roles.

Mount the datastore export. Add the following to your `/etc/fstab`:

```
192.168.1.1:/var/lib/one/ /var/lib/one/ nfs soft,intr,rsize=8192,wsiz=8192,noauto
```

**Warning:** Replace `192.168.1.1` with the IP of the frontend.

Mount the NFS share:

```
# mount /var/lib/one/
```

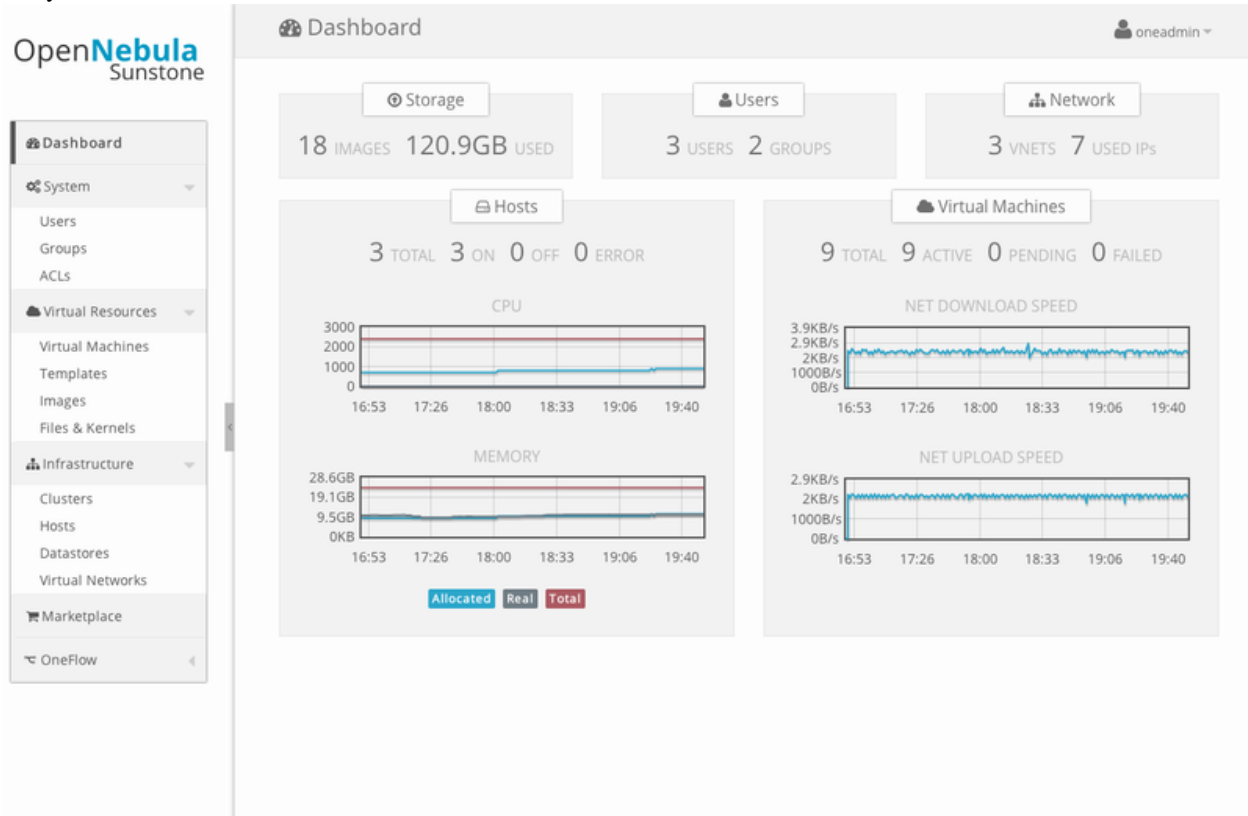
If the above command fails or hangs, it could be a firewall issue.



## 2.2.4 Step 3. Basic Usage

**Warning:** All the operations in this section can be done using Sunstone instead of the command line. Point your browser to: `http://frontend:9869`.

The default password for the `oneadmin` user can be found in `~/ .one/one_auth` which is randomly generated on every installation.



To interact with OpenNebula, you have to do it from the `oneadmin` account in the frontend. We will assume all the following commands are performed from that account. To login as `oneadmin` execute `su - oneadmin`.

### 3.1. Adding a Host

To start running VMs, you should first register a worker node for OpenNebula.

Issue this command for each one of your nodes. Replace `localhost` with your node's hostname.

```
$ onehost create localhost -i xen -v xen -n dummy
```

Run `onehost list` until it's set to on. If it fails you probably have something wrong in your ssh configuration. Take a look at `/var/log/one/oned.log`.

### 3.2. Adding virtual resources

Once it's working you need to create a network, an image and a virtual machine template.

To create networks, we need to create first a network template file `mynetwork.one` that contains:

```
NAME = "private"
TYPE = FIXED

BRIDGE = br0

LEASES = [ IP=192.168.0.100 ]
LEASES = [ IP=192.168.0.101 ]
LEASES = [ IP=192.168.0.102 ]
```

**Warning:** Replace the leases with free IPs in your host's network. You can add any number of leases.

Now we can move ahead and create the resources in OpenNebula:

```
$ onevnet create mynetwork.one

    --path "http://appliances.cl2g.com/CentOS-6.5/centos6.5.qcow2.gz" \
$ oneimage create --name "CentOS-6.5_x86_64" \
    --path "http://172.16.77.1/vm-images/CentOS65.qcow2" \
    --driver qcow2 \
    --datastore default

$ onetemplate create --name "CentOS-6.5" --cpu 1 --vcpu 1--memory 512 \
    --arch x86_64 --disk "CentOS-6.5_x86_64" --nic "private" --vnc \
    --ssh
```

You will need to wait until the image is ready to be used. Monitor its state by running `oneimage list`.

We must specify the desired bootloader to the template we just created. To do so execute the following command:

```
$ EDITOR=vi onetemplate update CentOS-6.5
```

Add a new line to the OS section of the template that specifies the bootloader:

```
OS=[
  BOOTLOADER = "pygrub",
  ARCH="x86_64" ]
```

In order to dynamically add ssh keys to Virtual Machines we must add our ssh key to the user template, by editing the user template:

```
$ EDITOR=vi oneuser update oneadmin
```

Add a new line like the following to the template:

```
SSH_PUBLIC_KEY="ssh-dss AAAAB3NzaC1kc3MAAACBANBWTQmm4Gt..."
```

Substitute the value above with the output of `cat ~/.ssh/id_dsa.pub`.

### 3.3. Running a Virtual Machine

To run a Virtual Machine, you will need to instantiate a template:

```
$ onetemplate instantiate "CentOS-6.5" --name "My Scratch VM"
```

Execute `onevm list` and watch the virtual machine going from PENDING to PROLOG to RUNNING. If the vm fails, check the reason in the log: `/var/log/one/<VM_ID>/vm.log`.

## 2.2.5 Further information

- *Planning the Installation*
- *Installing the Software*
- FAQs. Good for troubleshooting
- *Main Documentation*

## 2.3 Quickstart: OpenNebula on CentOS 6 and ESX 5.x

This guide aids in the process of quickly get a VMware-based OpenNebula cloud up and running on CentOS 6. After following this guide, users will have a working OpenNebula with graphical interface (Sunstone), at least one hypervisor (host) and a running virtual machines. This is useful at the time of setting up pilot clouds, to quickly test new features and as base deployment to build a large infrastructure.

Throughout the installation there are two separate roles: Frontend and Virtualization Nodes. The Frontend server will execute the OpenNebula services, and the Nodes will be used to execute virtual machines.

### 2.3.1 Package Layout

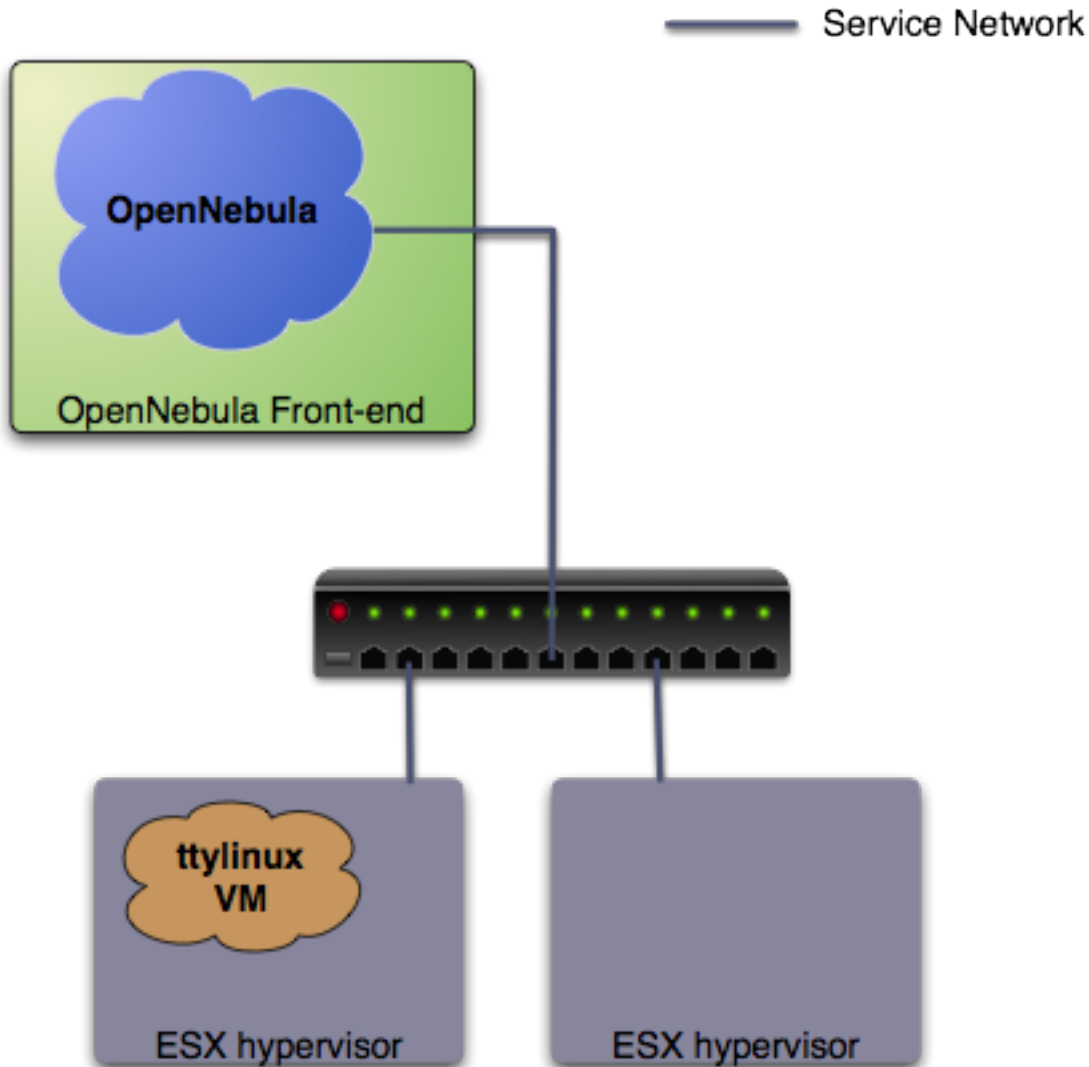
- `opennebula-server`: OpenNebula Daemons
- `opennebula`: OpenNebula CLI commands
- `opennebula-sunstone`: OpenNebula's web GUI
- `opennebula-java`: OpenNebula Java API
- `opennebula-node-kvm`: Installs dependencies required by OpenNebula in the nodes
- `opennebula-gate`: Send information from Virtual Machines to OpenNebula
- `opennebula-flow`: Manage OpenNebula Services
- `opennebula-context`: Package for OpenNebula Guests

Additionally `opennebula-common` and `opennebula-ruby` exist but they're intended to be used as dependencies. `opennebula-occi`, which is RESTful service to manage the cloud, is included in the `opennebula-sunstone` package.

### 2.3.2 Step 1. Infrastructure Set-up

The infrastructure needs to be set up in a similar fashion as the one depicted in the figure.

**Warning:** A ESX version 5.0 was used to create this guide. This guide may be useful for other versions of ESX, although the configuration (and therefore your mileage) may vary.



In this guide it is assumed that at least two physical servers are available, one to host the OpenNebula front-end and one to be used as a ESX virtualization node (this is the one you need to configure in the following section). The figure depicts one more ESX host, to show that the pilot cloud is ready to grow just by adding more virtualization nodes.

### Front-End

- **Operating System:** Centos 6.4
- **Required extra repository:** EPEL
- **Required packages:** NFS, libvirt

Let's install the repository (as root) and required packages

```
# yum install http://dl.fedoraproject.org/pub/epel/6/i386/epel-release-6-8.noarch.rpm
# cat << EOT > /etc/yum.repos.d/opennebula.repo [opennebula] name=opennebula
baseurl=http://downloads.opennebula.org/repo/CentOS/6/stable/x86_64 enabled=1 gpgcheck=0 EOT
# yum install nfs-utils nfs-utils-lib libvirt
```

### Virtualization node

- **Operating System:** ESX 5.0

**Warning:** The ESX hosts needs to be configured. To achieve this, you will need access to a Windows machine with the Virtual Infrastructure Client (vSphere client) install. The VI client can be downloaded from the ESX node, by pointing a browser to its IP.

**Warning:** The ESX hosts need to be properly licensed, with write access to the exported API (as the Evaluation license does). More information on valid licenses [here](#).

### 2.3.3 Step 2. OpenNebula Front-end Set-up

#### 2.1 OpenNebula installation

With the repository added, installing OpenNebula is straightforward (as root):

```
# yum install opennebula-server opennebula-sunstone
```

**Warning:** Do not start OpenNebula at this point, some pre configuration needs to be done. Starting OpenNebula is not due until [here](#).

Find out the uid and gid of oneadmin, we will need it for the next section:

```
$ id oneadmin
uid=499(oneadmin) gid=498(oneadmin)
```

In order to avoid problems, we recommend to disable SELinux for the pilot cloud front-end (sometimes it is the root of all evil):

```
# vi /etc/sysconfig/selinux
...
SELINUX=disabled
...

# setenforce 0
# getenforce
Permissive
```

#### 2.2 NFS configuration

The front-end needs to export via NFS two datastores (the system and the images datastore). This is required just so the ESX has access to two different datastores, and this guides uses NFS exported from the front-end to achieve this. This can be seamlessly replaced with two iSCSI backed datastores or even two local hard disks. In any case, we will use the ‘vmfs’ drivers to manage both datastores, independently of the storage backend. See the *VMFS Datastore Guide* for more details.

Let’s configure the NFS server. You will need to allow incoming connections, here we will simply stop iptables (as root):

```
$ sudo su - oneadmin

$ sudo vi /etc/exports
/var/lib/one/datastores/0 *(rw, sync, no_subtree_check, root_squash, anonuid=499, anongid=498)
/var/lib/one/datastores/1 *(rw, sync, no_subtree_check, root_squash, anonuid=499, anongid=498)

$ sudo service iptables stop
$ sudo service nfs start
```

```
$ sudo exportfs -a
```

**Warning:** Make sure **anonuid** and **anongid** are set to the oneadmin uid and gid.

### 2.3 Networking

There must be connection between the front-end and the ESX node. This can be tested with the ping command:

```
$ ping <esx-ip>
```

### 2.3.4 Step 3. VMware Virtualization Node Set-up

This is probably the step that involves more work to get the pilot cloud up and running, but it is crucial to ensure its correct functioning. The ESX that is going to be used as worker node needs the following steps:

#### 3.1 Creation of a oneadmin user

With the VI client connected to the ESX host, go to the “local Users & Groups” and add a new user like shown in the figure (**the UID is important, it needs to match the one of the front-end.**). Make sure that you are selecting the “Grant shell to this user” checkbox, and write down the password you enter.

**Edit User - oneadmin**

User Information

Login:  UID:

User Name:

User name and UID are optional

Change password

Password:

Confirm:

Shell Access

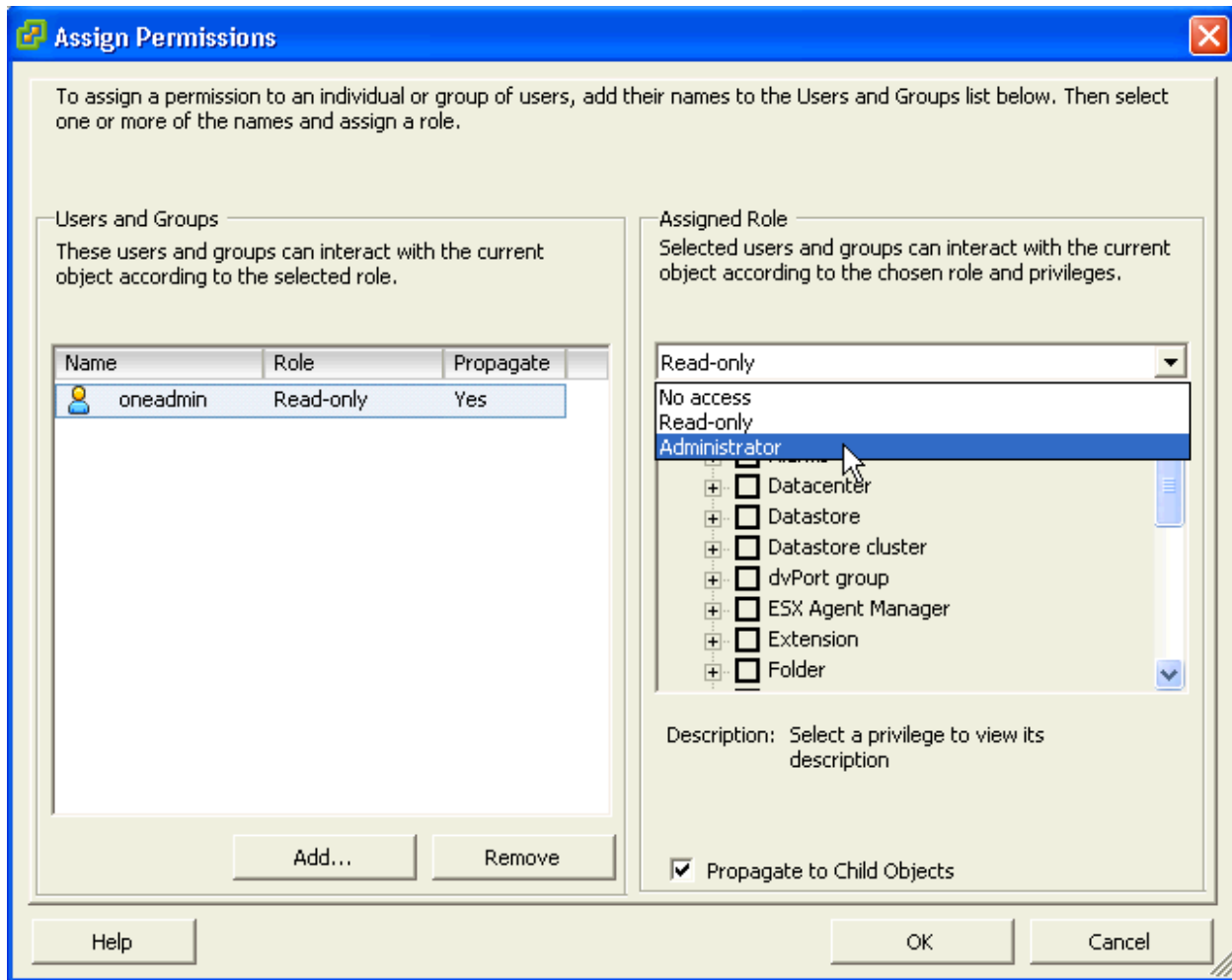
Grant shell access to this user

Group membership

Group:

- daemon
- nfsnobody
- root
- users

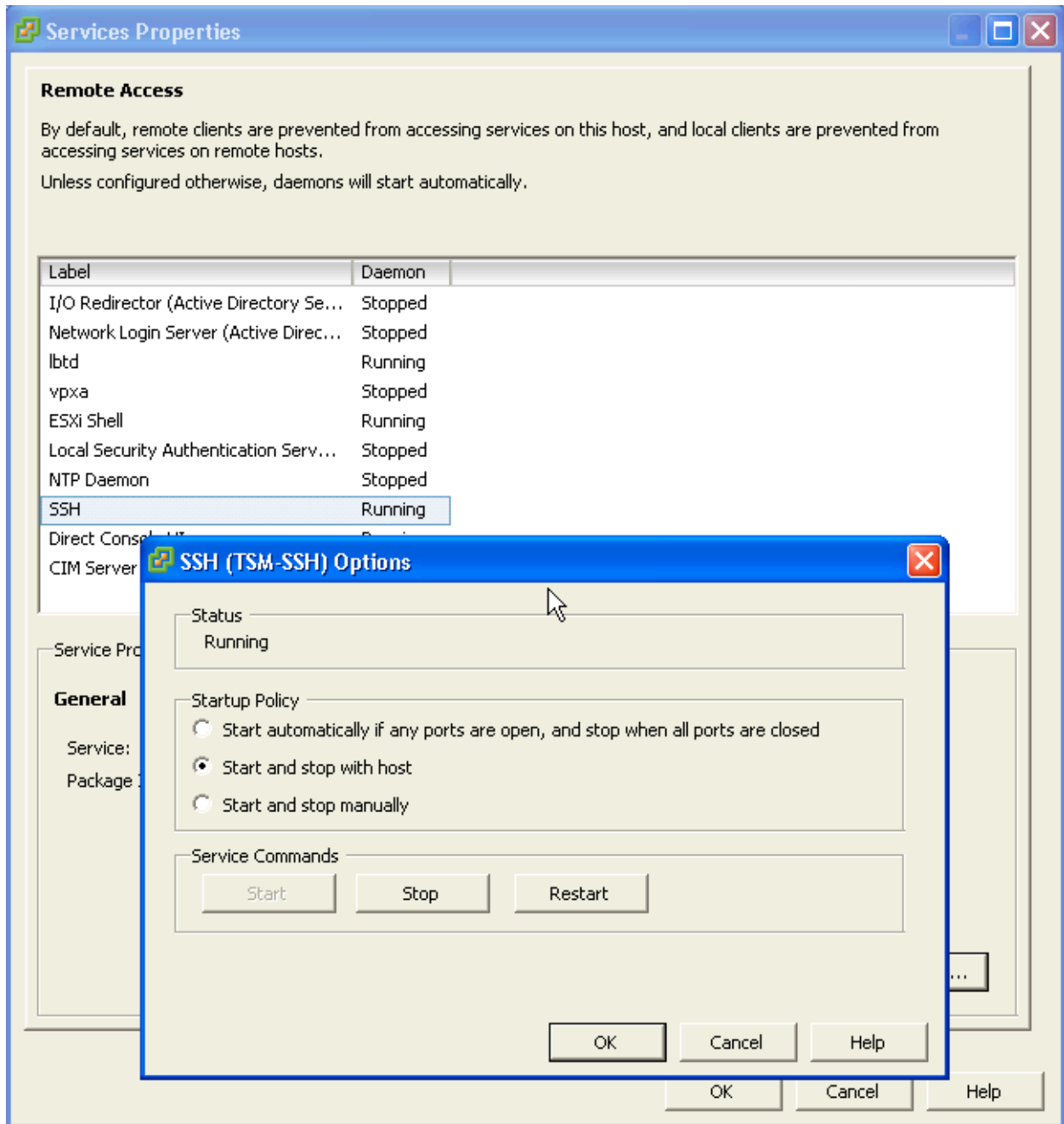
Afterwards, go to the “Permissions” tab and assign the “Administrator” Role to oneadmin (right click → Add Permission...).



### 3.2 Grant ssh access

Again in the VI client go to Configuration → Security Profile → Services Properties (Upper right). Click on the SSH label, select the “Options” button, and then “Start”. You can set it to start and stop with the host, as seen on the picture.





Then the following needs to be done:

- Connect via ssh to the OpenNebula front-end as the oneadmin user. Copy the output of the following command to the clipboard:

```
$ ssh-keygen
Enter an empty passphrase

$ cat .ssh/id_rsa.pub
```

- Connect via ssh to the ESX worker node (as oneadmin). Run the following from the front-end:

```
$ ssh <esx-ip>
  Enter the password you set in the step 3.1

$ su

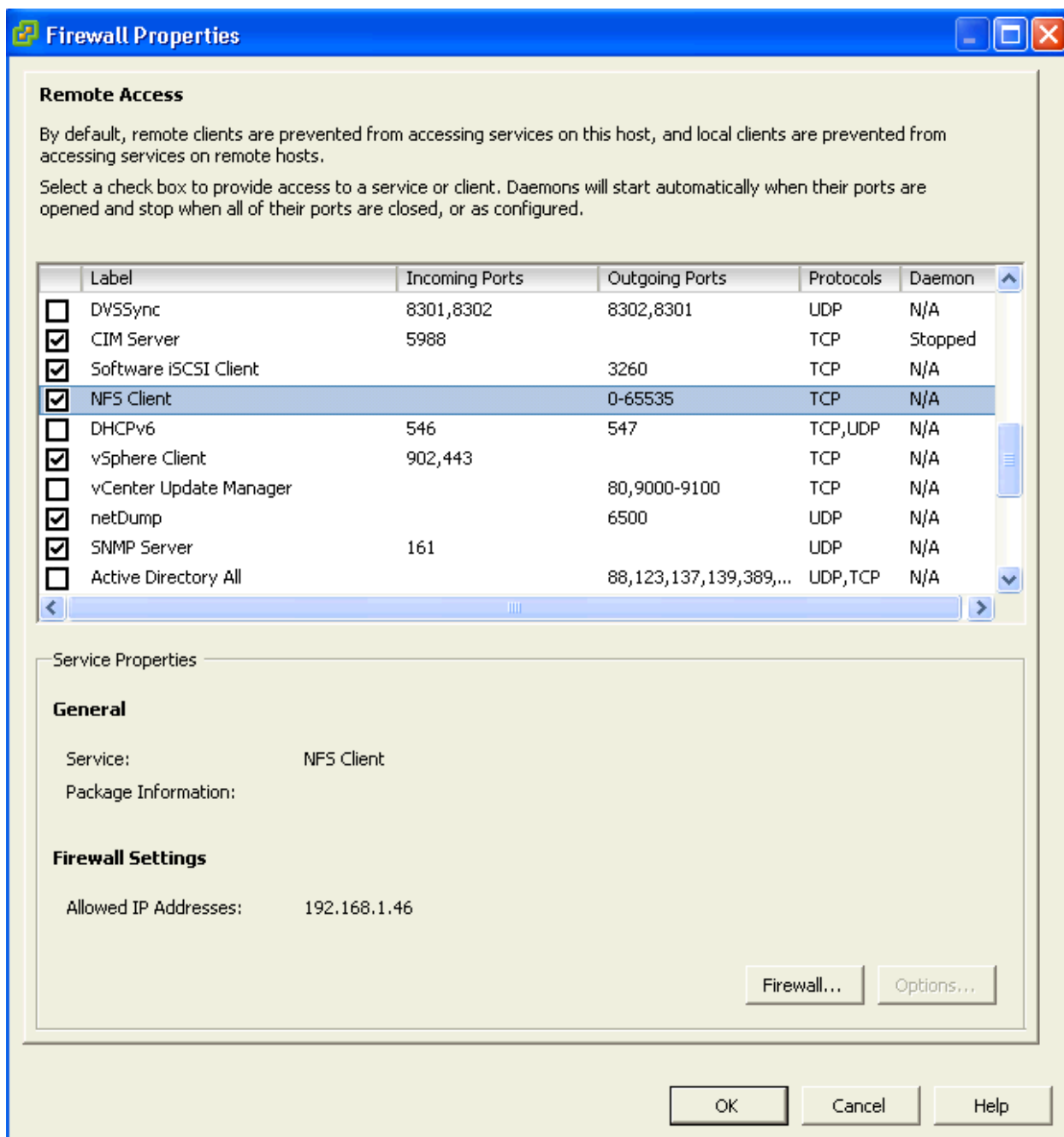
# mkdir /etc/ssh/keys-oneadmin
# chmod 755 /etc/ssh/keys-oneadmin
# vi /etc/ssh/keys-oneadmin/authorized_keys
paste here the contents of oneadmin's id_rsa.pub and exit vi
# chown oneadmin /etc/ssh/keys-oneadmin/authorized_keys
# chmod 600 /etc/ssh/keys-oneadmin/authorized_keys
# chmod +s /sbin/vmkfstools /bin/vim-cmd      # This is needed to create volatile disks
```

- Now oneadmin should be able to ssh without been prompted for a password

```
$ ssh <esx-ip>
```

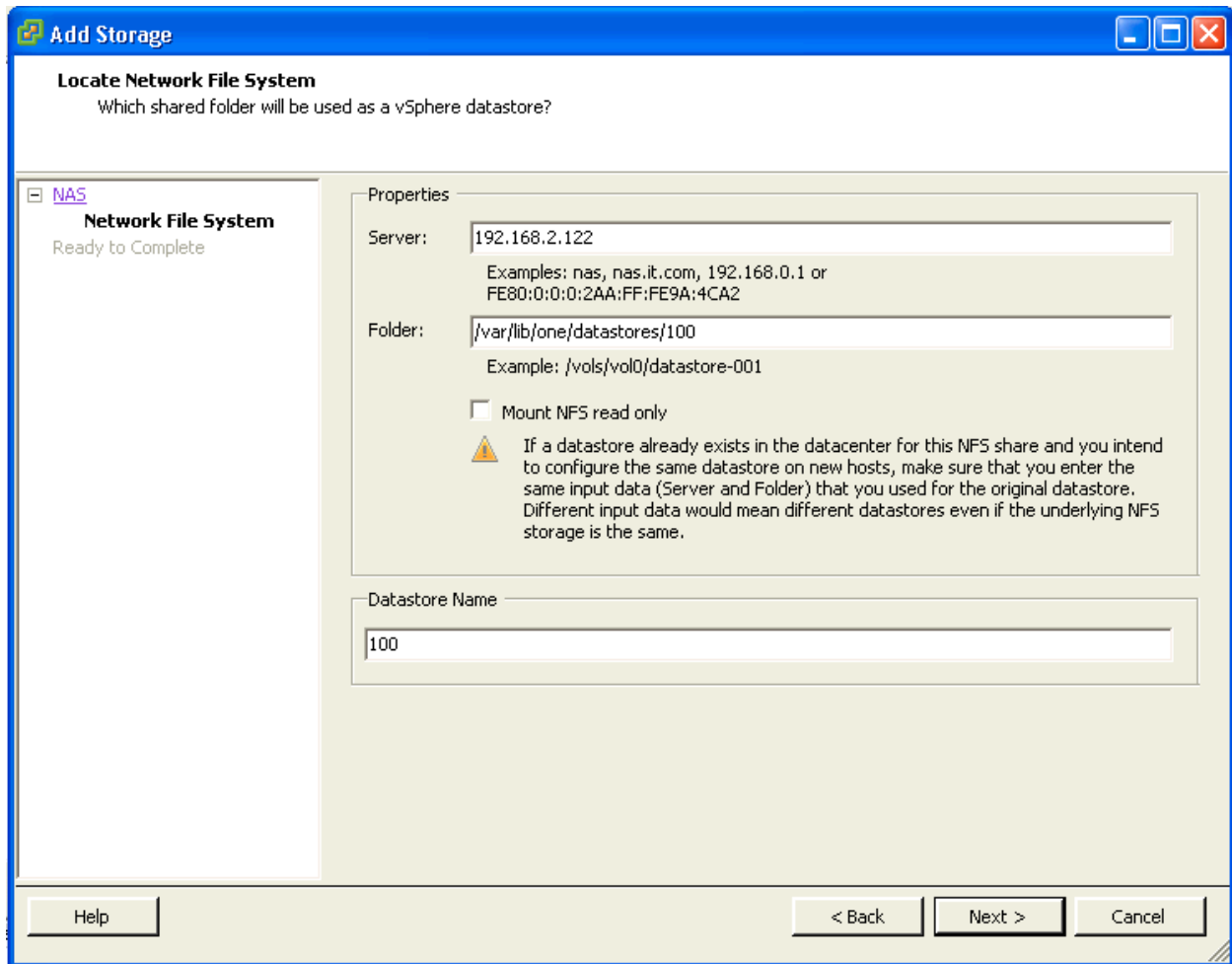
### 3.3 Mount datastores

We need now to mount the two datastores exported by default by the OpenNebula front-end. First, you need to make sure that the firewall will allow the NFS Client to connect to the front-end. Go to Configuration → Software → Security Profile, and enable the row NFS Client:



Again in the VI client, go to Configuration → Storage → Add Storage (Upper right). We need to add two datastores (**0** and **1**). The picture shows the details for the datastore **100**, to add datastore **0** and **1** simply change the reference from **100** to **0** and then **1** in the Folder and Datastore Name textboxes.

Please note that the IP of the server displayed may not correspond with your value, which has to be the IP your front-end uses to connect to the ESX.



The paths to be used as input:

```
/var/lib/one/datastores/0
```

```
/var/lib/one/datastores/1
```

More info on *datastores* and different possible configurations.

### 3.4 Configure VNC

Open an ssh connection to the ESX as root, and:

```
# cd /etc/vmware
# chown -R root firewall/
# chmod 7777 firewall/
# cd firewall/
# chmod 7777 service.xml
```

Add the following to `/etc/vmware/firewall/service.xml`

```
# vi /etc/vmware/firewall/service.xml
```

**Warning:** The service id must be the last service id+1. It will depend on your firewall configuration

```

<!-- VNC -->
<service id="0033">
  <id>VNC</id>
  <rule id='0000'>
    <direction>outbound</direction>
    <protocol>tcp</protocol>
    <porttype>dst</porttype>
    <port>
      <begin>5800</begin>
      <end>5999</end>
    </port>
  </rule>
  <rule id='0001'>
    <direction>inbound</direction>
    <protocol>tcp</protocol>
    <porttype>dst</porttype>
    <port>
      <begin>5800</begin>
      <end>5999</end>
    </port>
  </rule>
  <enabled>true</enabled>
  <required>>false</required>
</service>

```

Refresh the firewall

```

# /sbin/esxcli network firewall refresh
# /sbin/esxcli network firewall ruleset list

```

## 2.3.5 Step 4. OpenNebula Configuration

Let's configure OpenNebula in the front-end to allow it to use the ESX hypervisor. The following must be run under the "oneadmin" account.

### 4.1 Configure oned and Sunstone

Edit `/etc/one/oned.conf` with "sudo" and uncomment the following:

```

#*****
# DataStore Configuration
#*****
# DATASTORE_LOCATION: *Default* Path for Datastores in the hosts. It IS the
# same for all the hosts in the cluster. DATASTORE_LOCATION IS ONLY FOR THE
# HOSTS AND *NOT* THE FRONT-END. It defaults to /var/lib/one/datastores (or
# $ONE_LOCATION/var/datastores in self-contained mode)
#
# DATASTORE_BASE_PATH: This is the base path for the SOURCE attribute of
# the images registered in a Datastore. This is a default value, that can be
# changed when the datastore is created.
#*****

DATASTORE_LOCATION = /vmfs/volumes

DATASTORE_BASE_PATH = /vmfs/volumes

#-----
# VMware Information Driver Manager Configuration

```

```
#-----  
IM_MAD = [  
    name      = "vmware",  
    executable = "one_im_sh",  
    arguments  = "-c -t 15 -r 0 vmware" ]  
  
#-----  
# VMware Virtualization Driver Manager Configuration  
#-----  
VM_MAD = [  
    name      = "vmware",  
    executable = "one_vmm_sh",  
    arguments  = "-t 15 -r 0 vmware -s sh",  
    default    = "vmm_exec/vmm_exec_vmware.conf",  
    type       = "vmware" ]
```

Edit `/etc/one/sunstone-server.conf` with “sudo” and allow incoming connections from any IP:

```
sudo vi /etc/one/sunstone-server.conf  
  
# Server Configuration  
#  
:host: 0.0.0.0  
:port: 9869
```

### 4.2 Add the ESX credentials

```
$ sudo vi /etc/one/vmwarerc  
<Add the ESX oneadmin password, set in section 3.1>  
# Username and password of the VMware hypervisor  
:username: "oneadmin"  
:password: "password"
```

<b>Warning:</b> Do not edit <code>:libvirt_uri:</code> , the HOST placeholder is needed by the drivers
--

### 4.3 Start OpenNebula

Start OpenNebula and Sunstone as **oneadmin**

```
$ one start  
$ sunstone-server start
```

If no error message is shown, then everything went smooth!

### 4.4 Configure physical resources

Let’s configure both system and image datastores:

```
$ onedatastore update 0  
SHARED="YES"  
TM_MAD="vmfs"  
TYPE="SYSTEM_DS"  
BASE_PATH="/vmfs/volumes"  
  
$ onedatastore update 1  
TM_MAD="vmfs"  
DS_MAD="vmfs"  
BASE_PATH="/vmfs/volumes"  
CLONE_TARGET="SYSTEM"  
DISK_TYPE="FILE"  
LN_TARGET="NONE"
```

```
TYPE="IMAGE_DS"
BRIDGE_LIST="esx-ip"

$ onedatastore chmod 1 644
```

And the ESX Host:

```
$ onehost create <esx-ip> -i vmware -v vmware -n dummy
```

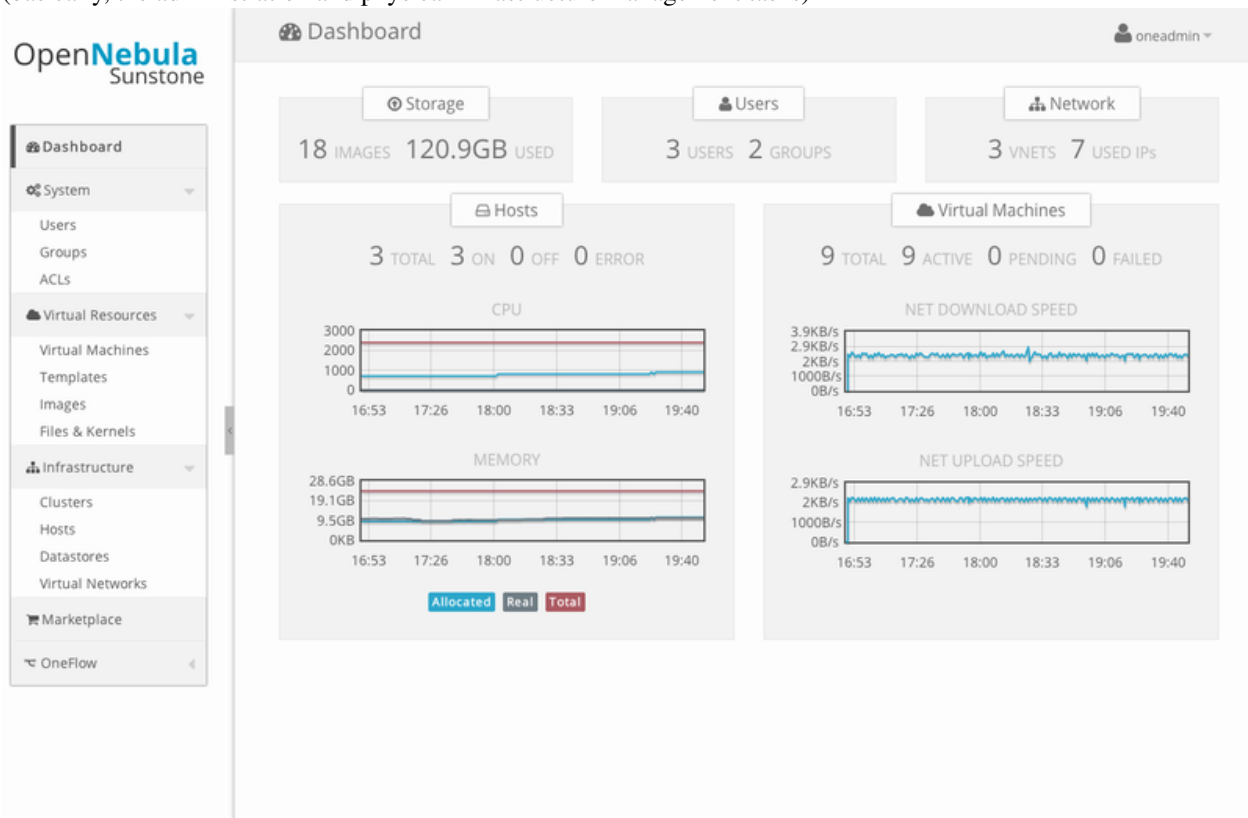
#### 4.5 Create a regular cloud user

```
$ oneuser create oneuser <mypassword>
```

### 2.3.6 Step 5. Using the Cloud through Sunstone

Ok, so now that everything is in place, let's start using your brand new OpenNebula cloud! Use your browser to access Sunstone. The URL would be `http://@IP-of-the-front-end@:9869`

Once you introduce the credentials for the “oneuser” user (with the chosen password in the previous section) you will get to see the Sunstone dashboard. You can also log in as “oneadmin”, you will notice the access to more functionality (basically, the administration and physical infrastructure management tasks)



It is time to launch our first VM. Let's use one of the pre created appliances found in the [marketplace](#).

Log in as “oneuser”, go to the Marketplace tab in Sunstone (in the left menu), and select the “ttylinux-VMware” row. Click on the “Import to local infrastructure” button in the upper right, and set the new image a name (use “ttylinux - VMware”) and place it in the “VMwareImages” datastore. If you go to the Virtual Resources/Image tab, you will see that the new Image will eventually change its status from LOCKED to READY.

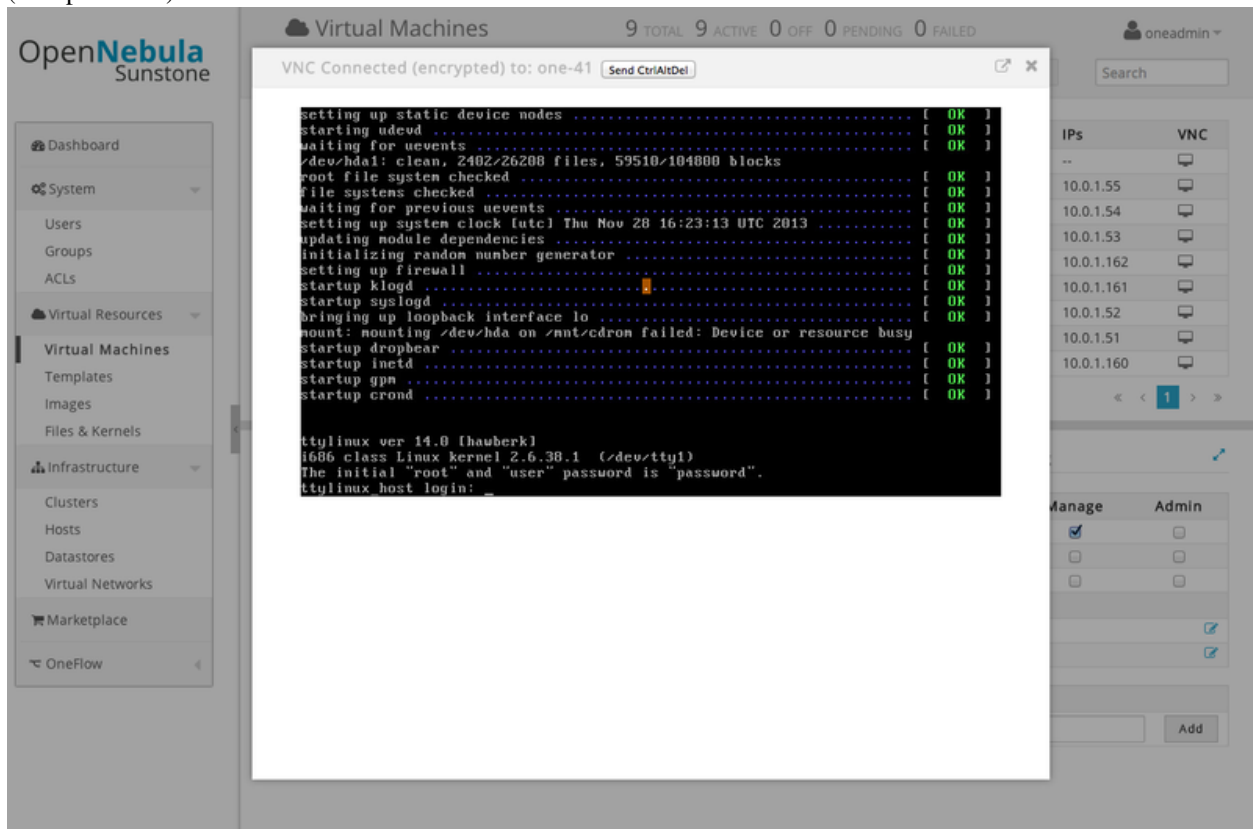
Now we need to create a template that uses this image. Go to the Virtual Resources/Templates tab, click on "+Create" and follow the wizard, or use the "Advanced mode" tab of the wizard to paste the following:

```
NAME      = "ttylinux"
CPU       = "1"
MEMORY   = "512"

DISK      = [
    IMAGE      = "ttylinux - VMware",
    IMAGE_UNAME = "oneuser"
]

GRAPHICS = [
    TYPE      = "vnc",
    LISTEN    = "0.0.0.0"
]
```

Select the newly created template and click on the Instantiate button. You can now proceed to the "Virtual Machines" tab. Once the VM is in state RUNNING you can click on the VNC icon and you should see the ttylinux login (root/password).



Please note that the minimal ttylinux VM does not come with the VMware Tools, and cannot be gracefully shutdown. Use the "Cancel" action instead.

And that's it! You have now a fully functional pilot cloud. You can now create your own virtual machines, or import other appliances from the marketplace, like [Centos 6.2](#).

Enjoy!



### 2.3.7 Step 6. Next Steps

- Follow the *VMware Virtualization Driver Guide* for the complete installation and tuning reference, and how to enable the disk attach/detach functionality, and vMotion live migration.
- OpenNebula can use *VMware native networks* to provide network isolation through VLAN tagging.

**Warning:** Did we miss something? Please [let us know!](#)

## 2.4 Quickstart: OpenNebula on Ubuntu 14.04 and KVM

The purpose of this guide is to provide users with step by step guide to install OpenNebula using Ubuntu 14.04 as the operating system and KVM as the hypervisor.

After following this guide, users will have a working OpenNebula with graphical interface (Sunstone), at least one hypervisor (host) and a running virtual machines. This is useful at the time of setting up pilot clouds, to quickly test new features and as base deployment to build a large informations<sup>o</sup>structure.

Throughout the installation there are two separate roles: **Frontend** and **Nodes**. The Frontend server will execute the OpenNebula services, and the Nodes will be used to execute virtual machines. Please not that **it is possible** to follow this guide with just one host combining both the Frontend and Nodes roles in a single server. However it is recommended execute virtual machines in hosts with virtualization extensions. To test if your host supports virtualization extensions, please run:

```
grep -E 'svm|vmx' /proc/cpuinfo
```

If you don't get any output you probably don't have virtualization extensions supported/enabled in your server.

### 2.4.1 Package Layout

- **opennebula-common:** Provides the user and common files
- **libopennebula-ruby:** All ruby libraries
- **opennebula-node:** Prepares a node as an opennebula-node
- **opennebula-sunstone:** OpenNebula Sunstone Web Interface
- **opennebula-tools:** Command Line interface
- **opennebula-gate:** Gate server that enables communication between VMs and OpenNebula
- **opennebula-flow:** Manages services and elasticity
- **opennebula:** OpenNebula Daemon

### 2.4.2 Step 1. Installation in the Frontend

**Warning:** Commands prefixed by # are meant to be run as `root`. Commands prefixed by \$ must be run as `oneadmin`.

### 1.1. Install the repo

Add the OpenNebula repository:

```
# wget -q -O- http://downloads.opennebula.org/repo/Ubuntu/repo.key | apt-key add -
# echo "deb http://downloads.opennebula.org/repo/Ubuntu/14.04 stable opennebula" \
  > /etc/apt/sources.list.d/opennebula.list
```

### 1.2. Install the required packages

```
# apt-get update
# apt-get install opennebula opennebula-sunstone nfs-kernel-server
```

### 1.3. Configure and Start the services

There are two main processes that must be started, the main OpenNebula daemon: `oned`, and the graphical user interface: `sunstone`.

Sunstone listens only in the loopback interface by default for security reasons. To change it edit `/etc/one/sunstone-server.conf` and change `:host: 127.0.0.1` to `:host: 0.0.0.0`.

Now we must restart the Sunstone:

```
# /etc/init.d/opennebula-sunstone restart
```

### 1.4. Configure NFS

**Warning:** Skip this section if you are using a single server for both the frontend and worker node roles.

Export `/var/lib/one/` from the frontend to the worker nodes. To do so add the following to the `/etc/exports` file in the frontend:

```
/var/lib/one/ *(rw, sync, no_subtree_check, root_squash)
```

Refresh the NFS exports by doing:

```
# service nfs-kernel-server restart
```

### 1.5. Configure SSH Public Key

OpenNebula will need to SSH passwordlessly from any node (including the frontend) to any other node.

To do so run the following commands:

```
# su - oneadmin
$ cp ~/.ssh/id_rsa.pub ~/.ssh/authorized_keys
```

Add the following snippet to `~/.ssh/config` so it doesn't prompt to add the keys to the `known_hosts` file:

```
$ cat << EOT > ~/.ssh/config
Host *
    StrictHostKeyChecking no
    UserKnownHostsFile /dev/null
```

```
EOT
$ chmod 600 ~/.ssh/config
```

## 2.4.3 Step 2. Installation in the Nodes

### 2.1. Install the repo

Add the OpenNebula repository:

```
# wget -q -O- http://downloads.opennebula.org/repo/Ubuntu/repo.key | apt-key add -
# echo "deb http://downloads.opennebula.org/repo/Ubuntu/14.04 stable opennebula" > \
    /etc/apt/sources.list.d/opennebula.list
```

### 2.2. Install the required packages

```
# apt-get update
# apt-get install opennebula-node nfs-common bridge-utils
```

### 2.3. Configure the Network

**Warning:** Backup all the files that are modified in this section before making changes to them.

You will need to have your main interface, typically `eth0`, connected to a bridge. The name of the bridge should be the same in all nodes.

If you were using DHCP for your `eth0` interface, replace `/etc/network/interfaces` with:

```
auto lo
iface lo inet loopback

auto br0
iface br0 inet dhcp
    bridge_ports eth0
    bridge_fd 9
    bridge_hello 2
    bridge_maxage 12
    bridge_stp off
```

If you were using a static IP addresses instead, use this other template:

```
auto lo
iface lo inet loopback

auto br0
iface br0 inet static
    address 192.168.0.10
    network 192.168.0.0
    netmask 255.255.255.0
    broadcast 192.168.0.255
    gateway 192.168.0.1
    bridge_ports eth0
    bridge_fd 9
    bridge_hello 2
```

```
bridge_maxage 12
bridge_stp off
```

After these changes, restart the network:

```
# /etc/init.d/networking restart
```

### 2.4. Configure NFS

**Warning:** Skip this section if you are using a single server for both the frontend and worker node roles.

Mount the datastore export. Add the following to your `/etc/fstab`:

```
192.168.1.1:/var/lib/one/ /var/lib/one/ nfs soft,intr,rsize=8192,wsiz=8192,noauto
```

**Warning:** Replace `192.168.1.1` with the IP of the frontend.

Mount the NFS share:

```
# mount /var/lib/one/
```

If the above command fails or hangs, it could be a firewall issue.

### 2.5. Configure Qemu

The `oneadmin` user must be able to manage `libvirt` as root:

```
# cat << EOT > /etc/libvirt/qemu.conf
user = "oneadmin"
group = "oneadmin"
dynamic_ownership = 0
EOT
```

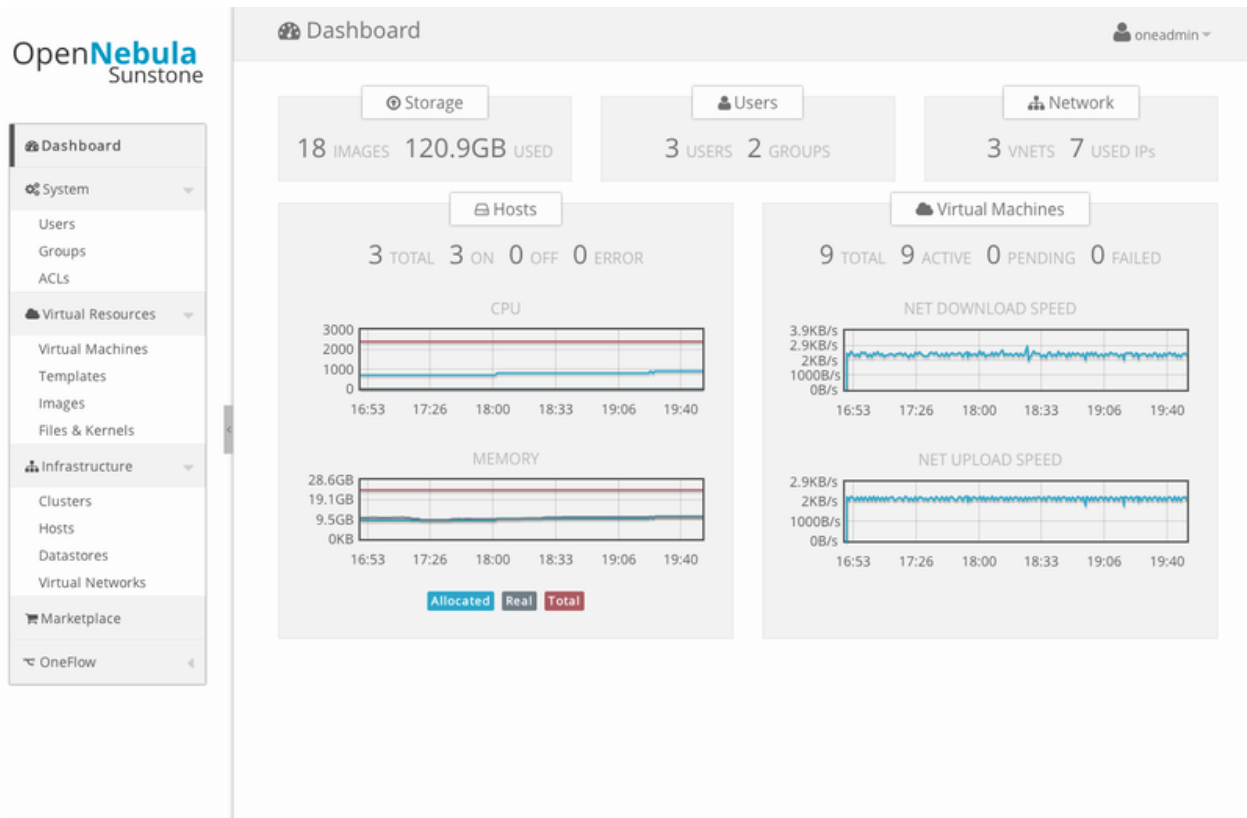
Restart `libvirt` to capture these changes:

```
# service libvirt-bin restart
```

#### 2.4.4 Step 3. Basic Usage

**Warning:** All the operations in this section can be done using Sunstone instead of the command line. Point your browser to: `http://frontend:9869`.

The default password for the `oneadmin` user can be found in `~/ .one/one_auth` which is randomly generated on every installation.



To interact with OpenNebula, you have to do it from the `oneadmin` account in the frontend. We will assume all the following commands are performed from that account. To login as `oneadmin` execute `su - oneadmin`.

### 3.1. Adding a Host

To start running VMs, you should first register a worker node for OpenNebula.

Issue this command for each one of your nodes. Replace `localhost` with your node's hostname.

```
$ onehost create localhost -i kvm -v kvm -n dummy
```

Run `onehost list` until it's set to on. If it fails you probably have something wrong in your ssh configuration. Take a look at `/var/log/one/oned.log`.

### 3.2. Adding virtual resources

Once it's working you need to create a network, an image and a virtual machine template.

To create networks, we need to create first a network template file `mynetwork.one` that contains:

```
NAME = "private"
TYPE = FIXED

BRIDGE = br0

LEASES = [ IP=192.168.0.100 ]
LEASES = [ IP=192.168.0.101 ]
LEASES = [ IP=192.168.0.102 ]
```

**Warning:** Replace the leases with free IPs in your host's network. You can add any number of leases.

Now we can move ahead and create the resources in OpenNebula:

```
$ onevnet create mynetwork.one

$ oneimage create --name "CentOS-6.5_x86_64" \
  --path "http://appliances.c12g.com/CentOS-6.5/centos6.5.qcow2.gz" \
  --driver qcow2 \
  --datastore default

$ onetemplate create --name "CentOS-6.5" --cpu 1 --vcpu 1 --memory 512 \
  --arch x86_64 --disk "CentOS-6.5_x86_64" --nic "private" --vnc \
  --ssh
```

You will need to wait until the image is ready to be used. Monitor its state by running `oneimage list`.

In order to dynamically add ssh keys to Virtual Machines we must add our ssh key to the user template, by editing the user template:

```
$ EDITOR=vi oneuser update oneadmin
```

Add a new line like the following to the template:

```
SSH_PUBLIC_KEY="ssh-dss AAAAB3NzaC1kc3MAAACBANBWTQmm4Gt..."
```

Substitute the value above with the output of `cat ~/.ssh/id_dsa.pub`.

### 3.3. Running a Virtual Machine

To run a Virtual Machine, you will need to instantiate a template:

```
$ onetemplate instantiate "CentOS-6.5" --name "My Scratch VM"
```

Execute `onevm list` and watch the virtual machine going from PENDING to PROLOG to RUNNING. If the vm fails, check the reason in the log: `/var/log/one/<VM_ID>/vm.log`.

## 2.4.5 Further information

- [Planning the Installation](#)
- [Installing the Software](#)
- [FAQs](#). Good for troubleshooting
- [Main Documentation](#)

## 2.5 Quickstart: Create Your First vDC

This guide will provide a quick example of how to partition your cloud for a vDC. In short, a vDC is a group of users with part of the physical resources assigned to them. The *Understanding OpenNebula* guide explains the OpenNebula provisioning model in detail.

## 2.5.1 Step 1. Create a Cluster

We will first create a *cluster*, ‘web-dev’, where we can group hosts, datastores and virtual networks for the new vDC.

```
$ onehost list
ID NAME                CLUSTER  RVM    ALLOCATED_CPU    ALLOCATED_MEM  STAT
0  host01              web-dev  0      0 / 200 (0%)    0K / 7.5G (0%) on
1  host02              web-dev  0      0 / 200 (0%)    0K / 7.5G (0%) on
2  host03              -        0      0 / 200 (0%)    0K / 7.5G (0%) on
3  host04              -        0      0 / 200 (0%)    0K / 7.5G (0%) on

$ onedatastore list
ID NAME                SIZE AVAIL CLUSTER  IMAGES TYPE DS      TM
0  system              113.3G 25%  web-dev  0  sys  -      shared
1  default              113.3G 25%  web-dev  1  img  fs      shared
2  files                113.3G 25%  -        0  fil  fs      ssh

$ onevnet list
ID USER    GROUP    NAME    CLUSTER  TYPE BRIDGE  LEASES
0  oneadmin oneadmin private web-dev   R  virbr0  0
```

The screenshot shows the 'Create Cluster' dialog in the OpenNebula web interface. The 'Name' field is filled with 'web-dev'. The 'Hosts' tab is active, showing a table of available hosts. Hosts 'host01' and 'host02' are selected. The 'Create' button is visible at the bottom right of the dialog.

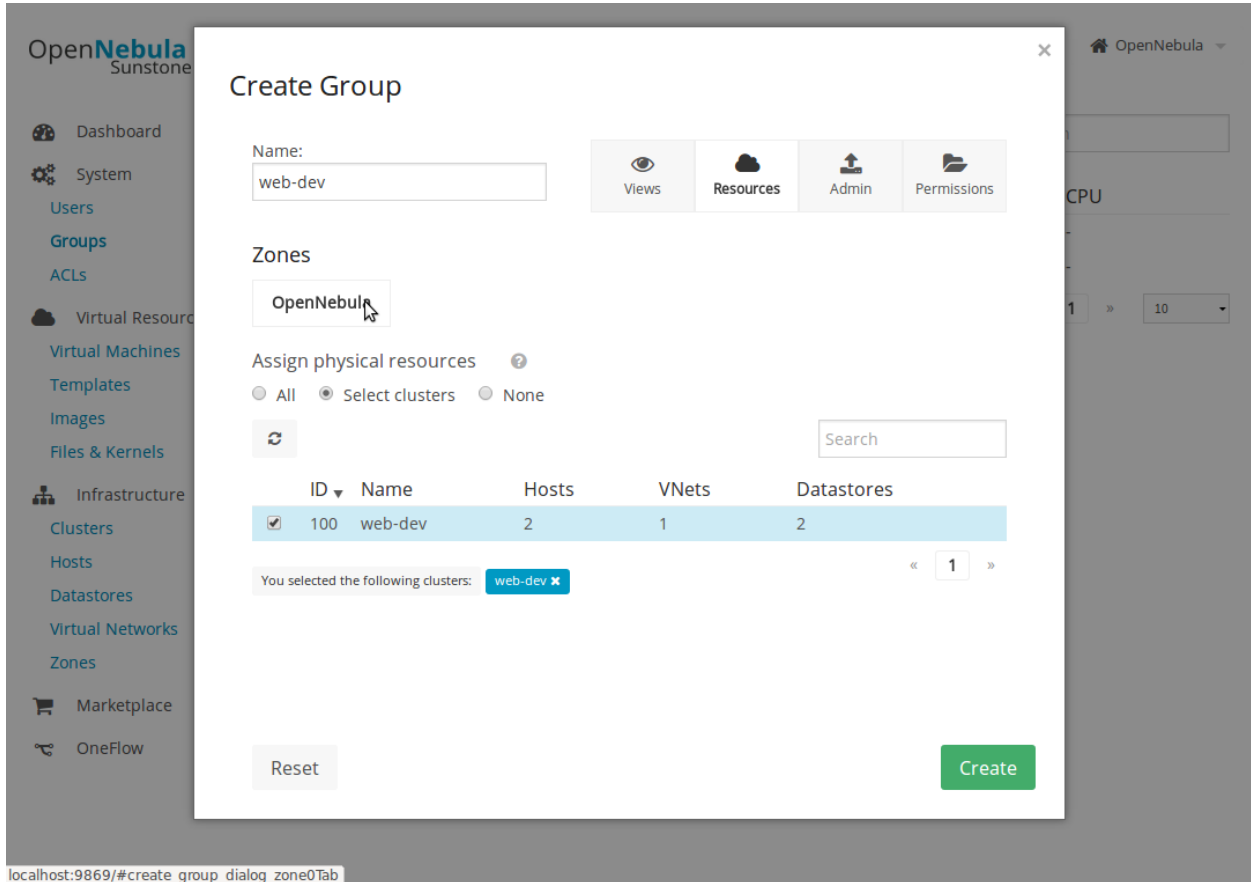
ID	Name	Cluster	RVMs	Allocated CPU	Allocated MEM	Status
3	host04	-	0	0 / 200 (0%)	0KB / 7.5GB (0%)	ON
2	host03	-	0	0 / 200 (0%)	0KB / 7.5GB (0%)	ON
1	host02	-	0	0 / 200 (0%)	0KB / 7.5GB (0%)	ON
0	host01	-	0	0 / 200 (0%)	0KB / 7.5GB (0%)	ON

## 2.5.2 Step 2. Create a vDC Group

We can now create the new *group*, named also ‘web-dev’. This group, or vDC, will have a special admin user, ‘web-dev-admin’.

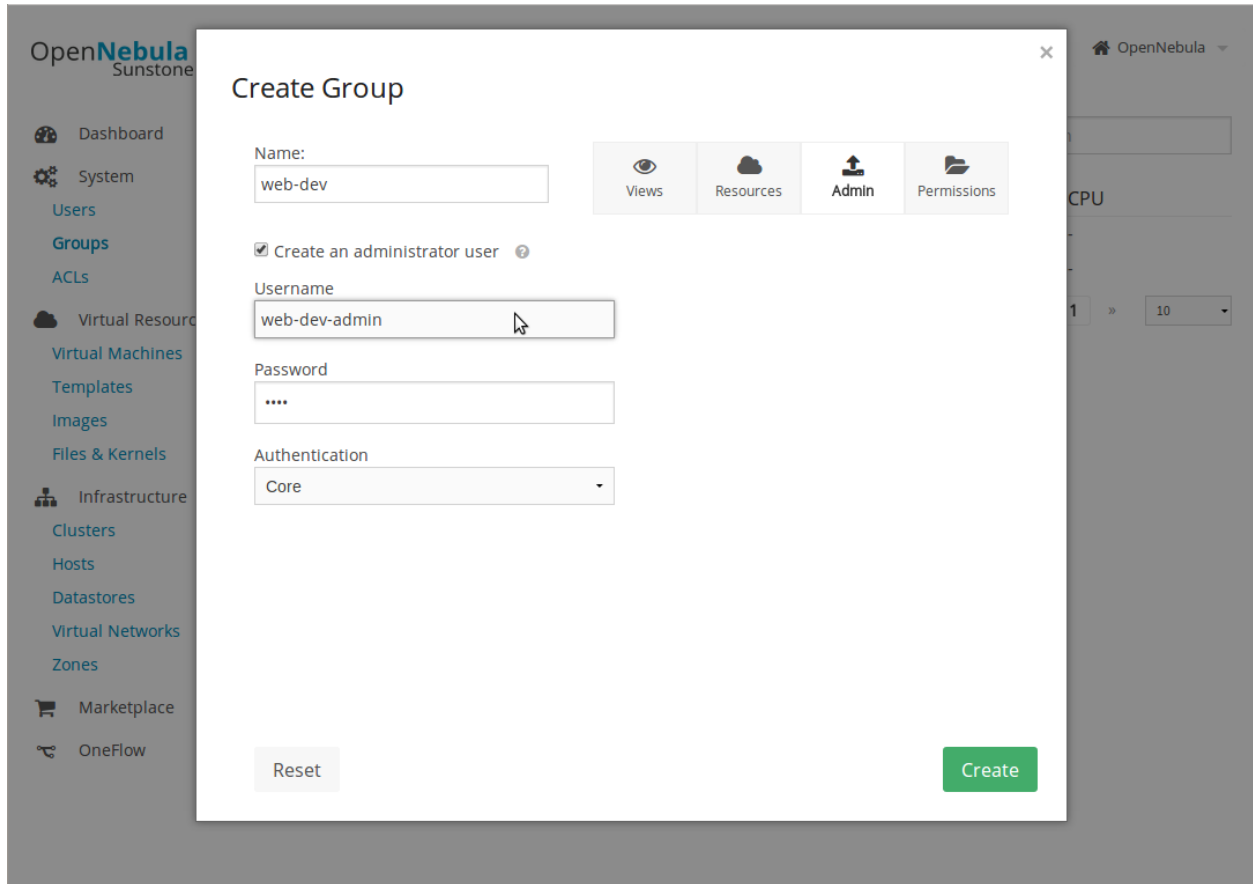
```
$ onegroup create --name web-dev --admin_user web-dev-admin --admin_password abcd  
ID: 100
```

```
$ onegroup add_provider 100 0 web-dev
```



localhost:9869/#create\_group\_dialog\_zone0Tab





### 2.5.3 Step 3. Optionally, Set Quotas

The cloud administrator can set *usage quotas* for the vDC. In this case, we will put a limit of 10 VMs.

```
$ onegroup show web-dev
GROUP 100 INFORMATION
ID           : 100
NAME        : web-dev

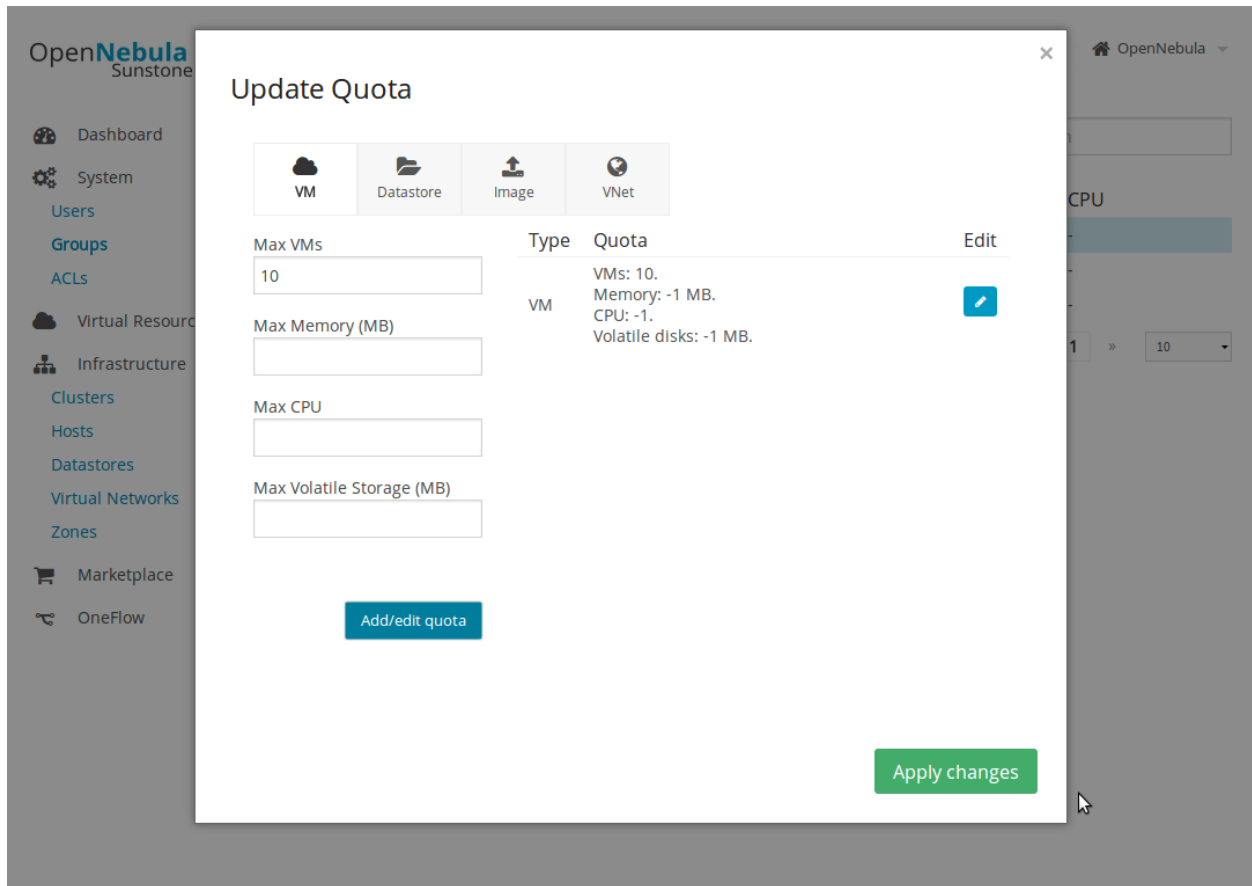
GROUP TEMPLATE
GROUP_ADMINS="web-dev-admin"
GROUP_ADMIN_VIEWS="vdcadmin"
SUNSTONE_VIEWS="cloud"

USERS
ID
2

RESOURCE PROVIDERS
ZONE CLUSTER
0 100

RESOURCE USAGE & QUOTAS

NUMBER OF VMs      MEMORY      CPU      VOLATILE_SIZE
0 / 10             0M / 0M    0.00 /   0.00    0M / 0M
```



## 2.5.4 Step 4. Prepare Virtual Resources for the Users

At this point, the cloud administrator can also prepare working Templates and Images for the vDC users.

```
$ onetemplate chgrp ubuntu web-dev
```

OpenNebula Sunstone

Template 0

oneadmin OpenNebula

Dashboard System Users Groups ACLs Virtual Resources Virtual Machines Templates Images Files & Kernels Infrastructure Clusters Hosts Datastores Virtual Networks Zones Marketplace OneFlow

Update Instantiate Clone

Info Template

Information	Permissions:	Use	Manage	Admin
ID	Owner	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Name	Group	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Register time	Other	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Ownership				
Owner	oneadmin			<input type="checkbox"/>
Group	0: oneadmin 1: users 100: web-dev			<input type="checkbox"/>

OpenNebula 4.5.80 by C12G Labs.

## 2.5.5 Reference for End Users

The vDC admin uses an interface similar to the cloud administrator, but without any information about the physical infrastructure. He will be able to create new users inside the vDC, monitor their resources, and create new Templates for them. The vDC admin can also decide to configure quota limits for each user.

Refer your vDC admin user to the *vDC Admin View Guide*.

End users access OpenNebula through a simplified instantiate, where they can launch their own VMs from the Templates prepared by the administrator. Users can also save the changes they make to their machines. This view is self explanatory, you can read more about it in the *Cloud View Guide*.



Create



VMs



Templates



jdoe



Log out



OpenNebula

## Virtual Machines

ubuntu-0

x1 - 1GB

ubuntu

192.168.122.2

**RUNNING** 53s ago

« 1 » 6




Create Virtual Machine

OpenNebula 4.5.80 by C12G Labs.



 jdoe



-  SSH Key
-  Quotas
-  Settings



You can add an SSH key to your account which will be used as the preferred method of access for new Virtual Machines

Add SSH Key



Add SSH Key