OpenNebula.org

OpenNebula 4.4 Design and Installation Guide

OpenNebula Project

January 07, 2014

CONTENTS

CHAPTER

ONE

GETTING STARTED

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. This introductory guide gives an overview of OpenNebula and summarizes its main benefits for the different stakeholders involved in a cloud computing infrastructure.

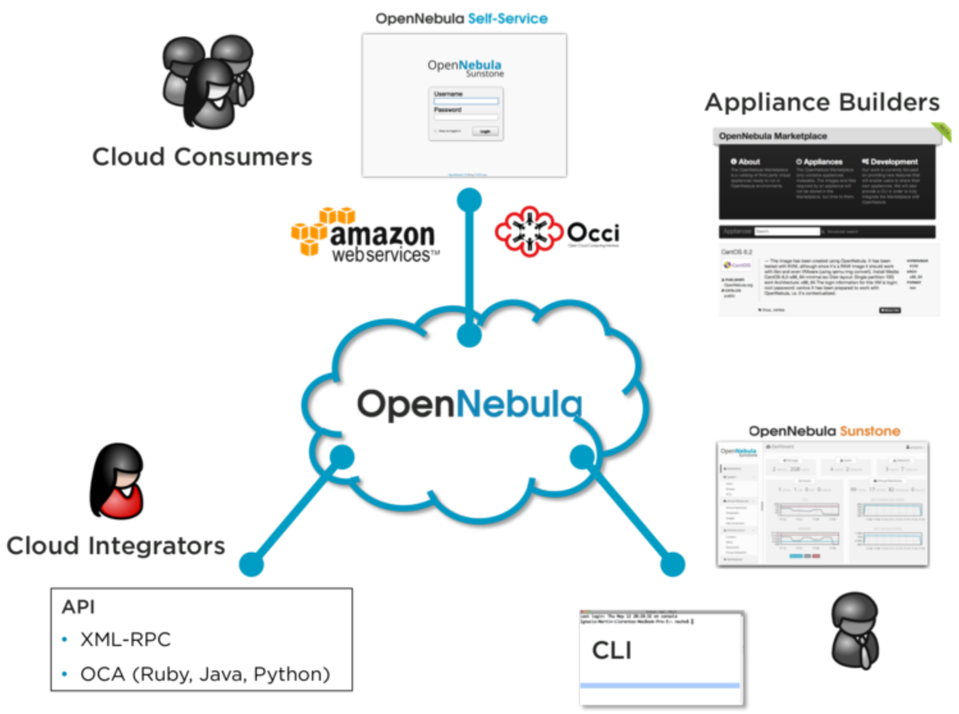
1.1.1 What Are the Key Features Provided by OpenNebula?

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:

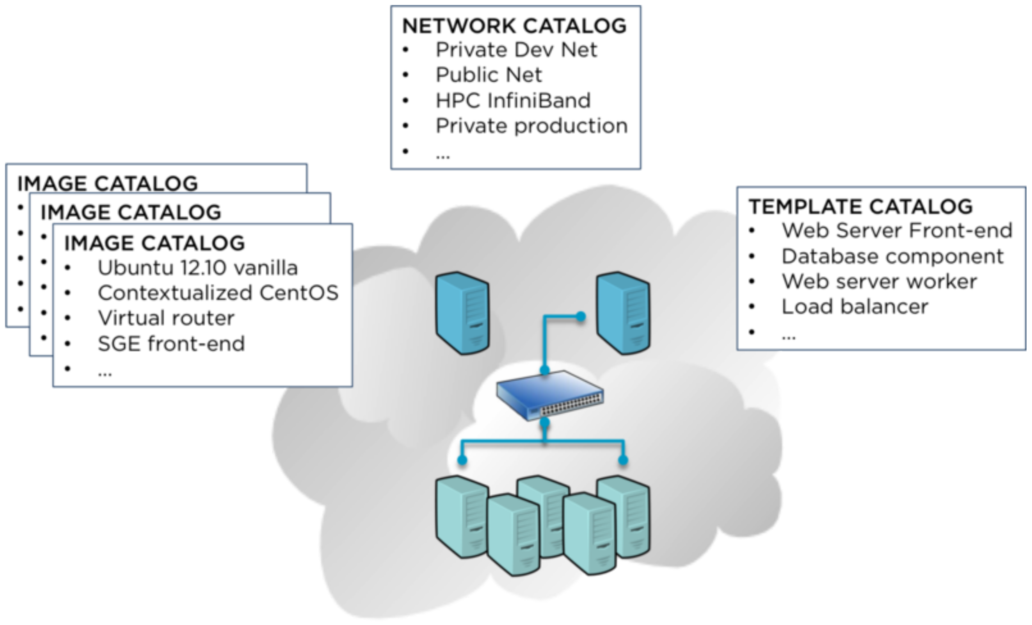
- Cloud interfaces for **Cloud Consumers**, like the *OCCI* and *EC2 Query and EBS* interfaces, and a simple *Sunstone cloud user view* that can be used as a self-service portal.
- Administration interfaces for **Cloud Advanced Users and Operators**, like a Unix-like *command line interface* and the 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*.

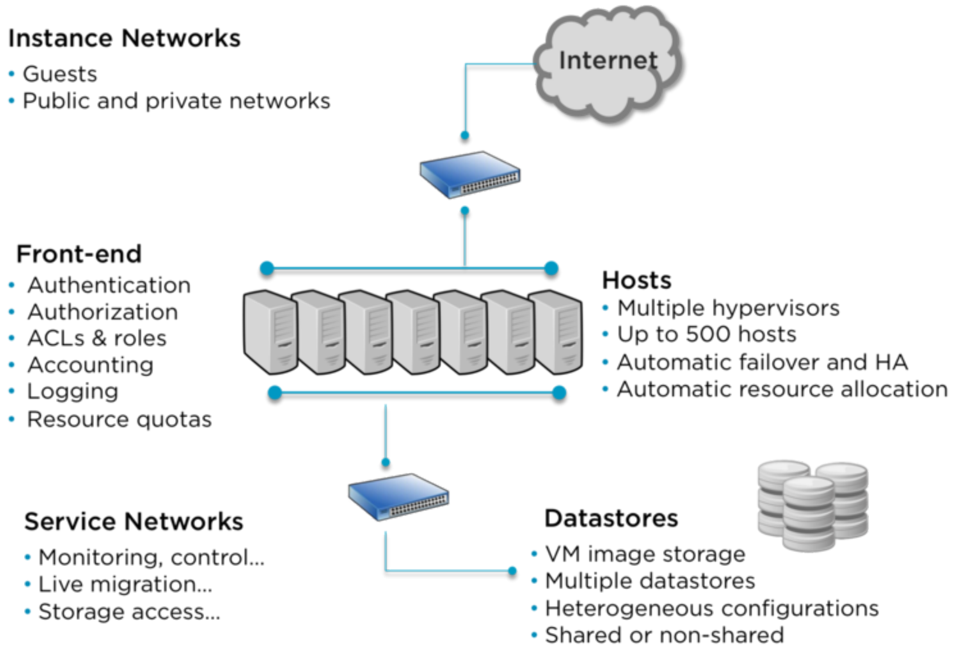


1.1.4 What Does OpenNebula Offer to Cloud Operators?

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 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 a 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 OpenNebula Zones component (*oZones*) allows for the centralized management of multiple instances of OpenNebula, called *Zones*, 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

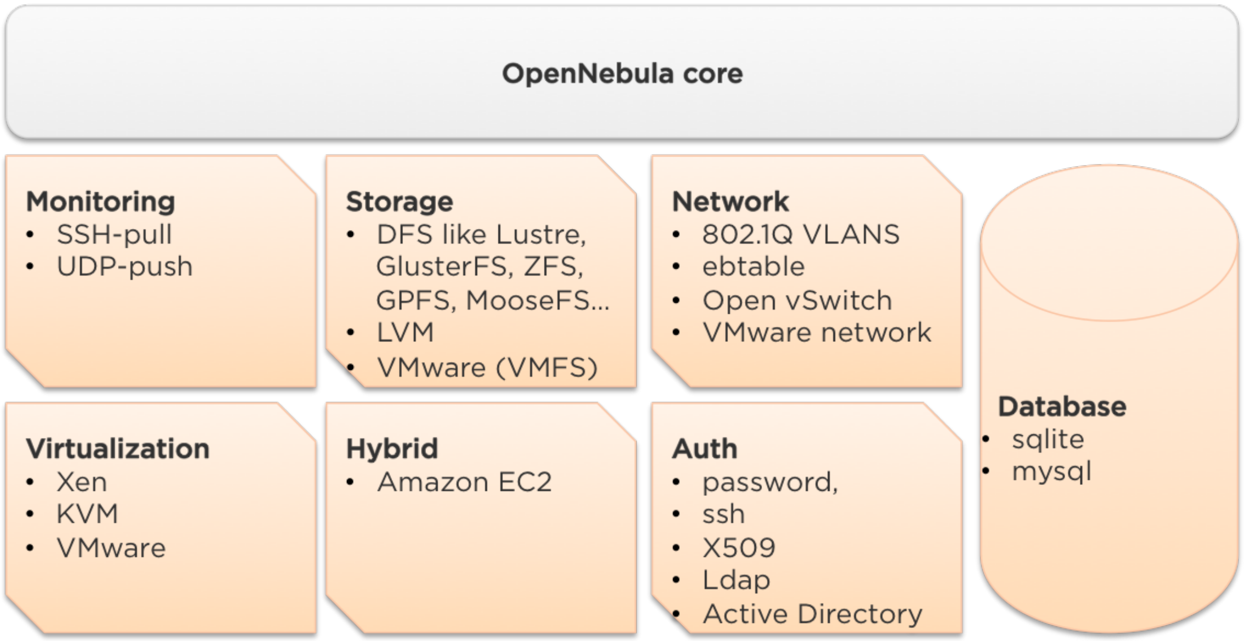


1.1.5 What Does OpenNebula Offer to Cloud Builders?

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*, *ebtables*, *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.

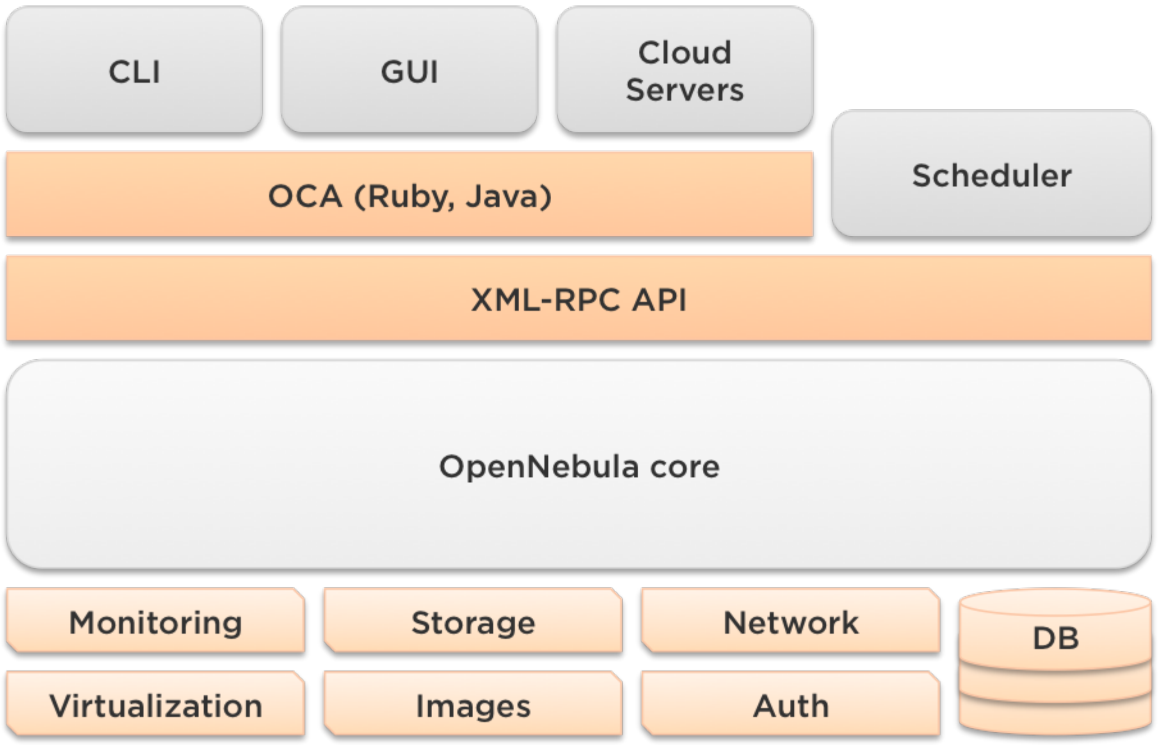
- **Cloud Bursting**: Out of the box connectors are shipped to support *Amazon EC2* cloudbursting.



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*.
- **oZones API** used to *programatically manage OpenNebula Zones and Virtual Data Centers*.
- **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 OpenNebula 4.4 Detailed Features and Functionality

This section describes the **detailed features and functionality of the latest version of OpenNebula (v4.4)** for the management of private clouds and datacenter virtualization(*). It includes links to the different parts of the documentation and the web site that provide extended information about each feature. We also provide a summarized table of key features.

1.2.1 Powerful User Security Management

- Secure and efficient *Users and Groups Subsystem* for authentication and authorization of requests with complete functionality for user management: create, delete, show...
- *Pluggable authentication and authorization* based on *passwords*, *ssh rsa keypairs*, *X509 certificates*, *LDAP* or *Active Directory*
- Special authentication mechanisms for *SunStone (OpenNebula GUI)* and the *Cloud Services (EC2 and OCCI)*
- Authorization framework with *fine-grained ACLs* that allows multiple-role support for different types of users and administrators, delegated control to authorized users, secure isolated multi-tenant environments, and easy resource (VM template, VM image, VM instance, virtual network and host) sharing

1.2.2 Advanced Multi-tenancy with Group Management

- Administrators can *groups users* into organizations that can represent different projects, division...
- Each group have *configurable access to shared resources* so enabling a multi-tenant environment with multiple groups sharing the same infrastructure

- Configuration of special *users that are restricted to public cloud APIs* (e.g. EC2 or OCCI)
- Complete functionality for management of groups: create, delete, show...
- Multiple group support, with the ability to define primary and secondary groups.

1.2.3 On-demand Provision of Virtual Data Centers

- A *Virtual Data Centers (VDC)* is a fully-isolated virtual infrastructure environment where a group of users, under the control of the VDC administrator, can create and manage compute, storage and networking capacity
- Support for the creation and management of multiples VDCs within the same logical cluster and zone
- Advanced multi-tenancy with complete functionality for management of VDCs: create, delete, show...

1.2.4 Advanced Control and Monitoring of Virtual Infrastructure

- *Image Repository Subsystem* with catalog and complete functionality for VM image management: list, publish, unpublish, show, enable, disable, register, update, saveas, delete, clone...
- *Template Repository Subsystem* with catalog and complete functionality for VM template management: add, delete, list, duplicate...
- *Full control of VM instance life-cycle* and complete functionality for VM instance management: submit, deploy, migrate, livemigrate, reschedule, stop, save, resume, cancel, shutdown, restart, reboot, delete, monitor, list, power-on, power-off,...
- Advanced functionality for VM dynamic management like *system and disk snapshotting*, *capacity resizing*, or *NIC hotplugging*
- *Programmable VM operations*, so allowing users to schedule actions
- Volume hotplugging to easily hot plug a volatile disk created on-the-fly or an existing image from a Datastore to a running VM
- *Broad network virtualization capabilities* with traffic isolation, ranged or fixed networks, definition of generic attributes to define multi-tier services consisting of groups of inter-connected VMs, and complete functionality for virtual network management to interconnect VM instances: create, delete, monitor, list...
- *IPv6 support* with definition site and global unicast addresses
- Configurable *system accounting statistics* 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
- Tagging of users, VM images and virtual networks with arbitrary metadata that can be later used by other components
- *User defined VM tags* to simplify VM management and to store application specific data
- *Plain files datastore* to store kernels, ramdisks and files to be used in context. The whole set of OpenNebula features applies, e.g. ACLs, ownership...

1.2.5 Complete Virtual Machine Configuration

- Complete *definition of VM attributes and requirements*
- Support for automatic configuration of VMs with advanced *contextualization mechanisms*
- *Cloud-init* support

- *Hook Manager* to trigger administration scripts upon VM state change
- Wide range of guest operating system including Microsoft Windows and Linux
- *Flexible network defintion*
- *Configuration of firewall for VMs* to specify a set of black/white TCP/UDP ports

1.2.6 Advanced Control and Monitoring of Physical Infrastructure

- *Configurable to deploy public, private and hybrid clouds*
- *Host Management Subsystem* with complete functionality for management of physical hosts: create, delete, enable, disable, monitor, list...
- Dynamic creation of *clusters* as a logical set of physical resources, namely: hosts, networks and data stores, within each zone
- Highly scalable and extensible built-in *monitoring subsystem*

1.2.7 Broad Commodity and Enterprise Platform Support

- Hypervisor agnostic *Virtualization Subsystem* with broad hypervisor support (*Xen*, *KVM* and *VMware*), centralized management of environments with multiple hypervisors, and support for multiple hypervisors within the same physical box
- *Storage Subsystem* with support for multiple data stores to balance I/O operations between storage servers, or to define different SLA policies (e.g. backup) and performance features for different VM types or users
- *Storage Subsystem* supporting any backend configuration with different datastore types: *file system datastore*, to store disk images in a file form and with image transferring using ssh or shared file systems (NFS, GlusterFS, Lustre...), *LVM* to store disk images in a block device form, *Ceph* for distributed block device, and *VMware datastore* specialized for the VMware hypervisor that handle the vmdk format and with support for VMFS
- Flexible *Network Subsystem* with integration with *Ebtable*, *Open vSwitch* and *802.1Q tagging*
- *Virtual Router* fully integrated with OpenNebula to provide basic L3 services like NATting, DHCP, DNS...

1.2.8 Distributed Resource Optimization

- Powerful and flexible *requirement/rank matchmaker scheduler* providing automatic initial VM placement for the definition of workload and resource-aware allocation policies such as packing, striping, load-aware, affinity-aware...
- *Advanced requirement expressions* with cluster attributes for VM placement, affinity policies, any host attribute for scheduling expressions, and scheduler feedback through VM tags
- Powerful and flexible *requirement/rank matchmaker scheduler* for storage load balancing to distribute efficiently the I/O of the VMs across different disks, LUNs or several storage backends
- *Resource quota management* to allocate, track and limit computing, storage and networking resource utilization
- Support for *cgroups* on KVM to enforce VM CPU usage as described in the VM Template

1.2.9 Centralized Management of Multiple Zones

- *Single access point and centralized management for multiple instances of OpenNebula*
- *Federation of multiple OpenNebula zones* for scalability, isolation or multiple-site support
- Support for the creation and management of multiples clusters within the same zone
- Complete functionality for management of zones: create, delete, show, list...

1.2.10 High Availability

- Persistent database backend with support for high availability configurations
- *Configurable behavior in the event of host, VM, or OpenNebula instance failure to provide an easy to use and cost-effective failover solution*
- Support for *high availability architectures*

1.2.11 Community Virtual Appliance Marketplace

- Marketplace with an online catalog where individuals and organizations can quickly distribute and deploy virtual appliances ready-to-run on OpenNebula cloud environments
- *Marketplace is fully integrated with OpenNebula* so any user of an OpenNebula cloud can find and deploy virtual appliances in a single click through familiar tools like the SunStone GUI or the OpenNebula CLI

1.2.12 Management of Multi-tier Applications

- *Automatic execution of multi-tiered applications* with complete functionality for the management of groups of virtual machines as a single entity: list, delete, scale up, scale down, shutdown... and the management of Service Templates: create, show, delete, instantiate...
- *Automatic deployment and undeployment of Virtual Machines* according to their dependencies in the Service Template
- Provide configurable services from a catalog and self-service portal
- Enable tight, efficient administrative control
- Complete integration with the OpenNebula's User Security Management system
- Computing resources can be tracked and limited using OpenNebula's *Resource Quota Management*
- *Automatic scaling of multi-tiered applications* according to performance metrics and time schedule

1.2.13 Gain Insight into Cloud Applications

- *OneGate allows Virtual Machine guests to push monitoring information to OpenNebula*
- With a security token the VMs can call back home and report guest and/or application status in a simple way, that can be easily queried through OpenNebula interfaces (Sunstone, CLI or API).
- Users and administrators can use it to gather metrics, detect problems in their applications, and trigger *OneFlow auto-scaling rules*

1.2.14 Hybrid Cloud Computing and Cloud Bursting

- *Extension of the local private infrastructure with resources from remote clouds*
- *Support for Amazon EC2* with most of the EC2 features like tags, security groups or VPC; and simultaneous access to multiple remote clouds

1.2.15 Standard Cloud Interfaces and Simple Self-Service Portal for Cloud Consumers

- *Transform your local infrastructure into a public cloud by exposing REST-based interfaces*
- *OGF OCCI service*, the emerging cloud API standard, and *client tools*
- *AWS EC2 API service*, the de facto cloud API standard, with *compatibility with EC2 ecosystem tools* and *client tools*
- Support for simultaneously exposing multiple cloud APIs
- *Self-service provisioning portal implemented as a user view of Sunstone* to allow non-IT end users to easily create, deploy and manage compute, storage and network resources

1.2.16 Rich Command Line and Web Interfaces for Cloud Administrators

- *Unix-like Command Line Interface* to manage all resources: users, VM images, VM templates, VM instances, virtual networks, zones, VDCs, physical hosts, accounting, authentication, authorization...
- *Easy-to-use Sunstone Graphical Interface* providing usage graphics and statistics with cloudwatch-like functionality, VNC support, different system views for different roles, catalog access, multiple-zone management...
- *Sunstone is easily customizable* to define multiple cloud views for different user groups

1.2.17 Multiple Deployment Options

- *Easy to install and update* with packages for most common Linux distributions
- Available in most popular Linux distributions
- *Optional building from source code*
- *System features a small footprint*, less than 10Mb
- *Detailed log files* with *syslog support* for the different components that maintain a record of significant changes

1.2.18 Easy Extension and Integration

- Modular and extensible architecture to fit into any existing datacenter
- Customizable drivers for the main subsystems to easily leverage existing IT infrastructure and system management products: *Virtualization*, *Storage*, *Monitoring*, *Network*, *Auth* and *Hybrid Cloud*
- New drivers can be easily written in any language
- Plugin support to easily extend SunStone Graphical Interface with additional tabs to better integrate Cloud and VM management with each site own operations and tools
- Easily customizable self-service portal for cloud consumers

- *Configuration and tuning parameters* to adjust behavior of the cloud management instance to the requirements of the environment and use cases
- Fully open-source technology available under Apache license
- Powerful and extensible low-level cloud API in *Ruby* and *JAVA* and *XMLRPC API*
- A Ruby API to build applications on top of the Zones/VDC component *ZONA, the ZONes Api*
- OpenNebula Add-on Catalog with components enhancing the functionality provided by OpenNebula

1.2.19 Reliability, Efficiency and Massive Scalability

- Automated testing process for functionality, scalability, performance, robustness and stability
- Technology matured through an active and engaged community
- Proven on large scale infrastructures consisting of tens of thousands of cores and VMs
- Highly scalable database back-end with support for *MySQL* and SQLite
- Virtualization drivers adjusted for maximum scalability
- Very efficient core developed in C++ language

(*) *Because OpenNebula leverages the functionality exposed by the underlying platform services, its functionality and performance may be affected by the limitations imposed by those services.*

- *The list of features may change on the different platform configurations*
- *Not all platform configurations exhibit a similar performance and stability*
- *The features may change to offer users more features and integration with other virtualization and cloud components*
- *The features may change due to changes in the functionality provided by underlying virtualization services*

1.3 Glossary

1.3.1 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.

1.3.2 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.

1.3.3 OpenNebula Management

- **ACL**: Access Control List. Check *the managing ACL rules guide*.
- **oneadmin**: Special administrative account. See the *Users and Groups guide*.
- **oZones**: The *susbsystem* in OpenNebula that manages zones.

CHAPTER

TWO

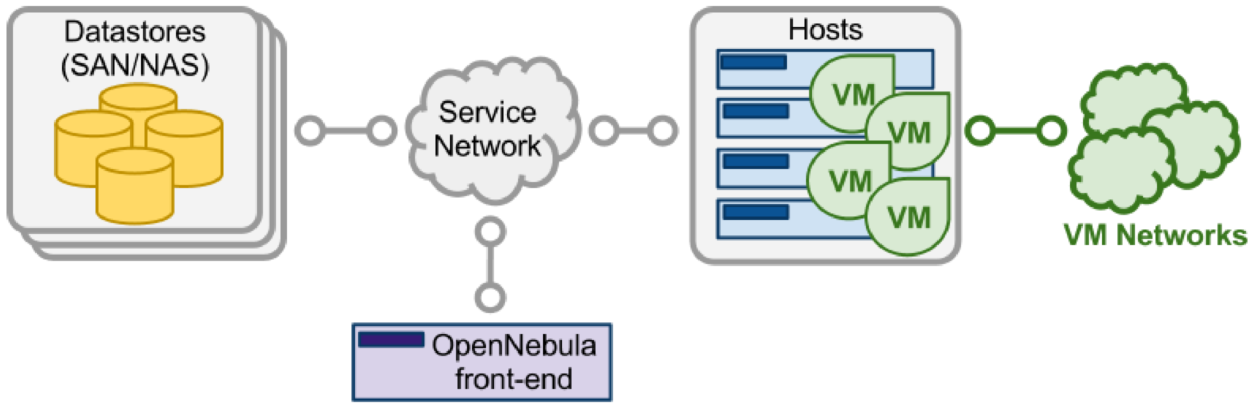
BUILDING YOUR CLOUD

2.1 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.

2.1.1 Architectural Overview

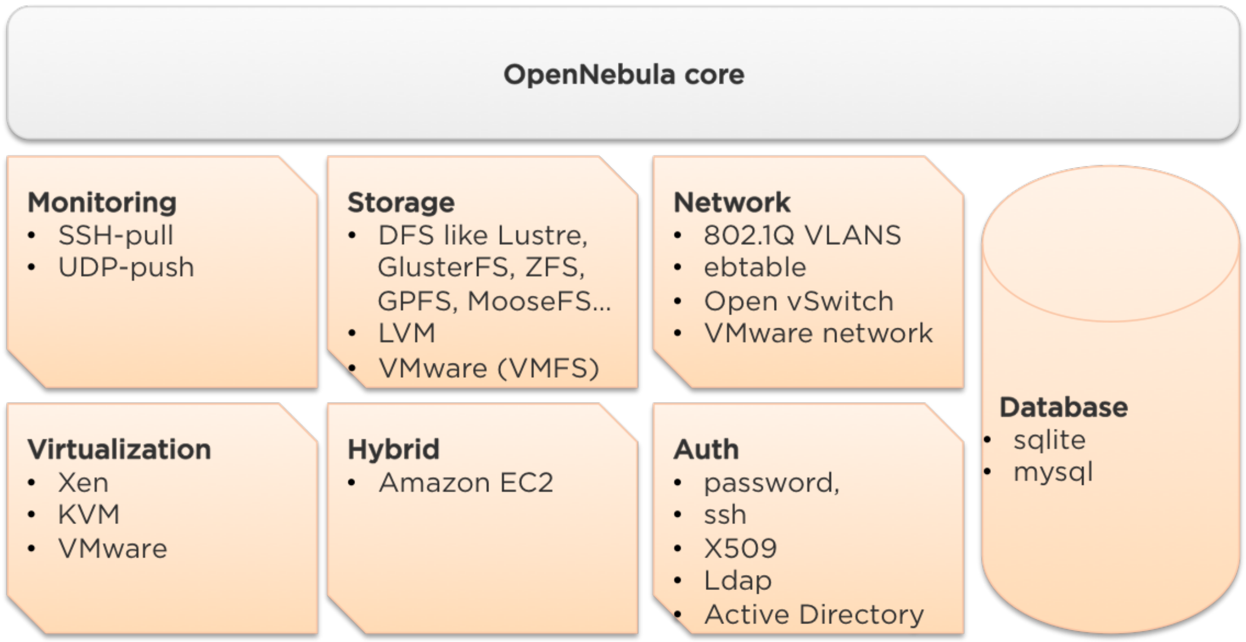
OpenNebula assumes that your physical infrastructure adopts a classical cluster-like architecture with a front-end, and a set of hosts where Virtual Machines (VM) will be executed. There is at least one physical network joining all the hosts with the front-end.



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.



2.1.2 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. The OpenNebula *Zones (oZones)* component allows for the centralized management of multiple instances of OpenNebula (zones), managing in turn potentially different administrative domains. You may also find interesting the following guide about *how to tune OpenNebula for large deployments*.

2.1.3 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 >= 4.4.

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

2.1.4 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*

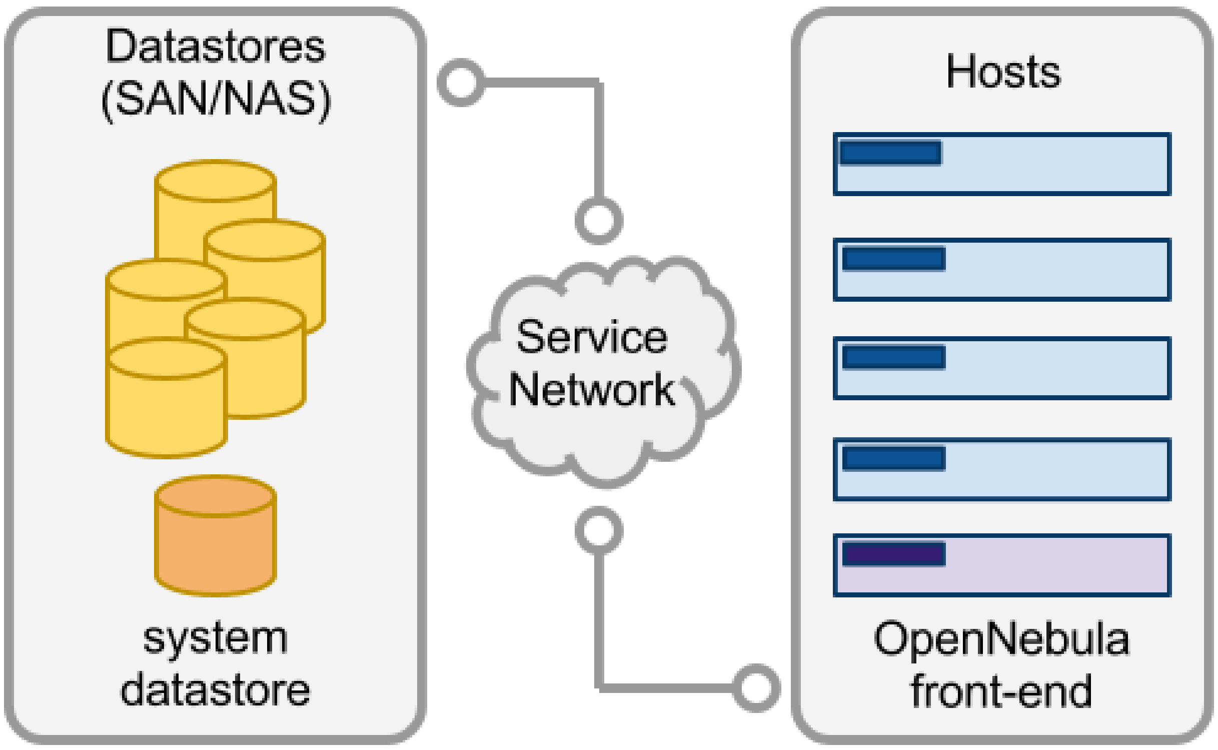
Warning: Default: OpenNebula is configured to interact with hosts running KVM.

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*.

2.1.5 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 snapshoted.
- *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.

2.1.6 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.
- *ebtables*: restrict network access through Ebtables 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.

2.1.7 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.

2.1.8 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.

- *Multiple Zone and Virtual Data Centers*: The OpenNebula Zones (oZones) component allows for the centralized management of multiple instances of OpenNebula (zones), managing in turn potentially different administrative domains. These zones can be effectively shared through the Virtual DataCenter (VDC) abstraction. A VDC is a set of virtual resources (images, VM templates, virtual networks and virtual machines) and users that manage those virtual resources, all sustained by infrastructure resources offered by OpenNebula.
- *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 OCCI*.
- *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.

2.2 Installing the Software

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

2.2.1 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
```

To install a CentOS/RHEL OpenNebula front-end with **packages downloaded from our page**, untar the tar.gz in the front-end and run:

```
# tar xvzf CentOS-6-opennebula-<OpenNebula Version>.tar.gz
# sudo yum localinstall opennebula-server opennebula-sunstone opennebula-ruby
```

CentOS/RHEL Package Description

These are the packages available for this distribution:

- **opennebula-server**: Main OpenNebula daemon, scheduler, etc
- **opennebula-sunstone**: OpenNebula Sunstone, EC2, OCCI
- **opennebula-ozones**: OpenNebula OZones
- **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

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-zones**: OpenNebula OZones (depends on **opennebula**)

- **opennebula-sunstone**: OpenNebula Sunstone (depends on **opennebula**)

1.3. Installing on Debian/Ubuntu

Also the JSON ruby library packaged with Debian 6 is not compatible with ozones. 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..
```

Ubuntu 13.04

```
# echo "deb http://downloads.opennebula.org/repo/Ubuntu/13.04 stable opennebula" > /etc/apt/sources..
```

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

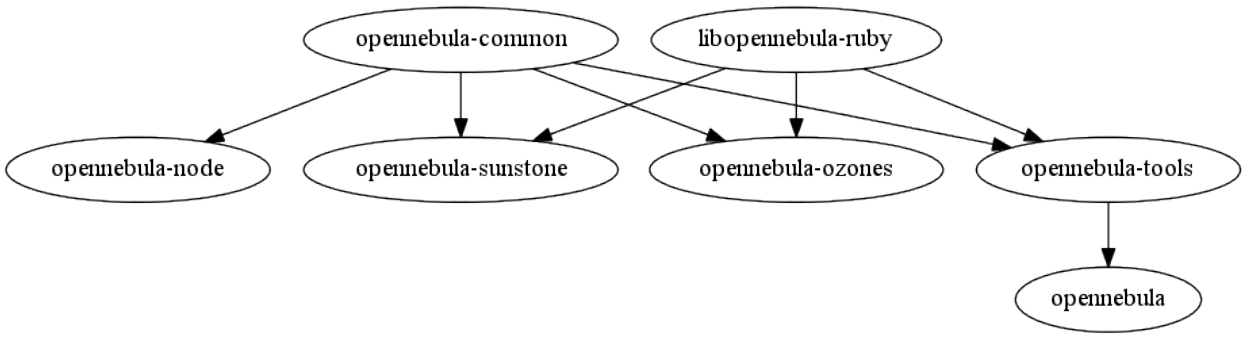
```
# apt-get update
# apt-get install opennebula opennebula-sunstone
```

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

```
$ sudo dpkg -i opennebula opennebula-sunstone
$ sudo apt-get install -f
```

Debian/Ubuntu Package Description

These are the packages available for these distributions:



- **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

2.2.2 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.

For **cloud bursting**, a newer nokogiri gem than the on packed by current distros is required. If you are planning to use cloud bursting, you need to install nokogiri >= 1.4.4 prior to run `install_gems`

```
# sudo gem install nokogiri -v 1.4.4
```

2.2.3 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`!

2.2.4 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 CPU     MEM        HOSTNAME         TIME
```

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

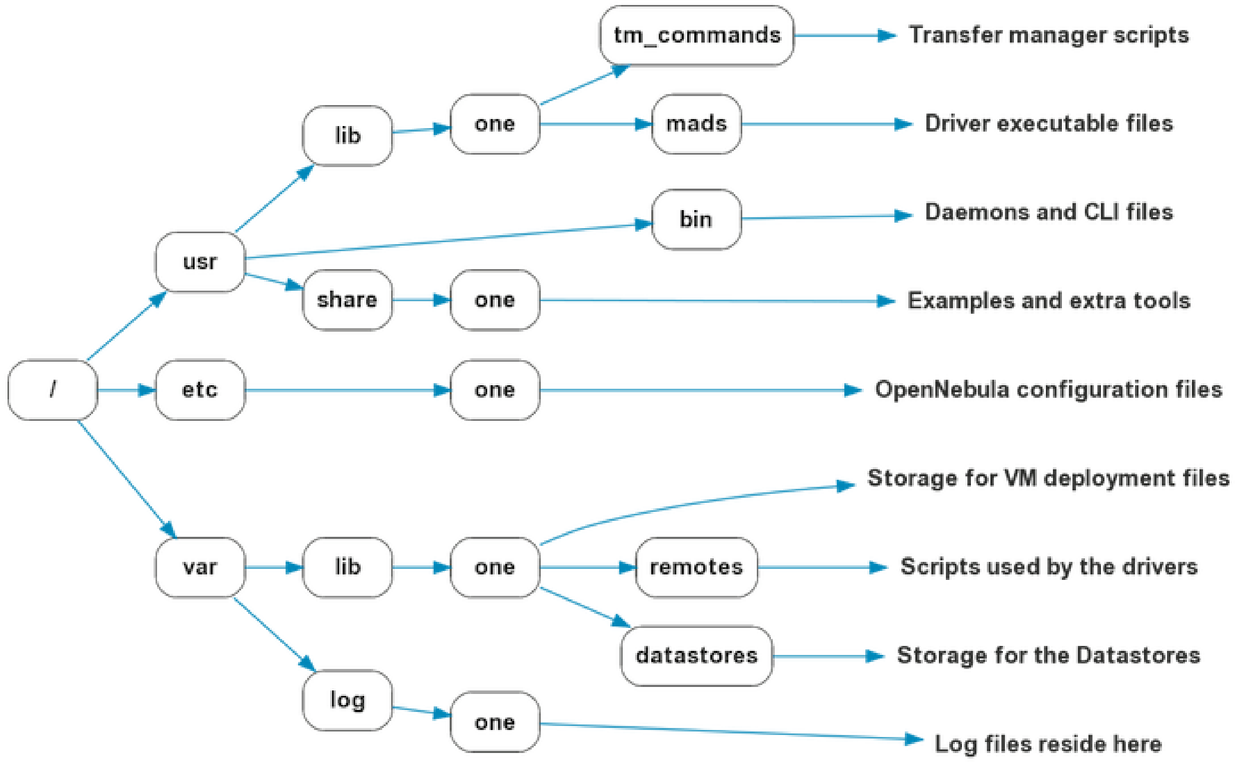
```
$ onevm list
Connection refused - connect(2)
```

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]: Bootstraping OpenNebula database.
```

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



2.2.5 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, execute the following as root:

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

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, execute the following as root:

```
$ sudo dpkg -i opennebula-node-kvm
$ sudo apt-get install -f
```

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

Warning: Due to the Debian packaging policy, there are some paths which are different in the Debian/Ubuntu packages with respect to OpenNebula's documentation. In particular:

- /usr/share/one/examples/ => /usr/share/doc/opennebula/examples/
- /usr/share/one/ => /usr/share/opennebula/

2.2.6 Step 6. Manual Configuration of Unix Accounts

Warning: This step can be skipped if you have installed the kvm node 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 frontend.

- 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.

2.2.7 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
```

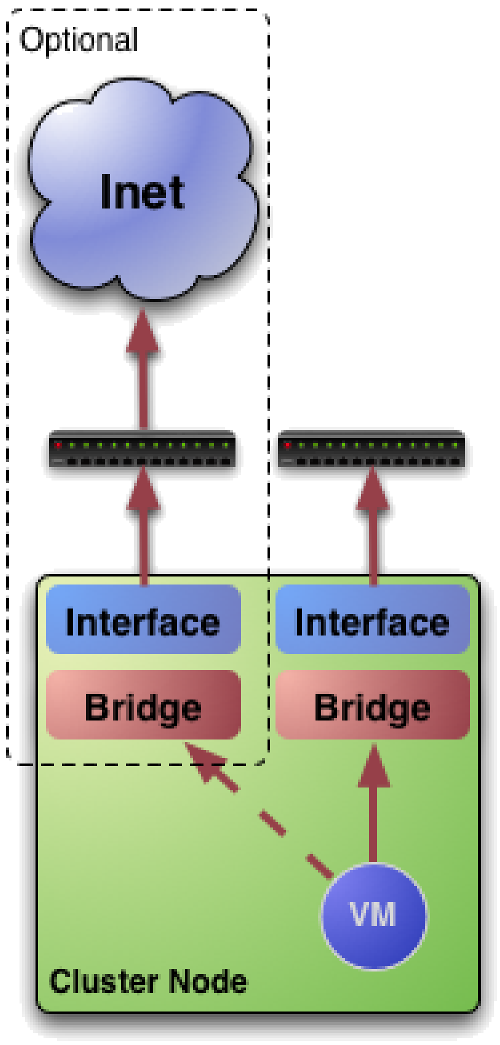
- Tell ssh client to not ask before adding hosts to `known_hosts` file. Also it is a good idea to reduced the connection timeout in case of network problems. This is configured into `~/.ssh/config`, see `man ssh_config` for a complete reference.:

```
$ 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.

2.2.8 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
```

2.2.9 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 ore 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.

2.2.10 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*.

2.2.11 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*.

2.3 Upgrading from Previous Versions

This guide describes the installation procedure for systems that are already running a 2.x or 3.x OpenNebula. The upgrade will preserve all current users, hosts, resources and configurations; for both Sqlite and MySQL backends.

Read the *Compatibility Guide* and Release Notes to know what is new in OpenNebula 4.4.

Warning: With the new *multi-system DS* functionality, it is now required that the system DS is also part of the cluster. If you are using System DS 0 for Hosts inside a Cluster, any VM saved (stop, suspend, undeploy) **will not be able to be resumed after the upgrade process**.

Warning: Two drivers available in 4.2 are now discontinued: **ganglia** and **iscsi**.

- **iscsi** drivers have been moved out of the main OpenNebula distribution and are available (although not supported) as an addon.
- **ganglia** drivers have been moved out of the main OpenNebula distribution and are available (although not supported) as an addon.

2.3.1 Preparation

Before proceeding, make sure you don't have any VMs in a transient state (prolog, migr, epil, save). Wait until these VMs get to a final state (runn, suspended, stopped, done). Check the *Managing Virtual Machines guide* for more information on the VM life-cycle.

Stop OpenNebula and any other related services you may have running: EC2, OCCI, and Sunstone. As `oneadmin`, in the front-end:

```
$ sunstone-server stop
$ oneflow-server stop
$ econe-server stop
$ occi-server stop
$ one stop
```

2.3.2 Backup

Backup the configuration files located in **/etc/one**. You don't need to do a manual backup of your database, the onedb command will perform one automatically.

2.3.3 Installation

Follow the *Platform Notes* and the *Installation guide*, taking into account that you will already have configured the passwordless ssh access for oneadmin.

It is highly recommended **not to keep** your current `oned.conf`, and update the `oned.conf` file shipped with OpenNebula 4.4 to your setup. If for any reason you plan to preserve your current `oned.conf` file, read the *Compatibility Guide* and the complete oned.conf reference for 4.2 and *4.4* versions.

Warning: If you are upgrading from a version prior to 4.2, read the 3.8 upgrade guide, 4.0 upgrade guide and 4.2 upgrade guide for specific notes.

2.3.4 Database Upgrade

The database schema and contents are incompatible between versions. The OpenNebula daemon checks the existing DB version, and will fail to start if the version found is not the one expected, with the message 'Database version mismatch'.

You can upgrade the existing DB with the 'onedb' command. You can specify any Sqlite or MySQL database. Check the *onedb reference* for more information.

Warning: Make sure at this point that OpenNebula is not running. If you installed from packages, the service may have been started automatically.

After you install the latest OpenNebula, and fix any possible conflicts in oned.conf, you can issue the 'onedb upgrade -v' command. The connection parameters have to be supplied with the command line options, see the *onedb manpage* for more information. Some examples:

```
$ onedb upgrade -v --sqlite /var/lib/one/one.db
```

```
$ onedb upgrade -v -S localhost -u oneadmin -p oneadmin -d opennebula
```

If everything goes well, you should get an output similar to this one:

```
$ onedb upgrade -v -u oneadmin -d opennebula
MySQL Password:
Version read:
4.0.1 : Database migrated from 3.8.0 to 4.2.0 (OpenNebula 4.2.0) by onedb command.

MySQL dump stored in /var/lib/one/mysql_localhost_opennebula.sql
Use 'onedb restore' or restore the DB using the mysql command:
mysql -u user -h server -P port db_name < backup_file
  > Running migrator /usr/lib/one/ruby/onedb/4.2.0_to_4.3.80.rb
  > Done

  > Running migrator /usr/lib/one/ruby/onedb/4.3.80_to_4.3.85.rb
  > Done

  > Running migrator /usr/lib/one/ruby/onedb/4.3.85_to_4.3.90.rb
  > Done

  > Running migrator /usr/lib/one/ruby/onedb/4.3.90_to_4.4.0.rb
  > Done

Database migrated from 4.2.0 to 4.4.0 (OpenNebula 4.4.0) by onedb command.
```

If you receive the message "ATTENTION: manual intervention required", read the section *Manual Intervention Required* below.

Warning: Make sure you keep the backup file. If you face any issues, the onedb command can restore this backup, but it won't downgrade databases to previous versions.

2.3.5 Check DB Consistency

After the upgrade is completed, you should run the command `onedb fsck`.

First, move the 4.2 backup file created by the upgrade command to a save place.

```
$ mv /var/lib/one/mysql_localhost_opennebula.sql /path/for/one-backups/
```

Then execute the following command:

```
$ onedb fsck -S localhost -u oneadmin -p oneadmin -d opennebula
MySQL dump stored in /var/lib/one/mysql_localhost_opennebula.sql
Use 'onedb restore' or restore the DB using the mysql command:
mysql -u user -h server -P port db_name < backup_file

Total errors found: 0
```

2.3.6 Update the Drivers

You should be able now to start OpenNebula as usual, running 'one start' as oneadmin. At this point, execute `onehost sync` to update the new drivers in the hosts.

Warning: Doing `onehost sync` is important. If the monitorization drivers are not updated, the hosts will behave erratically.

2.3.7 Setting new System DS

With the new *multi-system DS* functionality, it is now required that the system DS is also part of the cluster. If you are using System DS 0 for Hosts inside a Cluster, any VM saved (stop, suspend, undeploy) **will not be able to be resumed after the upgrade process**.

You will need to have at least one system DS in each cluster. If you don't already, create new system DS with the same definition as the system DS 0 (TM_MAD driver). Depending on your setup this may or may not require additional configuration on the hosts.

You may also try to recover saved VMs (stop, suspend, undeploy) following the steps described in this thread of the users mailing list.

2.3.8 Testing

OpenNebula will continue the monitoring and management of your previous Hosts and VMs.

As a measure of caution, look for any error messages in oned.log, and check that all drivers are loaded successfully. After that, keep an eye on oned.log while you issue the onevm, onevnet, oneimage, oneuser, onehost **list** commands. Try also using the **show** subcommand for some resources.

2.3.9 Restoring the Previous Version

If for any reason you need to restore your previous OpenNebula, follow these steps:

- With OpenNebula 4.4 still installed, restore the DB backup using 'onedb restore -f'
- Uninstall OpenNebula 4.4, and install again your previous version.
- Copy back the backup of /etc/one you did to restore your configuration.

2.3.10 Known Issues

If the MySQL database password contains specials characters, such as `@` or `#`, the onedb command will fail to connect to it.

The workaround is to temporarily change the oneadmin's password to an ASCII string. The set password statement can be used for this:

```
$ mysql -u oneadmin -p

mysql> SET PASSWORD = PASSWORD('newpass');
```

2.3.11 Manual Intervention Required

If you have a datastore configured to use a tm driver not included in the OpenNebula distribution, the onedb upgrade command will show you this message:

```
ATTENTION: manual intervention required

The Datastore <id> <name> is using the
custom TM MAD '<tm_mad>'. You will need to define new
configuration parameters in oned.conf for this driver, see
http://opennebula.org/documentation:rel4.4:upgrade
```

In OpenNebula 4.4, each tm_mad driver has a TM_MAD_CONF section in oned.conf. If you developed the driver, it should be fairly easy to define the required information looking at the existing ones:

```
# The  configuration for each driver is defined in TM_MAD_CONF. These
# values are used when creating a new datastore and should not be modified
# since they define the datastore behaviour.
#   name      : name of the transfer driver, listed in the -d option of the
#               TM_MAD section
#   ln_target : determines how the persistent images will be cloned when
#               a new VM is instantiated.
#       NONE: The image will be linked and no more storage capacity will be used
#       SELF: The image will be cloned in the Images datastore
#       SYSTEM: The image will be cloned in the System datastore
#   clone_target : determines how the non persistent images will be
#                  cloned when a new VM is instantiated.
#       NONE: The image will be linked and no more storage capacity will be used
#       SELF: The image will be cloned in the Images datastore
#       SYSTEM: The image will be cloned in the System datastore
#   shared : determines if the storage holding the system datastore is shared
#            among the different hosts or not. Valid values: "yes" or "no"

TM_MAD_CONF = [
    name        = "lvm",
    ln_target   = "NONE",
    clone_target= "SELF",
    shared      = "yes"
]
```

CHAPTER

THREE

QUICK STARTS

3.1 Quickstart: OpenNebula 4.4 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.

3.1.1 Package Layout

- opennebula-server: OpenNebula Daemons
- opennebula: OpenNebula CLI commands
- opennebula-sunstone: OpenNebula's web GUI
- opennebula-ozones: 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.

3.1.2 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
```

3.1.3 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.

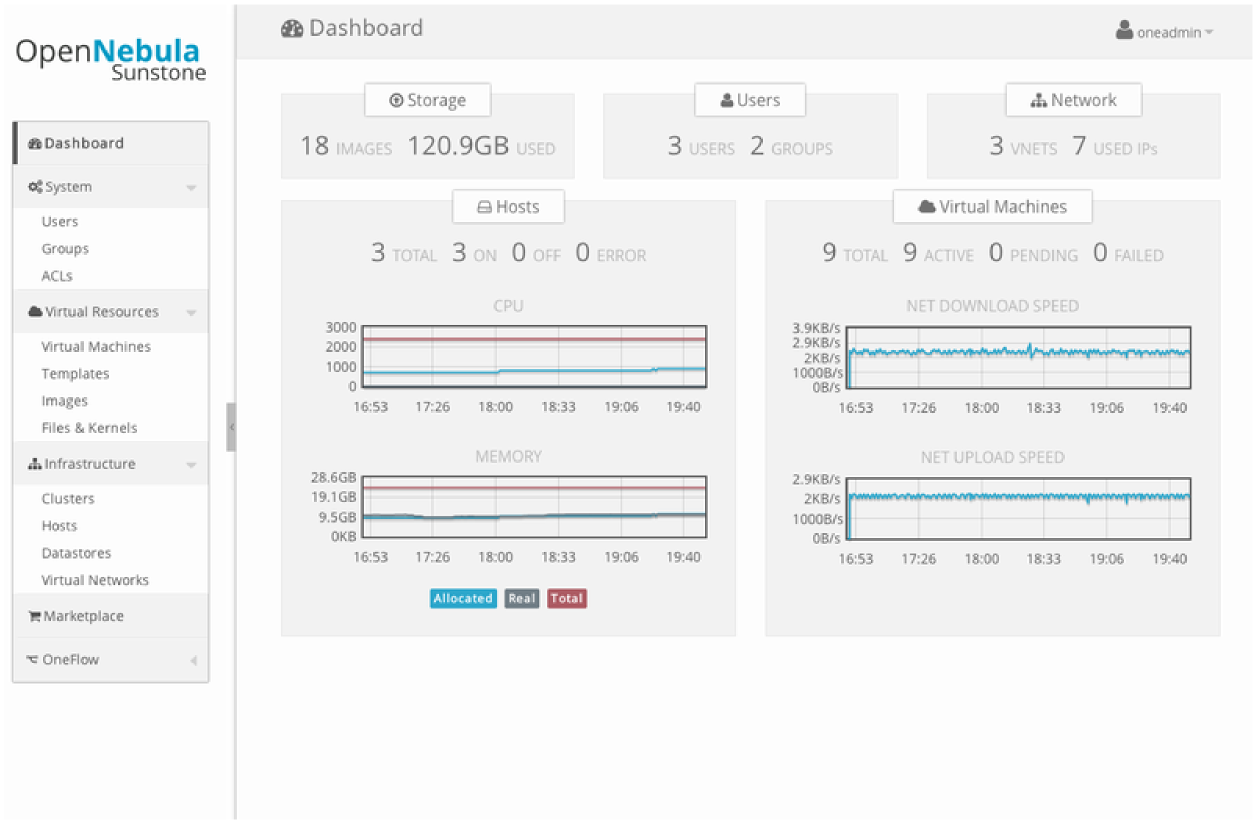
Mount the NFS share:

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

3.1.4 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-6.4_x86_64" \
    --path "http://us.cloud.centos.org/i/one/c6-x86_64-20130910-1.qcow2.bz2" \
    --driver qcow2 \
    --datastore default

$ onetemplate create --name "CentOS-6.4" --cpu 1 --vcpu 1 --memory 512 \
    --arch x86_64 --disk "CentOS-6.4_x86_64" --nic "private" --vnc \
    --ssh
```

(The image will be downloaded from http://wiki.centos.org/Cloud/OpenNebula)

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-6.4" --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`.

3.1.5 Further information

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

3.2 Quickstart: OpenNebula 4.4 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.

3.2.1 Package Layout

- opennebula-server: OpenNebula Daemons
- opennebula: OpenNebula CLI commands
- opennebula-sunstone: OpenNebula's web GUI
- opennebula-ozones: 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.

3.2.2 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
```

3.2.3 Step 2. Installation in the Nodes

Warning: The process to install Xen might change in the future. Please refer to the CentOS documenation 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
```

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 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.

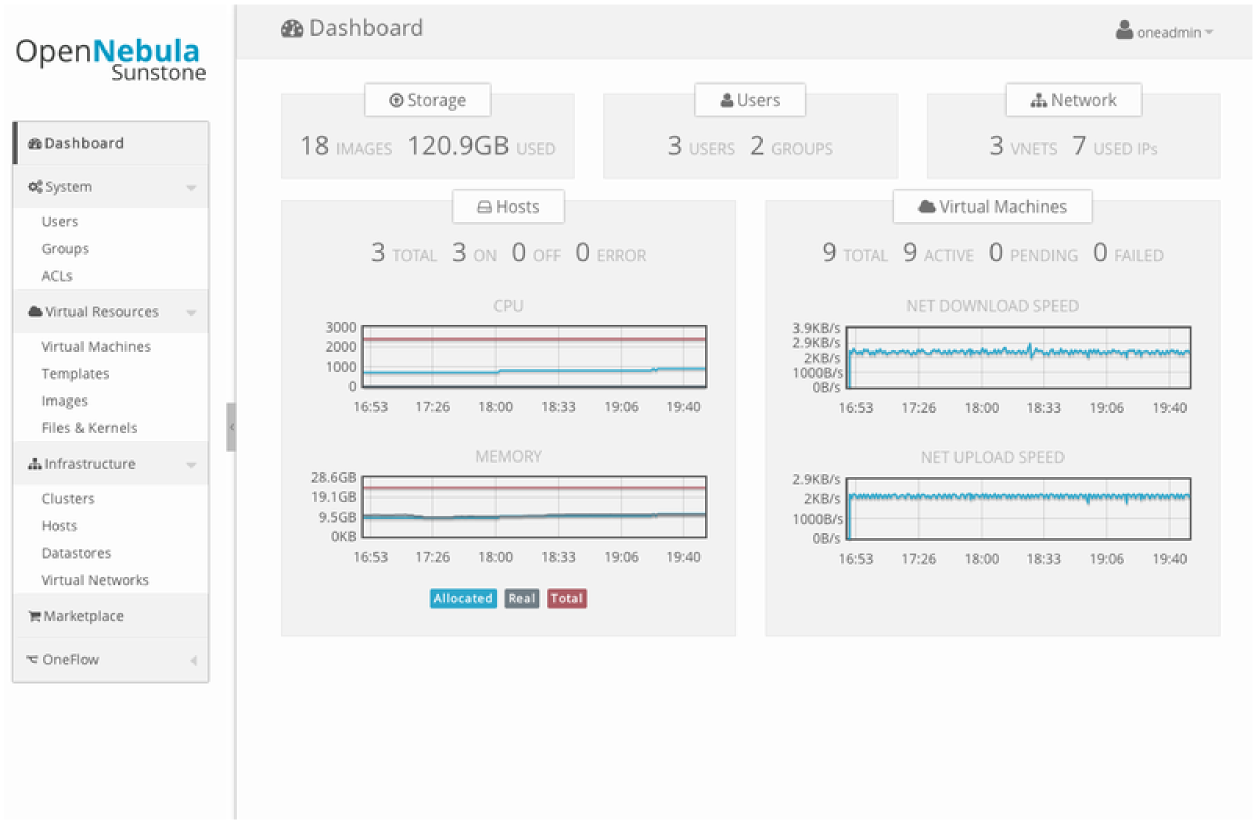
Mount the NFS share:

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

3.2.4 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

$ oneimage create --name "CentOS-6.4_x86_64" \
    --path "http://us.cloud.centos.org/i/one/c6-x86_64-20130910-1.qcow2.bz2" \
    --driver qcow2 \
    --datastore default

$ onetemplate create --name "CentOS-6.4" --cpu 1 --vcpu 1 --memory 512 \
    --arch x86_64 --disk "CentOS-6.4_x86_64" --nic "private" --vnc \
    --ssh
```

(The image will be downloaded from http://wiki.centos.org/Cloud/OpenNebula)

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-6.4
```

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-6.4" --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`.

3.2.5 Further information

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

3.3 Quickstart: OpenNebula 4.4 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. 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.

3.3.1 Package Layout

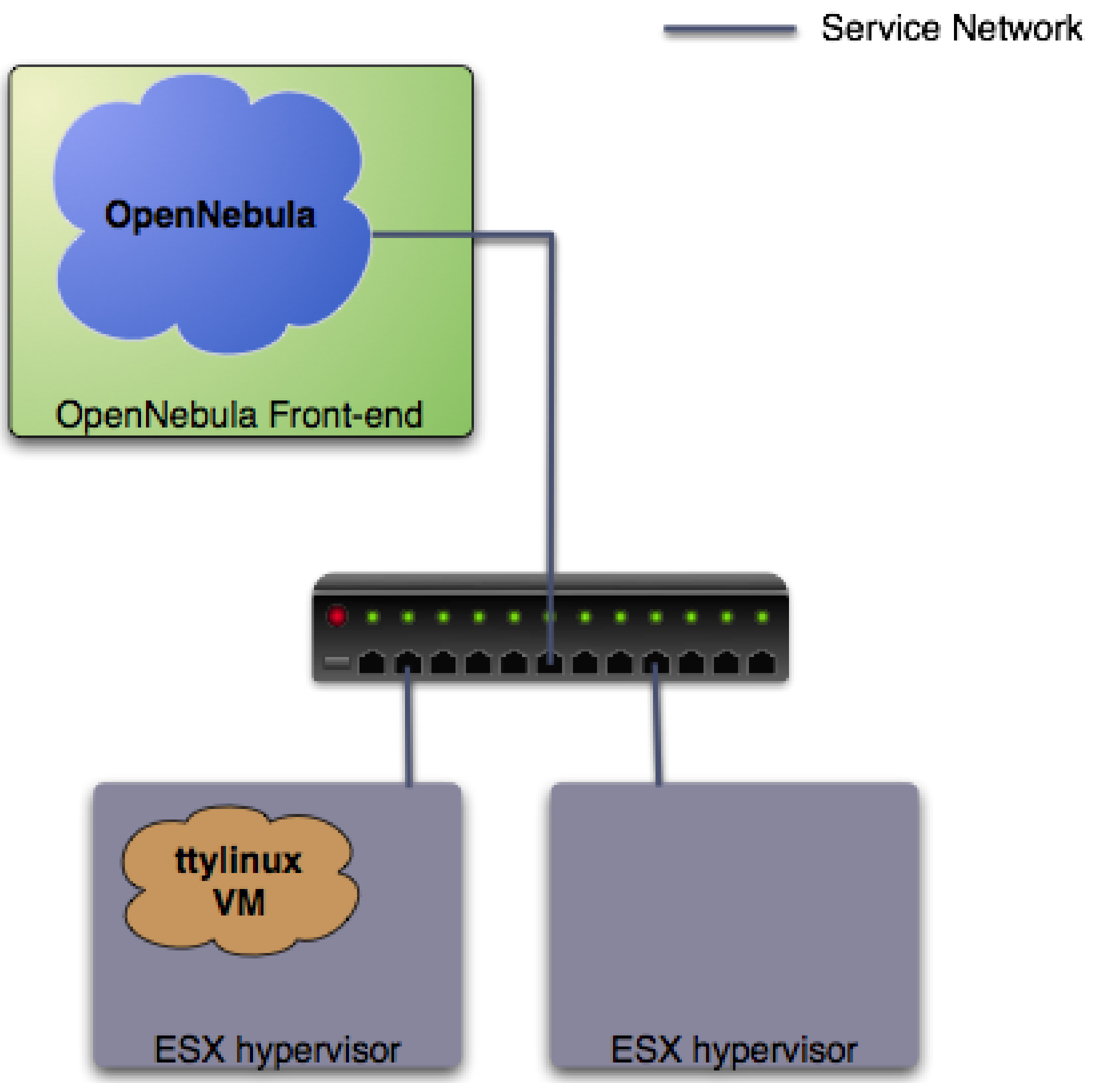
- opennebula-server: OpenNebula Daemons
- opennebula: OpenNebula CLI commands
- opennebula-sunstone: OpenNebula's web GUI
- opennebula-ozones: OpenNebula's web GUI
- opennebula-java: OpenNebula Java API
- opennebula-node-kvm: Installs dependencies required by OpenNebula in the nodes
- opennebula-gate: 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.

3.3.2 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

```
$ sudo rpm -Uvh http://download.fedoraproject.org/pub/epel/6/i386/epel-release-6-7.noarch.rpm
$ sudo 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.

3.3.3 Step 2. OpenNebula Front-end Set-up

2.1 OpenNebula installation

The first step is to install OpenNebula in the front-end. Please download OpeNebula 4.4 from here, choosing the CentOS package.

Once it is downloaded to the front-end, you need to untar it:

```
$ tar xvzf CentOS-6-opennebula-4.0.0-1.tar.gz
```

And then install all the needed packages:

```
$ sudo yum localinstall opennebula-4.4.0/*.rpm
```

Let's install noVNC to gain access to the VMs:

```
$ sudo /usr/share/one/install_novnc.sh
```

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). Follow these instructions:

```
$ sudo vi /etc/sysconfig/selinux
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=disabled
# SELINUXTYPE= can take one of these two values:
#     targeted - Targeted processes are protected,
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted

$ sudo setenforce 0

$ sudo 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:

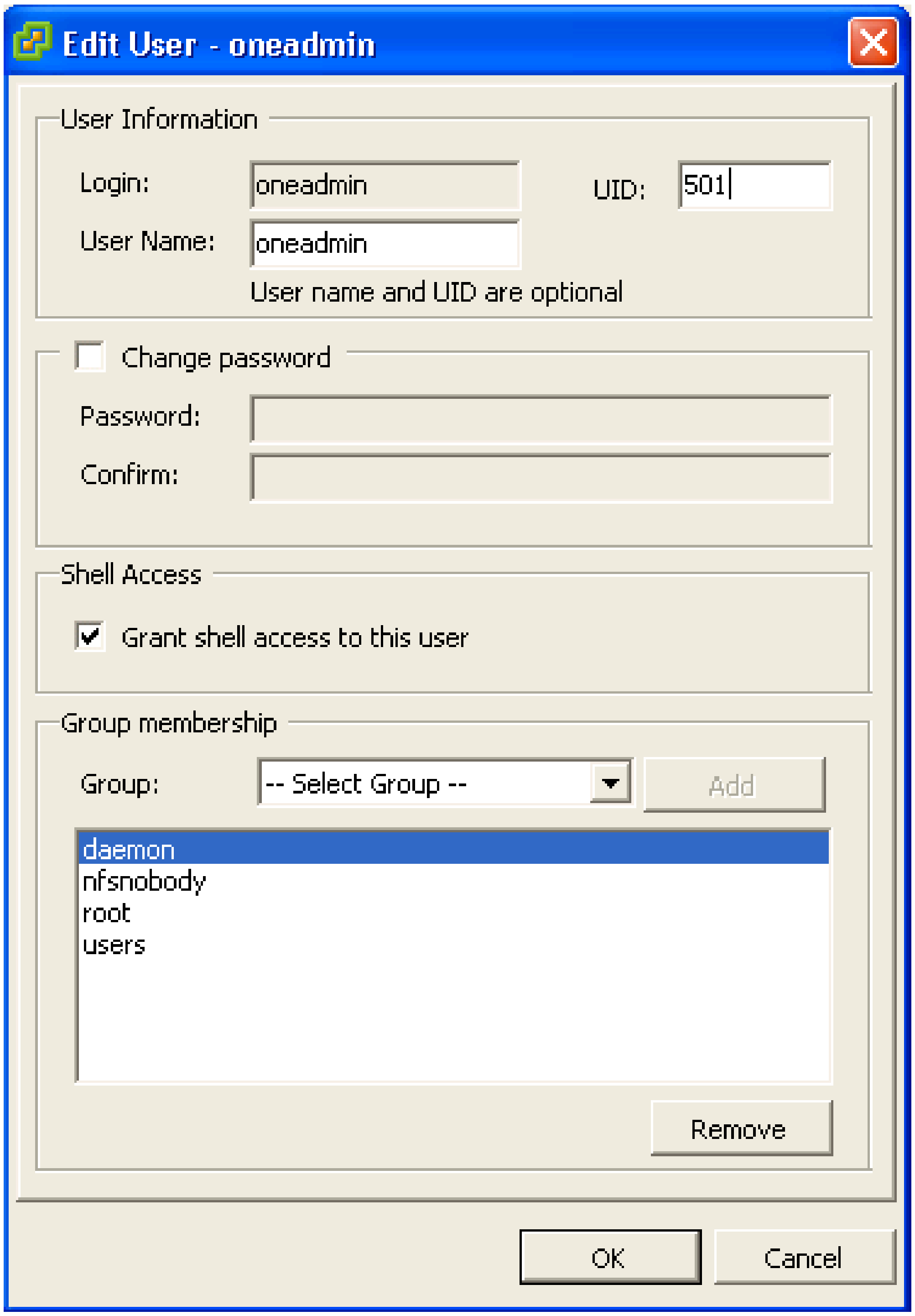
```
$ ping <esx-ip>
```

3.3.4 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: oneadmin   UID: 501

User Name: oneadmin

User name and UID are optional

☐ Change password

Password:

Confirm:

Shell Access

☑ Grant shell access to this user

Group membership

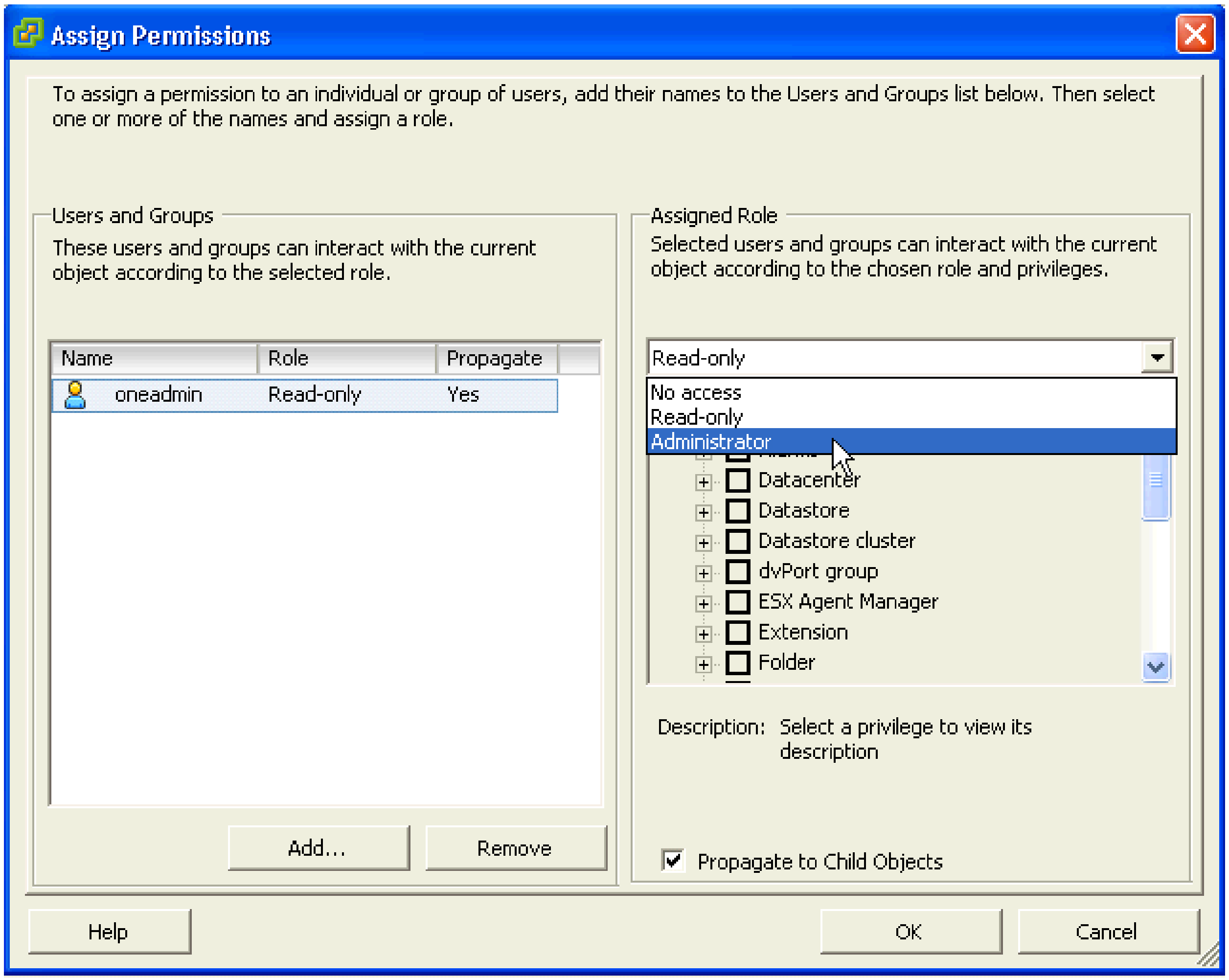
Group: -- Select Group --   Add

daemon
nfsnobody
root
users

Remove

OK   Cancel

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



Assign Permissions

To assign a permission to an individual or group of users, add their names to the Users and Groups list below. Then select one or more of the names and assign a role.

Users and Groups

These users and groups can interact with the current object according to the selected role.

Name	Role	Propagate
oneadmin	Read-only	Yes

Assigned Role

Selected users and groups can interact with the current object according to the chosen role and privileges.

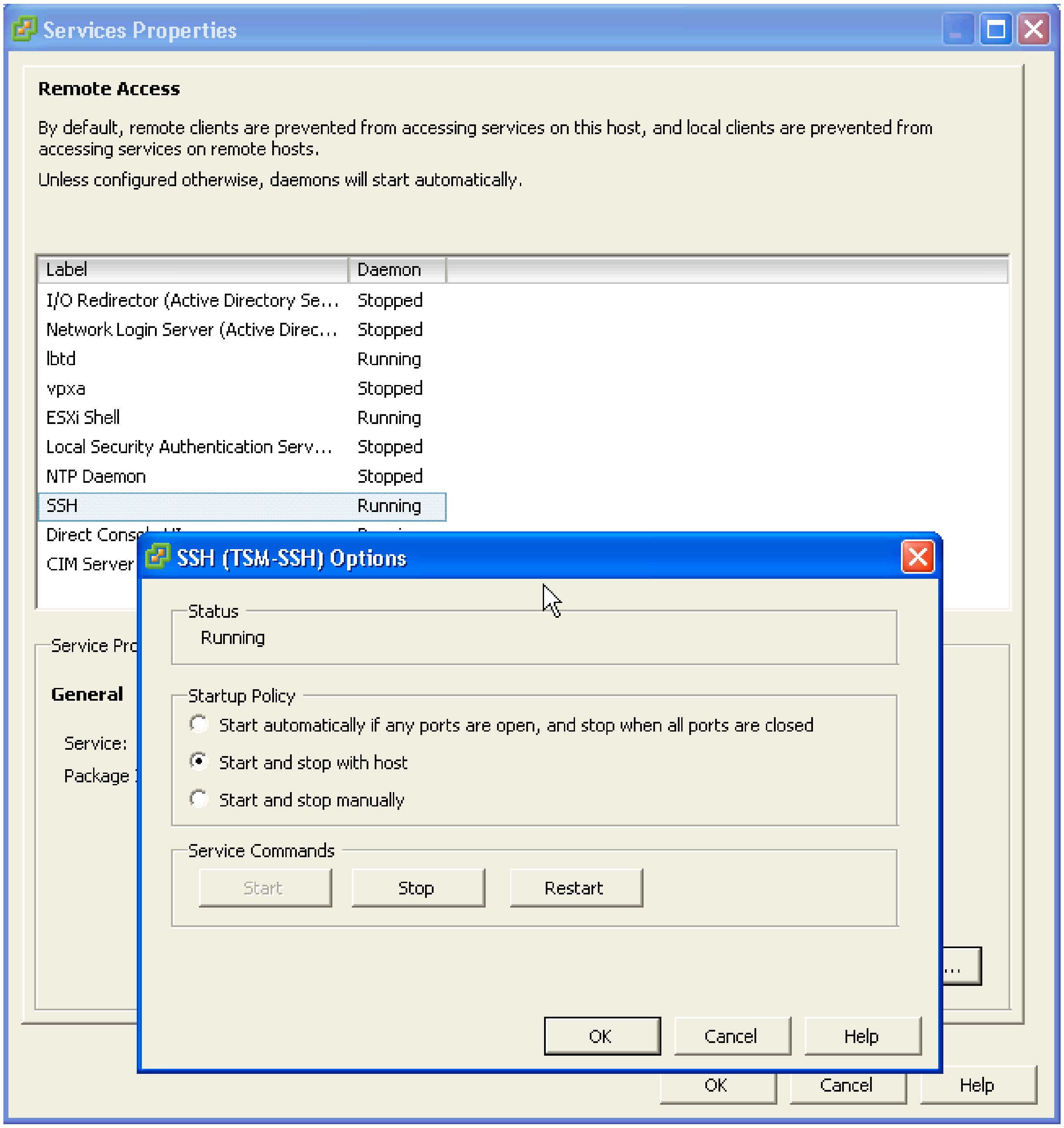
Read-only

Description: Select a privilege to view its description

Propagate to Child Objects

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.


Services Properties
Remote Access
By default, remote clients are prevented from accessing services on this host, and local clients are prevented from accessing services on remote hosts.
Unless configured otherwise, daemons will start automatically.
Label
Daemon
I/O Redirector (Active Directory Se... Stopped
Network Login Server (Active Direc... Stopped
lbtd Running
vpxa Stopped
ESXi Shell Running
Local Security Authentication Serv... Stopped
NTP Daemon Stopped
SSH Running
Direct Conso
CIM Server
Service Pro
General
Service:
Package
SSH (TSM-SSH) Options
Status
Running
Startup Policy
Start automatically if any ports are open, and stop when all ports are closed
Start and stop with host
Start and stop manually
Service Commands
Start
Stop
Restart
OK
Cancel
Help
OK
Cancel
Help

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:

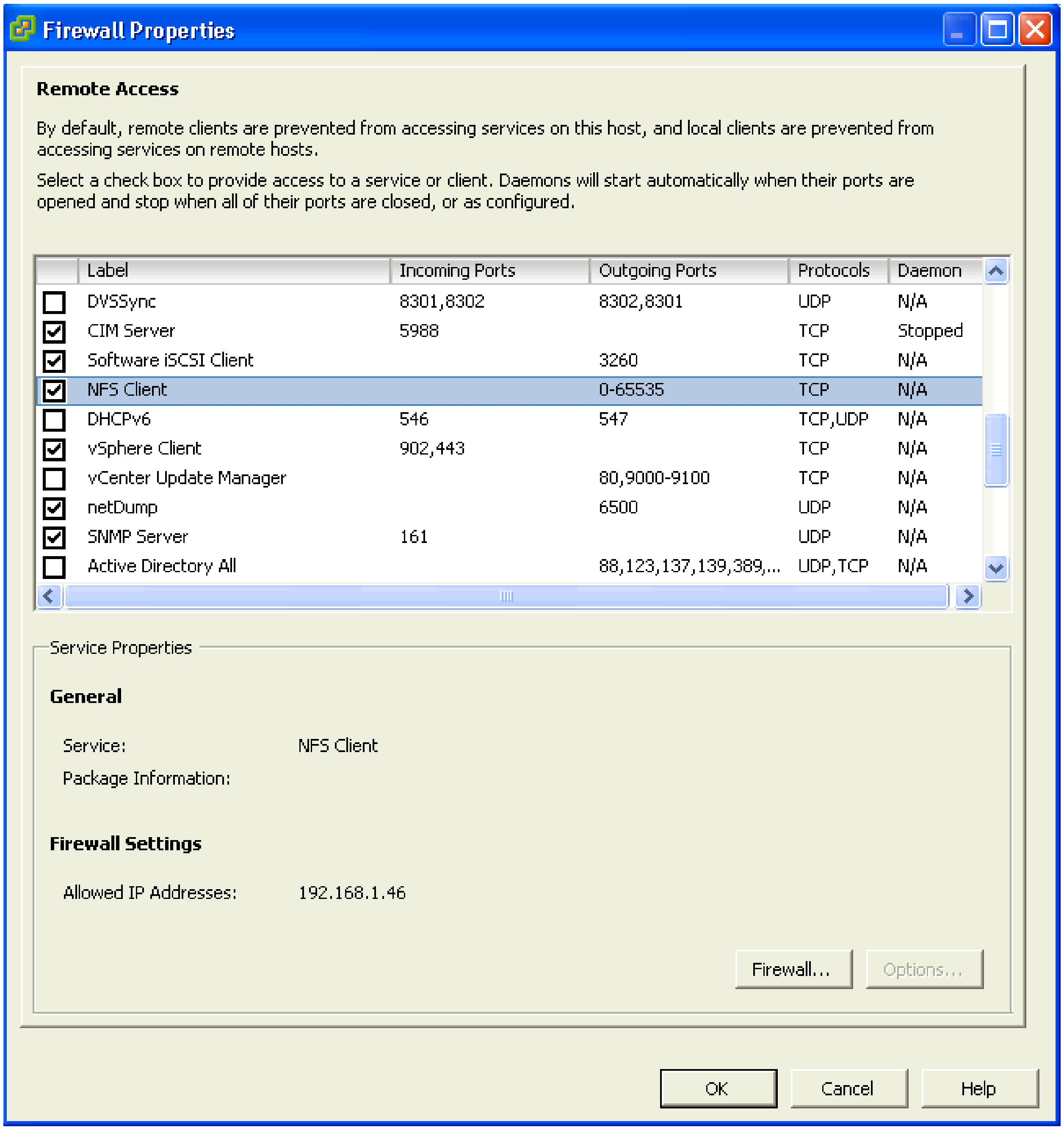
```
$ ssh <esx-ip>
 Enter the password you set in the step 3.1

$ 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:



Remote Access

By default, remote clients are prevented from accessing services on this host, and local clients are prevented from accessing services on remote hosts.

Select a check box to provide access to a service or client. Daemons will start automatically when their ports are opened and stop when all of their ports are closed, or as configured.

	Label	Incoming Ports	Outgoing Ports	Protocols	Daemon
☐	DVSSync	8301,8302	8302,8301	UDP	N/A
☑	CIM Server	5988		TCP	Stopped
☑	Software iSCSI Client		3260	TCP	N/A
☑	NFS Client		0-65535	TCP	N/A
☐	DHCPv6	546	547	TCP,UDP	N/A
☑	vSphere Client	902,443		TCP	N/A
☐	vCenter Update Manager		80,9000-9100	TCP	N/A
☑	netDump		6500	UDP	N/A
☑	SNMP Server	161		UDP	N/A
☐	Active Directory All		88,123,137,139,389,...	UDP,TCP	N/A

Service Properties

General

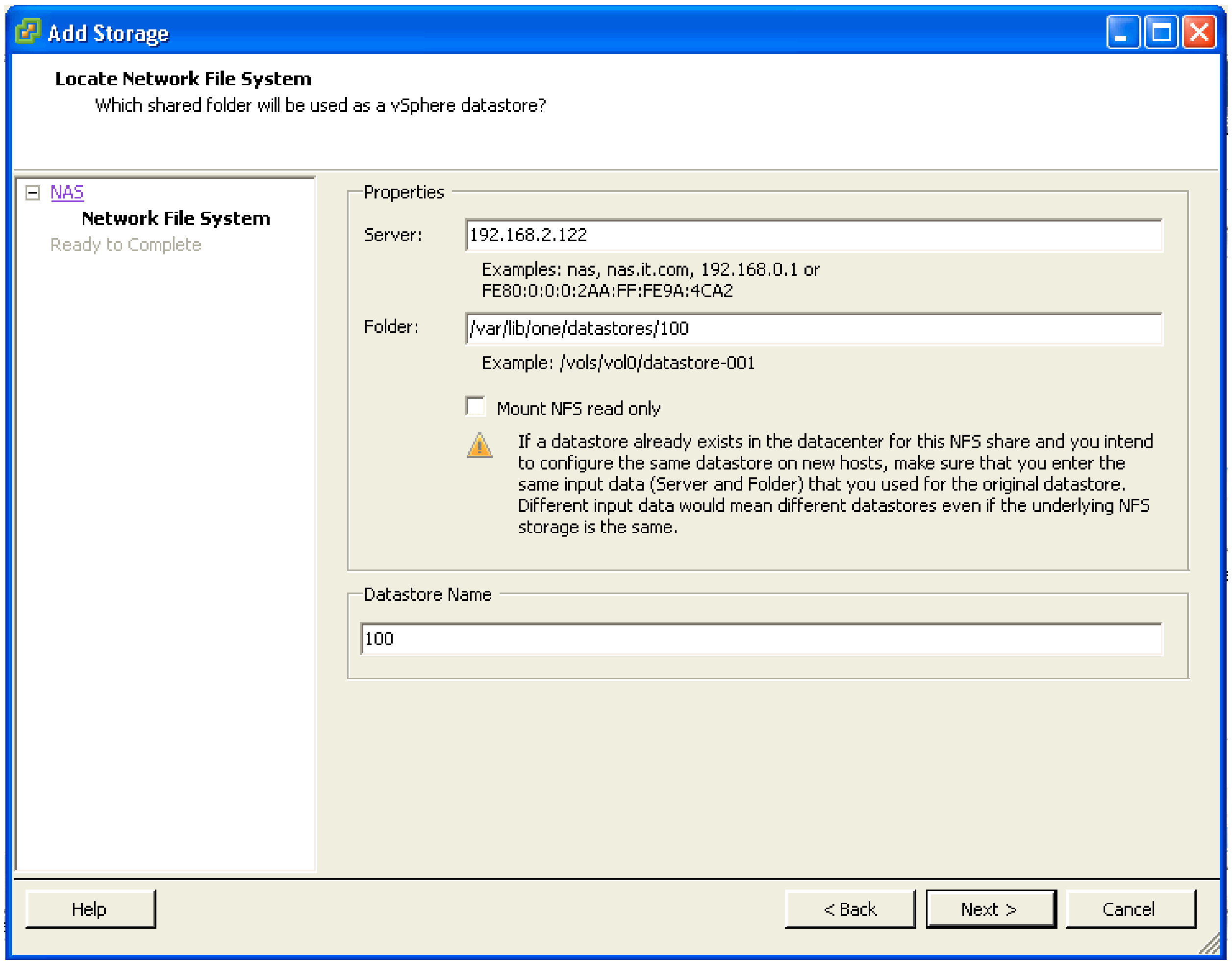
Service:	NFS Client
Package Information:	

Firewall Settings

Allowed IP Addresses:	192.168.1.46

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


Add Storage

Locate Network File System

Which shared folder will be used as a vSphere datastore?

Properties

Server: 192.168.2.122

Examples: nas, nas.it.com, 192.168.0.1 or FE80:0:0:0:2AA:FF:FE9A:4CA2

Folder: /var/lib/one/datastores/100

Example: /vols/vol0/datastore-001

Mount NFS read only

If a datastore already exists in the datacenter for this NFS share and you intend to configure the same datastore on new hosts, make sure that you enter the same input data (Server and Folder) that you used for the original datastore. Different input data would mean different datastores even if the underlying NFS storage is the same.

Datastore Name

100

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
```

3.3.5 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"
```

Warning: Do not edit `:libvirt_uri:`, 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
```

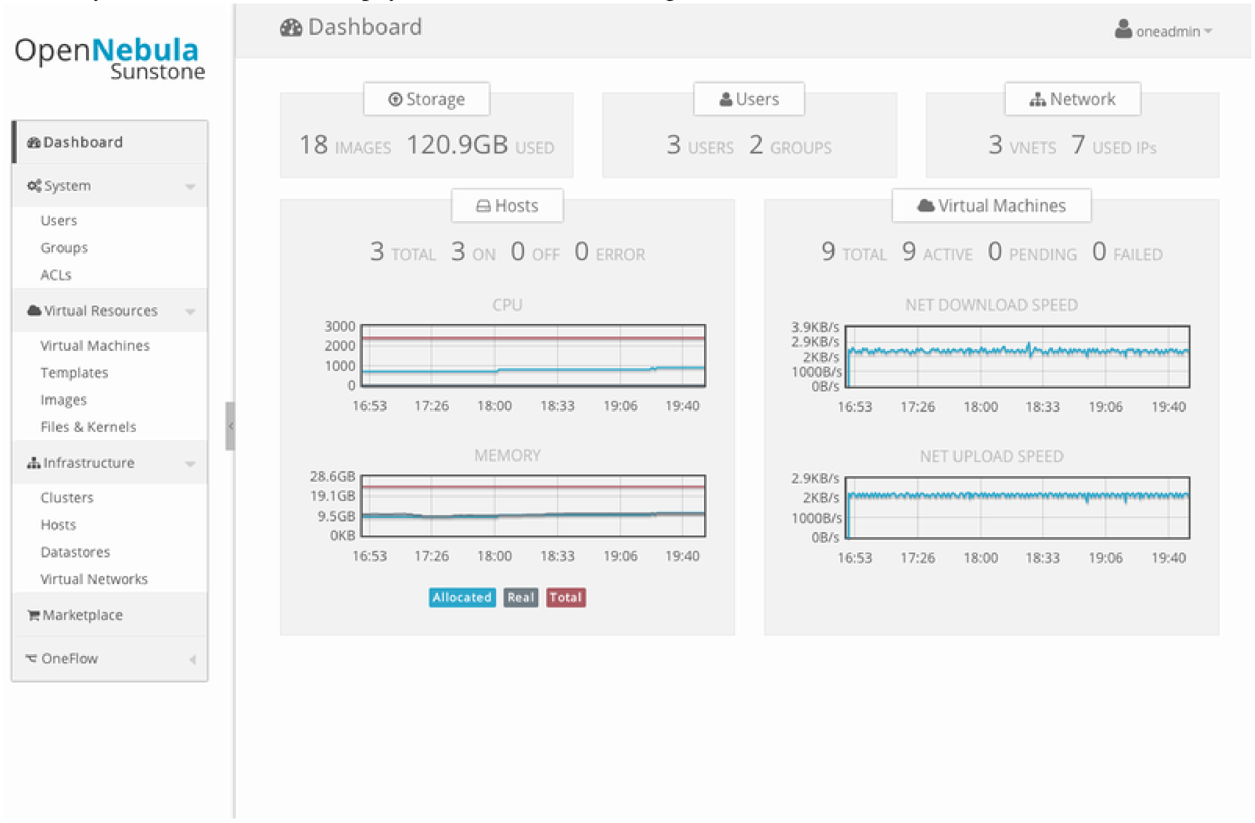
4.5 Create a regular cloud user

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

3.3.6 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)



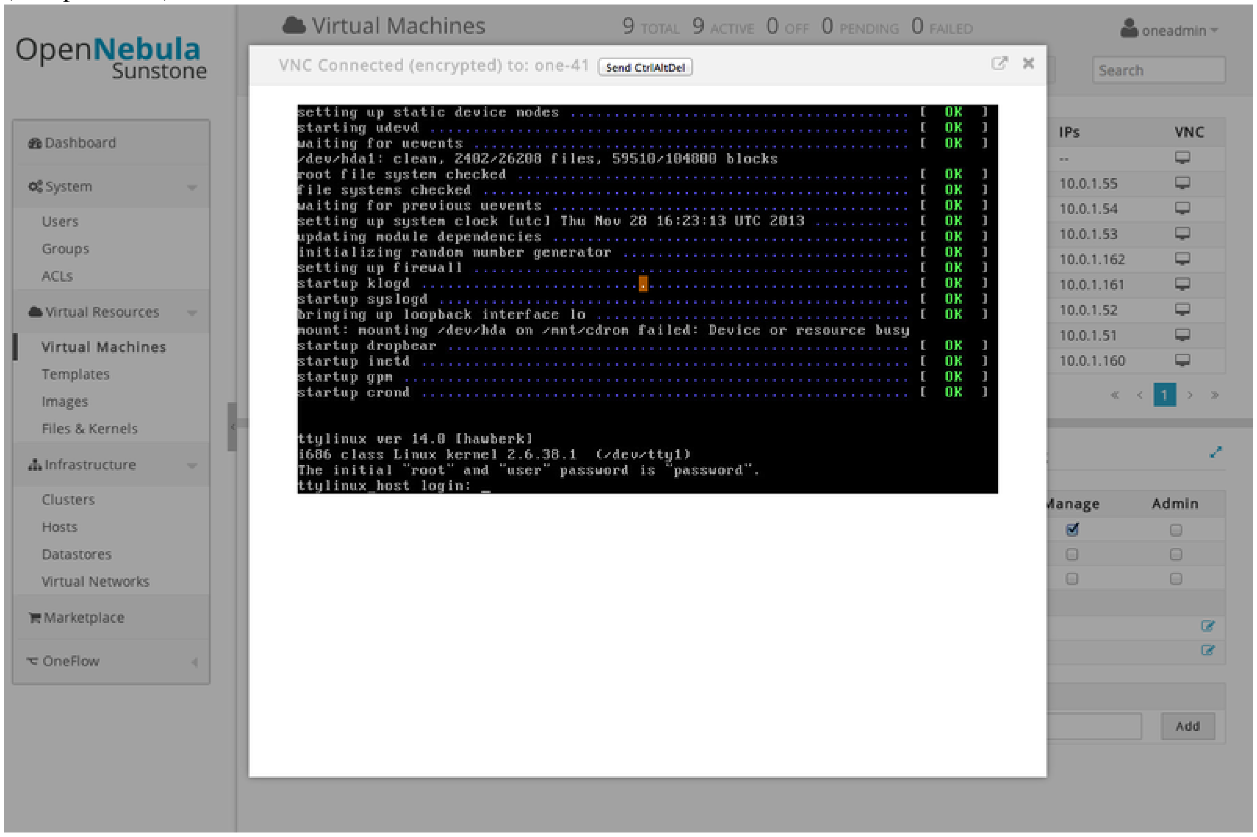
It is time to launch our first VM. Let's use one of the pre created appliances found in the marketplace.

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).



```
setting up static device nodes ........................................ [  OK  ]
starting udevd ........................................................ [  OK  ]
waiting for uevents ................................................... [  OK  ]
/dev/hda1: clean, 2402/26208 files, 59510/104800 blocks
root file system checked .............................................. [  OK  ]
file systems checked .................................................. [  OK  ]
waiting for previous uevents .......................................... [  OK  ]
setting up system clock [utc] Thu Nov 28 16:23:13 UTC 2013 ............ [  OK  ]
updating module dependencies .......................................... [  OK  ]
initializing random number generator .................................. [  OK  ]
setting up firewall ................................................... [  OK  ]
startup klogd ......................................................... [  OK  ]
startup syslogd ....................................................... [  OK  ]
bringing up loopback interface lo ..................................... [  OK  ]
mount: mounting /dev/hda on /mnt/cdrom failed: Device or resource busy
startup dropbear ...................................................... [  OK  ]
startup inetd ......................................................... [  OK  ]
startup gpm ........................................................... [  OK  ]
startup crond ......................................................... [  OK  ]


ttylinux ver 14.0 [hawberk]
i686 class Linux kernel 2.6.38.1  (/dev/tty1)
The initial "root" and "user" password is "password".
ttylinux_host login: _
```

IPs	VNC
--	
10.0.1.55	
10.0.1.54	
10.0.1.53	
10.0.1.162	
10.0.1.161	
10.0.1.52	
10.0.1.51	
10.0.1.160	

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!

3.3.7 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!

3.4 Quickstart: OpenNebula 4.4 on Ubuntu 12.04 and KVM

The purpose of this guide is to provide users with step by step guide to install OpenNebula using Ubuntu 12.04 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 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.

3.4.1 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

3.4.2 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/12.04 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
```

3.4.3 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/12.04 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 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
```

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

```
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
```

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 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/
```

2.5. Configure Qemu

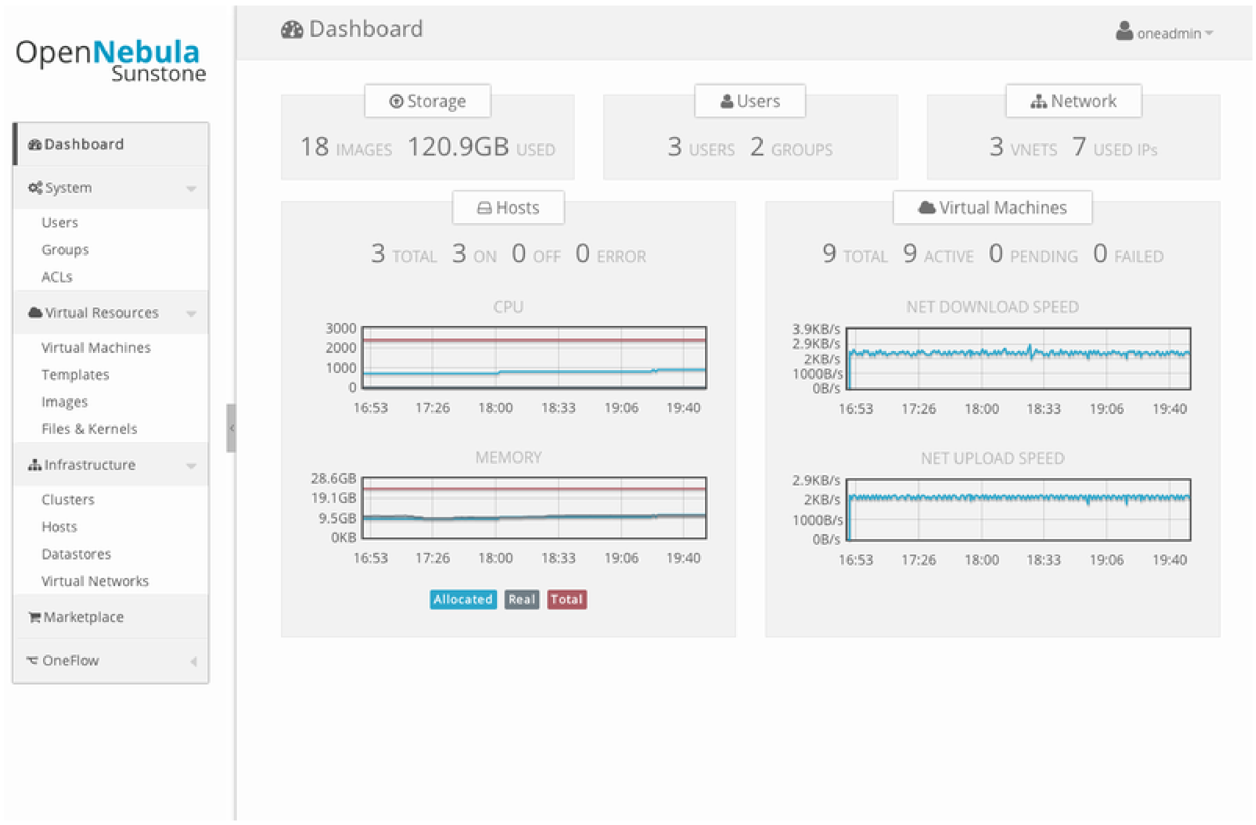
The `oneadmin` user must be able to manage libvirt as root:

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

3.4.4 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-6.4_x86_64" \
    --path "http://us.cloud.centos.org/i/one/c6-x86_64-20130910-1.qcow2.bz2" \
    --driver qcow2 \
    --datastore default

$ onetemplate create --name "CentOS-6.4" --cpu 1 --vcpu 1 --memory 512 \
    --arch x86_64 --disk "CentOS-6.4_x86_64" --nic "private" --vnc \
    --ssh
```

(The image will be downloaded from http://wiki.centos.org/Cloud/OpenNebula)

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-6.4" --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`.

3.4.5 Further information

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