



OpenNebula 5.0 Deployment guide

Release 5.0.1

OpenNebula Systems

Jun 27, 2016

This document is being provided by OpenNebula Systems under the Creative Commons Attribution-NonCommercial-Share Alike License.

THE DOCUMENT IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE DOCUMENT.

1	Cloud Design	1
1.1	Overview	1
1.2	Open Cloud Architecture	1
1.3	VMware Cloud Architecture	7
1.4	OpenNebula Provisioning Model	13
2	OpenNebula Installation	19
2.1	Overview	19
2.2	Front-end Installation	19
2.3	MySQL Setup	24
3	Node Installation	26
3.1	Overview	26
3.2	KVM Node Installation	27
3.3	vCenter Node Installation	33
3.4	Verify your Installation	37
4	Authentication Setup	45
4.1	Overview	45
4.2	SSH Authentication	46
4.3	x509 Authentication	48
4.4	LDAP Authentication	51
5	Sunstone Setup	56
5.1	Overview	56
5.2	Sunstone Installation & Configuration	57
5.3	Sunstone Views	65
5.4	User Security and Authentication	74
5.5	Cloud Servers Authentication	79
5.6	Configuring Sunstone for Large Deployments	82
6	VMware Infrastructure Setup	88
6.1	Overview	88
6.2	vCenter Node	88
6.3	vCenter Datastore	93
6.4	vCenter Networking	95
7	Open Cloud Host Setup	98
7.1	Overview	98
7.2	KVM Driver	98
7.3	Monitoring	107

7.4	PCI Passthrough	110
8	Open Cloud Storage Setup	116
8.1	Overview	116
8.2	Filesystem Datastore	118
8.3	Ceph Datastore	122
8.4	LVM Datastore	126
8.5	Raw Device Mapping (RDM) Datastore	128
8.6	iSCSI - Libvirt Datastore	130
8.7	The Kernels & Files Datastore	132
9	Open Cloud Networking Setup	134
9.1	Overview	134
9.2	Node Setup	135
9.3	Bridged Networking	136
9.4	802.1Q VLAN Networks	138
9.5	VXLAN Networks	139
9.6	Open vSwitch Networks	140
10	References	142
10.1	Overview	142
10.2	ONED Configuration	142
10.3	Logging & Debugging	156
10.4	Onedb Tool	159
10.5	Large Deployments	162

CLOUD DESIGN

1.1 Overview

The first step of building a reliable, useful and successful cloud is to decide a clear design. This design needs to be aligned with the expected use of the cloud, and it needs to describe which data center components are going to be part of the cloud. This comprises i) all the infrastructure components such as networking, storage, authorization and virtualization back-ends, as well as the ii) planned dimension of the cloud (characteristics of the workload, numbers of users and so on) and the iii) provisioning workflow, ie, how end users are going to be isolated and using the cloud.

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 Chapter provides information to plan an OpenNebula cloud based on *KVM* or *vCenter*. With this information, you will be able to easily architect and dimension your deployment, as well as understand the technologies involved in the management of virtualized resources and their relationship.

1.1.1 How Should I Read This Chapter

This is the first Chapter to read, as it introduces the needed concepts to correctly define a cloud architecture.

Within this Chapter, as first step a design of the cloud and its dimension should be drafted. For KVM clouds proceed to *Open Cloud Architecture* and for vCenter clouds read *VMware Cloud Architecture*.

Then you could read the *OpenNebula Provisioning Model* to identify the wanted model to provision resources to end users. In a small installation with a few hosts, you can skip this provisioning model guide and use OpenNebula without giving much thought to infrastructure partitioning and provisioning. But for medium and large deployments you will probably want to provide some level of isolation and structure.

Once the cloud architecture has been designed the next step would be to learn how to install the *OpenNebula front-end*.

1.1.2 Hypervisor Compatibility

Section	Compatibility
<i>Open Cloud Architecture</i>	This Section applies to KVM.
<i>VMware Cloud Architecture</i>	This Section applies to vCenter.
<i>OpenNebula Provisioning Model</i>	This Section applies to both KVM and vCenter.

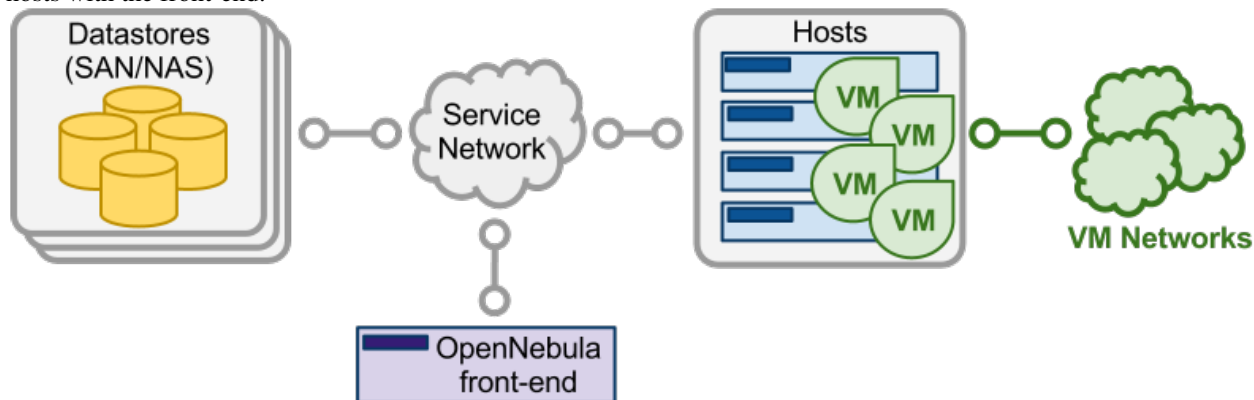
1.2 Open Cloud Architecture

Enterprise cloud computing is the next step in the evolution of data center (DC) virtualization. OpenNebula is a simple but feature-rich and flexible solution to build and manage enterprise clouds and virtualized DCs, that combines existing

virtualization technologies with advanced features for multi-tenancy, automatic provision and elasticity. OpenNebula follows a bottom-up approach driven by sysadmins, devops and users real needs.

1.2.1 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.



A cloud architecture is defined by three components: storage, networking and virtualization. Therefore, 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 Section briefly describes the different choices that you can make for the management of the different subsystems. If your specific services are not supported we recommend to check the drivers available in the [Add-on Catalog](#). We also provide information and support about how to develop new drivers.

1.2.2 Dimensioning the Cloud

The dimension of a cloud infrastructure can be directly inferred from the expected workload in terms of VMs that the cloud infrastructure must sustain. This workload is also tricky to estimate, but this is a crucial exercise to build an efficient cloud.

The main aspects to take into account at the time of dimensioning the OpenNebula cloud follows.

OpenNebula front-end

The minimum recommended specs are for the OpenNebula front-end are:

Resources	Minimum Recommended configuration
Memory	2 GB
CPU	1 CPU (2 cores)
Disk Size	100 GB
Network	2 NICS

The maximum number of servers (virtualization hosts) that can be managed by a single OpenNebula instance strongly depends on the performance and scalability of the underlying platform infrastructure, mainly the storage subsystem. The general recommendation is that no more than 500 servers managed by a single instance, but there are users with 1,000 servers in each zone. Related to this, read the section about *how to tune OpenNebula for large deployments*.

KVM nodes

Regarding the dimensions of the KVM virtualization nodes:

- **CPU:** without overcommitment, each CPU core assigned to a VM must exist as a physical CPU core. By example, for a workload of 40 VMs with 2 CPUs, the cloud will need 80 physical CPUs. These 80 physical CPUs can be spread among different hosts: 10 servers with 8 cores each, or 5 server of 16 cores each. With overcommitment, however, CPU dimension can be planned ahead, using the `CPU` and `VCPU` attributes: `CPU` states physical CPUs assigned to the VM, while `VCPU` states virtual CPUs to be presented to the guest OS.
- **MEMORY:** Planning for memory is straightforward, as by default *there is no overcommitment of memory* in OpenNebula. It is always a good practice to count 10% of overhead by the hypervisor (this is not an absolute upper limit, it depends on the hypervisor). So, in order to sustain a VM workload of 45 VMs with 2GB of RAM each, 90GB of physical memory is needed. The number of hosts is important, as each one will incur a 10% overhead due to the hypervisors. For instance, 10 hypervisors with 10GB RAM each will contribute with 9GB each (10% of 10GB = 1GB), so they will be able to sustain the estimated workload. The rule of thumb is having at least 1GB per core, but this also depends on the expected workload.

Storage

It is important to understand how OpenNebula uses storage, mainly the difference between system and image datastore.

- The **image datastore** is where OpenNebula stores all the images registered that can be used to create VMs, so the rule of thumb is to devote enough space for all the images that OpenNebula will have registered.
- The **system datastore** is where the VMs that are currently running store their disks. It is trickier to estimate correctly since volatile disks come into play with no counterpart in the image datastore (volatile disks are created on the fly in the hypervisor).

One valid approach is to limit the storage available to users by defining quotas in the number of maximum VMs and also the Max Volatile Storage a user can demand, and ensuring enough system and image datastore space to comply with the limit set in the quotas. In any case, OpenNebula allows cloud administrators to add more system and images datastores if needed.

Dimensioning storage is a critical aspect, as it is usually the cloud bottleneck. It very much depends on the underlying technology. As an example, in Ceph for a medium size cloud at least three servers are needed for storage with 5 disks each of 1TB, 16Gb of RAM, 2 CPUs of 4 cores each and at least 2 NICs.

Network

Networking needs to be carefully designed to ensure reliability in the cloud infrastructure. The recommendation is having 2 NICs in the front-end (public and service) (or 3 NICs depending on the storage backend, access to the storage network may be needed) 4 NICs present in each virtualization node: private, public, service and storage networks. Less NICs can be needed depending on the storage and networking configuration.

1.2.3 Front-End

The machine that holds the OpenNebula installation is called the front-end. This machine needs network connectivity to all the hosts, and possibly access to the storage Datastores (either by direct mount or network). The base installation of OpenNebula takes less than 150MB.

OpenNebula services include:

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

- Advanced components: OneFlow, OneGate, econe, ...

Note: 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 Availability OpenNebula Section.

If you need to federate several datacenters, with a different OpenNebula instance managing the resources but needing a common authentication schema, check the Federation Section.

1.2.4 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 information is sent according to the following process: each host periodically sends monitoring data to the front-end 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 check the *the Monitoring Section* for more details.

1.2.5 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 life-cycle.

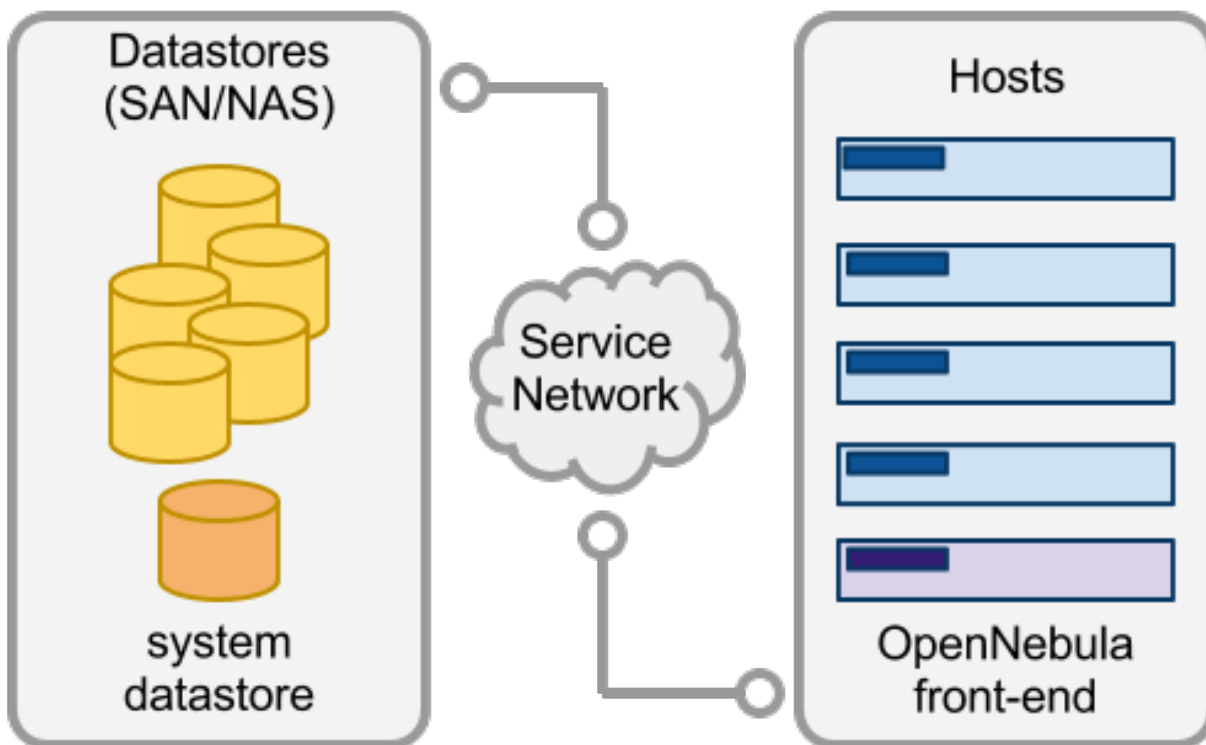
OpenNebula natively supports one open source hypervisor, the *KVM* hypervisor, and OpenNebula is configured by default to interact with hosts running KVM.

Ideally, the configuration of the nodes will be homogeneous in terms of the software components installed, the oneadmin administration user, accessible storage and network connectivity. This may not always be the case, and homogeneous hosts can be grouped in OpenNebula clusters

If you are interested in fail-over protection against hardware and operating system outages within your virtualized IT environment, check the Virtual Machines High Availability Section.

1.2.6 Storage

OpenNebula uses *Datastores* to store VMs' disk images. A datastore is any storage medium, typically 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, its 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:** to store the disk images repository. Disk images are moved, or cloned to/from the System Datastore when the VMs are deployed or shutdown, or when disks are attached or snapshotted.
- *File Datastore:* a special datastore used to store plain files, not disk images. These files can be used as kernels, ramdisks or context files.

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

- *Filesystem:* to store disk images in a file form. There are three types: ssh, shared and qcow.
- *LVM:* to use 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 default system and images datastores are configured to use a filesystem with the ssh transfer drivers.

Please check the *Storage Chapter* for more details.

1.2.7 Networking

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

- **Service Network:** used 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;
- **Instance Network:** offers network connectivity to the VMs across the different hosts. To make an effective use of your VM deployments, you will 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): doesn't perform any network operation, and firewalling rules are also ignored.
- **fw:** firewalling rules are applied, but networking isolation is ignored.
- **802.1Q:** restrict network access through VLAN tagging, which requires support by the hardware switches.
- **ebtables:** restrict network access through Ebtables rules. No special hardware configuration required.
- **ovswitch:** restrict network access with [Open vSwitch Virtual Switch](#).
- **vxlan:** segment a VLAN in isolated networks using the VXLAN encapsulation protocol.

Please check the [Networking Chapter](#) to find out more information about the networking technologies supported by OpenNebula.

1.2.8 Authentication

The following authentication methods are supported to access OpenNebula:

- Built-in User/Password
- [SSH Authentication](#)
- [X509 Authentication](#)
- [LDAP Authentication](#) (and Active Directory)

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

Please check the [Authentication Chapter](#) to find out more information about the authentication technologies supported by OpenNebula.

1.2.9 Advanced Components

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

- **Multi-VM Applications and Auto-scaling:** OneFlow allows users and administrators to define, execute and manage multi-tiered applications, or services composed of interconnected Virtual Machines with deployment dependencies between them. Each group of Virtual Machines is deployed and managed as a single entity, and is completely integrated with the advanced OpenNebula user and group management.
- **Cloud Bursting:** Cloud bursting is a model in which the local resources of a Private Cloud are combined with resources from remote Cloud providers. Such support for cloud bursting enables highly scalable hosting environments.

- **Public Cloud:** Cloud interfaces can be added to your Private Cloud if you want to provide partners or external users with access to your infrastructure, or to sell your overcapacity. The following interface provide a simple and remote management of cloud (virtual) resources at a high abstraction level: Amazon EC2 and EBS APIs.
- **Application Insight:** OneGate allows Virtual Machine guests to push monitoring information to OpenNebula. Users and administrators can use it to gather metrics, detect problems in their applications, and trigger OneFlow auto-scaling rules.

1.3 VMware Cloud Architecture

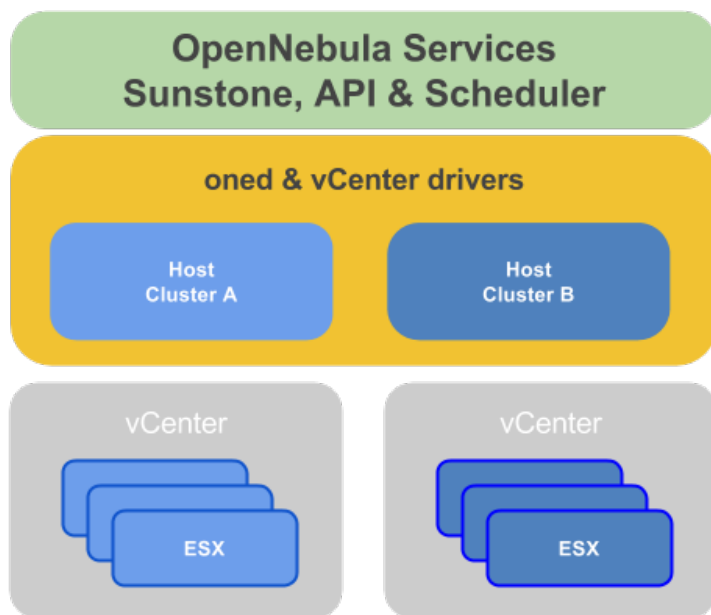
OpenNebula is intended for companies willing to create a self-service cloud environment on top of their VMware infrastructure without having to abandon their investment in VMware and retool the entire stack. In these environments, OpenNebula seamlessly integrates with existing vCenter infrastructures to leverage advanced features -such as vMotion, HA or DRS scheduling- provided by the VMware vSphere product family. OpenNebula exposes a multi-tenant, cloud-like provisioning layer on top of vCenter, including features like virtual data centers, data center federation or hybrid cloud computing to connect in-house vCenter infrastructures with public clouds.

OpenNebula over vCenter is intended for companies that want to keep VMware management tools, procedures and workflows. For these companies, throwing away VMware and retooling the entire stack is not the answer. However, as they consider moving beyond virtualization toward a private cloud, they can choose to either invest more in VMware, or proceed on a tactically challenging but strategically rewarding path of open.

1.3.1 Architectural Overview

OpenNebula assumes that your physical infrastructure adopts a classical cluster-like architecture with a front-end, and a set of vCenter instances grouping ESX hosts where Virtual Machines (VM) will be executed. There is at least one physical network joining all the vCenters and ESX hosts with the front-end. Connection from the front-end to vCenter is for management purposes, whereas the connection from the front-end to the ESX hosts is to support the VNC connections.

The VMware vCenter drivers enable OpenNebula to access one or more vCenter servers that manages one or more ESX Clusters. Each ESX Cluster is presented in OpenNebula as an aggregated hypervisor. Note that OpenNebula scheduling decisions are therefore made at ESX Cluster level, vCenter then uses the DRS component to select the actual ESX host and Datastore to deploy the Virtual Machine.



A cloud architecture is defined by three components: storage, networking and virtualization. Therefore, the basic components of an OpenNebula cloud 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 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 Section briefly describes the different choices that you can make for the management of the different subsystems. If your specific services are not supported we recommend to check the drivers available in the [Add-on Catalog](#). We also provide information and support about how to develop new drivers.

1.3.2 Dimensioning the Cloud

The dimension of a cloud infrastructure can be directly inferred from the expected workload in terms of VMs that the cloud infrastructure must sustain. This workload is also tricky to estimate, but this is a crucial exercise to build an efficient cloud.

OpenNebula front-end

The minimum recommended specs are for the OpenNebula front-end are:

Resource	Minimum Recommended configuration
Memory	2 GB
CPU	1 CPU (2 cores)
Disk Size	100 GB
Network	2 NICs

When running on a front-end with the minimums described in the above table, OpenNebula is able to manage a vCenter infrastructure of the following characteristics:

- Up to 4 vCenters
- Up to 40 ESXs managed by each vCenter
- Up to 1.000 VMs in total, each vCenter managing up to 250 VMs

ESX nodes

Regarding the dimensions of the ESX virtualization nodes:

- **CPU:** without overcommitment, each CPU core assigned to a VM must exist as a physical CPU core. By example, for a workload of 40 VMs with 2 CPUs, the cloud will need 80 physical CPUs. These 80 physical CPUs can be spread among different hosts: 10 servers with 8 cores each, or 5 server of 16 cores each. With overcommitment, however, CPU dimension can be planned ahead, using the `CPU` and `VCPU` attributes: `CPU` states physical CPUs assigned to the VM, while `VCPU` states virtual CPUs to be presented to the guest OS.
- **MEMORY:** Planning for memory is straightforward, as by default *there is no overcommitment of memory* in OpenNebula. It is always a good practice to count 10% of overhead by the hypervisor (this is not an absolute upper limit, it depends on the hypervisor). So, in order to sustain a VM workload of 45 VMs with 2GB of RAM each, 90GB of physical memory is needed. The number of hosts is important, as each one will incur a 10% overhead due to the hypervisors. For instance, 10 hypervisors with 10GB RAM each will contribute with 9GB each (10% of 10GB = 1GB), so they will be able to sustain the estimated workload. The rule of thumb is having at least 1GB per core, but this also depends on the expected workload.

Storage

Dimensioning storage is a critical aspect, as it is usually the cloud bottleneck. OpenNebula can manage any datastore that is mounted in the ESX and visible in vCenter. The datastore used by a VM can be fixed by the cloud admin or delegated to the cloud user. It is important to ensure that enough space is available for new VMs, otherwise its creation process will fail. One valid approach is to limit the storage available to users by defining quotas in the number of maximum VMs, and ensuring enough datastore space to comply with the limit set in the quotas. In any case, OpenNebula allows cloud administrators to add more datastores if needed.

Network

Networking needs to be carefully designed to ensure reliability in the cloud infrastructure. The recommendation is having 2 NICs in the front-end (service and public network) 4 NICs present in each ESX node: private, public, service and storage networks. Less NICs can be needed depending on the storage and networking configuration.

1.3.3 Front-End

The machine that holds the OpenNebula installation is called the front-end. This machine needs network connectivity to all the vCenter and ESX hosts. The base installation of OpenNebula takes less than 150MB.

OpenNebula services include:

- Management daemon (`oned`) and scheduler (`mm_sched`)
- Web interface server (`sunstone-server`)
- Advanced components: OneFlow, OneGate, econe, ...

Note: 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 Availability OpenNebula Section.

1.3.4 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 probes in the front-end provided by OpenNebula.

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

1.3.5 Virtualization Hosts

The VMware vCenter drivers enable OpenNebula to access one or more vCenter servers that manages one or more ESX Clusters. Each ESX Cluster is presented in OpenNebula as an aggregated hypervisor. The Virtualization Subsystem is the component in charge of talking with vCenter and taking the actions needed for each step in the VM life-cycle. All the management operations are issued by the front-end to vCenter, except the VNC connection that is performed directly from the front-end to the ESX where a particular VM is running.

OpenNebula natively supports *vCenter* hypervisor, vCenter drivers need to be configured in the OpenNebula front-end.

If you are interested in fail-over protection against hardware and operating system outages within your virtualized IT environment, check the Virtual Machines High Availability Section.

1.3.6 Storage

OpenNebula interacts as a consumer of vCenter storage, and as such, supports all the storage devices supported by *ESX*. When a VM is instantiated from a VM Template, the datastore associated with the VM template is chosen. If DRS is enabled, then vCenter will pick the optimal Datastore to deploy the VM. Alternatively, the datastore used by a VM can be fixed by the cloud admin or delegated to the cloud user.

vCenter/ESX Datastores can be represented in OpenNebula to create, clone and/or upload VMDKs. The vCenter/ESX datastore representation in OpenNebula is described in the *vCenter datastore Section*.

1.3.7 Networking

Networking in OpenNebula is handled by creating or importing Virtual Network representations of vCenter Networks and Distributed vSwitches. In this way, new VMs with defined network interfaces will be bound by OpenNebula to these Networks and/or Distributed vSwitches. OpenNebula can create a new logical layer of these vCenter Networks and Distributed vSwitches, in particular three types of Address Ranges can be defined per Virtual Network representing the vCenter network resources: plain Ethernet, IPv4 and IPv6. This networking information can be passed to the VMs through the contextualization process.

Please check the *Networking Chapter* to find out more information about the networking support in vCenter infrastructures by OpenNebula.

1.3.8 Authentication

The following authentication methods are supported to access OpenNebula:

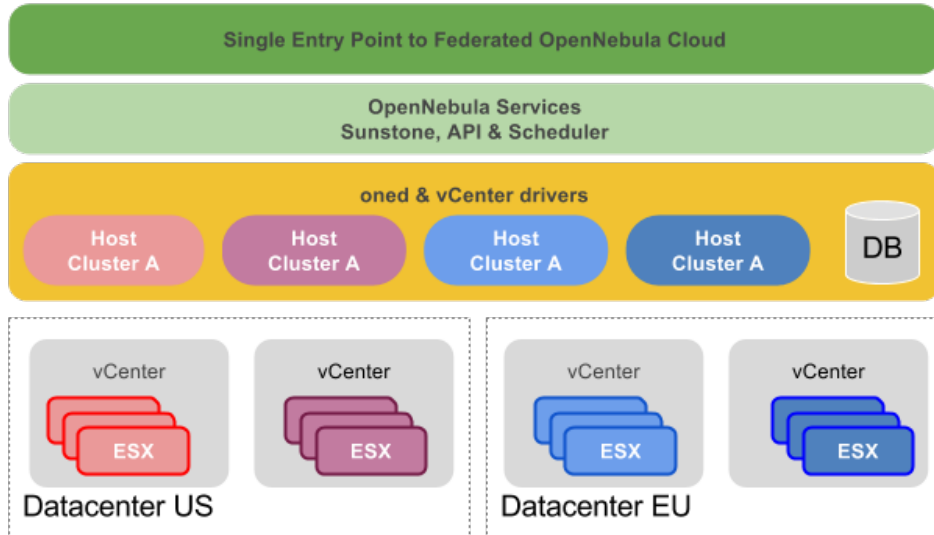
- Built-in User/Password
- *SSH Authentication*
- *X509 Authentication*
- *LDAP Authentication* (and Active Directory)

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

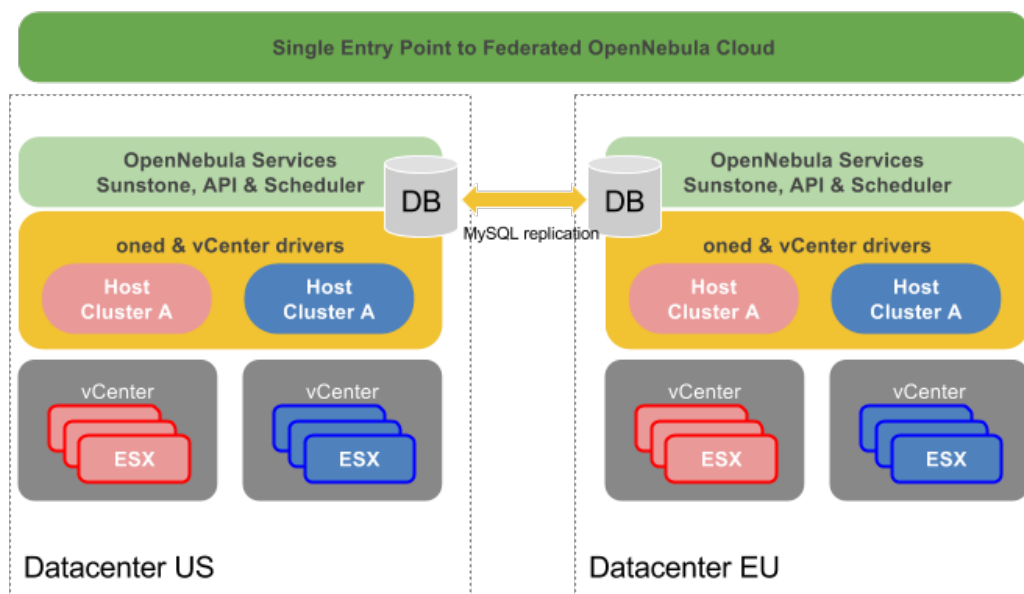
Please check the *Authentication Chapter* to find out more information about the authentication technologies supported by OpenNebula.

1.3.9 Multi-Datacenter Deployments

OpenNebula interacts with the vCenter instances by interfacing with its SOAP API exclusively. This characteristic enables architectures where the OpenNebula instance and the vCenter environment are located in different datacenters. A single OpenNebula instances can orchestrate several vCenter instances remotely located in different data centers. Connectivity between data centers needs to have low latency in order to have a reliable management of vCenter from OpenNebula.



When administration domains need to be isolated or the interconnection between datacenters does not allow a single controlling entity, OpenNebula can be configured in a federation. Each OpenNebula instance of the federation is called a Zone, one of them configured as master and the others as slaves. An OpenNebula federation is a tightly coupled integration, all the instances will share the same user accounts, groups, and permissions configuration. Federation allows end users to consume resources allocated by the federation administrators regardless of their geographic location. The integration is seamless, meaning that a user logged into the Sunstone web interface of a Zone will not have to log out and enter the address of another Zone. Sunstone allows to change the active Zone at any time, and it will automatically redirect the requests to the right OpenNebula at the target Zone. For more information, check the Federation Section.



1.3.10 Advanced Components

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

- **Multi-VM Applications and Auto-scaling:** OneFlow allows users and administrators to define, execute and manage services composed of interconnected Virtual Machines with deployment dependencies between them. Each group of Virtual Machines is deployed and managed as a single entity, and is completely integrated with the advanced OpenNebula user and group management.
- **Cloud Bursting:** Cloud bursting is a model in which the local resources of a Private Cloud are combined with resources from remote Cloud providers. Such support for cloud bursting enables highly scalable hosting environments.
- **Public Cloud:** Cloud interfaces can be added to your Private Cloud if you want to provide partners or external users with access to your infrastructure, or to sell your overcapacity. The following interface provide a simple and remote management of cloud (virtual) resources at a high abstraction level: Amazon EC2 and EBS APIs.
- **Application Insight:** OneGate allows Virtual Machine guests to push monitoring information to OpenNebula. Users and administrators can use it to gather metrics, detect problems in their applications, and trigger OneFlow auto-scaling rules.

1.4 OpenNebula Provisioning Model

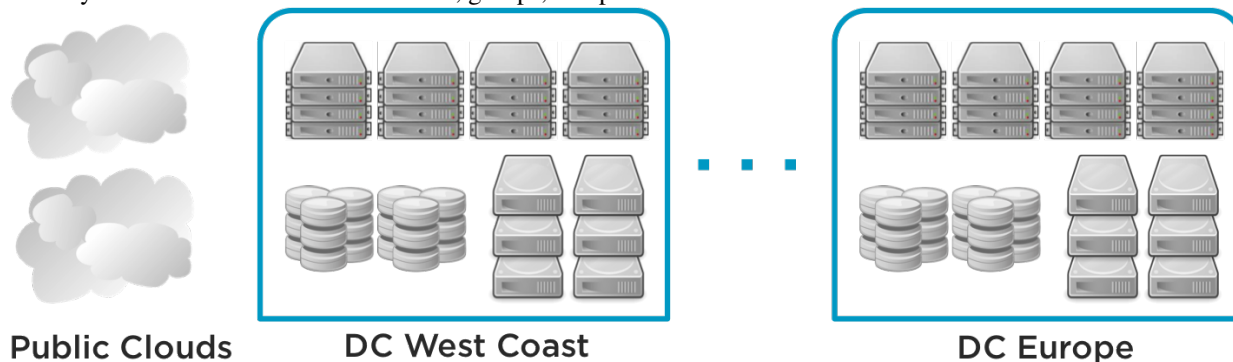
In a small installation with a few hosts, you can use OpenNebula without giving much thought to infrastructure partitioning and provisioning. But for medium and large deployments you will probably want to provide some level of isolation and structure.

This Section is meant for cloud architects, builders and administrators, to help them understand the OpenNebula model for managing and provisioning virtual resources. This model is a result of our collaboration with our user community throughout the life of the project.

1.4.1 The Infrastructure Perspective

Common large IT shops have multiple Data Centers (DCs), each one running its own OpenNebula instance and consisting of several physical Clusters of infrastructure resources (Hosts, Networks and Datastores). These Clusters could present different architectures and software/hardware execution environments to fulfill the needs of different workload profiles. Moreover, many organizations have access to external public clouds to build hybrid cloud scenarios where the private capacity of the Data Centers is supplemented with resources from external clouds, like Amazon AWS, to address peaks of demand.

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



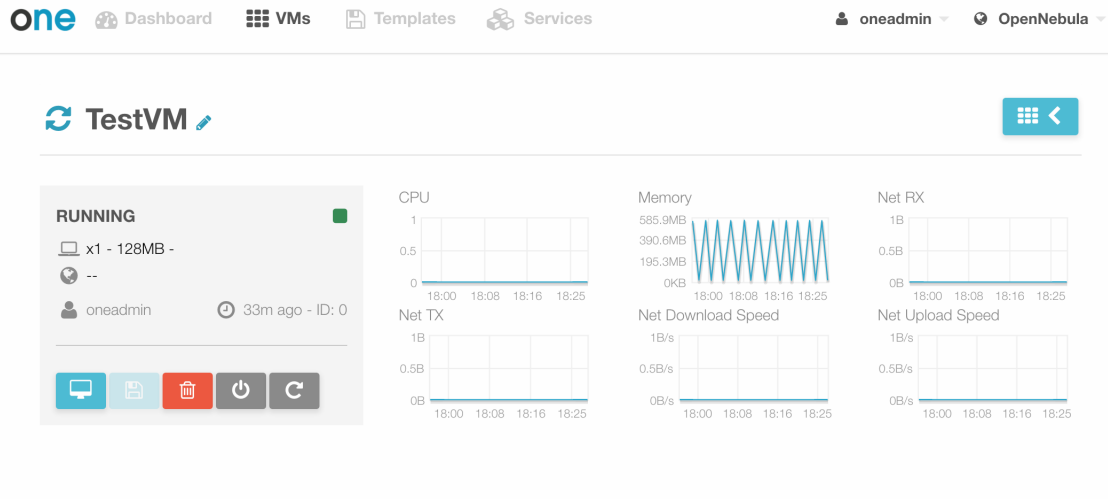
1.4.2 The Organizational Perspective

Users are organized into Groups (similar to what other environments call Projects, Domains, Tenants...). A Group is an authorization boundary that can be seen as a business unit if you are considering it as a private cloud or as a complete new company if it is a public cloud. While Clusters are used to group Physical Resources according to common characteristics such as networking topology or physical location, Virtual Data Centers (VDCs) allow to create “logical” pools of Physical Resources (which could belong to different Clusters and Zones) and allocate them to user Groups.

A VDC is a fully-isolated virtual infrastructure environment where a Group of users (or optionally several Groups of users), under the control of a Group admin, can create and manage compute and storage capacity. The users in the Group, including the Group admin, would only see the virtual resources and not the underlying physical infrastructure. The Physical Resources allocated to the Group are managed by the cloud administrator through a VDC. These resources grouped in the VDC can be dedicated exclusively to the Group, providing isolation at the physical level too.

The privileges of the Group users and the admin regarding the operations over the virtual resources created by other users can be configured. For example, in the Advanced Cloud Provisioning Case described below, the users can instantiate virtual machine templates to create their machines, while the admins of the Group have full control over other users’ resources and can also create new users in the Group.

Users can access their resources through any of the existing OpenNebula interfaces, such as the CLI, Sunstone Cloud View, OCA, or the AWS APIs. Group admins can manage their Groups through the CLI or the Group Admin View in Sunstone. Cloud administrators can manage the Groups through the CLI or Sunstone.



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

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

1.4.3 Examples of Provisioning Use Cases

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

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

For example, you can think Web Development, Human Resources, and Big Data Analysis as business units represented by Groups in a private OpenNebula cloud, and allocate them resources from your DCs and public clouds in order to create three different VDCs.

- **VDC BLUE:** VDC that allocates (ClusterA-DC_West_Coast + Cloudbursting) to Web Development
- **VDC RED:** VDC that allocates (ClusterB-DC_West_Coast + ClusterA-DC_Europe + Cloudbursting) to Human Resources
- **VDC GREEN:** VDC that allocates (ClusterC-DC_West_Coast + ClusterB-DC_Europe) to Big Data Analysis

Web Development



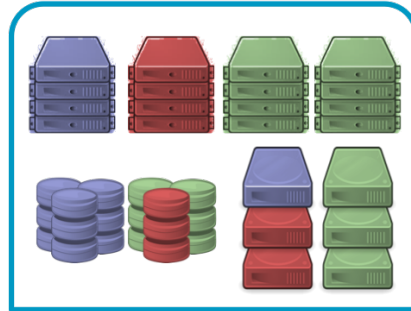
Human Resources



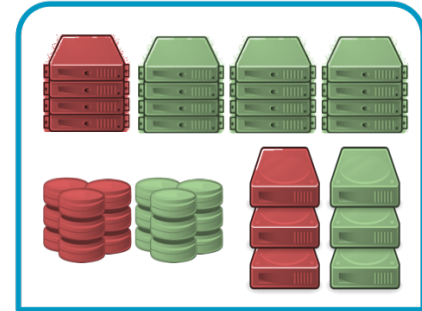
Big Data Analysis



Public Clouds



DC West Coast



DC Europe

1.4.4 Cloud Provisioning Scenarios

OpenNebula has three predefined User roles to implement three typical enterprise cloud scenarios:

- Data center infrastructure management
- Simple cloud provisioning model
- Advanced cloud provisioning model

In these three scenarios, Cloud Administrators manage the physical infrastructure, creates Users and VDCs, prepares base templates and images for Users, etc

Cloud Administrators typically access the cloud using the CLI or the Admin View of Sunstone.

Role	Capabilities
Cloud Admin.	<ul style="list-style-type: none"> • Operates the Cloud infrastructure (i.e. computing nodes, networking fabric, storage servers) • Creates and manages OpenNebula infrastructure resources: Hosts, Virtual Networks, Datastores • Creates and manages Multi-VM Applications (Services) • Creates new Groups and VDCs • Assigns Groups and physical resources to a VDC and sets quota limits • Defines base instance types to be used by the users. These types define the capacity of the VMs (memory, cpu and additional storage) and connectivity. • Prepare VM images to be used by the users • Monitor the status and health of the cloud • Generate activity reports

Data Center Infrastructure Management

This model is used to manage data center virtualization and to integrate and federate existing IT assets that can be in different data centers. In this usage model, Users are familiar with virtualization concepts. Except for the infrastructure resources, the web interface offers the same operations available to the Cloud Admin. These are “Advanced Users” that could be considered also as “Limited Cloud Administrators”.

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

Groups are used by the Cloud Administrator to isolate users, which are combined with VDCs to have allocated resources, but are not offered on-demand.

These “Advanced Users” typically access the cloud by using the CLI or the User View of Sunstone. This is not the default model configured for the group Users.

Role	Capabilities
Advanced User	<ul style="list-style-type: none"> • Instantiates VMs using their own templates • Creates new templates and images • Manages their VMs, including advanced life-cycle features • Creates and manages Multi-VM Application (Services) • Check their usage and quotas • Upload SSH keys to access the VMs

Simple Cloud Provisioning

In the simple infrastructure provisioning model, the Cloud offers infrastructure as a service to individual Users. Users are considered as “Cloud Users” or “Cloud Consumers”, being much more limited in their operations. These Users access a very intuitive simplified web interface that allows them to launch Virtual Machines from predefined Templates.

They can access their VMs, and perform basic operations like shutdown. The changes made to a VM disk can be saved back, but new Images cannot be created from scratch.

Groups are used by the Cloud Administrator to isolate users, which are combined with VDCs to have allocated resources, but are not offered on-demand.

These “Cloud Users” typically access the cloud by using the Cloud View of Sunstone. This is the default model configured for the group Users.

Role	Capabilities
Cloud User	<ul style="list-style-type: none"> • Instantiates VMs using the templates defined by the Cloud Admins and the images defined by the Cloud Admins or Group Admins. • Instantiates VMs using their own Images saved from a previous running VM • Manages their VMs, including <ul style="list-style-type: none"> – reboot – power off/on (short-term switching-off) – delete – save a VM into a new Template – obtain basic monitor information and status (including IP addresses) • Delete any previous VM template and disk snapshot • Check user account usage and quotas • Upload SSH keys to access the VMs

Advanced Cloud Provisioning

The advanced provisioning model is an extension of the previous one where the cloud provider offers VDCs on demand to Groups of Users (projects, companies, departments or business units). Each Group can define one or more users as Group Admins. These admins can create new users inside the Group, and also manage the resources of the rest of the users. A Group Admin may, for example, shutdown a VM from other user to free group quota usage.

These Group Admins typically access the cloud by using the Group Admin View of Sunstone.

The Group Users have the capabilities described in the previous scenario and typically access the cloud by using the Cloud View of Sunstone.

Role	Capabilities
Group Admin.	<ul style="list-style-type: none"> • Creates new users in the Group • Operates on the Group’s virtual machines and disk images • Share Saved Templates with the members of the Group • Checks Group usage and quotas

OPENNEBULA INSTALLATION

2.1 Overview

The Front-end is the central part of an OpenNebula installation. This is the machine where the server software is installed and where you connect to manage your cloud. It can be a physical node or a Virtual Machine.

2.1.1 How Should I Read This Chapter

Before reading this chapter make sure you have read and understood the *Cloud Design* chapter.

The aim of this chapter is to give you a quick-start guide to deploy OpenNebula. This is the simplest possible installation, but it is also the foundation for a more complex setup, with Advanced Components (like Host and VM High Availability, Cloud Bursting, etc...).

First you should read the *Front-end Installation* section. Note that by default it uses a SQLite database that is not recommended for production so, if this is not a small proof of concept, while following the Installation section, you should enable *MySQL*.

After reading this chapter, read the *Node Installation* chapter next in order to add hypervisors to your cloud.

2.1.2 Hypervisor Compatibility

Section	Compatibility
<i>Front-end Installation</i>	This Section applies to both KVM and vCenter.
<i>MySQL Setup</i>	This Section applies to both KVM and vCenter.
Scheduler	This Section applies to both KVM and vCenter.

2.2 Front-end Installation

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

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: you can add **our package repositories** to your system, or visit the [software menu](#) to **download the latest package** for your Linux distribution.

If there are no packages for your distribution, head to the [Building from Source Code](#) guide.

2.2.1 Step 1. Disable SELinux in CentOS/RHEL 7

SELinux can cause some problems, like not trusting `oneadmin` user's SSH credentials. You can disable it changing in the file `/etc/selinux/config` this line:

```
SELINUX=disabled
```

After this file is changed reboot the machine.

2.2.2 Step 2. Add OpenNebula Repositories

CentOS/RHEL 7

To add OpenNebula repository execute the following as root:

```
# cat << EOT > /etc/yum.repos.d/opennebula.repo
[opennebula]
name=opennebula
baseurl=http://downloads.opennebula.org/repo/5.0/CentOS/7/x86_64
enabled=1
gpgcheck=0
EOT
```

Debian/Ubuntu

To add OpenNebula repository on Debian/Ubuntu execute as root:

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

Debian 8

```
# echo "deb http://downloads.opennebula.org/repo/5.0/Debian/8 stable opennebula" > /
↳etc/apt/sources.list.d/opennebula.list
```

Ubuntu 14.04

```
# echo "deb http://downloads.opennebula.org/repo/5.0/Ubuntu/14.04 stable opennebula"
↳> /etc/apt/sources.list.d/opennebula.list
```

Ubuntu 16.04

```
# echo "deb http://downloads.opennebula.org/repo/5.0/Ubuntu/16.04 stable opennebula"
↳> /etc/apt/sources.list.d/opennebula.list
```

2.2.3 Step 3. Installing the Software

Installing on CentOS/RHEL 7

Before installing:

- Activate the [EPEL](#) repo. In CentOS this can be done with the following command:


```
# yum install epel-release
```

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.

```
# yum install opennebula-server opennebula-sunstone opennebula-ruby opennebula-gate_
↪opennebula-flow
```

CentOS/RHEL Package Description

These are the packages available for this distribution:

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

Note: The files located in `/etc/one` and `/var/lib/one/remotes` are marked as configuration files.

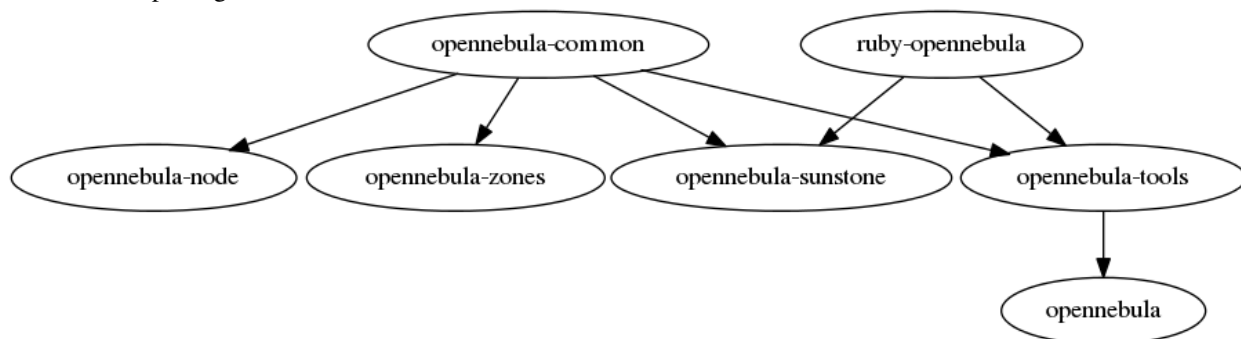
Installing on Debian/Ubuntu

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

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

Debian/Ubuntu Package Description

These are the packages available for these distributions:



- **opennebula-common**: Provides the user and common files.
- **ruby-opennebula**: Ruby API.

- **libopennebula-java**: Java API.
- **libopennebula-java-doc**: Java API Documentation.
- **opennebula-node**: Prepares a node as an opennebula-node.
- **opennebula-sunstone**: *Sunstone* (the GUI).
- **opennebula-tools**: Command Line interface.
- **opennebula-gate**: OneGate server that enables communication between VMs and OpenNebula.
- **opennebula-flow**: OneFlow manages services and elasticity.
- **opennebula**: OpenNebula Daemon.

Note: Besides `/etc/one`, the following files are marked as configuration files:

- `/var/lib/one/remotes/datastore/ceph/ceph.conf`
 - `/var/lib/one/remotes/vnm/OpenNebulaNetwork.conf`
-

2.2.4 Step 4. 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.

2.2.5 Step 5. Enabling MySQL/MariaDB (Optional)

You can skip this step if you just want to deploy OpenNebula as quickly as possible. However if you are deploying this for production or in a more serious environment, make sure you read the *MySQL Setup* section.

Note that it is possible to switch from SQLite to MySQL, but since it's more cumbersome to migrate databases, we suggest that if in doubt, use MySQL from the start.

2.2.6 Step 6. Starting OpenNebula

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

The `/var/lib/one/.one/one_auth` file will have been created with a randomly-generated password. It should contain the following: `oneadmin:<password>`. Feel free to change the password before starting OpenNebula. For example:

```
$ echo "oneadmin:mypassword" > ~/.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:

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

2.2.7 Step 7. Verifying the Installation

After OpenNebula is started for the first time, you should check that the commands can connect to the OpenNebula daemon. You can do this in the Linux CLI or in the graphical user interface: Sunstone.

Linux CLI

In the Front-end, run the following command as `oneadmin`:

```
$ oneuser show
USER 0 INFORMATION
ID           : 0
NAME        : oneadmin
GROUP       : oneadmin
PASSWORD    : 3bc15c8aae3e4124dd409035f32ea2fd6835efc9
AUTH_DRIVER : core
ENABLED     : Yes

USER TEMPLATE
TOKEN_PASSWORD="ec21d27e2fe4f9ed08a396cbd47b08b8e0a4ca3c"

RESOURCE USAGE & QUOTAS
```

If you get an error message, then the OpenNebula daemon could not be started properly:

```
$ oneuser show
Failed to open TCP connection to localhost:2633 (Connection refused - connect(2) for
↪ "localhost" port 2633)
```

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].

Sunstone

Now you can try to log in into Sunstone web interface. To do this point your browser to `http://<fontend_address>:9869`. If everything is OK you will be greeted with a login page. The user is `oneadmin` and the password is the one in the file `/var/lib/one/.one/one_auth` in your Front-end.

If the page does not load, make sure you check `/var/log/one/sunstone.log` and `/var/log/one/sunstone.error`. Also, make sure TCP port 9869 is allowed through the firewall.

Directory Structure

The following table lists some notable paths that are available in your Front-end after the installation:

Path	Description
<code>/etc/one/</code>	Configuration Files
<code>/var/log/one/</code>	Log files, notably: <code>oned.log</code> , <code>sched.log</code> , <code>sunstone.log</code> and <code><vmid>.log</code>
<code>/var/lib/one/</code>	<code>oneadmin</code> home directory
<code>/var/lib/one/datastores/<dsid>/</code>	Storage for the datastores
<code>/var/lib/one/vms/<vmid>/</code>	Action files for VMs (deployment file, transfer manager scripts, etc...)
<code>/var/lib/one/.one/one_auth</code>	<code>oneadmin</code> credentials
<code>/var/lib/one/remotes/</code>	Probes and scripts that will be synced to the Hosts
<code>/var/lib/one/remotes/hooks/</code>	Hook scripts
<code>/var/lib/one/remotes/vmm/</code>	Virtual Machine Manager Driver scripts
<code>/var/lib/one/remotes/auth/</code>	Authentication Driver scripts
<code>/var/lib/one/remotes/im/</code>	Information Manager (monitoring) Driver scripts
<code>/var/lib/one/remotes/market/</code>	MarketPlace Driver scripts
<code>/var/lib/one/remotes/datastore/</code>	Datastore Driver scripts
<code>/var/lib/one/remotes/vnm/</code>	Networking Driver scripts
<code>/var/lib/one/remotes/tm/</code>	Transfer Manager Driver scripts

2.2.8 Step 8. Next steps

Now that you have successfully started your OpenNebula service, head over to the *Node Installation* chapter in order to add hypervisors to your cloud.

2.3 MySQL Setup

The MySQL/MariaDB back-end is an alternative to the default SQLite back-end. In this guide and in the rest of OpenNebula's documentation and configuration files we will refer to this database as the MySQL, however OpenNebula you can use either MySQL or MariaDB.

The two back-ends cannot coexist (SQLite and MySQL), and you will have to decide which one is going to be used while planning your OpenNebula installation.

Note: If you are planning to install OpenNebula with MySQL back-end, please follow this guide *prior* to start OpenNebula the first time to avoid problems with `oneadmin` and `serveradmin` credentials.

2.3.1 Installation

First of all, you need a working MySQL server. You can either deploy one for the OpenNebula installation or reuse any existing MySQL already deployed and accessible by the Front-end.

Configuring MySQL

You need to add a new user and grant it privileges on the `opennebula` database. This new database doesn't need to exist, OpenNebula will create it the first time you run it.

Assuming you are going to use the default values, log in to your MySQL server and issue the following commands:

```
$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor. [...]

mysql> GRANT ALL PRIVILEGES ON opennebula.* TO 'oneadmin' IDENTIFIED BY '<thepassword>'
↵;
Query OK, 0 rows affected (0.00 sec)
```

Visit the [MySQL documentation](#) to learn how to manage accounts.

Now configure the transaction isolation level:

```
mysql> SET GLOBAL TRANSACTION ISOLATION LEVEL READ COMMITTED;
```

Configuring OpenNebula

Before you run OpenNebula for the first time, you need to set in `oned.conf` the connection details, and the database you have granted privileges on.

```
# Sample configuration for MySQL
DB = [ backend = "mysql",
        server  = "localhost",
        port    = 0,
        user    = "oneadmin",
        passwd  = "<thepassword>",
        db_name = "opennebula" ]
```

Fields:

- **server**: URL of the machine running the MySQL server.
- **port**: port for the connection to the server. If set to 0, the default port is used.
- **user**: MySQL user-name.
- **passwd**: MySQL password.
- **db_name**: Name of the MySQL database OpenNebula will use.

2.3.2 Using OpenNebula with MySQL

After this installation and configuration process you can use OpenNebula as usual.

NODE INSTALLATION

3.1 Overview

After OpenNebula Front-end is correctly setup the next step is preparing the hosts where the VMs are going to run.

3.1.1 How Should I Read This Chapter

Make sure you have properly *installed the Front-end* before reading this chapter.

This chapter focuses on the minimal node installation you need to follow in order to finish deploying OpenNebula. Concepts like storage and network have been simplified. Therefore feel free to follow the other specific chapters in order to configure other subsystems like networking and storage.

Note that you can follow this chapter without reading any other guides and you will be ready, by the end of it, to deploy a Virtual Machine. If in the future you want to switch to other storage or networking technologies you will be able to do so.

After installing the nodes and *verifying your installation*, you can either start using your cloud or configure more components:

- *Authenticaton*. (Optional) For integrating OpenNebula with LDAP/AD, or securing it further with other authentication technologies.
- *Sunstone*. OpenNebula GUI should working and accessible at this stage, but by reading this guide you will learn about specific enhanced configurations for Sunstone.

If your cloud is KVM based you should also follow:

- *Open Cloud Host Setup*.
- *Open Cloud Storage Setup*.
- *Open Cloud Networking Setup*.

Otherwise, if it's VMware based:

- Head over to the *VMware Infrastructure Setup* chapter.

3.1.2 Hypervisor Compatibility

Section	Compatibility
<i>KVM Node Installation</i>	This Section applies to KVM.
<i>vCenter Node Installation</i>	This Section applies to vCenter.
<i>Verify your Installation</i>	This Section applies to both vCenter and KVM.

3.2 KVM Node Installation

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

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: you can add **our package repositories** to your system, or visit the [software menu](#) to **download the latest package** for your Linux distribution.

3.2.1 Step 1. Add OpenNebula Repositories

CentOS/RHEL 7

To add OpenNebula repository execute the following as root:

```
# cat << EOT > /etc/yum.repos.d/opennebula.repo
[opennebula]
name=opennebula
baseurl=http://downloads.opennebula.org/repo/5.0/CentOS/7/x86_64
enabled=1
gpgcheck=0
EOT
```

Debian/Ubuntu

To add OpenNebula repository on Debian/Ubuntu execute as root:

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

Debian 8

```
# echo "deb http://downloads.opennebula.org/repo/5.0/Debian/8 stable opennebula" > /
↳etc/apt/sources.list.d/opennebula.list
```

Ubuntu 14.04

```
# echo "deb http://downloads.opennebula.org/repo/5.0/Ubuntu/14.04 stable opennebula"
↳> /etc/apt/sources.list.d/opennebula.list
```

Ubuntu 16.04

```
# echo "deb http://downloads.opennebula.org/repo/5.0/Ubuntu/16.04 stable opennebula"
↳> /etc/apt/sources.list.d/opennebula.list
```

3.2.2 Step 2. Installing the Software

Installing on CentOS/RHEL

Execute the following commands to install the node package and restart libvirt to use the OpenNebula provided configuration file:

```
$ sudo yum install opennebula-node-kvm
$ sudo service libvirtd restart
```

For further configuration, check the specific guide: *KVM*.

Installing on Debian/Ubuntu

Execute the following commands to install the node package and restart libvirt to use the OpenNebula provided configuration file:

```
$ sudo apt-get install opennebula-node
$ sudo service libvirtd restart # debian
$ sudo service libvirt-bin restart # ubuntu
```

For further configuration check the specific guide: *KVM*.

3.2.3 Step 3. Disable SELinux in CentOS/RHEL 7

SELinux can cause some problems, like not trusting `oneadmin` user's SSH credentials. You can disable it changing in the file `/etc/selinux/config` this line:

```
SELINUX=disabled
```

After this file is changed reboot the machine.

3.2.4 Step 4. Configure Passwordless SSH

OpenNebula Front-end connects to the hypervisor Hosts using SSH. You must distribute the public key of `oneadmin` user from all machines to the file `/var/lib/one/.ssh/authorized_keys` in all the machines. There are many methods to achieve the distribution of the SSH keys, ultimately the administrator should choose a method (the recommendation is to use a configuration management system). In this guide we are going to manually scp the SSH keys.

When the package was installed in the Front-end, an SSH key was generated and the `authorized_keys` populated. We will sync the `id_rsa`, `id_rsa.pub` and `authorized_keys` from the Front-end to the nodes. Additionally we need to create a `known_hosts` file and sync it as well to the nodes. To create the `known_hosts` file, we have to execute this command as user `oneadmin` in the Front-end with all the node names as parameters:

```
$ ssh-keyscan <node1> <node2> <node3> ... >> /var/lib/one/.ssh/known_hosts
```

Now we need to copy the directory `/var/lib/one/.ssh` to all the nodes. The easiest way is to set a temporary password to `oneadmin` in all the hosts and copy the directory from the Front-end:

```
$ scp -rp /var/lib/one/.ssh <node1>:/var/lib/one/
$ scp -rp /var/lib/one/.ssh <node2>:/var/lib/one/
$ scp -rp /var/lib/one/.ssh <node3>:/var/lib/one/
$ ...
```

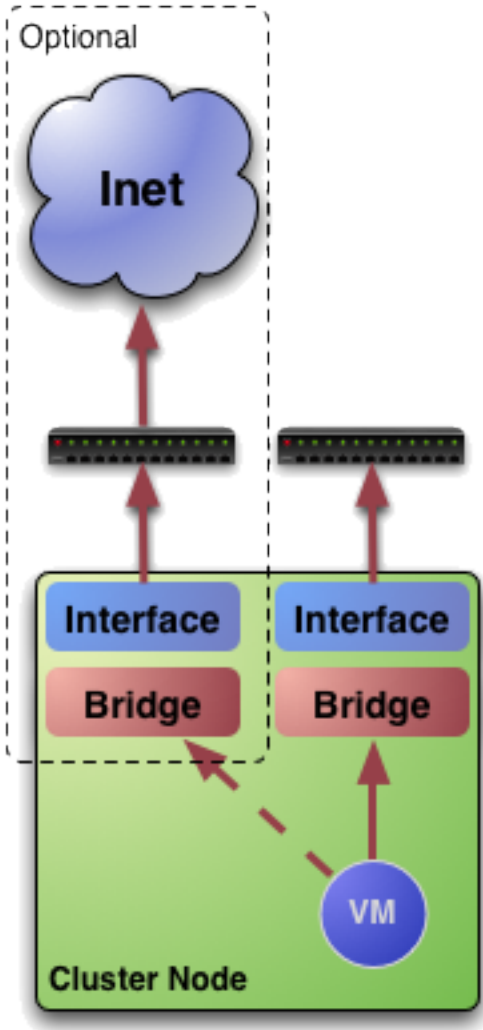
You should verify that connecting from the Front-end, as user `oneadmin`, to the nodes, and from the nodes to the Front-end, does not ask password:


```
$ ssh <node1>
$ ssh <frontend>
$ exit
$ exit

$ ssh <node2>
$ ssh <frontend>
$ exit
$ exit

$ ssh <node3>
$ ssh <frontend>
$ exit
$ exit
```

3.2.5 Step 5. Networking Configuration



A network connection is needed by the OpenNebula Front-end daemons to access the hosts to manage and monitor

the Hosts, and to transfer the Image files. It is highly recommended to use a dedicated network for this purpose.

There are various network models (please check the [Networking](#) chapter to find out the networking technologies supported by OpenNebula).

You may want to use the simplest network model that corresponds to the *bridged* drivers. For this driver, you will need to setup a linux bridge and include a physical device to the bridge. Later on, when defining the network in OpenNebula, you will specify the name of this bridge and OpenNebula will know that it should connect the VM to this bridge, thus giving it connectivity with the physical network device connected to the bridge. For example, a typical host with two physical networks, one for public IP addresses (attached to an `eth0` NIC for example) and the other for private virtual LANs (NIC `eth1` for example) should have two bridges:

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

Note: Remember that this is only required in the Hosts, not in the Front-end. Also remember that it is not important the exact name of the resources (`br0`, `br1`, etc...), however it's important that the bridges and NICs have the same name in all the Hosts.

3.2.6 Step 6. Storage Configuration

You can skip this step entirely if you just want to try out OpenNebula, as it will come configured by default in such a way that it uses the local storage of the Front-end to store Images, and the local storage of the hypervisors as storage for the running VMs.

However, if you want to set-up another storage configuration at this stage, like Ceph, NFS, LVM, etc, you should read the [Open Cloud Storage](#) chapter.

3.2.7 Step 8. Adding a Host to OpenNebula

In this step we will register the node we have installed in the OpenNebula Front-end, so OpenNebula can launch VMs in it. This step can be done in the CLI or in Sunstone, the graphical user interface. Follow just one method, not both, as they accomplish the same.

To learn more about the host subsystem, read this [guide](#).

Adding a Host through Sunstone

Open the Sunstone as documented [here](#). In the left side menu go to Infrastructure -> Hosts. Click on the + button.

The screenshot shows the OpenNebula Hosts management interface. The left sidebar contains navigation options: Dashboard, Instances, Templates, Storage, Network, Infrastructure, Clusters, Hosts, Zones, System, and Settings. The main content area is titled 'Hosts' and features a toolbar with a green plus sign (circled in red), a refresh icon, and buttons for 'Select cluster', 'Enable', 'Disable', and 'Offline'. A search bar is also present. Below the toolbar is a table with columns: ID, Name, Cluster, RVMs, Allocated CPU, Allocated MEM, and Status. The table is currently empty, with a message 'There is no data available' and a cloud icon containing an 'i'. At the bottom, a summary shows '1 TOTAL', '1 ON', '0 OFF', and '0 ERROR'. A 'Support Not connected' box with a 'Sign in' button is visible in the bottom left.

The fill-in the fqdn of the node in the Hostname field.

The screenshot shows the 'Create Host' form in the OpenNebula interface. The left sidebar is the same as in the previous screenshot. The main content area is titled 'Create Host' and features a toolbar with a back icon, 'Reset', and 'Create' buttons. The form contains two dropdown menus: 'Type' (set to 'KVM') and 'Cluster' (set to '0: default'). Below these is a text input field for 'Hostname' containing the value 'node01', which is circled in red. A 'Support Not connected' box with a 'Sign in' button is visible in the bottom left.

Finally, return to the Hosts list, and check that the Host switch to ON status. It should take somewhere between 20s to 1m. Try clicking on the refresh button to check the status more frequently.

If the host turns to `err` state instead of `on`, check the `/var/log/one/oned.log`. Chances are it's a problem with the SSH!

Adding a Host through the CLI

To add a node to the cloud, run this command as `oneadmin` in the Front-end:

```
$ onehost create <node01> -i kvm -v kvm

$ onehost list
  ID NAME           CLUSTER  RVM   ALLOCATED_CPU  ALLOCATED_MEM  STAT
  -- --           -
  1 localhost      default   0      -                -             init

# After some time (20s - 1m)

$ onehost list
  ID NAME           CLUSTER  RVM   ALLOCATED_CPU  ALLOCATED_MEM  STAT
  -- --           -
  0 node01         default   0      0 / 400 (0%)   0K / 7.7G (0%) on
```

If the host turns to `err` state instead of `on`, check the `/var/log/one/oned.log`. Chances are it's a problem with the SSH!

3.2.8 Step 8. Import Currently Running VMs (Optional)

You can skip this step as importing VMs can be done at any moment, however, if you wish to see your previously deployed VMs in OpenNebula you can use the import VM functionality.

3.2.9 Step 9. Next steps

You can now jump to the optional *Verify your Installation* section in order to get to launch a test VM.

Otherwise, you are ready to start using your cloud or you could configure more components:

- *Authenticator*. (Optional) For integrating OpenNebula with LDAP/AD, or securing it further with other authentication technologies.
- *Sunstone*. OpenNebula GUI should be working and accessible at this stage, but by reading this guide you will learn about specific enhanced configurations for Sunstone.

If your cloud is KVM based you should also follow:

- *Open Cloud Host Setup*.
- *Open Cloud Storage Setup*.
- *Open Cloud Networking Setup*.

If it's VMware based:

- Head over to the *VMware Infrastructure Setup* chapter.

3.3 vCenter Node Installation

This Section lays out the requirements configuration needed in the vCenter and ESX instances in order to be managed by OpenNebula.

The VMware vCenter drivers enable OpenNebula to access one or more vCenter servers that manages one or more ESX Clusters. Each ESX Cluster is presented in OpenNebula as an aggregated hypervisor, i.e. as an OpenNebula Host. This means that the representation is one OpenNebula Host per ESX Cluster.

Note that OpenNebula scheduling decisions are therefore made at ESX Cluster level, vCenter then uses the DRS component to select the actual ESX host and Datastore to deploy the Virtual Machine, although the datastore can be explicitly selected from OpenNebula.

3.3.1 Requirements

The following must be met for a functional vCenter environment:

- vCenter 5.5 and/or 6.0, with at least one cluster aggregating at least one ESX 5.5 and/or 6.0 host.
- VMware tools are needed in the guestOS to enable several features (contextualization and networking feedback). Please install [VMware Tools \(for Windows\)](#) or [Open Virtual Machine Tools](#) (for *nix) in the guestOS.
- Define a vCenter user for OpenNebula. This vCenter user (let's call her `oneadmin`) needs to have access to the ESX clusters that OpenNebula will manage. In order to avoid problems, the hassle free approach is to **declare this oneadmin user as Administrator**.
- Alternatively, in some enterprise environments declaring the user as Administrator is not allowed, in that case, you will need to grant these permissions (please note that the following permissions related to operations are related to the use that OpenNebula does with this operations):

vCenter Operation	Privileges	Notes
CloneVM_Task	VirtualMachine.Provisioning.DeployTemplate	Creates a clone of a particular VM
Recon-figVM_Task	VirtualMachine.Interact.DeviceConnection VirtualMachine.Interact.SetCDMedia VirtualMachine.Interact.SetFloppyMedia VirtualMachine.Config.Rename VirtualMachine.Config.Annotation VirtualMachine.Config.AddExistingDisk VirtualMachine.Config.AddNewDisk VirtualMachine.Config.RemoveDisk VirtualMachine.Config.CPUCount VirtualMachine.Config.Memory VirtualMachine.Config.RawDevice VirtualMachine.Config.AddRemoveDevice VirtualMachine.Config.Settings VirtualMachine.Config.AdvancedConfig VirtualMachine.Config.SwapPlacement VirtualMachine.Config.HostUSBDevice VirtualMachine.Config.DiskExtend VirtualMachine.Config.ChangeTracking VirtualMachine.Provisioning.ReadCustSpecs VirtualMachine.Inventory.CreateFromExisting VirtualMachine.Inventory.CreateNew VirtualMachine.Inventory.Move VirtualMachine.Inventory.Register VirtualMachine.Inventory.Remove VirtualMachine.Inventory.Unregister DVSwitch.CanUse DVPortgroup.CanUse Datastore.AllocateSpace Datastore.BrowseDatastore Datastore.LowLevelFileOperations Datastore.RemoveFile Network.Assign Resource.AssignVirtualMachineToResourcePool	Reconfigures a particular virtual machine.
PowerOnVM_Task	VirtualMachine.Interact.PowerOn	Powers on a virtual machine
PowerOffVM_Task	VirtualMachine.Interact.PowerOff	Powers off a virtual machine
Destroy_Task	VirtualMachine.Inventory.Delete	Deletes a VM (including disks)
SuspendVM_Task	VirtualMachine.Interact.Suspend	Suspends a VM
RebootGuest	VirtualMachine.Interact.Reset	Reboots VM's guest Operating System
ResetVM_Task	VirtualMachine.Interact.Reset	Resets power on a virtual machine
ShutdownGuest	VirtualMachine.Interact.PowerOff	Shutdown guest Operating System
CreateSnapshot_Task	VirtualMachine.State.CreateSnapshot	Creates a new snapshot of a virtual machine.
RemoveSnapshot_Task	VirtualMachine.State.RemoveSnapshot	Removes a snapshot from a virtual machine
RevertToSnapshot_Task	VirtualMachine.State.RevertToSnapshot	Revert a virtual machine to a particular snapshot
CreateVirtualDisk_Task	Datastore.FileManagement	On all VMFS datastores represented by OpenNebula
CopyVirtualDisk_Task	Datastore.FileManagement	On all VMFS datastores represented by OpenNebula
DeleteVirtualDisk_Task	Datastore.FileManagement	On all VMFS datastores represented by OpenNebula

Note: For security reasons, you may define different users to access different ESX Clusters. A different user can be defined in OpenNebula per ESX cluster, which is encapsulated in OpenNebula as an OpenNebula host.

- All ESX hosts belonging to the same ESX cluster to be exposed to OpenNebula **must** share at least one datastore among them.
 - The ESX cluster **should** have DRS enabled. DRS is not required but it is recommended. OpenNebula does not schedule to the granularity of ESX hosts, DRS is needed to select the actual ESX host within the cluster, otherwise the VM will be launched in the ESX where the VM template has been created.
 - **Save as VM Templates those VMs that will be instantiated through the OpenNebula provisioning portal**
 - To enable VNC functionality, repeat the following procedure for each ESX:
 - In the vSphere client proceed to Home -> Inventory -> Hosts and Clusters
 - Select the ESX host, Configuration tab and select Security Profile in the Software category
 - In the Firewall section, select Edit. Enable GDB Server, then click OK
 - Make sure that the ESX hosts are reachable from the OpenNebula Front-end
-

Important: OpenNebula will **NOT** modify any vCenter configuration.

3.3.2 Configuration

There are a few simple steps needed to configure OpenNebula so it can interact with vCenter:

Step 1: Check connectivity

The OpenNebula Front-end needs network connectivity to all the vCenters that it is supposed to manage.

Additionally, to enable VNC access to the spawned Virtual Machines, the Front-end also needs network connectivity to all the ESX hosts

Step 2: Enable the drivers in the Front-end (oned.conf)

In order to configure OpenNebula to work with the vCenter drivers, the following sections need to be uncommented or added in the `/etc/one/oned.conf` file:

```
#-----
# vCenter Information Driver Manager Configuration
#   -r number of retries when monitoring a host
#   -t number of threads, i.e. number of hosts monitored at the same time
#-----
IM_MAD = [
    NAME           = "vcenter",
    SUNSTONE_NAME  = "VMWare vCenter",
    EXECUTABLE     = "one_im_sh",
    ARGUMENTS      = "-c -t 15 -r 0 vcenter" ]
#-----
#-----
# vCenter Virtualization Driver Manager Configuration
```

```

# -r number of retries when monitoring a host
# -t number of threads, i.e. number of hosts monitored at the same time
# -p more than one action per host in parallel, needs support from hypervisor
# -s <shell> to execute commands, bash by default
# -d default snapshot strategy. It can be either 'detach' or 'suspend'. It
#     defaults to 'suspend'.
#-----
VM_MAD = [
  NAME           = "vcenter",
  SUNSTONE_NAME  = "VMWare vCenter",
  EXECUTABLE     = "one_vmm_sh",
  ARGUMENTS     = "-p -t 15 -r 0 vcenter -s sh",
  default       = "vmm_exec/vmm_exec_vcenter.conf",
  TYPE          = "xml",
  IMPORTED_VMS_ACTIONS = "terminate, terminate-hard, hold, release, suspend,
    resume, delete, reboot, reboot-hard, resched, unresched, poweroff,
    poweroff-hard, disk-attach, disk-detach, nic-attach, nic-detach,
    snap-create, snap-delete"
]
#-----

```

As a Virtualization driver, the vCenter driver accept a series of parameters that control their execution. The parameters allowed are:

parameter	description
-r <num>	number of retries when executing an action
-t <num>	number of threads, i.e. number of actions done at the same time

See the Virtual Machine drivers reference for more information about these parameters, and how to customize and extend the drivers.

OpenNebula needs to be restarted afterwards, this can be done with the following command:

```
$ sudo service opennebula restart
```

Step 3: Importing vCenter Clusters

OpenNebula ships with a powerful CLI tool to import vCenter clusters, VM Templates, Networks and running VMs. The tools is self-explanatory, just set the credentials and FQDN/IP to access the vCenter host and follow on screen instructions. A sample section follows:

```

$ onehost list
  ID NAME                CLUSTER  RVM    ALLOCATED_CPU    ALLOCATED_MEM STAT

$ onevcenter hosts --vcenter <vcenter-host> --vuser <vcenter-username> --vpass
↔<vcenter-password>
Connecting to vCenter: <vcenter-host>...done!
Exploring vCenter resources...done!
Do you want to process datacenter Development [y/n]? y
  * Import cluster clusterA [y/n]? y
    OpenNebula host clusterA with id 0 successfully created.

  * Import cluster clusterB [y/n]? y
    OpenNebula host clusterB with id 1 successfully created.

$ onehost list
  ID NAME                CLUSTER  RVM    ALLOCATED_CPU    ALLOCATED_MEM STAT

```



```

0 clusterA - 0 - - init
1 clusterB - 0 - - init
$ onehost list
ID NAME CLUSTER RVM ALLOCATED_CPU ALLOCATED_MEM STAT
0 clusterA - 0 0 / 800 (0%) 0K / 16G (0%) on
1 clusterB - 0 - - - init
$ onehost list
ID NAME CLUSTER RVM ALLOCATED_CPU ALLOCATED_MEM STAT
0 clusterA - 0 0 / 800 (0%) 0K / 16G (0%) on
1 clusterB - 0 0 / 1600 (0%) 0K / 16G (0%) on

```

The following variables are added to the OpenNebula hosts representing ESX clusters:

Operation	Note
VCENTER_HOST	hostname or IP of the vCenter host
VCENTER_USER	Name of the vCenter user
VCENTER_PASSWORD	Password of the vCenter user

Once the vCenter cluster is monitored, OpenNebula will display any existing VM as Wild. These VMs can be *imported* and managed through OpenNebula.

Note: OpenNebula will create a special key at boot time and save it in `/var/lib/one/.one/one_key`. This key will be used as a private key to encrypt and decrypt all the passwords for all the vCenters that OpenNebula can access. Thus, the password shown in the OpenNebula host representing the vCenter is the original password encrypted with this special key.

Note: OpenNebula will add by default a `one-<vid>-` prefix to the name of the vCenter VMs it spawns, where `<vid>` is the id of the VM in OpenNebula. This value can be changed using a special attribute set in the vCenter cluster representation in OpenNebula, ie, the OpenNebula host. This attribute is called `VM_PREFIX` (which can be set in the OpenNebula host template), and will evaluate one variable, `$i`, to the id of the VM. A value of `one-$i-` in that parameter would have the same behavior as the default.

After this guide, you may want to *verify your installation* or learn how to setup the *vmware-based cloud infrastructure*.

Step 4: Next Steps

Jump to the *Verify your Installation* section in order to get to launch a test VM.

3.4 Verify your Installation

This chapter ends with this optional section, where you will be able to test your cloud by launching a virtual machine and test that everything is working correctly.

You should only follow the specific subsection for your hypervisor.

3.4.1 KVM based Cloud Verification

The goal of this subsection is to launch a small Virtual Machine, in this case a TTYLinux (which is a very small Virtual Machine just for testing purposes).

Requirements

To complete this section, you must have network connectivity to the Internet from the Front-end, in order to import an appliance from <http://marketplace.opennebula.systems>.

Step 1. Check that the Hosts are on

The Hosts that you have registered should be in ON status.

The screenshot displays the OpenNebula Hosts management interface. The main content area shows a table of hosts with the following data:

ID	Name	Cluster	RVMs	Allocated CPU	Allocated MEM	Status
2	node01	default	0	0 / 400 (0%)	0KB / 7.7GB (0%)	ON

The status 'ON' is circled in red. Below the table, it shows 'Showing 1 to 1 of 1 entries' and a summary: '1 TOTAL 1 ON 0 OFF 0 ERROR'. The left sidebar contains navigation options: Dashboard, Instances, Templates, Storage, Network, Infrastructure, Clusters, Hosts, Zones, System, and Settings. The top right shows the user 'oneadmin' and the system 'OpenNebula'.

Step 2. Import an appliance from the MarketPlace.

We need to add an Image to our Cloud. To do so we can do so by navigating to Storage -> Apps in the left side menu (1). You will see a list of Appliances, you can filter by `tty kvm` (2) in order to find the one we are going to use in this guide. After that select it (3) and click to button with an arrow pointing to OpenNebula (4) in order to import it.

The screenshot shows the OpenNebula interface. On the left sidebar, the 'Apps' menu item is circled in red. In the main 'Apps' section, the 'OpenNebula' button in the top toolbar is circled in red. Below it, the search input field containing 'ttylinux - kvm' is also circled in red. A table below shows one app entry with ID 0, owned by 'oneadmin', named 'ttylinux - kvm', with a size of 40MB and a state of 'READY'. A red circle highlights the checkbox for this entry.

ID	Owner	Group	Name	Size	State	Registration Time	Marketplace	Zone
0	oneadmin	oneadmin	ttylinux - kvm	40MB	READY	01:00:00 01/01/1970	OpenNebula Public	0

A dialog with the information of the appliance you have selected will pop up. You will need to select the datastore under the “Select the Datastore to store the resource” (1) and then click on the “Download” button (2)

Download App To OpenNebula

The screenshot shows the 'Download App To OpenNebula' dialog. The 'Download' button is circled in red. Below it, there are input fields for 'Name' and 'VM Template Name', both containing 'ttylinux - kvm'. A section titled 'Select the Datastore to store the resource' shows a table with one entry, ID 1, circled in red.

ID	Owner	Group	Name	Capacity	Cluster	Type	Status
1	oneadmin	oneadmin	default	158.9GB / 461.5GB (34%)	0	IMAGE	ON

Navigate now to the Storage -> Images tab and refresh until the Status of the Image switches to READY.

OpenNebula

Images

oneadmin OpenNebula

Dashboard

Instances

Templates

Storage

Datastores

Images

Files

MarketPlaces

Apps

Network

Infrastructure

System

Settings

Support
Not connected
Sign in

OpenNebula 4.90.0
by OpenNebula Systems.

MarketPlace Clone

Search

ID	Owner	Group	Name	Datastore	Type	Status	#VMS
0	oneadmin	oneadmin	ttylinux - kvm	default	OS	READY	0

Showing 1 to 1 of 1 entries

Previous 1 Next

Step 3. Instantiate a VM

Navigate to Templates -> VMs, then select the `ttylinux - kvm` template that has been created and click on the “Instantiate”.

OpenNebula

VM Templates

oneadmin OpenNebula

Dashboard

Instances

Templates

VMs

Services

Storage

Network

Infrastructure

System

Settings

Support
Not connected
Sign in

OpenNebula 4.90.0
by OpenNebula Systems.

Update **Instantiate** Clone

Search

ID	Owner	Group	Name	Registration time
0	oneadmin	oneadmin	ttylinux - kvm	19:10:47 27/05/2016

Showing 1 to 1 of 1 entries

Previous 1 Next

In this dialog simply click on the “Instantiate” button.

Instantiate VM Template

oneadmin OpenNebula

Instantiate as persistent

VM Name
 Number of instances
 Hold

Capacity

Memory MB

CPU
 VCPU

Disks

DISK 0: ttylinux - kvm MB

Network

Step 4. Test VNC access

Navigate to Instances -> VMs. You will see that after a while the VM switches to Status RUNNING (you might need to click on the refresh button). Once it does, you can click on the VNC icon (at the right side of the image below).

OpenNebula VMs oneadmin OpenNebula

ID	Owner	Group	Name	Status	Host	IPs	
<input checked="" type="checkbox"/> 0	oneadmin	oneadmin	ttylinux - kvm-0	RUNNING	node01	--	<input checked="" type="button" value="VNC"/>

Showing 1 to 1 of 1 entries

Previous 1 Next

1 TOTAL 1 ACTIVE 0 OFF 0 PENDING 0 FAILED

If the VM fails, click on the VM and then in the Log tab to see why it failed. Alternatively, you can look at the log file `/var/log/one/<vmid>.log`.

Step 5. Adding Network connectivity to your VM

As you might have noticed, this VM does not have networking yet, this is because it depends a lot in your network technology. You would need to follow these steps in order to add Networking to the template.

1. Read the *Networking* chapter to choose a network technology, unless you have chosen to use the dummy driver explained in the *node installation* section and you have configured the bridges properly.
2. Create a new Network in the Network -> Virtual Networks dialog, and fill in the details of your network: like the Bridge, or the Physical Device, the IP ranges, etc.
3. Select the `ttylinux - kvm` template and update it. In the Network section of the template, select the network that you created in the previous step.
4. Instantiate and connect to the VM.

3.4.2 vCenter based Cloud Verification

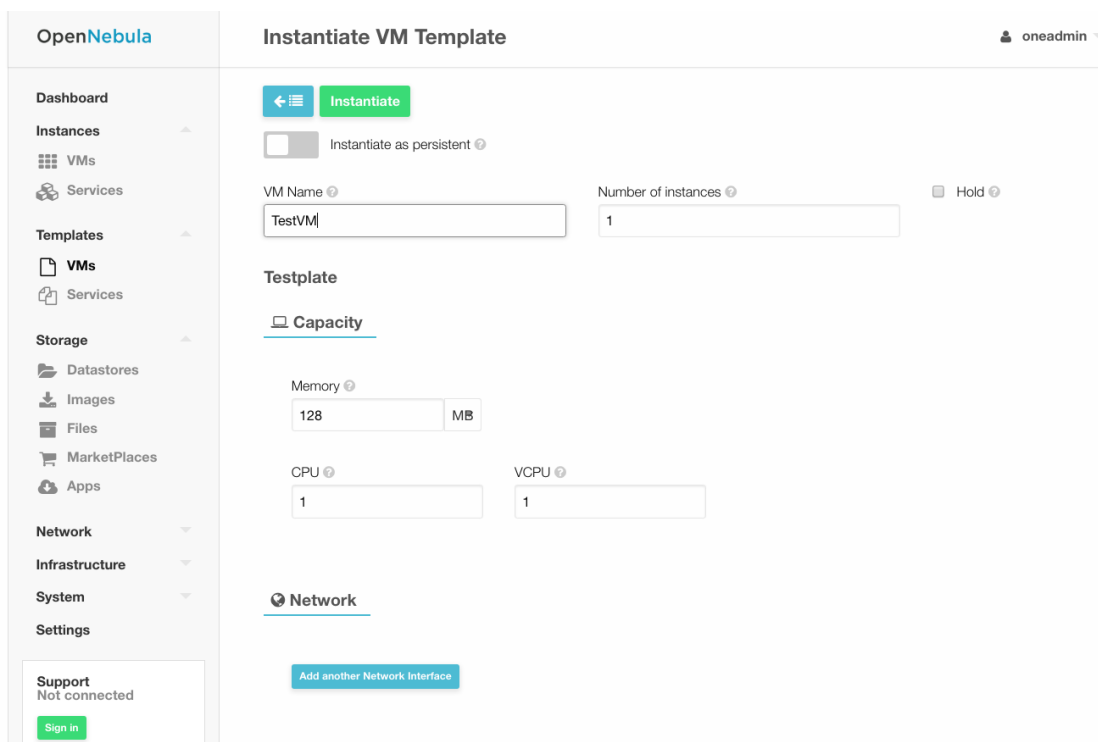
In order to verify the correct installation of your OpenNebula cloud, follow the next steps. The following assumes that Sunstone is up and running, as explained in the *front-end installation Section*. To access Sunstone, point your browser to `http://<fontend_address>:9869`.

Step 1. Import a VM Template

You will need a VM Template defined in vCenter, and then import it using the steps described in the *vCenter Node Section*.

Step 2. Instantiate the VM Template

You can easily instantiate the template to create a new VM from it using Sunstone. Proceed to the Templates tab of the left side menu, VMs Section, select the imported template and click on the Instantiate button.



Step 3. Check the VM is Running

The scheduler should place the VM in the vCenter cluster imported as part of the *vCenter Node Installation* Section.

After a few minutes (depending on the size of the disks defined by the VM Template), the state of the VM should be **RUNNING**. You can check the process in Sunstone in the Instances tab of the left side menu, VMs Section.

Once the VM is running, click on the VNC blue icon, and if you can see a console to the VM, congratulations! You have a fully functional OpenNebula cloud.



The next step would be to further configure the OpenNebula cloud to suits your needs. You can learn more in the *VMware Infrastructure Setup* guide.

3.4.3 Next steps

After this chapter, you are ready to start using your cloud or you could configure more components:

- *Authenticaton*. (Optional) For integrating OpenNebula with LDAP/AD, or securing it further with other authentication technologies.

- *Sunstone*. OpenNebula GUI should working and accessible at this stage, but by reading this guide you will learn about specific enhanced configurations for Sunstone.

If your cloud is KVM based you should also follow:

- *Open Cloud Host Setup*.
- *Open Cloud Storage Setup*.
- *Open Cloud Networking Setup*.

Otherwise, if it's VMware based:

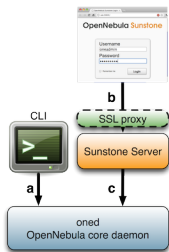
- Head over to the *VMware Infrastructure Setup* chapter.

AUTHENTICATION SETUP

4.1 Overview

OpenNebula comes by default with an internal user/password authentication system, see the Users & Groups Subsystem guide for more information. You can enable an external Authentication driver.

4.1.1 Authentication



In the figure to the right of this text you can see three authentication configurations you can customize in OpenNebula.

a) CLI/API Authentication

You can choose from the following authentication drivers to access OpenNebula from the command line:

- Built-in User/Password and token authentication
- *SSH Authentication*
- *X509 Authentication*
- *LDAP Authentication*

b) Sunstone Authentication

By default any authentication driver configured to work with OpenNebula can be used out-of-the-box with Sunstone. Additionally you can add SSL security to Sunstone as described in the *Sunstone documentation*

c) Servers Authentication

This method is designed to delegate the authentication process to high level tools interacting with OpenNebula. You'll be interested in this method if you are developing your own servers.

By default, OpenNebula ships with two servers: *Sunstone* and EC2. When a user interacts with one of them, the server authenticates the request and then forwards the requested operation to the OpenNebula daemon.

The forwarded requests are encrypted by default using a Symmetric Key mechanism. The following guide shows how to strengthen the security of these requests using x509 certificates. This is specially relevant if you are running your server in a machine other than the front-end.

- *Cloud Servers Authentication*

4.1.2 How Should I Read This Chapter

When designing the architecture of your cloud you will have to choose where to store user's credentials. Different authentication methods can be configured at the same time and selected per user basis. One big distinction between the different authentication methods is the ability to be used by API, CLI and/or only Sunstone (web interface).

Can be used with API, CLI and Sunstone:

- Built-in User/Password
- LDAP

Can be used only with API and CLI:

- SSH

Can be used only with Sunstone:

- X509

The following sections are self-contained so you can directly go to the guide that describes the configuration for the chosen auth method.

4.1.3 Hypervisor Compatibility

Section	Compatibility
Built-in User/Password and token authentication	This Section applies to both KVM and vCenter.
<i>SSH Authentication</i>	This Section applies to both KVM and vCenter.
<i>X509 Authentication</i>	This Section applies to both KVM and vCenter.
<i>LDAP Authentication</i>	This Section applies to both KVM and vCenter.
<i>Sunstone documentation</i>	This Section applies to both KVM and vCenter.

4.2 SSH Authentication

This guide will show you how to enable and use the SSH authentication for the OpenNebula CLI. Using this authentication method, users login to OpenNebula with a token encrypted with their private ssh keys.

4.2.1 Requirements

You don't need to install any additional software.

4.2.2 Considerations & Limitations

With the current release, this authentication method is only valid to interact with OpenNebula using the CLI.

4.2.3 Configuration

OpenNebula Configuration

The Auth MAD and ssh authentication is enabled by default. In case it does not work make sure that the authentication method is in the list of enabled methods.

```
AUTH_MAD = [
  executable = "one_auth_mad",
  authn = "ssh,x509,ldap,server_cipher,server_x509"
]
```

There is an external plain user/password authentication driver, and existing accounts will keep working as usual.

4.2.4 Usage

Create New Users

This authentication method uses standard ssh RSA key pairs for authentication. Users can create these files **if they don't exist** using this command:

```
$ ssh-keygen -t rsa
```

OpenNebula commands look for the files generated at the standard location (`$HOME/.ssh/id_rsa`) so it is a good idea not to change the default path. It is also a good idea to protect the private key with a password.

The users requesting a new account have to generate a public key and send it to the administrator. The way to extract it is the following:

```
$ oneuser key
Enter PEM pass phrase:
MIIBCAKCAQEApUO+JISjsf02rFVtDrlyar/34EoUoVETx0n+RqWNav+5wi+gHiPp3e03AfEkXzjDYi8F
voS4a4456f1OUQlQddfyPECn590eX8Zu4DH3gp1VUuDeeE8WJWyAzdK5hg6F+RdyP1pT26mnyunZB8Xd
b1l8seoIAQiOS6t1VfA8FrtwLGmdEETfttS9ukyGxw5vdTp1se/fcam+r9AXBR06zjc77x+DbRFbXcgI
1XIdpVrjCFL0fdN53L0aU7kTE9VNEXR×K8sPv1Nfx+FQWpX/HtH8ICs5WREsZGmXPAO/IkrSpMVg5taS
jie9JAQOMesjFIwgTWBUh6cNXuYsQ/5wIwIBIw==
```

The string written to the console must be sent to the administrator, so the can create the new user in a similar way as the default user/password authentication users.

The following command will create a new user with username 'newuser', assuming that the previous public key is saved in the text file `/tmp/pub_key`:

```
$ oneuser create newuser --ssh --read-file /tmp/pub_key
```

Instead of using the `--read-file` option, the public key could be specified as the second parameter.

If the administrator has access to the user's **private ssh key**, he can create new users with the following command:

```
$ oneuser create newuser --ssh --key /home/newuser/.ssh/id_rsa
```

Update Existing Users to SSH

You can change the authentication method of an existing user to SSH with the following commands:

```
$ oneuser chauth <id|name> ssh
$ oneuser passwd <id|name> --ssh --read-file /tmp/pub_key
```

As with the `create` command, you can specify the public key as the second parameter, or use the user's private key with the `--key` option.

User Login

Users must execute the 'oneuser login' command to generate a login token. The token will be stored in the `$ONE_AUTH` environment variable. The command requires the OpenNebula username, and the authentication method (`--ssh` in this case).

```
$ oneuser login newuser --ssh
```

The default ssh key is assumed to be in `~/.ssh/id_rsa`, otherwise the path can be specified with the `--key` option.

The generated token has a default **expiration time** of 10 hour. You can change that with the `--time` option.

4.3 x509 Authentication

This guide will show you how to enable and use the x509 certificates authentication with OpenNebula. The x509 certificates can be used in two different ways in OpenNebula.

The first option that is explained in this guide enables us to use certificates with the CLI. In this case the user will generate a login token with his private key, OpenNebula will validate the certificate and decrypt the token to authenticate the user.

The second option enables us to use certificates with Sunstone and the Public Cloud servers included in OpenNebula. In this case the authentication is leveraged to Apache or any other SSL capable HTTP proxy that has to be configured by the administrator. If this certificate is validated the server will encrypt those credentials using a server certificate and will send the token to OpenNebula.

4.3.1 Requirements

If you want to use the x509 certificates with Sunstone or one of the Public Clouds, you must deploy a SSL capable HTTP proxy on top of them in order to handle the certificate validation.

4.3.2 Considerations & Limitations

The X509 driver uses the certificate DN as user passwords. The x509 driver will remove any space in the certificate DN. This may cause problems in the unlikely situation that you are using a CA signing certificate subjects that only differ in spaces.

4.3.3 Configuration

The following table summarizes the available options for the x509 driver (`/etc/one/auth/x509_auth.conf`):

VARIABLE	VALUE
<code>:ca_dir</code>	Path to the trusted CA directory. It should contain the trusted CA's for the server, each CA certificate should be name <code>CA_hash.0</code>
<code>:check_crl</code>	By default, if you place CRL files in the CA directory in the form <code>CA_hash.r0</code> , OpenNebula will check them. You can enforce CRL checking by defining <code>:check_crl</code> , i.e. authentication will fail if no CRL file is found. You can always disable this feature by moving or renaming <code>.r0</code> files

Follow these steps to change oneadmin's authentication method to x509:

Warning: You should have another account in the oneadmin group, so you can revert these steps if the process fails.

- *Change the oneadmin password* to the oneadmin certificate DN.

```
$ oneuser chauth 0 x509 --x509 --cert /tmp/newcert.pem
```

- *Add trusted CA certificates* to the certificates directory

```
$ openssl x509 -noout -hash -in cacert.pem
78d0bbd8
$ sudo cp cacert.pem /etc/one/auth/certificates/78d0bbd8.0
```

- *Create a login* for oneadmin using the `-x509` option. This token has a default expiration time set to 1 hour, you can change this value using the option `-time`.

```
$ oneuser login oneadmin --x509 --cert newcert.pem --key newkey.pem
Enter PEM pass phrase:
export ONE_AUTH=/home/oneadmin/.one/one_x509
```

- Set `ONE_AUTH` to the x509 login file

```
$ export ONE_AUTH=/home/oneadmin/.one/one_x509
```

4.3.4 Usage

Add and Remove Trusted CA Certificates

You need to copy all trusted CA certificates to the certificates directory, renaming each of them as `<CA_hash>.0`. The hash can be obtained with the `openssl` command:

```
$ openssl x509 -noout -hash -in cacert.pem
78d0bbd8
$ sudo cp cacert.pem /etc/one/auth/certificates/78d0bbd8.0
```

To stop trusting a CA, simply remove its certificate from the certificates directory.

This process can be done without restarting OpenNebula, the driver will look for the certificates each time an authentication request is made.

Create New Users

The users requesting a new account have to send their certificate, signed by a trusted CA, to the administrator. The following command will create a new user with username 'newuser', assuming that the user's certificate is saved in the file /tmp/newcert.pem:

```
$ oneuser create newuser --x509 --cert /tmp/newcert.pem
```

This command will create a new user whose password contains the subject DN of his certificate. Therefore if the subject DN is known by the administrator the user can be created as follows:

```
$ oneuser create newuser --x509 "user_subject_DN"
```

Update Existing Users to x509 & Multiple DN

You can change the authentication method of an existing user to x509 with the following command:

- Using the user certificate:

```
$ oneuser chauth <id|name> x509 --x509 --cert /tmp/newcert.pem
```

- Using the user certificate subject DN:

```
$ oneuser chauth <id|name> x509 --x509 "user_subject_DN"
```

You can also map multiple certificates to the same OpenNebula account. Just add each certificate DN separated with '|' to the password field.

```
$ oneuser passwd <id|name> --x509 "/DC=es/O=one/CN=user|/DC=us/O=two/CN=user"
```

User Login

Users must execute the 'oneuser login' command to generate a login token. The token will be stored in the \$ONE_AUTH environment variable. The command requires the OpenNebula username, and the authentication method (--x509 in this case).

```
newuser@frontend $ oneuser login newuser --x509 --cert newcert.pem --key newkey.pem
Enter PEM pass phrase:
```

The generated token has a default **expiration time** of 10 hours. You can change that with the --time option.

4.3.5 Tuning & Extending

The x509 authentication method is just one of the drivers enabled in AUTH_MAD. All drivers are located in /var/lib/one/remotes/auth.

OpenNebula is configured to use x509 authentication by default. You can customize the enabled drivers in the AUTH_MAD attribute of *oned.conf*. More than one authentication method can be defined:

```
AUTH_MAD = [
  executable = "one_auth_mad",
  authn = "ssh,x509,ldap,server_cipher,server_x509"
]
```

4.3.6 Enabling x509 auth in Sunstone

Update the `/etc/one/sunstone-server.conf` `:auth` parameter to use the x509 auth:

```
:auth: x509
```

4.4 LDAP Authentication

The LDAP Authentication add-on permits users to have the same credentials as in LDAP, so effectively centralizing authentication. Enabling it will let any correctly authenticated LDAP user to use OpenNebula.

4.4.1 Prerequisites

Warning: This Add-on requires the `'net/ldap'` ruby library provided by the `'net-ldap'` gem.

This Add-on will not install any Ldap server or configure it in any way. It will not create, delete or modify any entry in the Ldap server it connects to. The only requirement is the ability to connect to an already running Ldap server and being able to perform a successful `ldapbind` operation and have a user able to perform searches of users, therefore no special attributes or values are required in the LDIF entry of the user authenticating.

4.4.2 Configuration

Configuration file for auth module is located at `/etc/one/auth/ldap_auth.conf`. This is the default configuration:

```
server 1:
  # Ldap user able to query, if not set connects as anonymous. For
  # Active Directory append the domain name. Example:
  # Administrator@my.domain.com
  #:user: 'admin'
  #:password: 'password'

  # Ldap authentication method
  :auth_method: :simple

  # Ldap server
  :host: localhost
  :port: 389

  # Uncomment this line for tls connections
  #:encryption: :simple_tls

  # base hierarchy where to search for users and groups
  :base: 'dc=domain'

  # group the users need to belong to. If not set any user will do
  #:group: 'cn=cloud,ou=groups,dc=domain'

  # field that holds the user name, if not set 'cn' will be used
  :user_field: 'cn'
```

```

# for Active Directory use this user_field instead
#:user_field: 'sAMAccountName'

# field name for group membership, by default it is 'member'
#:group_field: 'member'

# user field that that is in in the group group_field, if not set 'dn' will be_
↪used
#:user_group_field: 'dn'

# Generate mapping file from group template info
:mapping_generate: true

# Seconds a mapping file remain untouched until the next regeneration
:mapping_timeout: 300

# Name of the mapping file in OpenNebula var directory
:mapping_filename: server1.yaml

# Key from the OpenNebula template to map to an AD group
:mapping_key: GROUP_DN

# Default group ID used for users in an AD group not mapped
:mapping_default: 1

# this example server wont be called as it is not in the :order list
server 2:
  :auth_method: :simple
  :host: localhost
  :port: 389
  :base: 'dc=domain'
  #:group: 'cn=cloud,ou=groups,dc=domain'
  :user_field: 'cn'

# List the order the servers are queried
:order:
  - server 1
  #- server 2

```

The structure is a hash where any key different to `:order` will contain the configuration of one ldap server we want to query. The special key `:order` holds an array with the order we want to query the configured servers. Any server not listed in `:order` wont be queried.

VARIABLE	DESCRIPTION
:user	Name of the user that can query ldap. Do not set it if you can perform queries anonymously
:password	Password for the user defined in :user. Do not set if anonymous access is enabled
:auth_method	Can be set to :simple_tls if SSL connection is needed
:encryption	Can be set to :simple_tls if SSL connection is needed
:host	Host name of the ldap server
:port	Port of the ldap server
:base	Base leaf where to perform user searches
:group	If set the users need to belong to this group
:user_field	Field in ldap that holds the user name
:mapping_generate	Generate automatically a mapping file. It can be disabled in case it needs to be done manually
:mapping_time	Number of seconds between automatic mapping file generation
:mapping_file	Name of the mapping file. Should be different for each server
:mapping_key	Key in the group template used to generate the mapping file. It should hold the DN of the mapped group
:mapping_default	Default group used when no mapped group is found. Set to false in case you don't want the user to be authorized if it does not belong to a mapped group

To enable ldap authentication the described parameters should be configured. OpenNebula must be also configured to enable external authentication. Add this line in `/etc/one/oned.conf`

```
DEFAULT_AUTH = "ldap"
```

4.4.3 User Management

Using LDAP authentication module the administrator doesn't need to create users with `oneuser` command as this will be automatically done.

Users can store their credentials into `$ONE_AUTH` file (usually `$HOME/.one/one_auth`) in this fashion:

```
<user_dn>:ldap_password
```

where

- `<user_dn>` the DN of the user in the LDAP service
- `ldap_password` is the password of the user in the LDAP service

Alternatively a user can generate an authentication token using the `oneuser login` command, so there is no need to keep the ldap password in a plain file. Simply input the `ldap_password` when requested. More information on the management of login tokens and `$ONE_AUTH` file can be found in [Managing Users Guide](#).

DN's With Special Characters

When the user dn or password contains blank spaces the LDAP driver will escape them so they can be used to create OpenNebula users. Therefore, users needs to set up their `$ONE_AUTH` file accordingly.

Users can easily create escaped `$ONE_AUTH` tokens with the command `oneuser encode <user> [<password>]`, as an example:

```
$ oneuser encode 'cn=First Name,dc=institution,dc=country' 'pass word'
cn=First%20Name,dc=institution,dc=country:pass%20word
```

The output of this command should be put in the `$ONE_AUTH` file.

4.4.4 Active Directory

LDAP Auth drivers are able to connect to Active Directory. You will need:

- Active Directory server with support for simple user/password authentication.
- User with read permissions in the Active Directory user's tree.

You will need to change the following values in the configuration file (`/etc/one/auth/ldap_auth.conf`):

- `:user:` the Active Directory user with read permissions in the user's tree plus the domain. For example for user **Administrator** at domain **win.opennebula.org** you specify it as `Administrator@win.opennebula.org`
- `:password:` password of this user
- `:host:` hostname or IP of the Domain Controller
- `:base:` base DN to search for users. You need to decompose the full domain name and use each part as DN component. Example, for `win.opennebula.org` you will get the base DN: `DN=win, DN=opennebula, DN=org`
- `:user_field:` set it to `sAMAccountName`

`:group` parameter is still not supported for Active Directory, leave it commented.

4.4.5 Group Mapping

You can make new users belong to an specific group upon creation. To do this a mapping is generated from the LDAP group to an existing OpenNebula group. This system uses a mapping file specified by `:mapping_file` parameter and resides in `OpenNebula var` directory. The mapping file can be generated automatically using data in the group template that tells which LDAP group maps to that specific group. For example we can add in the group template this line:

```
GROUP_DN="CN=technicians,CN=Groups,DC=example,DC=com"
```

And in the `ldap` configuration file we set the `:mapping_key` to `GROUP_DN`. This tells the driver to look for the group DN in that template parameter. This mapping expires the number of seconds specified by `:mapping_timeout`. This is done so the authentication is not continually querying OpenNebula.

You can also disable the automatic generation of this file and do the mapping manually. The mapping file is in YAML format and contains a hash where the key is the LDAP's group DN and the value is the ID of the OpenNebula group. For example:

```
CN=technicians,CN=Groups,DC=example,DC=com: '100'
CN=Domain Admins,CN=Users,DC=example,DC=com: '101'
```

When several servers are configured you should have different `:mapping_key` and `:mapping_file` values for each one so they don't collide. For example:

```
internal:
  :mapping_file: internal.yaml
  :mapping_key: INTERNAL_GROUP_DN

external:
  :mapping_file: external.yaml
  :mapping_key: EXTERNAL_GROUP_DN
```

And in the OpenNebula group template you can define two mappings, one for each server:

```
INTERNAL_GROUP_DN="CN=technicians,CN=Groups,DC=internal,DC=com"  
EXTERNAL_GROUP_DN="CN=staff,DC=other-company,DC=com"
```

4.4.6 Enabling LDAP auth in Sunstone

Update the `/etc/one/sunstone-server.conf` `:auth` parameter to use the `opennebula`:

```
:auth: opennebula
```

Using this method the credentials provided in the login screen will be sent to the OpenNebula core and the authentication will be delegated to the OpenNebula auth system, using the specified driver for that user. Therefore any OpenNebula auth driver can be used through this method to authenticate the user (i.e: LDAP).

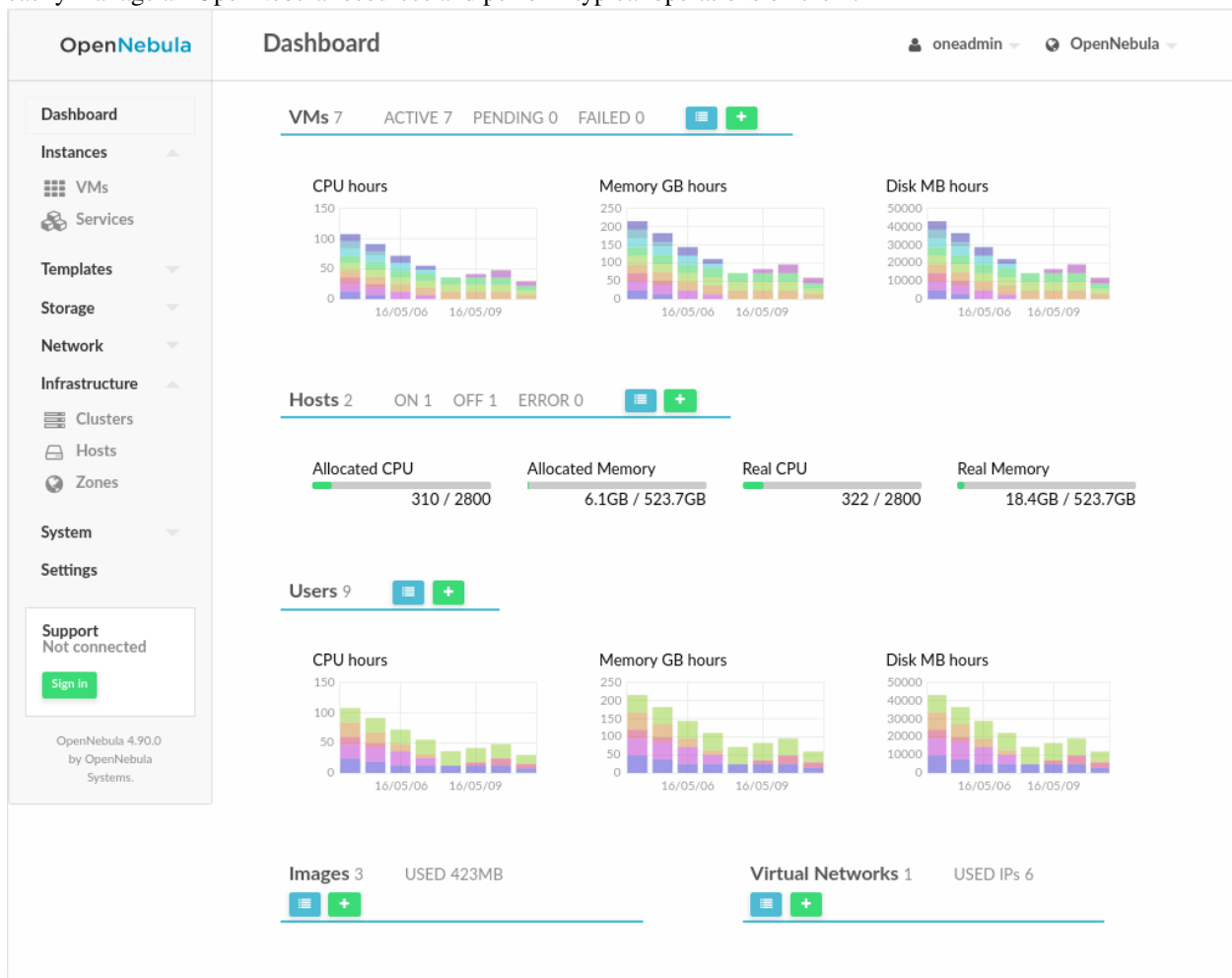
To automatically encode credentials as explained in *DN's with special characters* section also add this parameter to sunstone configuration:

```
:encode_user_password: true
```

SUNSTONE SETUP

5.1 Overview

OpenNebula Sunstone is a Graphical User Interface (GUI) intended for both end users and administrators that simplifies the typical management operations in private and hybrid cloud infrastructures. OpenNebula Sunstone allows to easily manage all OpenNebula resources and perform typical operations on them.



5.1.1 How Should I Read this Chapter

The *Sunstone Installation & Configuration* section describes the configuration and customization options for Sunstone. After Sunstone is running, you can define different sunstone behaviors for each user role in the *Sunstone Views* section. For more information on how to customize and extend you Sunstone deployment use the following links:

- *Security & Authentication Methods*, improve security with x509 authentication and SSL
- *Cloud Servers Authentication*, advanced reference about the security between Sunstone and OpenNebula.
- *Advanced Deployments*, improving scalability and isolating the server

5.1.2 Hypervisor Compatibility

Sunstone is available for all the hypervisors. When using vCenter, the cloud admin should enable the `admin_vcenter`, `groupadmin_vcenter` and `cloud_vcenter` *Sunstone views*.

5.2 Sunstone Installation & Configuration

5.2.1 Requirements

You must have an OpenNebula site properly configured and running to use OpenNebula Sunstone, be sure to check the *OpenNebula Installation and Configuration Guides* to set up your private cloud first. This section also assumes that you are familiar with the configuration and use of OpenNebula.

OpenNebula Sunstone was installed during the OpenNebula installation. If you followed the *installation guide* then you already have all ruby gem requirements. Otherwise, run the `install_gem` script as root:

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

The Sunstone Operation Center offers the possibility of starting a VNC/SPICE session to a Virtual Machine. This is done by using a VNC/SPICE websocket-based client (noVNC) on the client side and a VNC proxy translating and redirecting the connections on the server-side.

Warning: The SPICE Web client is a prototype and is limited in function. More information of this component can be found in the following [link](#)

Warning: Make sure that there is free space in sunstone's log directory or it will die silently. By default the log directory is `/var/log/one`.

Requirements:

- Websockets-enabled browser (optional): Firefox and Chrome support websockets. In some versions of Firefox manual activation is required. If websockets are not enabled, flash emulation will be used.
- Installing the python-numpy package is recommended for a better vnc performance.

5.2.2 Considerations & Limitations

OpenNebula Sunstone supports Chrome, Firefox and Internet Explorer 11. Other browsers are not supported and may not work well.

Note: Internet Explorer is **not** supported with the Compatibility Mode enabled, since it emulates IE7 which is not supported.

5.2.3 Configuration

sunstone-server.conf

The Sunstone configuration file can be found at `/etc/one/sunstone-server.conf`. It uses YAML syntax to define some options:

Available options are:

Option	Description
:tmpdir	Uploaded images will be temporally stored in this folder before being copied to OpenNebula
:one_xmlrpc	OpenNebula daemon host and port
:host	IP address on which the server will listen on. 0.0.0.0 by default.
:port	Port on which the server will listen. 9869 by default.
:sessions	Method of keeping user sessions. It can be <code>memory</code> or <code>memcache</code> . For server that spawn more than one process (like Passenger or Unicorn) <code>memcache</code> should be used
:mem-cache_host	Host where memcached server resides
:mem-cache_port	Port of memcached server
:mem-cache_namespace	memcache namespace where to store sessions. Useful when memcached server is used by more services
:debug_level	Log debug level: 0 = ERROR, 1 = WARNING, 2 = INFO, 3 = DEBUG
:env	Execution environment for Sunstone. <code>dev</code> , Instead of pulling the minified js all the files will be pulled (<code>app/main.js</code>). Check the Building from Source guide in the docs, for details on how to run Sunstone in development. <code>prod</code> , the minified js will be used (<code>dist/main.js</code>)
:auth	Authentication driver for incoming requests. Possible values are <code>sunstone</code> , <code>openebula</code> , <code>remote</code> and <code>x509</code> . Check authentication methods for more info
:core_auth	Authentication driver to communicate with OpenNebula core. Possible values are <code>x509</code> or <code>cipher</code> . Check cloud_auth for more information
:encode_user_password	For external authentication drivers, such as LDAP. Performs a URL encoding on the credentials word to OpenNebula, e.g. <code>secret%20password</code> . This only works with “openebula” auth.
:lang	Default language for the Sunstone interface. This is the default language that will be used if user has not defined a variable LANG with a different valid value its user template
:vnc_proxy_port	Base port for the VNC proxy. The proxy will run on this port as long as Sunstone server does. 29876 by default.
:vnc_proxy_support_wss	<code>only</code> , only. If enabled, the proxy will be set up with a certificate and a key to use secure websockets. If set to <code>only</code> the proxy will only accept encrypted connections, otherwise it will accept both encrypted or unencrypted ones.
:vnc_proxy_cert	Full path to certificate file for wss connections.
:vnc_proxy_key	Full path to key file. Not necessary if key is included in certificate.
:vnc_proxy_ipv6	Enable ipv6 for novnc. (true or false)
:vnc_request_password	Request VNC password for external windows, by default it will not be requested (true or false)
:table_order	Default table order, resources get ordered by ID in <code>asc</code> or <code>desc</code> order.
:market-place_username	Username credential to connect to the Marketplace.
:market-place_password	Password to connect to the Marketplace.
:market-place_url	Endpoint to connect to the Marketplace. If commented, a 503 <code>service unavailable</code> error will be returned to clients.
:one-flow_server	Endpoint to connect to the OneFlow server.
:routes	List of files containing custom routes to be loaded. Check server plugins for more info.

Starting Sunstone

To start Sunstone just issue the following command as `oneadmin`

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

You can find the Sunstone server log file in `/var/log/one/sunstone.log`. Errors are logged in `/var/log/one/sunstone.error`.

5.2.4 Commercial Support Integration

We are aware that in production environments, access to professional, efficient support is a must, and this is why we have introduced an integrated tab in Sunstone to access [OpenNebula Systems](#) (the company behind OpenNebula, formerly C12G) professional support. In this way, support ticket management can be performed through Sunstone, avoiding disruption of work and enhancing productivity.

The screenshot shows the Sunstone interface with the 'Commercial Support Requests' tab selected. The sidebar on the left lists various system components, with 'Support' highlighted and marked as 'Not connected'. The main content area provides information about the support subscription, including a list of benefits such as problem diagnosis, solving unexpected problems, performance tuning, and answering 'how to' questions. A 'Commercial Support' login form is displayed, featuring input fields for 'Email' (containing 'sysadmin@opennebula.org') and 'Password', a 'Sign in' button, and a 'Get an account' button. Below the form are links for 'Documentation' and 'Community'.

This tab can be disabled in the sunstone views configuration files

This tab and can be disabled in each one of the *view yaml files*.

```
enabled_tabs:
  [ ... ]
  #- support-tab
```

5.2.5 Troubleshooting

Cannot connect to OneFlow server

The Service instances and templates tabs may show the following message:

```
Cannot connect to OneFlow server
```


You need to start the OneFlow component following this section, or disable the Service and Service Templates menu entries in the *Sunstone views yaml files*.

VNC Troubleshooting

When clicking the VNC icon, the process of starting a session begins:

- A request is made and if a VNC session is possible, Sunstone server will add the VM Host to the list of allowed vnc session targets and create a random token associated to it.
- The server responds with the session token, then a noVNC dialog pops up.
- The VNC console embedded in this dialog will try to connect to the proxy either using websockets (default) or emulating them using Flash. Only connections providing the right token will be successful. The token expires and cannot be reused.

There can be multiple reasons that may prevent noVNC from correctly connecting to the machines. Here's a checklist of common problems:

- noVNC requires Python ≥ 2.5 for the websockets proxy to work. You may also need additional modules as `python2<version>-numpy`.
- You can retrieve useful information from `/var/log/one/novnc.log`
- You must have a GRAPHICS section in the VM template enabling VNC, as stated in the documentation. Make sure the attribute IP is set correctly (0.0.0.0 to allow connections from everywhere), otherwise, no connections will be allowed from the outside.
- Your browser must support websockets, and have them enabled. This is the default in current Chrome and Firefox, but former versions of Firefox (i.e. 3.5) required manual activation. Otherwise Flash emulation will be used.
- Make sure there are not firewalls blocking the connections. The proxy will redirect the websocket data from the VNC proxy port to the VNC port stated in the template of the VM. The value of the proxy port is defined in `sunstone-server.conf`.
- Make sure that you can connect directly from Sunstone frontend to the VM using a normal VNC client tools such as `vncviewer`.

- When using secure websockets, make sure that your certificate and key (if not included in certificate) are correctly set in Sunstone configuration files. Note that your certificate must be valid and trusted for the wss connection to work. If you are working with a certificate that it is not accepted by the browser, you can manually add it to the browser trust-list visiting `https://sunstone.server.address:vnc_proxy_port`. The browser will warn that the certificate is not secure and prompt you to manually trust it.
- If your connection is very, very, very slow, there might be a token expiration issue. Please try the manual proxy launch as described below to check it.
- Doesn't work yet? Try launching Sunstone, killing the websockify proxy and relaunching the proxy manually in a console window with the command that is logged at the beginning of `/var/log/one/novnc.log`. You must generate a lock file containing the PID of the python process in `/var/lock/one/.novnc.lock`. Leave it running and click on the VNC icon on Sunstone for the same VM again. You should see some output from the proxy in the console and hopefully the cause of why the connection does not work.
- Please contact the support forum only when you have gone through the suggestion above and provide full sunstone logs, shown errors and any relevant information of your infrastructure (if there are Firewalls etc)
- The message "SecurityError: The operation is insecure." is usually related to a Same-Origin-Policy problem. If you have Sunstone TLS secured and try to connect to an insecure websocket for VNC, Firefox blocks that. For Firefox, you need to have both connections secured to not get this error. And don't use a self-signed certificate for the server, this would raise the error again (you can setup your own little CA, that works, but don't use a self-signed server certificate). The other option would be to go into the Firefox config (`about:config`) and set "network.websocket.allowInsecureFromHTTPS" to "true".

5.2.6 Tuning & Extending

Internationalization and Languages

Sunstone supports multiple languages. If you want to contribute a new language, make corrections or complete a translation, you can visit our:

- [Transifex project page](#)

Translating through Transifex is easy and quick. All translations should be submitted via Transifex.

Users can update or contribute translations anytime. Prior to every release, normally after the beta release, a call for translations will be made in the forum. Then the source strings will be updated in Transifex so all the translations can be updated to the latest OpenNebula version. Translation with an acceptable level of completeness will be added to the final OpenNebula release.

Customize the VM Logos

The VM Templates have an image logo to identify the guest OS. To modify the list of available logos, or to add new ones, edit `/etc/one/sunstone-logos.yaml`.

```
- { 'name': "Arch Linux",      'path': "images/logos/arch.png" }
- { 'name': "CentOS",        'path': "images/logos/centos.png" }
- { 'name': "Debian",        'path': "images/logos/debian.png" }
- { 'name': "Fedora",        'path': "images/logos/fedora.png" }
- { 'name': "Linux",         'path': "images/logos/linux.png" }
- { 'name': "Redhat",        'path': "images/logos/redhat.png" }
- { 'name': "Ubuntu",        'path': "images/logos/ubuntu.png" }
- { 'name': "Windows XP/2003", 'path': "images/logos/windowsxp.png" }
- { 'name': "Windows 8",     'path': "images/logos/windows8.png" }
```

Create VM Template

Branding the Sunstone Portal

You can easily add your logos to the login and main screens by updating the `logo:` attribute as follows:

- The login screen is defined in the `/etc/one/sunstone-views.yaml`.
- The logo of the main UI screen is defined for each view in *the view yaml file*.

sunstone-views.yaml

OpenNebula Sunstone can be adapted to different user roles. For example, it will only show the resources the users have access to. Its behavior can be customized and extended via *views*.

The preferred method to select which views are available to each group is to update the group configuration from Sunstone; as described in *Sunstone Views section*. There is also the `/etc/one/sunstone-views.yaml` file that defines an alternative method to set the view for each user or group.

Sunstone will calculate the views available to each user using:

- From all the groups the user belongs to, the views defined inside each group are combined and presented to the user
- If no views are available from the user's group, the defaults would be fetched from `/etc/one/sunstone-views.yaml`. Here, views can be defined for:
 - Each user (`users:` section), list each user and the set of views available for her.
 - Each group (`groups:` section), list the set of views for the group.
 - The default view, if a user is not listed in the `users:` section, nor its group in the `groups:` section, the default views will be used.
 - The default views for group admins, if a group admin user is not listed in the `users:` section, nor its group in the `groups:` section, the `default_groupadmin` views will be used.

By default users in the `oneadmin` group have access to all views, and users in the `users` group can use the `cloud` view.

The following `/etc/one/sunstone-views.yaml` example enables the user (`user.yaml`) and the cloud (`cloud.yaml`) views for helen and the cloud (`cloud.yaml`) view for group `cloud-users`. If more than one view is available for a given user the first one is the default.

```

---
logo: images/opennebula-sunstone-v4.0.png
users:
  helen:
    - cloud
    - user
groups:
  cloud-users:
    - cloud
default:
  - user
default_groupadmin:
  - groupadmin
  - cloud

```

A Different Endpoint for Each View

OpenNebula *Sunstone views* can be adapted to deploy a different endpoint for each kind of user. For example if you want an endpoint for the admins and a different one for the cloud users. You will just have to deploy a *new sunstone server* and set a default view for each sunstone instance:

```

# Admin sunstone
cat /etc/one/sunstone-server.conf
...
:host: admin.sunstone.com
...

cat /etc/one/sunstone-views.yaml
...
users:
groups:
default:
  - admin

```

```

# Users sunstone
cat /etc/one/sunstone-server.conf
...
:host: user.sunstone.com
...

cat /etc/one/sunstone-views.yaml
...
users:
groups:
default:
  - user

```

5.3 Sunstone Views

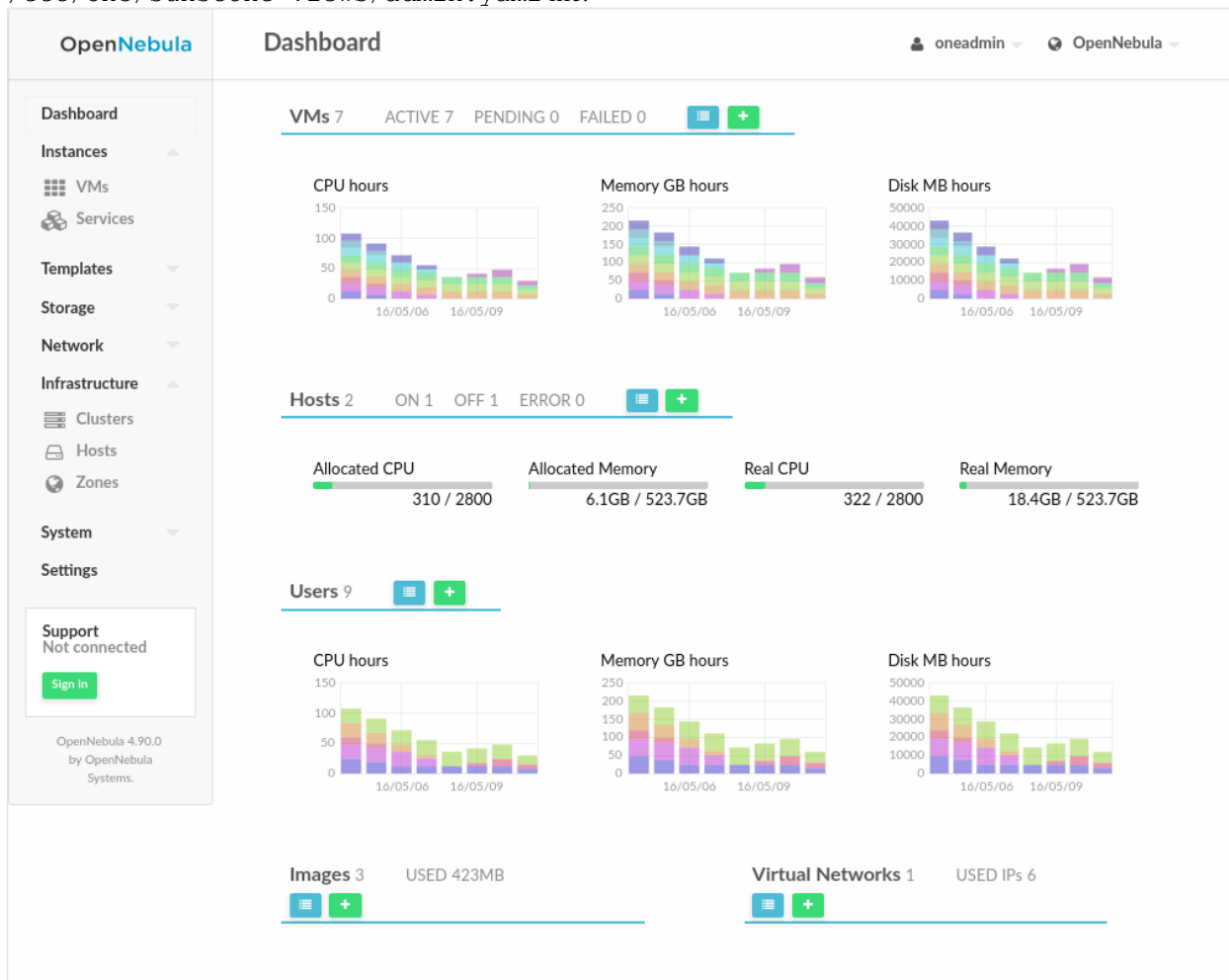
Using the OpenNebula Sunstone Views you will be able to provide a simplified UI aimed at end-users of an OpenNebula cloud. The OpenNebula Sunstone Views are fully customizable, so you can easily enable or disable specific information tabs or action buttons. You can define multiple views for different user groups. Each view defines a set of UI components so each user just accesses and views the relevant parts of the cloud for her role.

The OpenNebula Sunstone Views can be grouped into two different layouts. On one hand, the classic Sunstone layout exposes a complete view of the cloud, allowing administrators and advanced users to have full control of any physical or virtual resource of the cloud. On the other hand, the cloud layout exposes a simplified version of the cloud where end-users will be able to manage any virtual resource of the cloud, without taking care of the physical resources management.

5.3.1 Default Views

Admin View

This view provides full control of the cloud. Details can be configured in the `/etc/one/sunstone-views/admin.yaml` file.



Admin vCenter View

Based on the Admin View. It is designed to present the valid operations against a vCenter infrastructure to a cloud administrator. Details can be configured in the `/etc/one/sunstone-views/admin_vcenter.yaml` file.

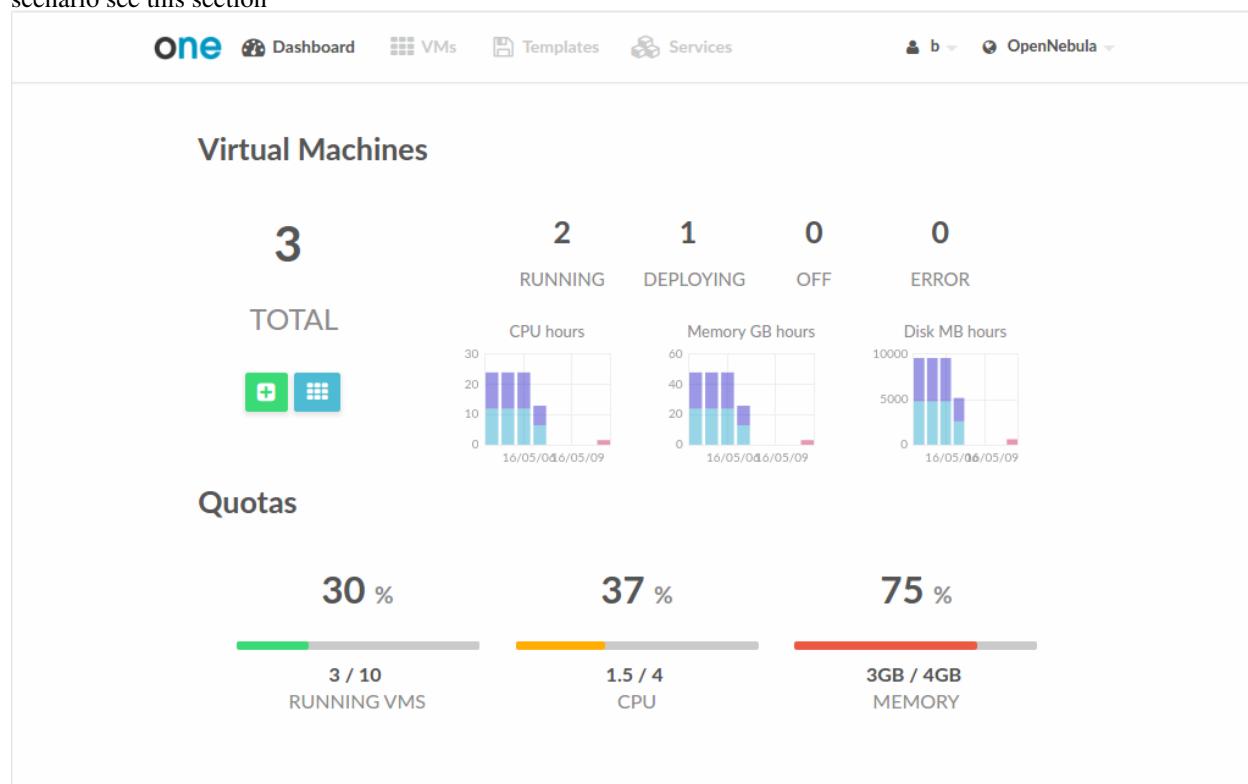
Group Admin View

Based on the Admin View. It provides control of all the resources belonging to a group, but with no access to resources outside that group, that is, restricted to the physical and virtual resources of the group. This view features the ability to create new users within the group as well as set and keep track of user quotas. For more information on how to configure this scenario see this section

Cloud View

This is a simplified view mainly intended for end-users that just require a portal where they can provision new virtual machines easily from pre-defined Templates. For more information about this view, please check the `/etc/one/sunstone-views/cloud.yaml` file.

In this scenario the cloud administrator must prepare a set of templates and images and make them available to the cloud users. These Templates must be ready to be instantiated. Before using them, users can optionally customize the VM capacity, add new network interfaces and provide values required by the template. Thereby, the user doesn't have to know any details of the infrastructure such as networking or storage. For more information on how to configure this scenario see this section

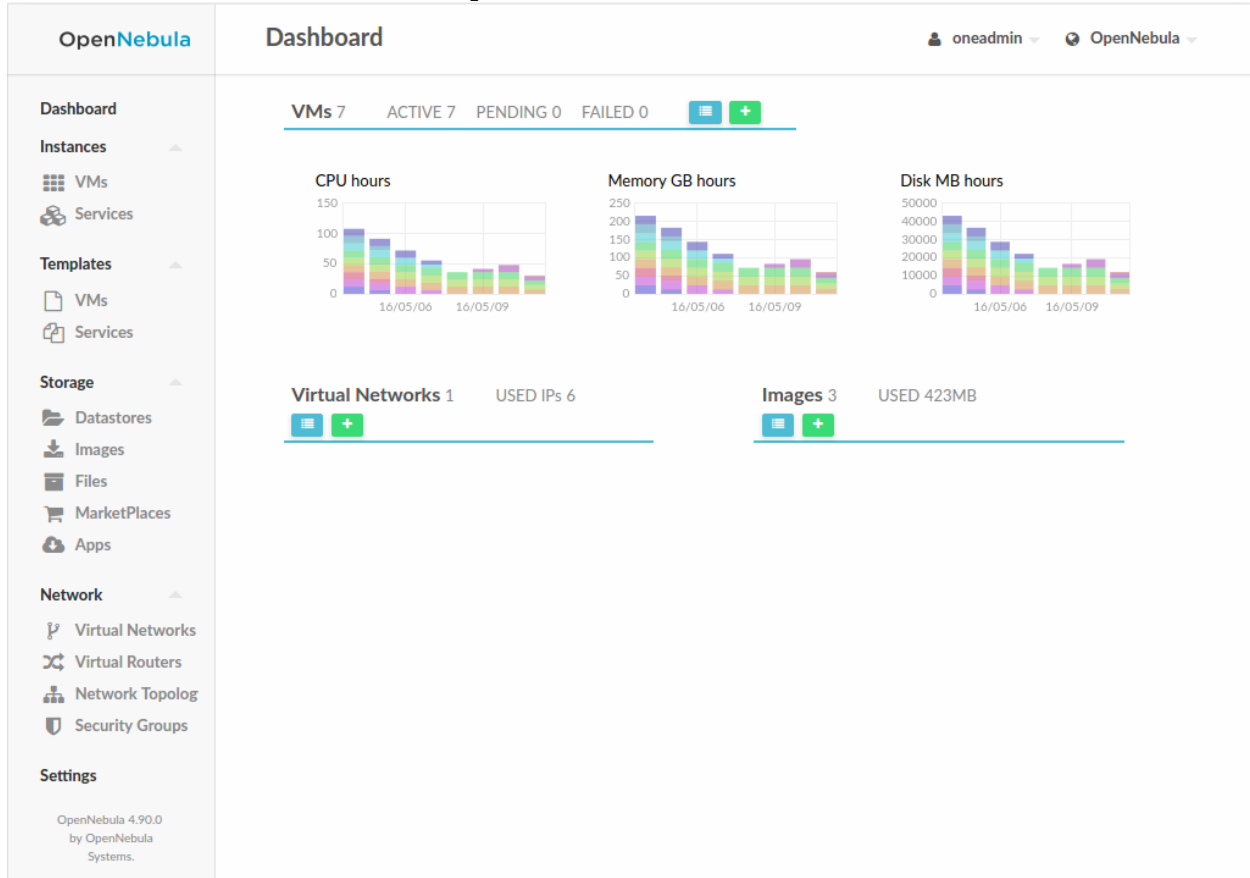


vCenter Cloud View

Based on the Cloud View, this view is designed to present the valid operations against a vCenter infrastructure to a cloud end-user.

User View

Based on the Admin View, it is an advanced user view. It is intended for users that need access to more actions than the limited set available in the cloud view. Users will not be able to manage nor retrieve the hosts and clusters of the cloud. They will be able to see Datastores and Virtual Networks in order to use them when creating a new Image or Virtual Machine, but they will not be able to create new ones. Details can be configured in the `/etc/one/sunstone-views/user.yaml` file.



5.3.2 Configuring Access to the Views

By default, the admin view is only available to the `oneadmin` group. New users will be included in the `users` group and will use the default `cloud` view. The views assigned to a given group can be defined in the group creation form or updating an existing group to implement different OpenNebula models. For more information on the different OpenNebula models please check the [Understanding OpenNebula documentation](#).

5.3.3 Usage

Sunstone users can change their current view from the top-right drop-down menu:

They can also configure several options from the settings tab:

- Views: change between the different available views

- Language: select the language that they want to use for the UI.
- Use secure websockets for VNC: Try to connect using secure websockets when starting VNC sessions.
- Display Name: If the user wishes to customize the username that is shown in Sunstone it is possible to so by adding a special parameter named `SUNSTONE_DISPLAY_NAME` with the desired value. It is worth noting that Cloud Administrators may want to automate this with a hook on user create in order to fetch the user name from outside OpenNebula.

These options are saved in the user template, as well as other hidden settings like for instance the attribute that lets Sunstone remember the number of items displayed in the datatables per user. If not defined, defaults from `/etc/one/sunstone-server.conf` are taken.

The screenshot shows the OpenNebula Sunstone interface. On the left is a navigation sidebar with options like Dashboard, Instances, Templates, Storage, Network, Infrastructure, System, and Settings. The main content area is titled 'Settings' and shows user configuration for 'oneadmin'. It includes sections for 'Information' (ID, Name, Group, Authentication driver, Password, Table Order, Language, View) and 'Attributes' (TOKEN_PASSWORD). There are also buttons for 'Update password' and 'Add'.

5.3.4 Defining a New OpenNebula Sunstone View or Customizing an Existing one

View definitions are placed in the `/etc/one/sunstone-views` directory. Each view is defined by a configuration file, in the form:

```
<view_name>.yaml
```

The name of the view will be the filename without the yaml extension.

```
/etc/one/
...
|-- sunstone-views/
|   |-- admin.yaml      <--- the admin view
|   `-- cloud.yaml     <--- the cloud view
`-- sunstone-views.yaml
...
```

Note: The easiest way to create a custom view is to copy the `admin.yaml` or `cloud.yaml` file and then harden it

as needed.

Admin View Customization

The contents of a view file specifies the enabled features, visible tabs, and enabled actions

For the dashboard, the following widgets can be configured:

```
# The following widgets can be used inside any of the '_per_row' settings
# bellow. As the name suggest, the widgets will be scaled to fit one,
# two, or three per row. The footer uses the widgets at full size, i.e.
# one per row.
#
# - storage
# - users
# - network
# - hosts
# - vms
# - groupquotas
# - quotas
panel_tabs:
actions:
    Dashboard.refresh: false
    Sunstone.toggle_top: false
widgets_one_per_row:
    - vms
    - hosts
    - users
widgets_three_per_row:
widgets_two_per_row:
    - storage
    - network
widgets_one_footer:
```

Inside features there are two settings:

- `showback`: When this is false, all Showback features are hidden. The monthly report tables, and the cost for new VMs in the create VM wizard.
- `secgroups`: If true, the create VM wizard will allow to add security groups to each network interface.

```
features:
    # True to show showback monthly reports, and VM cost
    showback: true

    # Allows to change the security groups for each network interface
    # on the VM creation dialog
    secgroups: true
```

This file also defines the tabs available in the view (note: tab is one of the main sections of the UI, those in the left-side menu). Each tab can be enabled or disabled by updating the `enabled_tabs`: attribute. For example to disable the Clusters tab, comment the `clusters-tab` entry:

```
enabled_tabs:
    - dashboard-tab
    - instances-top-tab
    - vms-tab
```

```

- oneflow-services-tab
- templates-top-tab
- templates-tab
- oneflow-templates-tab
- storage-top-tab
- datastores-tab
- images-tab
- files-tab
- marketplaces-tab
- marketplaceapps-tab
- network-top-tab
- vnets-tab
- vrouters-tab
- vnets-topology-tab
- secgroups-tab
- infrastructure-top-tab
#- clusters-tab
- hosts-tab
- zones-tab
- system-top-tab
- users-tab
- groups-tab
- vdc-tab
- acls-tab
- settings-tab
- support-tab

```

Each tab can be tuned by selecting:

- The individual resource tabs available (`panel_tabs`: attribute) in the tab, these are the tabs activated when an object is selected (e.g. the information, or capacity tabs in the Virtual Machines tab).
- The columns shown in the main information table (`table_columns`: attribute).
- The action buttons available to the view (`actions`: attribute).

The attributes in each of the above sections should be self-explanatory. As an example, the following section defines a simplified datastore tab, without the info `panel_tab` and no action buttons:

```

datastores-tab:
  panel_tabs:
    datastore_info_tab: false
    datastore_image_tab: true
    datastore_clusters_tab: false
  table_columns:
    - 0      # Checkbox
    - 1      # ID
    - 2      # Owner
    - 3      # Group
    - 4      # Name
    - 5      # Capacity
    - 6      # Cluster
    #- 7      # Basepath
    #- 8      # TM
    #- 9      # DS
    - 10     # Type
    - 11     # Status
    #- 12     # Labels
  actions:

```

```
Datastore.refresh: true
Datastore.create_dialog: false
Datastore.import_dialog: false
Datastore.addtocluster: false
Datastore.rename: false
Datastore.chown: false
Datastore.chgrp: false
Datastore.chmod: false
Datastore.delete: false
Datastore.enable: false
Datastore.disable: false
```

Cloud View Customization

The cloud layout can also be customized by changing the corresponding `/etc/one/sunstone-views/` yml files. In this file you can customize the options available when instantiating a new template, the dashboard setup or the resources available for cloud users.

Features

- `showback`: When this is false, all Showback features are hidden. The monthly report tables, and the cost for new VMs in the create VM wizard.
- `secgroups`: If true, the create VM wizard will allow to add security groups to each network interface.

```
features:
  # True to show showback monthly reports, and VM cost
  showback: true

  # Allows to change the security groups for each network interface
  # on the VM creation dialog
  secgroups: true
```

Resources

The list of VMs is always visible. The list of VM Templates and OneFlow Services can be hidden with the `provision_tabs` setting.

```
tabs:
  provision-tab:
    provision_tabs:
      flows: true
      templates: true
```

Dashboard

The dashboard can be configured to show user's quotas, group quotas, overview of user VMs, overview of group VMs

```
tabs:
  dashboard:
    # Connected user's quotas
```

```

quotas: true
# Overview of connected user's VMs
vms: true
# Group's quotas
vdcquotas: false
# Overview of group's VMs
vdcvms: false

```

Create VM Wizard

The create VM wizard can be configured with the following options:

```

tabs:
  create_vm:
    # True to allow capacity (CPU, MEMORY, VCPU) customization
    capacity_select: true
    # True to allow NIC customization
    network_select: true
    # True to allow DISK size customization
    disk_resize: true

```

Actions

The actions available for a given VM can be customized and extended by modifying the yaml file. You can even insert VM panels from the admin view into this view, for example to use the disk snapshots or scheduled actions.

- Hiding the delete button

```

tabs:
  provision-tab:
    ...
    actions: &provisionactions
    ...
    VM.shutdown_hard: false
    VM.delete: false

```

- Using undeploy instead of power off

```

tabs:
  provision-tab:
    ...
    actions: &provisionactions
    ...
    VM.poweroff: false
    VM.poweroff_hard: false
    VM.undeploy: true
    VM.undeploy_hard: true

```

- Adding panels from the admin view, for example the disk snapshots tab

```

tabs:
  provision-tab:
    panel_tabs:
    ...

```

```

    vm_snapshot_tab: true
    ...
    ...
    actions: &provisionactions
    ...
    VM.disk_snapshot_create: true
    VM.disk_snapshot_revert: true
    VM.disk_snapshot_delete: true

```

oneadmin OpenNebula

VMs Templates Services

ttylinux virtio-35

RUNNING

x0.1 - 128MB - ttylinux-vd

oneadmin 21h ago - ID: 35

Storage Actions

ID	Target	Image / Format-Size	Size	Persistent	Actions
0	vda	ttylinux-vd	27MB/24MB	YES	Revert Delete
0		16:02:15 02/10/2015 -/24MB	Updated packages		
1		16:02:37 02/10/2015 -/24MB	Release Candidate		

5.4 User Security and Authentication

By default Sunstone works with the core authentication method (user and password) although you can configure any authentication mechanism supported by OpenNebula. In this section you will learn how to enable other authentication methods and how to secure the Sunstone connections through SSL.

5.4.1 Authentication Methods

Authentication is two-folded:

- **Web client and Sunstone server.** Authentication is based on the credentials stored in the OpenNebula database for the user. Depending on the type of this credentials the authentication method can be: sunstone, x509 and opennebula (supporting LDAP or other custom methods).
- **Sunstone server and OpenNebula core.** The requests of a user are forwarded to the core daemon, including the original user name. Each request is signed with the credentials of an special `server` user. This authentication

mechanism is based either in symmetric key cryptography (default) or x509 certificates. Details on how to configure these methods can be found in the [Cloud Authentication section](#).

The following sections details the client-to-Sunstone server authentication methods.

Basic Auth

In the basic mode, username and password are matched to those in OpenNebula's database in order to authorize the user at the time of login. Rack cookie-based sessions are then used to authenticate and authorize the requests.

To enable this login method, set the `:auth:` option of `/etc/one/sunstone-server.conf` to `sunstone`:

```
:auth: sunstone
```

OpenNebula Auth

Using this method the credentials included in the header will be sent to the OpenNebula core and the authentication will be delegated to the OpenNebula auth system, using the specified driver for that user. Therefore any OpenNebula auth driver can be used through this method to authenticate the user (i.e: LDAP). The sunstone configuration is:

```
:auth: opennebula
```

x509 Auth

This method performs the login to OpenNebula based on a x509 certificate DN (Distinguished Name). The DN is extracted from the certificate and matched to the password value in the user database.

The user password has to be changed running one of the following commands:

```
$ oneuser chauth new_user x509 "/C=ES/O=ONE/OU=DEV/CN=clouduser"
```

or the same command using a certificate file:

```
$ oneuser chauth new_user --x509 --cert /tmp/my_cert.pem
```

New users with this authentication method should be created as follows:

```
$ oneuser create new_user "/C=ES/O=ONE/OU=DEV/CN=clouduser" --driver x509
```

or using a certificate file:

```
$ oneuser create new_user --x509 --cert /tmp/my_cert.pem
```

To enable this login method, set the `:auth:` option of `/etc/one/sunstone-server.conf` to `x509`:

```
:auth: x509
```

The login screen will not display the username and password fields anymore, as all information is fetched from the user certificate:



Note that OpenNebula will not verify that the user is holding a valid certificate at the time of login: this is expected to be done by the external container of the Sunstone server (normally Apache), whose job is to tell the user's browser that the site requires a user certificate and to check that the certificate is consistently signed by the chosen Certificate Authority (CA).

Warning: Sunstone x509 auth method only handles the authentication of the user at the time of login. Authentication of the user certificate is a complementary setup, which can rely on Apache.

Remote Auth

This method is similar to x509 auth. It performs the login to OpenNebula based on a Kerberos `REMOTE_USER`. The `USER@DOMAIN` is extracted from `REMOTE_USER` variable and matched to the password value in the user database. To use Kerberos authentication users needs to be configured with the public driver. Note that this will prevent users to authenticate through the XML-RPC interface, only Sunstone access will be granted to these users.

To update existing users to use the Kerberos authentication change the driver to public and update the password as follows:

```
$ oneuser chauth new_user public "new_user@DOMAIN"
```

New users with this authentication method should be created as follows:

```
$ oneuser create new_user "new_user@DOMAIN" --driver public
```

To enable this login method, set the `:auth:` option of `/etc/one/sunstone-server.conf` to `remote`:


```
:auth: remote
```

The login screen will not display the username and password fields anymore, as all information is fetched from Kerberos server or a remote authentication service.

Note that OpenNebula will not verify that the user is holding a valid Kerberos ticket at the time of login: this is expected to be done by the external container of the Sunstone server (normally Apache), whose job is to tell the user's browser that the site requires a valid ticket to login.

Warning: Sunstone remote auth method only handles the authentication of the user at the time of login. Authentication of the remote ticket is a complementary setup, which can rely on Apache.

5.4.2 Configuring a SSL Proxy

OpenNebula Sunstone runs natively just on normal HTTP connections. If the extra security provided by SSL is needed, a proxy can be set up to handle the SSL connection that forwards the petition to the Sunstone server and takes back the answer to the client.

This set up needs:

- A server certificate for the SSL connections
- An HTTP proxy that understands SSL
- OpenNebula Sunstone configuration to accept petitions from the proxy

If you want to try out the SSL setup easily, you can find in the following lines an example to set a self-signed certificate to be used by a web server configured to act as an HTTP proxy to a correctly configured OpenNebula Sunstone.

Let's assume the server where the proxy is going to be started is called `cloudserver.org`. Therefore, the steps are:

Step 1: Server Certificate (Snakeoil)

We are going to generate a snakeoil certificate. If using an Ubuntu system follow the next steps (otherwise your mileage may vary, but not a lot):

- Install the `ssl-cert` package

```
# apt-get install ssl-cert
```

- Generate the certificate

```
# /usr/sbin/make-ssl-cert generate-default-snakeoil
```

- As we are using `lighttpd`, we need to append the private key with the certificate to obtain a server certificate valid to `lighttpd`

```
# cat /etc/ssl/private/ssl-cert-snakeoil.key /etc/ssl/certs/ssl-cert-snakeoil.pem > /
↳ etc/lighttpd/server.pem
```

Step 2: SSL HTTP Proxy

lighttpd

You will need to edit the `/etc/lighttpd/lighttpd.conf` configuration file and

- Add the following modules (if not present already)
 - `mod_access`
 - `mod_alias`
 - `mod_proxy`
 - `mod_accesslog`
 - `mod_compress`
- Change the server port to 443 if you are going to run `lighttpd` as root, or any number above 1024 otherwise:

```
server.port                = 8443
```

- Add the proxy module section:

```
#### proxy module
## read proxy.txt for more info
proxy.server              = ( "" =>
                            ( "" =>
                              (
                                "host" => "127.0.0.1",
                                "port" => 9869
                              )
                            )
                          )

#### SSL engine
ssl.engine                 = "enable"
ssl.pemfile                = "/etc/lighttpd/server.pem"
```

The host must be the server hostname of the computer running the Sunstone server, and the port the one that the Sunstone Server is running on.

nginx

You will need to configure a new virtual host in `nginx`. Depending on the operating system and the method of installation, `nginx` loads virtual host configurations from either `/etc/nginx/conf.d` or `/etc/nginx/sites-enabled`.

- A sample `cloudserver.org` virtual host is presented next:

```
#### OpenNebula Sunstone upstream
upstream sunstone {
    server 127.0.0.1:9869;
}

#### cloudserver.org HTTP virtual host
server {
    listen 80;
```

```

server_name cloudserver.org;

### Permanent redirect to HTTPS (optional)
return 301 https://$server_name:8443;
}

#### cloudserver.org HTTPS virtual host
server {
    listen 8443;
    server_name cloudserver.org;

    ### SSL Parameters
    ssl on;
    ssl_certificate /etc/ssl/certs/ssl-cert-snakeoil.pem;
    ssl_certificate_key /etc/ssl/private/ssl-cert-snakeoil.key;

    ### Proxy requests to upstream
    location / {
        proxy_pass http://sunstone;
    }
}

```

The IP address and port number used in `upstream` must be the one of the server Sunstone is running on. On typical installations the nginx master process is run as user `root` so you don't need to modify the HTTPS port.

Step 3: Sunstone Configuration

Edit `/etc/one/sunstone-server.conf` to listen at `localhost:9869`.

```

:host: 127.0.0.1
:port: 9869

```

Once the proxy server is started, OpenNebula Sunstone requests using HTTPS URIs can be directed to `https://cloudserver.org:8443`, that will then be unencrypted, passed to `localhost`, port `9869`, satisfied (hopefully), encrypted again and then passed back to the client.

5.5 Cloud Servers Authentication

Todo

Mention oneflow

OpenNebula ships with two servers: *Sunstone* and EC2. When a user interacts with one of them, the server authenticates the request and then forwards the requested operation to the OpenNebula daemon.

The forwarded requests between the servers and the core daemon include the original user name, and are signed with the credentials of an special `server` user.

In this guide this request forwarding mechanism is explained, and how it is secured with a symmetric-key algorithm or x509 certificates.

5.5.1 Server Users

The *Sunstone* and EC2 services communicate with the core using a `server` user. OpenNebula creates the **serveradmin** account at bootstrap, with the authentication driver **server_cipher** (symmetric key).

This `server` user uses a special authentication mechanism that allows the servers to perform an operation on behalf of other user.

You can strengthen the security of the requests from the servers to the core daemon changing the serveruser's driver to **server_x509**. This is specially relevant if you are running your server in a machine other than the frontend.

Please note that you can have as many users with a **server_*** driver as you need. For example, you may want to have Sunstone configured with a user with **server_x509** driver, and EC2 with **server_cipher**.

5.5.2 Symmetric Key

Enable

This mechanism is enabled by default, you will have a user named **serveradmin** with driver **server_cipher**.

To use it, you need a user with the driver **server_cipher**. Enable it in the relevant configuration file in `/etc/one`:

- *Sunstone*: `/etc/one/sunstone-server.conf`
- EC2: `/etc/one/econe.conf`

```
:core_auth: cipher
```

Configure

You must update the configuration files in `/var/lib/one/.one` if you change the `serveradmin`'s password, or create a different user with the **server_cipher** driver.

```
$ ls -l /var/lib/one/.one
ec2_auth
sunstone_auth

$ cat /var/lib/one/.one/sunstone_auth
serveradmin:1612b78a4843647a4b541346f678f9e1b43bbcf9
```

Warning: `serveradmin` password is hashed in the database. You can use the `--sha1` flag when issuing `oneuser passwd` command for this user.

Warning: When Sunstone is running in a different machine than `oned` you should use an SSL connection. This can be archived with an SSL proxy like `stunnel` or `apache/nginx` acting as proxy. After securing OpenNebula XMLRPC connection configure Sunstone to use `https` with the proxy port:

```
:one_xmlrpc: https://frontend:2634/RPC2
```

5.5.3 x509 Encryption

Enable

To enable it, change the authentication driver of the **serveradmin** user, or create a new user with the driver **server_x509**:

```
$ oneuser chauth serveradmin server_x509
$ oneuser passwd serveradmin --x509 --cert usercert.pem
```

The serveradmin account should look like:

```
$ oneuser list
  ID GROUP      NAME              AUTH
↪PASSWORD
  0  oneadmin  oneadmin         core
↪c24783ba96a35464632a624d9f829136edc0175e
  1  oneadmin  serveradmin     server_x          /C=ES/O=ONE/OU=DEV/
↪CN=server
```

You need to edit `/etc/one/auth/server_x509_auth.conf` and uncomment all the fields. The defaults should work:

```
# User to be used for x509 server authentication
:srv_user: serveradmin

# Path to the certificate used by the OpenNebula Services
# Certificates must be in PEM format
:one_cert: "/etc/one/auth/cert.pem"
:one_key:  "/etc/one/auth/pk.pem"
```

Copy the certificate and the private key to the paths set in `:one_cert:` and `:one_key:`, or simply update the paths.

Then edit the relevant configuration file in `/etc/one`:

- **Sunstone:** `/etc/one/sunstone-server.conf`
- **EC2:** `/etc/one/econe.conf`

```
:core_auth: x509
```

Configure

To trust the serveradmin certificate, `/etc/one/auth/cert.pem` if you used the default path, the CA's certificate must be added to the `ca_dir` defined in `/etc/one/auth/x509_auth.conf`. See the [x509 Authentication guide for more information](#).

```
$ openssl x509 -noout -hash -in cacert.pem
78d0bbd8

$ sudo cp cacert.pem /etc/one/auth/certificates/78d0bbd8.0
```

5.5.4 Tuning & Extending

Files

You can find the drivers in these paths:

- `/var/lib/one/remotes/auth/server_cipher/authenticate`
- `/var/lib/one/remotes/auth/server_server/authenticate`

Authentication Session String

OpenNebula users with the driver **server_cipher** or **server_x509** use a special authentication session string (the first parameter of the XML-RPC calls). A regular authentication token is in the form:

```
username:secret
```

Whereas a user with a **server_*** driver must use this token format:

```
username:target_username:secret
```

The core daemon understands a request with this authentication session token as “perform this operation on behalf of target_user”. The `secret` part of the token is signed with one of the two mechanisms explained before.

5.6 Configuring Sunstone for Large Deployments

Low to medium enterprise clouds will typically deploy Sunstone in a single machine along with the OpenNebula daemons. However this simple deployment can be improved by:

- Isolating the access from Web clients to the Sunstone server. This can be achieved by deploying the Sunstone server in a separated machine.
- Improve the scalability of the server for large user pools. Usually deploying sunstone in a separate application container in one or more hosts.

Check also the [api scalability guide](#) as these of the tips also have an impact on Sunstone performance.

5.6.1 Deploying Sunstone in a Different Machine

By default the Sunstone server is configured to run in the frontend, but you are able to install the Sunstone server in a machine different from the frontend.

- You will need to install only the sunstone server packages in the machine that will be running the server. If you are installing from source use the `-s` option for the `install.sh` script.
- Make sure `:one_xmlrpc:` variable in `sunstone-server.conf` points to the right place where OpenNebula frontend is running, You can also leave it undefined and export `ONE_XMLRPC` environment variable.
- Provide the serveradmin credentials in the following file `/var/lib/one/.one/sunstone_auth`. If you changed the serveradmin password please check the [Cloud Servers Authentication guide](#).
- If you want to upload files to OpenNebula, you will have to share the upload directory (`/var/tmp` by default) between sunstone and oned. Some server do not take into account the `TMPDIR` env var and this directory must be defined in the configuration file, for example in Passenger (`client_body_temp_path`)

```
$ cat /var/lib/one/.one/sunstone_auth
serveradmin:1612b78a4843647a4b541346f678f9e1b43bbcf9
```

Using this setup the VirtualMachine logs will not be available. If you need to retrieve this information you must deploy the server in the frontend

5.6.2 Running Sunstone Inside Another Webserver

Self contained deployment of Sunstone (using `sunstone-server` script) is ok for small to medium installations. This is no longer true when the service has lots of concurrent users and the number of objects in the system is high (for example, more than 2000 simultaneous virtual machines).

Sunstone server was modified to be able to run as a `rack` server. This makes it suitable to run in any web server that supports this protocol. In ruby world this is the standard supported by most web servers. We now can select web servers that support spawning multiple processes like `unicorn` or embedding the service inside `apache` or `nginx` web servers using the `Passenger` module. Another benefit will be the ability to run Sunstone in several servers and balance the load between them.

Warning: Deploying Sunstone behind a proxy in a federated environment requires some specific configuration to properly handle the Sunstone headers required by the Federation.

- **nginx:** enable `underscores_in_headers on;` and `proxy_pass_request_headers on;`

Configuring memcached

When using one on these web servers the use of a `memcached` server is necessary. Sunstone needs to store user sessions so it does not ask for user/password for every action. By default Sunstone is configured to use memory sessions, that is, the sessions are stored in the process memory. Thin and webrick web servers do not spawn new processes but new threads all of them have access to that session pool. When using more than one process to server Sunstone there must be a service that stores this information and can be accessed by all the processes. In this case we will need to install `memcached`. It comes with most distributions and its default configuration should be ok. We will also need to install ruby libraries to be able to access it. The rubygem library needed is `memcache-client`. If there is no package for your distribution with this ruby library you can install it using rubygems:

```
$ sudo gem install memcache-client
```

Then you will have to change in sunstone configuration (`/etc/one/sunstone-server.conf`) the value of `:sessions` to `memcache`.

If you want to use `novnc` you need to have it running. You can start this service with the command:

```
$ novnc-server start
```

Another thing you have to take into account is the user on which the server will run. The installation sets the permissions for `oneadmin` user and group and files like the Sunstone configuration and credentials can not be read by other users. Apache usually runs as `www-data` user and group so to let the server run as this user the group of these files must be changed, for example:

```
$ chgrp www-data /etc/one/sunstone-server.conf
$ chgrp www-data /etc/one/sunstone-plugins.yaml
$ chgrp www-data /var/lib/one/.one/sunstone_auth
$ chmod a+x /var/lib/one
$ chmod a+x /var/lib/one/.one
```

```
$ chgrp www-data /var/log/one/sunstone*
$ chmod g+w /var/log/one/sunstone*
```

We advise to use Passenger in your installation but we will show you how to run Sunstone inside unicorn web server as an example.

For more information on web servers that support rack and more information about it you can check the [rack documentation](#) page. You can alternatively check a [list of ruby web servers](#).

Running Sunstone with Unicorn

To get more information about this web server you can go to its [web page](#). It is a multi process web server that spawns new processes to deal with requests.

The installation is done using rubygems (or with your package manager if it is available):

```
$ sudo gem install unicorn
```

In the directory where Sunstone files reside (`/usr/lib/one/sunstone` or `/usr/share/opennebula/sunstone`) there is a file called `config.ru`. This file is specific for rack applications and tells how to run the application. To start a new server using `unicorn` you can run this command from that directory:

```
$ unicorn -p 9869
```

Default unicorn configuration should be ok for most installations but a configuration file can be created to tune it. For example, to tell unicorn to spawn 4 processes and write `stderr` to `/tmp/unicorn.log` we can create a file called `unicorn.conf` that contains:

```
worker_processes 4
logger debug
stderr_path '/tmp/unicorn.log'
```

and start the server and daemonize it using:

```
$ unicorn -d -p 9869 -c unicorn.conf
```

You can find more information about the configuration options in the [unicorn documentation](#).

Running Sunstone with Passenger in Apache

[Phusion Passenger](#) is a module for [Apache](#) and [Nginx](#) web servers that runs ruby rack applications. This can be used to run Sunstone server and will manage all its life cycle. If you are already using one of these servers or just feel comfortable with one of them we encourage you to use this method. This kind of deployment adds better concurrency and lets us add an https endpoint.

We will provide the instructions for Apache web server but the steps will be similar for nginx following [Passenger documentation](#).

First thing you have to do is install Phusion Passenger. For this you can use pre-made packages for your distribution or follow the [installation instructions](#) from their web page. The installation is self explanatory and will guide you in all the process, follow them and you will be ready to run Sunstone.

Next thing we have to do is configure the virtual host that will run our Sunstone server. We have to point to the `public` directory from the Sunstone installation, here is an example:


```

<VirtualHost *:80>
  ServerName sunstone-server
  PassengerUser oneadmin
  # !!! Be sure to point DocumentRoot to 'public'!
  DocumentRoot /usr/lib/one/sunstone/public
  <Directory /usr/lib/one/sunstone/public>
    # This relaxes Apache security settings.
    AllowOverride all
    # MultiViews must be turned off.
    Options -MultiViews
    # Uncomment this if you're on Apache >= 2.4:
    #Require all granted
  </Directory>
</VirtualHost>

```

Note: When you're experiencing login problems you might want to set `PassengerMaxInstancesPerApp 1` in your passenger configuration or try memcached since Sunstone does not support sessions across multiple server instances.

Now the configuration should be ready, restart -or reload apache configuration- to start the application and point to the virtual host to check if everything is running.

Running Sunstone behind nginx SSL Proxy

How to set things up with nginx ssl proxy for sunstone and encrypted vnc.

```

# No squealing.
server_tokens off;

# OpenNebula Sunstone upstream
upstream sunstone {
  server 127.0.0.1:9869;
}

# HTTP virtual host, redirect to HTTPS
server {
  listen 80 default_server;
  return 301 https://$server_name:443;
}

# HTTPS virtual host, proxy to Sunstone
server {
  listen 443 ssl default_server;
  ssl_certificate /etc/ssl/certs/opennebula-certchain.pem;
  ssl_certificate_key /etc/ssl/private/opennebula-key.pem;
  ssl_stapling on;
}

```

And this is the changes that have to be made to `sunstone-server.conf`:

```

UI Settings

:vnc_proxy_port: 29876
:vnc_proxy_support_wss: only
:vnc_proxy_cert: /etc/one/ssl/opennebula-certchain.pem

```

```
:vnc_proxy_key: /etc/one/ssl/opennebula-key.pem
:vnc_proxy_ipv6: false
```

If using a selfsigned cert, the connection to VNC window in Sunstone will fail, either get a real cert, or manually accept the selfsigned cert in your browser before trying it with Sunstone. Now, VNC sessions should show “encrypted” in the title.

Running Sunstone with Passenger using FreeIPA/Kerberos auth in Apache

It is also possible to use Sunstone `remote` authentication with Apache and Passenger. The configuration in this case is quite similar to Passenger configuration but we must include the Apache auth module line. How to configure freeIPA server and Kerberos is outside of the scope of this document, you can get more info in [FreeIPA Apache setup example](#)

As example to include Kerberos authentication we can use two different modules: `mod_auth_gssapi` or `mod_authnz_pam` And generate the keytab for http service, here is an example with Passenger:

```
LoadModule auth_gssapi_module modules/mod_auth_gssapi.so

<VirtualHost *:80>
  ServerName sunstone-server
  PassengerUser oneadmin
  # !!! Be sure to point DocumentRoot to 'public'!
  DocumentRoot /usr/lib/one/sunstone/public
  <Directory /usr/lib/one/sunstone/public>
    # Only is possible to access to this dir using a valid ticket
    AuthType GSSAPI
    AuthName "EXAMPLE.COM login"
    GssapiCredStore keytab:/etc/http.keytab
    Require valid-user
    ErrorDocument 401 '<html><meta http-equiv="refresh" content="0; URL=https://
↪yourdomain"><body>Kerberos authentication did not pass.</body></html>'
    AllowOverride all
    # MultiViews must be turned off.
    Options -MultiViews
  </Directory>
</VirtualHost>
```

Note: User must generate a valid ticket running `kinit` to get acces to Sunstone service. You can also set a custom 401 document to warn users about any authentication failure.

Now our configuration is ready to use Passenger and Kerberos, restart -or reload apache configuration- and point to the virtual host using a valid ticket to check if everything is running.

Running Sunstone in Multiple Servers

You can run Sunstone in several servers and use a load balancer that connects to them. Make sure you are using `memcache` for sessions and both Sunstone servers connect to the same `memcached` server. To do this change the parameter `:memcache_host` in the configuration file. Also make sure that both Sunstone instances connect to the same OpenNebula server.

MarketPlace

If you plan on using the MarketPlaceApp download functionality the Sunstone server(s) will need access to the MarketPlace backends.

If you are using Phusion Passenger, take the following recommendations into account:

- Set `PassengerResponseBufferHighWatermark` to `0`.
- Increase `PassengerMaxPoolSize`. Each MarketPlaceApp download will take one of this application processes.
- If Passenger Enterprise is available, set `PassengerConcurrencyModel` to `thread`.

If you are using another backend than Passenger, please port these recommendations to your backend.

VMWARE INFRASTRUCTURE SETUP

6.1 Overview

After configuring the OpenNebula front-end and the vCenter nodes, the next step is to learn what capabilities can be leveraged from the vCenter infrastructure and fine tune the OpenNebula cloud to make use of them.

The Virtualization Subsystem is the component in charge of talking with the hypervisor and taking the actions needed for each step in the VM life-cycle. This Chapter gives a detailed view of the vCenter drivers, the resources it manages and how to setup OpenNebula to leverage different vCenter features.

6.1.1 How Should I Read This Chapter

You should be reading this chapter after performing the *vCenter node install*.

This Chapter is organized in the *vCenter Node Section*, which introduces the vCenter integration approach under the point of view of OpenNebula, with description of how to import, create and use VM Templates, resource pools, limitations and so on; the *Networking Setup Section* section that glosses over the network consumption and usage and then the *Datastore Setup Section* which introduces the concepts of OpenNebula datastores as related to vCenter datastores, and the VMDK image management made by OpenNebula.

After reading this Chapter, you can delve on advanced topics like OpenNebula upgrade, logging, scalability in the *Reference Chapter*. The next step should be proceeding to the Operations guide to learn how the Cloud users can consume the cloud resources that have been set up.

6.1.2 Hypervisor Compatibility

All this Chapter applies exclusively to vCenter hypervisor.

6.2 vCenter Node

The vCenter driver for OpenNebula enables the interaction with vCenter to control the life-cycle of vCenter resources such as Virtual Machines, Templates, Networks and VMDKs.

6.2.1 OpenNebula approach to vCenter interaction

OpenNebula consumes resources from vCenter, and with exceptions (VMs and VMDKs) does not create these resources, but rather offers mechanisms to import them into OpenNebula to be controlled.

Virtual Machines are deployed from VMware VM Templates that **must exist previously in vCenter**. There is a one-to-one relationship between each VMware VM Template and the equivalent OpenNebula Template. Users will then instantiate the OpenNebula Templates where you can easily build from any provisioning strategy (e.g. access control, quota...).

Networking is handled by creating Virtual Network representations of the vCenter networks. OpenNebula additionally can handle on top of these networks three types of Address Ranges: Ethernet, IPv4 and IPv6. This networking information can be passed to the VMs through the contextualization process.

Therefore there is no need to convert your current Virtual Machines or import/export them through any process; once ready just save them as VM Templates in vCenter, following [this procedure](#).

There are different behavior of the vCenter resources when deleted in OpenNebula. The following resources are NOT deleted in vCenter when deleted in OpenNebula:

- VM Templates
- Networks
- Datastores

The following resource are deleted in vCenter when deleted in OpenNebula:

- Images
- Virtual Machines

Note: After a VM Template is cloned and booted into a vCenter Cluster it can access VMware advanced features and it can be managed through the OpenNebula provisioning portal -to control the life-cycle, add/remove NICs, make snapshots- or through vCenter (e.g. to move the VM to another datastore or migrate it to another ESX). OpenNebula will poll vCenter to detect these changes and update its internal representation accordingly.

6.2.2 Considerations & Limitations

- **Unsupported Operations:** The following operations are **NOT** supported on vCenter VMs managed by OpenNebula, although they can be performed through vCenter:

Operation	Note
migrate	VMs cannot be migrated between ESX clusters
disk snapshots	Only system snapshots are available for vCenter VMs

- **No Security Groups:** Firewall rules as defined in Security Groups cannot be enforced in vCenter VMs.
- OpenNebula treats **snapshots** a tad different than VMware. OpenNebula assumes that they are independent, whereas VMware builds them incrementally. This means that OpenNebula will still present snapshots that are no longer valid if one of their parent snapshots are deleted, and thus revert operations applied upon them will fail.
- **No files in context:** Passing entire files to VMs is not supported, but all the other CONTEXT sections will be honored.
- Cluster names cannot contain spaces.
- Image names cannot contain spaces.
- Datastore names cannot contain spaces.
- vCenter credential password cannot have more than 22 characters.

- If you are running Sunstone using nginx/apache you will have to forward the following headers to be able to interact with vCenter, `HTTP_X_VCENTER_USER`, `HTTP_X_VCENTER_PASSWORD` and `HTTP_X_VCENTER_HOST` (or, alternatively, `X_VCENTER_USER`, `X_VCENTER_PASSWORD` and `X_VCENTER_HOST`). For example in nginx you have to add the following attrs to the server section of your nginx file: (underscores_in_headers on; proxy_pass_request_headers on;).
- Attaching a new CDROM ISO will add a new (or change the existing) ISO to an already existing CDROM drive that needs to be present in the VM.

6.2.3 Configuring

The vCenter virtualization driver configuration file is located in `/etc/one/vmm_exec/vmm_exec_vcenter.conf`. This file is home for default values for OpenNebula VM templates.

It is generally a good idea to place defaults for the vCenter-specific attributes, that is, attributes mandatory in the vCenter driver that are not mandatory for other hypervisors. Non mandatory attributes for vCenter but specific to them are also recommended to have a default.

6.2.4 Importing vCenter VM Templates and running VMs

The `onevcenter` tool can be used to import existing VM templates from the ESX clusters:

```
$ onevcenter templates --vcenter <vcenter-host> --vuser <vcenter-username> --vpass
↪<vcenter-password>

Connecting to vCenter: <vcenter-host>...done!

Looking for VM Templates...done!

Do you want to process datacenter Development [y/n]? y

* VM Template found:
  - Name      : ttyTemplate
  - UUID     : 421649f3-92d4-49b0-8b3e-358abd18b7dc
  - Cluster  : clusterA
  Import this VM template [y/n]? y
  OpenNebula template 4 created!

* VM Template found:
  - Name      : Template test
  - UUID     : 4216d5af-7c51-914c-33af-1747667c1019
  - Cluster  : clusterB
  Import this VM template [y/n]? y
  OpenNebula template 5 created!

$ onetemplate list
ID USER          GROUP          NAME          REGTIME
 4 oneadmin      oneadmin      ttyTemplate   09/22 11:54:33
 5 oneadmin      oneadmin      Template test 09/22 11:54:35

$ onetemplate show 5
TEMPLATE 5 INFORMATION
ID          : 5
NAME       : Template test
USER       : oneadmin
GROUP      : oneadmin
```

```
REGISTER TIME   : 09/22 11:54:35

PERMISSIONS
OWNER          : um-
GROUP         : ---
OTHER         : ---

TEMPLATE CONTENTS
CPU="1"
MEMORY="512"
PUBLIC_CLOUD=[
  TYPE="vcenter",
  VM_TEMPLATE="4216d5af-7c51-914c-33af-1747667c1019" ]
SCHED_REQUIREMENTS="NAME=\"devel\""
VCPU="1"
```

After a vCenter VM Template is imported as a OpenNebula VM Template, it can be modified to change the capacity in terms of CPU and MEMORY, the name, permissions, etc. It can also be enriched to add:

- New disks
- New network interfaces
- Context information

Before using your OpenNebula cloud you may want to read about the vCenter specifics.

To import existing VMs, the ‘onehost importvm’ command can be used. VMs in running state can be imported, and also VMs defined in vCenter that are not in power.on state (this will import the VMs in OpenNebula as in the poweroff state).

```
$ onehost show 0
HOST 0 INFORMATION
ID                : 0
NAME              : MyvCenterHost
CLUSTER          : -
[....]

WILD VIRTUAL MACHINES

                NAME                IMPORT_ID  CPU    MEMORY
RunningVM 4223cbb1-34a3-6a58-5ec7-a55db235ac64    1      1024
[....]

$ onehost importvm 0 RunningVM
$ onevm list
ID USER   GROUP   NAME           STAT  UCPU  UMEM HOST           TIME
 3 oneadmin oneadmin RunningVM      runn   0    590M MyvCenterHost 0d 01h02
```

After a Virtual Machine is imported, their life-cycle (including creation of snapshots) can be controlled through OpenNebula. The following operations *cannot* be performed on an imported VM:

- Recover –recreate
- Undeploy (and Undeploy –hard)
- Migrate (and Migrate –live)
- Stop

Running VMs with open VNC ports are imported with the ability to establish VNC connection to them via OpenNebula. To activate the VNC ports, you need to right click on the VM in vCenter while it is shut down and click on “Edit Settings”, and set the following `remotedisplay.*` settings:

- `remotedisplay.vnc.enabled` must be set to `TRUE`.
- `remotedisplay.vnc.ip` must be set to `0.0.0.0` (or alternatively, the IP of the OpenNebula front-end).
- `remotedisplay.vnc.port` must be set to a available VNC port number.

Also, network management operations are present like the ability to attach/detach network interfaces, as well as capacity (CPU and MEMORY) resizing operations and VNC connections if the ports are opened before hand. The same import mechanism is available graphically through Sunstone for hosts, networks, templates and running VMs. vCenter hosts can be imported using the vCenter host create dialog, and Networks and VM Templates through the Import button in the Virtual Networks and Templates tab respectively. Running and Powered Off VMs can be imported through the WILDS tab in the Host info tab.

Note: running VMS can only be imported after the vCenter host has been successfully acquired.

6.2.5 Resource Pool

OpenNebula can place VMs in different Resource Pools. There are two approaches to achieve this, fixed per Cluster basis or flexible per VM Template basis.

In the fixed per Cluster basis approach, the vCenter credentials that OpenNebula use can be confined into a Resource Pool, to allow only a fraction of the vCenter infrastructure to be used by OpenNebula users. The steps to confine OpenNebula users into a Resource Pool are:

- Create a new vCenter user
- Create a Resource Pool in vCenter and assign the subset of Datacenter hardware resources wanted to be exposed through OpenNebula
- Give vCenter user Resource Pool Administration rights over the Resource Pool
- Give vCenter user Resource Pool Administration (or equivalent) over the Datastores the VMs are going to be running on

Afterwards, these credentials can be used to add to OpenNebula the host representing the vCenter cluster. Add a new tag called `VCENTER_RESOURCE_POOL` to the host template representing the vCenter cluster (for instance, in the info tab of the host, or in the CLI), with the name of the Resource Pool.

The screenshot shows the OpenNebula web interface for a host named 'devel6'. The interface is divided into a sidebar on the left and a main content area. The sidebar contains navigation links for Dashboard, System, Virtual Resources, Infrastructure, and Support. The main content area is titled 'Host devel6' and has tabs for Info, Graphs, and VMs. The 'Info' tab is active, showing a table of host information and attributes.

Information		Capacity
id	5	Allocated Memory
Name	devel6	Allocated CPU
Cluster	-	Real Memory
State	MONITORED	Real CPU
IM MAD	vcenter	
VM MAD	vcenter	
VN MAD	dummy	

Attributes		
IM_MAD	vcenter	edit delete
PUBLIC_CLOUD	YES	edit delete
RESERVED_CPU		edit delete
RESERVED_MEM		edit delete
TOTALHOST	1	edit delete
TOTAL_ZOMBIES	2	edit delete
VCENTER_HOST	10.0.1.140	edit delete
VCENTER_RESOURCE_POOL	Dev6ResourcePool	edit delete

The second approach is more flexible in the sense that all Resource Pools defined in vCenter can be used, and the mechanism to select which one the VM is going to reside into can be defined using the attribute `RESOURCE_POOL` in the OpenNebula VM Template:

Nested Resource Pools can be represented using `'/'`. For instance, a Resource Pool `"RPChild"` nested under `"RPAncesor"` can be represented both in `VCENTER_RESOURCE_POOL` and `RESOURCE_POOL` attributes as `"RPAncesor/RPChild"`.

```
RESOURCE_POOL="RPAncesor/RPChild"
PUBLIC_CLOUD=[
  HOST="Cluster",
  TYPE="vcenter",
  VM_TEMPLATE="4223067b-ed9b-8f73-82ba-b1a98c3ff96e" ]
```

6.3 vCenter Datastore

The vCenter datastore allows the representation in OpenNebula of VMDK images available in vCenter datastores. It is a persistent only datastore, meaning that VMDK images are not cloned automatically by OpenNebula when a VM is instantiated. vCenter handles the VMDK image copies, so no system datastore is needed in OpenNebula, and only vCenter image datastores are allowed.

No system datastore is needed since the vCenter support in OpenNebula does not rely on transfer managers to copy VMDK images, but rather this is delegated to vCenter. When a VM Template is instantiated, vCenter performs the

VMDK copies, and deletes them after the VM ends its life-cycle. The OpenNebula vCenter datastore is a purely persistent images datastore to allow for VMDK cloning and enable disk attach/detach on running VMs.

The vCenter datastore in OpenNebula is tied to a vCenter OpenNebula host in the sense that all operations to be performed in the datastore are going to be performed through the vCenter instance associated to the OpenNebula host, which happens to hold the needed credentials to access the vCenter instance.

Creation of empty datablocks and VMDK image cloning are supported, as well as image deletion.

6.3.1 Limitations

- No support for snapshots in the vCenter datastore.
- Only one disk is allowed per directory in the vCenter datastores.
- Datastore names cannot contain spaces.
- Image names cannot contain spaces.

6.3.2 Requirements

- In order to use the vCenter datastore, all the ESX servers controlled by vCenter need to mount the same VMFS datastore with the same name.

6.3.3 Configuration

In order to create a OpenNebula vCenter datastore that represents a vCenter VMFS datastore, a new OpenNebula datastore needs to be created with the following attributes:

- The OpenNebula vCenter datastore name needs to be exactly the same as the vCenter VMFS datastore available in the ESX hosts.
- The specific attributes for this datastore driver are listed in the following table:

Attribute	Description
DS_MAD	Must be set to <code>vcenter</code>
TM_MAD	Must be set <code>vcenter</code>
VCENTER_CLUSTER	Name of the OpenNebula host that represents the vCenter cluster that groups the ESX hosts that mount the represented VMFS datastore
ADAPTER_TYPE	Default adapter type used by virtual disks to plug inherited to VMs for the images in the datastore. It is inherited by images and can be overwritten if specified explicitly in the image. Possible values (careful with the case): <code>lsiLogic</code> , <code>ide</code> , <code>busLogic</code> . More information in the VMware documentation . Known as “Bus adapter controller” in Sunstone.
DISK_TYPE	Type of disk to be created when a DATABLOCK is requested. This value is inherited from the datastore to the image but can be explicitly overwritten. The type of disk has implications on performance and occupied space. Values (careful with the case): <code>delta</code> , <code>eagerZeroedThick</code> , <code>flatMonolithic</code> , <code>preallocated</code> , <code>raw</code> , <code>rdm</code> , <code>rdmp</code> , <code>seSparse</code> , <code>sparse2Gb</code> , <code>sparseMonolithic</code> , <code>thick</code> , <code>thick2</code> . More information in the VMware documentation . Known as “Disk Provisioning Type” in Sunstone.

vCenter datastores can be represented in OpenNebula to achieve the following VM operations:

- Choose a different datastore for VM deployment
- Clone VMDKs images
- Create empty datablocks

- Delete VMDK images

All OpenNebula datastores are actively monitoring, and the scheduler will refuse to deploy a VM onto a vCenter datastore with insufficient free space.

The **onevcenter** tool can be used to import vCenter datastores:

```
$ onevcenter datastores --vuser <VCENTER_USER> --vpass <VCENTER_PASS> --vcenter
-><VCENTER_FQDN>

Connecting to vCenter: vcenter.vcenter3...done!

Looking for Datastores...done!

Do you want to process datacenter Datacenter [y/n]? y

* Datastore found:
  - Name      : datastore2
  - Total MB  : 132352
  - Free MB   : 130605
  - Cluster   : Cluster
  Import this Datastore [y/n]? y
  OpenNebula datastore 100 created!
```

6.3.4 Tuning and Extending

Drivers can be easily customized please refer to the specific guide for each datastore driver or to the Storage subsystem developer's guide.

However you may find the files you need to modify here:

- /var/lib/one/remotes/datastore/vcenter
- /var/lib/one/remotes/tm/vcenter

6.4 vCenter Networking

Virtual Networks from vCenter can be represented using OpenNebula networks, taking into account that the BRIDGE of the Virtual Network needs to match the name of the Network defined in vCenter. OpenNebula supports both “Port Groups” and “Distributed Port Groups”, and as such can consume any vCenter defined network resource (even those created by other networking components like for instance NSX).

Virtual Networks in vCenter can be created using the vCenter web client, with any specific configuration like for instance VLANs. OpenNebula will use these networks with the defined characteristics, but it cannot create new Virtual Networks in vCenter, but rather only OpenNebula representations of such Virtual Networks. OpenNebula additionally can handle on top of these networks three types of Address Ranges: Ethernet, IPv4 and IPv6.

vCenter VM Templates can define their own NICs, and OpenNebula will not manage them. However, any NIC added in the OpenNebula VM Template, or through the attach_nic operation, will be handled by OpenNebula, and as such it is subject to be detached and its information (IP, MAC, etc) is known by OpenNebula.

vCenter VM Templates with already defined NICs that reference Networks in vCenter will be imported without this information in OpenNebula. These NICs will be invisible for OpenNebula, and therefore cannot be detached from the VMs. The imported VM Templates in OpenNebula can be updated to add NICs from Virtual Networks imported from vCenter (being Networks or Distributed vSwitches). We recommend therefore to use VM Templates in vCenter without defined NICs, to add them later on in the OpenNebula VM Templates

6.4.1 Importing vCenter Networks

The `onevcenter` tool can be used to import existing Networks and distributed vSwitches from the ESX clusters:

```
$ onevcenter networks --vcenter <vcenter-host> --vuser <vcenter-username> --vpass
↪<vcenter-password>

Connecting to vCenter: <vcenter-host>...done!

Looking for vCenter networks...done!

Do you want to process datacenter vOneDatacenter [y/n]? y

* Network found:
  - Name      : MyvCenterNetwork
  - Type      : Port Group
  Import this Network [y/n]? y
  How many VMs are you planning to fit into this network [255]? 45
  What type of Virtual Network do you want to create (IPv[4],IPv[6],[E]thernet) ? E
  Please input the first MAC in the range [Enter for default]:
  OpenNebula virtual network 29 created with size 45!

$ onevnet list
  ID USER          GROUP      NAME                CLUSTER  BRIDGE  LEASES
  29 oneadmin       oneadmin   MyvCenterNetwork   -        MyFakeNe  0

$ onevnet show 29
VIRTUAL NETWORK 29 INFORMATION
ID              : 29
NAME            : MyvCenterNetwork
USER           : oneadmin
GROUP          : oneadmin
CLUSTER        : -
BRIDGE         : MyvCenterNetwork
VLAN           : No
USED LEASES    : 0

PERMISSIONS
OWNER          : um-
GROUP         : ---
OTHER         : ---

VIRTUAL NETWORK TEMPLATE
BRIDGE="MyvCenterNetwork"
PHYDEV=""
VCENTER_TYPE="Port Group"
VLAN="NO"
VLAN_ID=""

ADDRESS RANGE POOL
  AR TYPE    SIZE LEASES          MAC                IP                GLOBAL_PREFIX
  0 ETHER    45     0 02:00:97:7f:f0:87   -                -

LEASES
AR  OWNER          MAC                IP                IP6_GLOBAL
```

The same import mechanism is available graphically through Sunstone

OpenNebula

Dashboard

Instances

Templates

Storage

 Datastores

 Images

 MarketPlaces

 Apps

Network

 Virtual Networks

 Virtual Routers

 Network Topology

Infrastructure

System

Settings

Support
Not connected
[Sign in](#)

OpenNebula 4.90.0
by OpenNebula Systems.

Import vCenter Networks

oneadmin OpenNebula

[Reset](#) [Import](#)

Hostname: User: Password: [Get Networks](#)

Datacenter DataCenter Select All Expand Advanced Sections [Import Selected Networks](#) [Clear Imported Networks](#)

^
NSXDSwitch-DVUplinks-405 - Cluster - Distributed Port Group
Size: Type: MAC:

^
NSXDSwitchDefaultPortGroup - Cluster - Distributed Port Group
Size: Type: MAC:

^
Testing DPortGroup 2 Import - Cluster - Distributed Port Group
Size: Type: MAC:

OPEN CLOUD HOST SETUP

7.1 Overview

The Hosts are servers with a hypervisor installed (KVM) which execute the running Virtual Machines. These Hosts are managed by the KVM Driver, which will perform the actions needed to manage the VM and its life-cycle. This chapter analyses the KVM driver in detail, and will give you, amongst other things, the tools to configure and add KVM hosts into the OpenNebula Cloud.

7.1.1 How Should I Read This Chapter

Before reading this chapter, you should have already installed your *Frontend*, the *KVM Hosts* and have an OpenNebula cloud up and running with at least one virtualization node.

This chapter will focus on the configuration options for the Hosts.

- Read the *KVM driver* section in order to understand the procedure of configuring and managing kvm Hosts.
- In the *Monitoring* section, you can find information about how OpenNebula is monitoring its Hosts and Virtual Machines, and changes you can make in the configuration of that subsystem.
- You can read this section if you are interested in performing *PCI Passthrough*.

After reading this chapter, you should read the *Open Cloud Storage* chapter.

7.1.2 Hypervisor Compatibility

This chapter applies only to KVM.

Follow the *vCenter Node* section for a similar guide for vCenter.

7.2 KVM Driver

KVM (Kernel-based Virtual Machine) is the hypervisor for OpenNebula's *Open Cloud Architecture*. KVM is a complete virtualization system for Linux. It offers full virtualization, where each Virtual Machine interacts with its own virtualized hardware. This guide describes the use of the KVM with OpenNebula.

7.2.1 Requirements

The Hosts will need a CPU with Intel VT or AMD's AMD-V features, in order to support virtualization. KVM's [Preparing to use KVM](#) guide will clarify any doubts you may have regarding if your hardware supports KVM.

KVM will be installed and configured after following the [KVM Host Installation](#) section.

7.2.2 Considerations & Limitations

Try to use *virtio* whenever possible, both for networks and disks. Using emulated hardware, both for networks and disks, will have an impact in performance and will not expose all the available functionality. For instance, if you don't use *virtio* for the disk drivers, you will not be able to exceed a small number of devices connected to the controller, meaning that you have a limit when attaching disks, and it will not work while the VM is running (live disk-attach).

7.2.3 Configuration

KVM Configuration

The OpenNebula packages will configure KVM automatically, therefore you don't need to take any extra steps.

Drivers

The KVM driver is enabled by default in OpenNebula:

```
#-----
# KVM Virtualization Driver Manager Configuration
# -r number of retries when monitoring a host
# -t number of threads, i.e. number of hosts monitored at the same time
# -l <actions[=command_name]> actions executed locally, command can be
# overridden for each action.
#   Valid actions: deploy, shutdown, cancel, save, restore, migrate, poll
#   An example: "-l migrate=migrate_local,save"
# -p more than one action per host in parallel, needs support from hypervisor
# -s <shell> to execute remote commands, bash by default
#
# Note: You can use type = "qemu" to use qemu emulated guests, e.g. if your
# CPU does not have virtualization extensions or use nested Qemu-KVM hosts
#-----
VM_MAD = [
    NAME           = "kvm",
    SUNSTONE_NAME  = "KVM",
    EXECUTABLE     = "one_vmm_exec",
    ARGUMENTS      = "-t 15 -r 0 kvm",
    DEFAULT        = "vmm_exec/vmm_exec_kvm.conf",
    TYPE           = "kvm",
    KEEP_SNAPSHOTS= "no",
    IMPORTED_VMS_ACTIONS = "terminate, terminate-hard, hold, release, suspend,
        resume, delete, reboot, reboot-hard, resched, unresched, disk-attach,
        disk-detach, nic-attach, nic-detach, snap-create, snap-delete"
]
```

The configuration parameters: `-r`, `-t`, `-l`, `-p` and `-s` are already preconfigured with sane defaults. If you change them you will need to restart OpenNebula.

Read the Virtual Machine Drivers Reference for more information about these parameters, and how to customize and extend the drivers.

Driver Defaults

There are some attributes required for KVM to boot a VM. You can set a suitable defaults for them so, all the VMs get needed values. These attributes are set in `/etc/one/vmm_exec/vmm_exec_kvm.conf`. The following can be set for KVM:

- **EMULATOR:** path to the kvm executable.
- **OS:** attributes `KERNEL`, `INITRD`, `BOOT`, `ROOT`, `KERNEL_CMD`, `MACHINE` and `ARCH`.
- **VCPU**
- **FEATURES:** attributes `ACPI`, `PAE`.
- **DISK:** attributes `DRIVER` and `CACHE`. All disks will use that driver and caching algorithm.
- **NIC:** attribute `FILTER`.
- **RAW:** to add libvirt attributes to the domain XML file.
- **HYPERV:** to enable hyperv extensions.
- **SPICE:** to add default devices for SPICE.

For example:

```
OS          = [ ARCH = "x86_64" ]
FEATURES = [ PAE = "no", ACPI = "yes", APIC = "no", HYPERV = "no", GUEST_AGENT = "no" ]
DISK       = [ DRIVER = "raw" , CACHE = "none" ]
HYPERV_OPTIONS="<relaxed state='on'/><vapic state='on'/><spinlocks state='on' retries=
SPICE_OPTIONS="
  <video>
    <model type='qxl' heads='1'/>
  </video>
  <sound model='ich6' />
  <channel type='spicevmc'>
    <target type='virtio' name='com.redhat.spice.0'/>
  </channel>
  <redirdev bus='usb' type='spicevmc'/>
  <redirdev bus='usb' type='spicevmc'/>
  <redirdev bus='usb' type='spicevmc'/>"
```

Live-Migration for Other Cache settings

In case you are using disks with a cache setting different to none you may have problems with live migration depending on the libvirt version. You can enable the migration adding the `--unsafe` parameter to the `virsh` command. The file to change is `/var/lib/one/remotes/vmm/kvm/kvmrc`. Uncomment the following line, and execute `onehost sync --force` afterwards:

```
MIGRATE_OPTIONS=--unsafe
```


Configure the Timeouts (Optional)

Optionally, you can set a timeout for the VM Shutdown operation can be set up. This feature is useful when a VM gets stuck in Shutdown (or simply does not notice the shutdown command). By default, after the timeout time the VM will return to Running state but is can also be configured so the VM is destroyed after the grace time. This is configured in `/var/lib/one/remotes/vmm/kvm/kvmrc`:

```
# Seconds to wait after shutdown until timeout
export SHUTDOWN_TIMEOUT=300

# Uncomment this line to force VM cancellation after shutdown timeout
#export FORCE_DESTROY=yes
```

Working with cgroups (Optional)

Warning: This section outlines the configuration and use of cgroups with OpenNebula and libvirt/KVM. Please refer to the cgroups documentation of your Linux distribution for specific details.

Cgroups is a kernel feature that allows you to control the amount of resources allocated to a given process (among other things). This feature can be used to enforce the amount of CPU assigned to a VM, as defined in its template. So, thanks to cgroups a VM with CPU=0.5 will get half of the physical CPU cycles than a VM with CPU=1.0.

Cgroups can be also used to limit the overall amount of physical RAM that the VMs can use, so you can leave always a fraction to the host OS.

The following outlines the steps need to configure cgroups, this should be **performed in the hosts, not in the front-end**:

- Define where to mount the cgroup controller virtual file systems, at least memory and cpu are needed.
- (Optional) You may want to limit the total memory devoted to VMs. Create a group for the libvirt processes (VMs) and the total memory you want to assign to them. Be sure to assign libvirt processes to this group, e.g. with `CGROUP_DAEMON` or in `cgrules.conf`. Example:

```
# /etc/cgconfig.conf

group virt {
    memory {
        memory.limit_in_bytes = 5120M;
    }
}

mount {
    cpu      = /mnt/cgroups/cpu;
    memory  = /mnt/cgroups/memory;
    cpuset  = /mnt/cgroups/cpuset;
    devices = /mnt/cgroups/devices;
    blkio   = /mnt/cgroups/blkio;
    cpuacct = /mnt/cgroups/cpuacct;
}
```

```
# /etc/cgrules.conf

*:libvirtd      memory      virt/
```

- Enable cgroups support in libvirt by adding this configuration to `/etc/libvirt/qemu.conf`:

```
# /etc/libvirt/qemu.conf

cgroup_controllers = [ "cpu", "devices", "memory", "blkio", "cpuset", "cpuacct" ]
cgroup_device_acl = [
    "/dev/null", "/dev/full", "/dev/zero",
    "/dev/random", "/dev/urandom",
    "/dev/ptmx", "/dev/kvm", "/dev/kqemu",
    "/dev/rtc", "/dev/hpet", "/dev/vfio/vfio"
]
```

- After configuring the hosts start/restart the cgroups service then restart the libvirtd service.
- (Optional) If you have limited the amount of memory for VMs, you may want to set `RESERVED_MEM` parameter in host or cluster templates.

That's it. OpenNebula automatically generates a number of CPU shares proportional to the CPU attribute in the VM template. For example, consider a host running 2 VMs (73 and 74, with CPU=0.5 and CPU=1) respectively. If everything is properly configured you should see:

```
/mnt/cgroups/cpu/sysdefault/libvirt/qemu/
|-- cgroup.event_control
...
|-- cpu.shares
|-- cpu.stat
|-- notify_on_release
|-- one-73
|   |-- cgroup.clone_children
|   |-- cgroup.event_control
|   |-- cgroup.procs
|   |-- cpu.shares
|   ...
|   `-- vcpu0
|       |-- cgroup.clone_children
|       ...
|-- one-74
|   |-- cgroup.clone_children
|   |-- cgroup.event_control
|   |-- cgroup.procs
|   |-- cpu.shares
|   ...
|   `-- vcpu0
|       |-- cgroup.clone_children
|       ...
`-- tasks
```

and the cpu shares for each VM:

```
> cat /mnt/cgroups/cpu/sysdefault/libvirt/qemu/one-73/cpu.shares
512
> cat /mnt/cgroups/cpu/sysdefault/libvirt/qemu/one-74/cpu.shares
1024
```

VCPUs are not pinned so most probably the virtual process will be changing the core it is using. In an ideal case where the VM is alone in the physical host the total amount of CPU consumed will be equal to VCPU plus any overhead of virtualization (for example networking). In case there are more VMs in that physical node and is heavily used then the VMs will compete for physical CPU time. In this case cgroups will do a fair share of CPU time between VMs (a VM with CPU=2 will get double the time as a VM with CPU=1).

In case you are not overcommitting (CPU=VCPU) all the virtual CPUs will have one physical CPU (even if it's not pinned) so they could consume the number of VCPU assigned minus the virtualization overhead and any process running in the host OS.

7.2.4 Usage

KVM Specific Attributes

The following are template attributes specific to KVM, please refer to the template reference documentation for a complete list of the attributes supported to define a VM.

DISK

- **TYPE:** This attribute defines the type of the media to be exposed to the VM, possible values are: `disk` (default), `cdrom` or `floppy`. This attribute corresponds to the `media` option of the `-driver` argument of the `kvm` command.
- **DRIVER:** specifies the format of the disk image; possible values are `raw`, `qcow2`... This attribute corresponds to the `format` option of the `-driver` argument of the `kvm` command.
- **CACHE:** specifies the optional cache mechanism, possible values are `default`, `none`, `writethrough` and `writeback`.
- **IO:** set IO policy possible values are `threads` and `native`.

NIC

- **TARGET:** name for the tun device created for the VM. It corresponds to the `ifname` option of the `'-net'` argument of the `kvm` command.
- **SCRIPT:** name of a shell script to be executed after creating the tun device for the VM. It corresponds to the `script` option of the `'-net'` argument of the `kvm` command.
- **MODEL:** ethernet hardware to emulate. You can get the list of available models with this command:

```
$ kvm -net nic,model=? -nographic /dev/null
```

- **FILTER** to define a network filtering rule for the interface. Libvirt includes some predefined rules (e.g. `clean-traffic`) that can be used. [Check the Libvirt documentation](#) for more information, you can also list the rules in your system with:

```
$ virsh -c qemu:///system nwfilter-list
```

Graphics

If properly configured, libvirt and KVM can work with SPICE ([check this for more information](#)). To select it, just add to the `GRAPHICS` attribute:

- `TYPE = SPICE`

Enabling spice will also make the driver inject specific configuration for these machines. The configuration can be changed in the driver configuration file, variable `SPICE_OPTIONS`.

Virtio

Virtio is the framework for IO virtualization in KVM. You will need a linux kernel with the virtio drivers for the guest, check [the KVM documentation](#) for more info.

If you want to use the virtio drivers add the following attributes to your devices:

- DISK, add the attribute `DEV_PREFIX="vd"`
- NIC, add the attribute `MODE="virtio"`

Additional Attributes

The **raw** attribute offers the end user the possibility of passing by attributes not known by OpenNebula to KVM. Basically, everything placed here will be written literally into the KVM deployment file (**use libvirt xml format and semantics**).

```
RAW = [ type = "kvm",
        data = "<devices><serial type=\"pty\"><source path=\"/dev/pts/5\"/><target_
↪port=\"0\"/></serial><console type=\"pty\" tty=\"/dev/pts/5\"><source path=\"/dev/
↪pts/5\"/><target port=\"0\"/></console></devices>" ]
```

Disk/Nic Hotplugging

KVM supports hotplugging to the `virtio` and the `SCSI` buses. For disks, the bus the disk will be attached to is inferred from the `DEV_PREFIX` attribute of the disk template.

- `vd`: `virtio` (recommended).
- `sd`: `SCSI` (default).

If `TARGET` is passed instead of `DEV_PREFIX` the same rules apply (what happens behind the scenes is that OpenNebula generates a `TARGET` based on the `DEV_PREFIX` if no `TARGET` is provided).

The configuration for the default cache type on newly attached disks is configured in `/var/lib/one/remotes/vmm/kvm/kvmrc`:

```
# This parameter will set the default cache type for new attached disks. It
# will be used in case the attached disk does not have an specific cache
# method set (can be set using templates when attaching a disk).
DEFAULT_ATTACH_CACHE=none
```

For Disks and NICs, if the guest OS is a Linux flavor, the guest needs to be explicitly tell to rescan the PCI bus. This can be done issuing the following command as root:

```
# echo 1 > /sys/bus/pci/rescan
```

Enabling QEMU Guest Agent

QEMU Guest Agent allows the communication of some actions with the guest OS. This agent uses a virtio serial connection to send and receive commands. One of the interesting actions is that it allows to freeze the filesystem before doing an snapshot. This way the snapshot won't contain half written data. Filesystem freeze will only be used with `CEPH` and `qcow2` storage drivers.

The agent package needed in the Guest OS is available in most distributions. Is called `qemu-guest-agent` in most of them. If you need more information you can follow these links:

- https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Virtualization_Deployment_and_Administration_Guide/chap-QEMU_Guest_Agent.html
- http://wiki.libvirt.org/page/Qemu_guest_agent
- <http://wiki.qemu.org/Features/QAPI/GuestAgent>

To enable the communication channel with the guest agent this line must be present in `/etc/one/vmm_exec/vmm_exec_kvm.conf`:

```
RAW = "<devices><channel type='unix'><source mode='bind'></source><target type='virtio' name=
↳ 'org.qemu.guest_agent.0'></target></channel></devices>"
```

Importing VMs

VMs running on KVM hypervisors that were not launched through OpenNebula can be imported in OpenNebula. It is important to highlight that, besides the limitations explained in the host guide, the “Poweroff” operation is not available for these imported VMs in KVM.

7.2.5 Tuning & Extending

Multiple Actions per Host

Warning: This feature is experimental. Some modifications to the code must be done before this is a recommended setup.

By default the drivers use a unix socket to communicate with the libvirt daemon. This method can only be safely used by one process at a time. To make sure this happens the drivers are configured to send only one action per host at a time. For example, there will be only one deployment done per host at a given time.

This limitation can be solved configuring libvirt to accept TCP connections and OpenNebula to use this communication method.

Libvirt configuration

Here is described how to configure libvirtd to accept unencrypted and unauthenticated TCP connections in a CentOS 7 machine. For other setup check your distribution and libvirt documentation.

Change the file `/etc/libvirt/libvirtd.conf` in each of the hypervisors and make sure that these parameters are set and have the following values:

```
listen_tls = 0
listen_tcp = 1
tcp_port = "16509"
auth_tcp = "none"
```

You will also need to modify `/etc/sysconfig/libvirtd` and uncomment this line:

```
LIBVIRTD_ARGS="--listen"
```

After modifying these files the libvirt daemon must be restarted:

```
$ sudo systemctl restart libvirtd
```

OpenNebula configuration

The VMM driver must be configured so it allows more than one action to be executed per host. This can be done adding the parameter `-p` to the driver executable. This is done in `/etc/one/oned.conf` in the `VM_MAD` configuration section:

```
VM_MAD = [
  name       = "kvm",
  executable = "one_vmm_exec",
  arguments  = "-t 15 -r 0 kvm -p",
  default    = "vmm_exec/vmm_exec_kvm.conf",
  type       = "kvm" ]
```

Change the file `/var/lib/one/remotes/vmm/kvm/kvmrc` so set a TCP endpoint for libvirt communication:

```
export LIBVIRT_URI=qemu+tcp://localhost/system
```

The scheduler configuration should also be changed to let it deploy more than one VM per host. The file is located at `/etc/one/sched.conf` and the value to change is `MAX_HOST`. For example, to let the scheduler submit 10 VMs per host use this line:

```
MAX_HOST = 10
```

After this update the remote files in the nodes and restart opennebula:

```
$ onehost sync --force
$ sudo systemctl restart opennebula
```

Files and Parameters

The driver consists of the following files:

- `/usr/lib/one/mads/one_vmm_exec`: generic VMM driver.
- `/var/lib/one/remotes/vmm/kvm`: commands executed to perform actions.

And the following driver configuration files:

- `/etc/one/vmm_exec/vmm_exec_kvm.conf`: This file is home for default values for domain definitions (in other words, OpenNebula templates).

It is generally a good idea to place defaults for the KVM-specific attributes, that is, attributes mandatory in the KVM driver that are not mandatory for other hypervisors. Non mandatory attributes for KVM but specific to them are also recommended to have a default.

- `/var/lib/one/remotes/vmm/kvm/kvmrc`: This file holds instructions to be executed before the actual driver load to perform specific tasks or to pass environmental variables to the driver. The syntax used for the former is plain shell script that will be evaluated before the driver execution. For the latter, the syntax is the familiar:

```
ENVIRONMENT_VARIABLE=VALUE
```

The parameters that can be changed here are as follows:

Parameter	Description
LIBVIRT_URI	Connection string to libvirt
QEMU_PROTOCOL	Protocol used for live migrations
SHUTDOWN_TIME	Seconds to wait after shutdown until timeout
FORCE_DESTROY	Force VM cancellation after shutdown timeout
CANCEL_NO ACPI	Force VM's without ACPI enabled to be destroyed on shutdown
DEFAULT_ATTACH_CACHE	This parameter will set the default cache type for new attached disks. It will be used in case the attached disk does not have an specific cache method set (can be set using templates when attaching a disk).
MIGRATE_OPTIONS	Set options for the virsh migrate command

See the Virtual Machine drivers reference for more information.

7.2.6 Troubleshooting

image magic is incorrect

When trying to restore the VM from a suspended state this error is returned:

```
libvirtd1021: operation failed: image magic is incorrect
```

It can be fixed by applying:

```
options kvm_intel nested=0
options kvm_intel emulate_invalid_guest_state=0
options kvm ignore_msrs=1
```

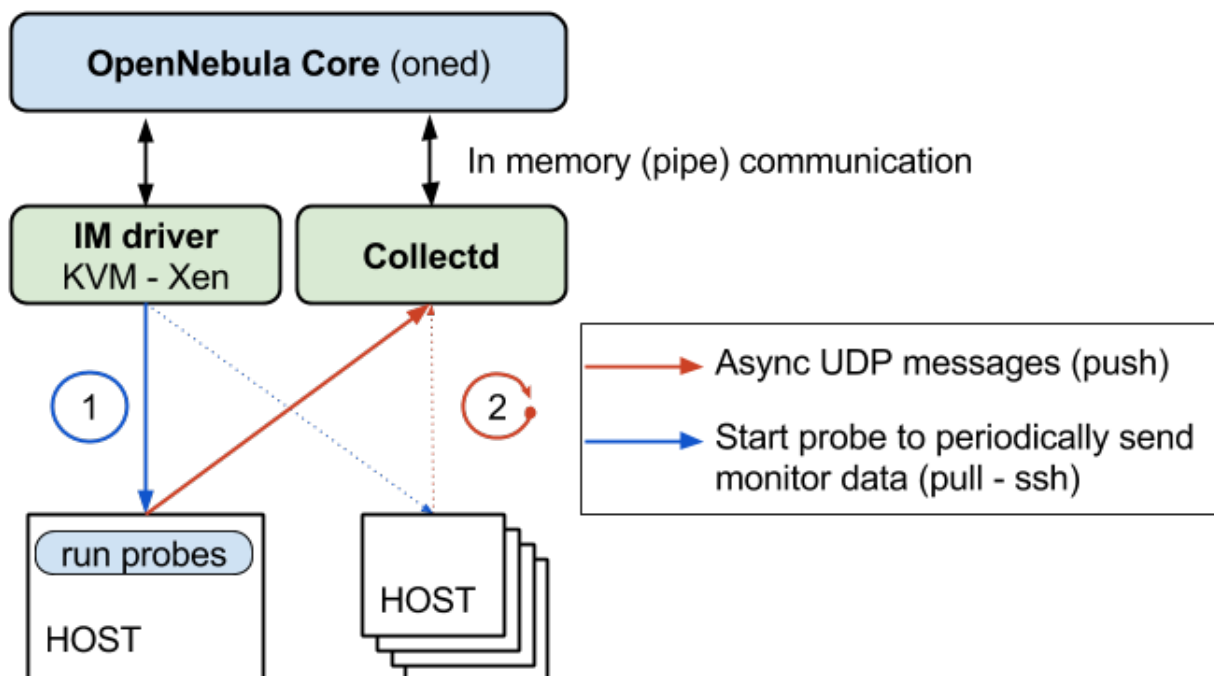
7.3 Monitoring

This section provides an overview of the OpenNebula monitoring subsystem. The monitoring subsystem gathers information relative to the Hosts and the Virtual Machines, such as the Host status, basic performance indicators, as well as Virtual Machine 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 using a push mechanism.

7.3.1 Overview

Each host periodically sends monitoring data via UDP to the Frontend which collects it and processes it in a dedicated module. This distributed monitoring system resembles the architecture of dedicated monitoring systems, using a lightweight communication protocol, and a push model.

OpenNebula starts a `collectd` daemon running in the Front-end that listens for UDP connections on port 4124. In the first monitoring cycle the OpenNebula connects to the host using `ssh` and starts a daemon that will execute the probe scripts and sends the collected data to the `collectd` daemon in the Frontend every specific amount of seconds (configurable with the `-i` option of the `collectd IM_MAD`). This way the monitoring subsystem doesn't need to make new `ssh` connections to receive data.



If the agent stops in a specific Host, OpenNebula will detect that no monitorization data is received from that hosts and will restart the probe with SSH.

7.3.2 Requirements

- The firewall of the Frontend (if enabled) must allow UDP packages incoming from the hosts on port 4124.

7.3.3 OpenNebula Configuration

Enabling the Drivers

To enable this monitoring system `/etc/one/oned.conf` must be configured with the following snippets:

collectd must be enabled both for KVM:

```

#-----
# Information Collector for KVM IM's.
#-----
# This driver CANNOT BE ASSIGNED TO A HOST, and needs to be used with KVM
# -h prints this help.
# -a Address to bind the collectd socket (defaults 0.0.0.0)
# -p UDP port to listen for monitor information (default 4124)
# -f Interval in seconds to flush collected information (default 5)
# -t Number of threads for the server (default 50)
# -i Time in seconds of the monitorization push cycle. This parameter must
#     be smaller than MONITORING_INTERVAL, otherwise push monitorization will
#     not be effective.
#-----
IM_MAD = [
    NAME           = "collectd",
    EXECUTABLE     = "collectd",

```



```
ARGUMENTS = "-p 4124 -f 5 -t 50 -i 20" ]
#-----
```

Valid arguments for this driver are:

- **-a:** Address to bind the collectd socket (defaults 0.0.0.0)
- **-p:** port number
- **-f:** Interval in seconds to flush collected information to OpenNebula (default 5)
- **-t:** Number of threads for the collectd server (default 50)
- **-i:** Time in seconds of the monitorization push cycle. This parameter must be smaller than MONITORING_INTERVAL (see below), otherwise push monitorization will not be effective.

KVM:

```
#-----
# KVM UDP-push Information Driver Manager Configuration
# -r number of retries when monitoring a host
# -t number of threads, i.e. number of hosts monitored at the same time
#-----
IM_MAD = [
    NAME           = "kvm",
    SUNSTONE_NAME  = "KVM",
    EXECUTABLE     = "one_im_ssh",
    ARGUMENTS      = "-r 3 -t 15 kvm" ]
#-----
```

The arguments passed to this driver are:

- **-r:** number of retries when monitoring a host
- **-t:** number of threads, i.e. number of hosts monitored at the same time

Monitoring Configuration Parameters

OpenNebula allows to customize the general behavior of the whole monitoring subsystem:

Parameter	Description
MONITORING_INTERVAL	Time in seconds between host and VM monitorization. It must have a value greater than the manager timer
HOST_PER_INTERVAL	Number of hosts monitored in each interval.

7.3.4 Troubleshooting

Healthy Monitoring System

Every (approximately) `monitoring_push_cycle` of seconds OpenNebula is receiving the monitoring data of every Virtual Machine and of a host like such:

```
Tue May 24 16:21:47 2016 [Z0][InM][D]: Host thost087 (0) successfully monitored.
Tue May 24 16:21:47 2016 [Z0][VMM][D]: VM 0 successfully monitored: STATE=a CPU=0.0
↪MEMORY=113404 NETRX=648 NETTX=398
Tue May 24 16:22:07 2016 [Z0][InM][D]: Host thost087 (0) successfully monitored.
Tue May 24 16:22:07 2016 [Z0][VMM][D]: VM 0 successfully monitored: STATE=a CPU=0.0
↪MEMORY=113516 NETRX=648 NETTX=468
```

```
Tue May 24 16:22:11 2016 [Z0][VMM][D]: VM 0 successfully monitored: DISK_
↪SIZE=[ID=0,SIZE=27] DISK_SIZE=[ID=1,SIZE=1]
Tue May 24 16:22:27 2016 [Z0][InM][D]: Host thost087 (0) successfully monitored.
Tue May 24 16:22:27 2016 [Z0][VMM][D]: VM 0 successfully monitored: STATE=a CPU=0.0_
↪MEMORY=113544 NETRX=648 NETTX=468
```

However, if in `oned.log` a host is being monitored **actively** periodically (every `MONITORING_INTERVAL` seconds) then the monitorization is **not** working correctly:

```
Tue May 24 16:24:23 2016 [Z0][InM][D]: Monitoring host thost087 (0)
Tue May 24 16:25:23 2016 [Z0][InM][D]: Monitoring host thost087 (0)
Tue May 24 16:26:23 2016 [Z0][InM][D]: Monitoring host thost087 (0)
```

If this is the case it's probably because OpenNebula is receiving probes faster than it can process. See the Tuning section to fix this.

Monitoring Probes

For the troubleshooting of errors produced during the execution of the monitoring probes, please refer to the *troubleshooting* section.

7.3.5 Tuning & Extending

Adjust Monitoring Interval Times

In order to tune your OpenNebula installation with appropriate values of the monitoring parameters you need to adjust the `-i` option of the `collectd IM_MAD` (the monitoring push cycle).

If the system is not working healthily it will be due to the database throughput since OpenNebula will write the monitoring information to a database, an amount of ~4KB per VM. If the number of virtual machines is too large and the monitoring push cycle too low, OpenNebula will not be able to write that amount of data to the database.

Driver Files

The probes are specialized programs that obtain the monitor metrics. Probes are defined for each hypervisor, and are located at `/var/lib/one/remotes/im/kvm-probes.d` for KVM.

You can easily write your own probes or modify existing ones, please see the Information Manager Drivers guide. Remember to synchronize the monitor probes in the hosts using `onehost sync` as described in the Managing Hosts guide.

7.4 PCI Passthrough

It is possible to discover PCI devices in the Hosts and assign them to Virtual Machines for the KVM hypervisor.

The setup and environment information is taken from [here](#). You can safely ignore all the VGA related sections, for PCI devices that are not graphic cards, or if you don't want to output video signal from them.

Warning: The overall setup state was extracted from a preconfigured Fedora 22 machine. **Configuration for your distro may be different.**

7.4.1 Requirements

- The host that is going to be used for virtualization needs to support **I/O MMU**. For Intel processors this is called VT-d and for AMD processors is called AMD-Vi. The instructions are made for Intel branded processors but the process should be very similar for AMD.
- kernel \geq 3.12

7.4.2 Machine Configuration (Hypervisor)

Kernel Configuration

The kernel must be configured to support I/O MMU and to blacklist any driver that could be accessing the PCI's that we want to use in our VMs. The parameter to enable I/O MMU is:

```
intel_iommu=on
```

We also need to tell the kernel to load the `vfio-pci` driver and blacklist the drivers for the selected cards. For example, for nvidia GPUs we can use these parameters:

```
rd.driver.pre=vfio-pci rd.driver.blacklist=nouveau
```

Loading vfio Driver in initrd

The modules for `vfio` must be added to `initrd`. The list of modules are `vfio` `vfio_iommu_type1` `vfio_pci` `vfio_virqfd`. For example, if your system uses `dracut` add the file `/etc/dracut.conf.d/local.conf` with this line:

```
add_drivers+="vfio vfio_iommu_type1 vfio_pci vfio_virqfd"
```

and regenerate `initrd`:

```
# dracut --force
```

Driver Blacklisting

The same blacklisting done in the kernel parameters must be done in the system configuration. `/etc/modprobe.d/blacklist.conf` for nvidia GPUs:

```
blacklist nouveau
blacklist lbm-nouveau
options nouveau modeset=0
alias nouveau off
alias lbm-nouveau off
```

Alongside this configuration `vfio` driver should be loaded passing the id of the PCI cards we want to attach to VMs. For example, for nvidia Grid K2 GPU we pass the id `10de:11bf`. File `/etc/modprobe.d/local.conf`:

```
options vfio-pci ids=10de:11bf
```

vfio Device Binding

I/O MMU separates PCI cards into groups to isolate memory operation between devices and VMs. To add the cards to vfio and assign a group to them we can use the scripts shared in the [aforementioned web page](#).

This script binds a card to vfio. It goes into `/usr/local/bin/vfio-bind`:

```
#!/bin/sh
modprobe vfio-pci
for dev in "$@"; do
    vendor=$(cat /sys/bus/pci/devices/$dev/vendor)
    device=$(cat /sys/bus/pci/devices/$dev/device)
    if [ -e /sys/bus/pci/devices/\$dev/driver ]; then
        echo $dev > /sys/bus/pci/devices/$dev/driver/unbind
    fi
    echo $vendor $device > /sys/bus/pci/drivers/vfio-pci/new_id
done
```

The configuration goes into `/etc/sysconfig/vfio-bind`. The cards are specified with PCI addresses. Addresses can be retrieved with `lspci` command. Make sure to prepend the domain that is usually 0000. For example:

```
DEVICES="0000:04:00.0 0000:05:00.0 0000:84:00.0 0000:85:00.0"
```

Here is a systemd script that executes the script. It can be written to `/etc/systemd/system/vfio-bind.service` and enabled:

```
[Unit]
Description=Binds devices to vfio-pci
After=syslog.target

[Service]
EnvironmentFile=-/etc/sysconfig/vfio-bind
Type=oneshot
RemainAfterExit=yes
ExecStart=-/usr/local/bin/vfio-bind $DEVICES

[Install]
WantedBy=multi-user.target
```

qemu Configuration

Now we need to give qemu access to the vfio devices for the groups assigned to the PCI cards. We can get a list of PCI cards and its I/O MMU group using this command:

```
# find /sys/kernel/iommu_groups/ -type l
```

In our example our cards have the groups 45, 46, 58 and 59 so we add this configuration to `/etc/libvirt/qemu.conf`:

```
cgroup_device_acl = [
    "/dev/null", "/dev/full", "/dev/zero",
    "/dev/random", "/dev/urandom",
    "/dev/ptmx", "/dev/kvm", "/dev/kqemu",
    "/dev/rtc", "/dev/hpet", "/dev/vfio/vfio",
    "/dev/vfio/45", "/dev/vfio/46", "/dev/vfio/58",
    "/dev/vfio/59"
]
```

7.4.3 Driver Configuration

The only configuration that is needed is the filter for the monitoring probe that gets the list of PCI cards. By default the probe lists all the cards available in a host. To narrow the list a filter configuration can be changed in `/var/lib/one/remotes/im/kvm-probes.d/pci.rb` and set a list with the same format as `lspci`:

```
# This variable contains the filters for PCI card monitoring. The format
# is the same as lspci and several filters can be added separated by commas.
# A nil filter will retrieve all PCI cards.
#
# From lspci help:
#   -d [<vendor>]:[<device>][:<class>]
#
# For example
#
# FILTER = '::0300' # all VGA cards
# FILTER = '10de::0300' # all NVIDIA VGA cards
# FILTER = '10de:11bf:0300' # only GK104GL [GRID K2]
# FILTER = '8086::0300,::0106' # all Intel VGA cards and any SATA controller
```

7.4.4 Usage

The basic workflow is to inspect the host information, either in the CLI or in Sunstone, to find out the available PCI devices, and to add the desired device to the template. PCI devices can be added by specifying `VENDOR`, `DEVICE` and `CLASS`, or simply `CLASS`. Note that OpenNebula will only deploy the VM in a host with the available PCI device. If no hosts match, an error message will appear in the Scheduler log.

CLI

A new table in `onehost show` command gives us the list of PCI devices per host. For example:

```
PCI DEVICES

  VM ADDR      TYPE                NAME
    00:00.0 8086:0a04:0600 Haswell-ULT DRAM Controller
    00:02.0 8086:0a16:0300 Haswell-ULT Integrated Graphics Controller
123 00:03.0 8086:0a0c:0403 Haswell-ULT HD Audio Controller
    00:14.0 8086:9c31:0c03 8 Series USB xHCI HC
    00:16.0 8086:9c3a:0780 8 Series HECI #0
    00:1b.0 8086:9c20:0403 8 Series HD Audio Controller
    00:1c.0 8086:9c10:0604 8 Series PCI Express Root Port 1
    00:1c.2 8086:9c14:0604 8 Series PCI Express Root Port 3
    00:1d.0 8086:9c26:0c03 8 Series USB EHCI #1
    00:1f.0 8086:9c43:0601 8 Series LPC Controller
    00:1f.2 8086:9c03:0106 8 Series SATA Controller 1 [AHCI mode]
    00:1f.3 8086:9c22:0c05 8 Series SMBus Controller
    02:00.0 8086:08b1:0280 Wireless 7260
```

- **VM:** The VM ID using that specific device. Empty if no VMs are using that device.
- **ADDR:** PCI Address.
- **TYPE:** Values describing the device. These are `VENDOR:DEVICE:CLASS`. These values are used when selecting a PCI device do to passthrough.
- **NAME:** Name of the PCI device.

To make use of one of the PCI devices in a VM a new option can be added selecting which device to use. For example this will ask for a Haswell-ULT HD Audio Controller:

```
PCI = [
  VENDOR = "8086",
  DEVICE = "0a0c",
  CLASS = "0403"
]
```


The device can be also specified without all the type values. For example, to get any PCI Express Root Ports this can be added to a VM template:

```
PCI = [
  CLASS = "0604"
]
```

More than one PCI options can be added to attach more than one PCI device to the VM.

Sunstone

In Sunstone the information is displayed in the **PCI** tab:



VM	PCI Address	Type	Name
	00:00.0	8086:1604:0600	Broadwell-U Host Bridge -OPI
	00:02.0	8086:1616:0300	Broadwell-U Integrated Graphics
	00:03.0	8086:160c:0403	Broadwell-U Audio Controller
	00:04.0	8086:1603:1180	Broadwell-U Camarillo Device
	00:14.0	8086:9cb1:0c03	Wildcat Point-LP USB xHCI Controller
	00:16.0	8086:9cba:0780	Wildcat Point-LP MEI Controller #1
	00:1b.0	8086:9ca0:0403	Wildcat Point-LP High Definition Audio Controller
	00:1c.0	8086:9c90:0604	Wildcat Point-LP PCI Express Root Port #1
	00:1c.3	8086:9c96:0604	Wildcat Point-LP PCI Express Root Port #4
	00:1d.0	8086:9ca6:0c03	Wildcat Point-LP USB EHCI Controller

To add a PCI device to a template, select the **Other** tab:

- General
- Storage
- Network
- OS Booting
- Input/Output
- Context
- Scheduling
- Hybrid
- Other

RAW data

TYPE

DATA ?

PCI Devices

Vendor

Device

Class



+ Add PCI device

OPEN CLOUD STORAGE SETUP

8.1 Overview

8.1.1 Datastore Types

OpenNebula storage is structured around the Datastore concept. A Datastore is any storage medium to store disk images. OpenNebula features three different datastore types:

- The **Images Datastore**, stores the images repository.
- The **System Datastore** holds disk for running virtual machines. Disk are moved, or cloned to/from the Images datastore when the VMs are deployed or terminated; or when disks are attached or snapshotted.
- The **Files & Kernels Datastore** to store plain files and not disk images. The plain files can be used as kernels, ram-disks or context files. *See details here.*

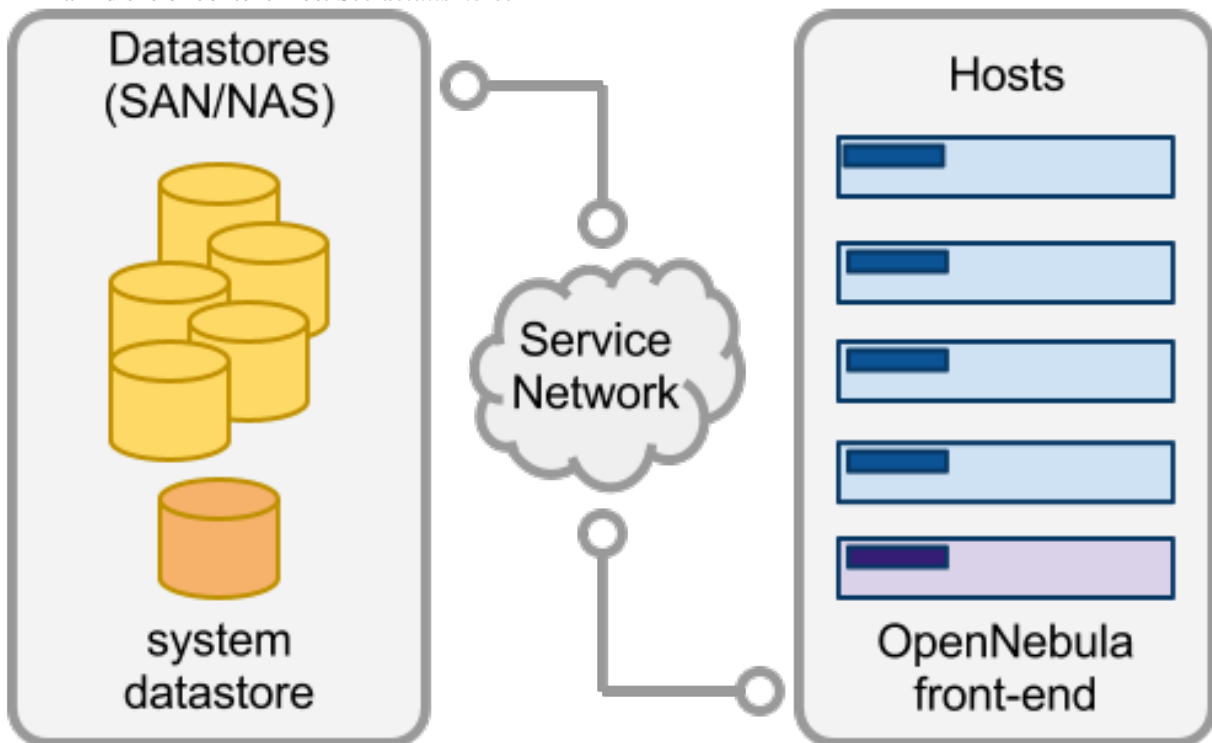


Image Datastores

There are different Image Datastores depending on how the images are stored on the underlying storage technology:

- *Filesystem*, to store images in a file form.
- *LVM*, to store images in LVM logical volumes.
- *Ceph*, to store images using Ceph block devices.
- *Raw Device Mapping*, to direct attach to the virtual machine existing block devices in the nodes.
- *iSCSI - Libvirt Datastore*, to access iSCSI devices through the built-in qemu support.

Disk images are transferred between the Image and System datastores by the transfer manager (TM) drivers. These drivers are specialized pieces of software that perform low-level storage operations. The following table summarizes the available transfer modes for each datastore:

Datastore	Image to System Datastore disk transfers methods
Filesystem	<ul style="list-style-type: none"> • shared, images are exported in a shared filesystem • ssh, images are copied using the ssh protocol • qcow2, like <i>shared</i> but specialized for the qcow2 format
Ceph	<ul style="list-style-type: none"> • ceph, all images are exported in Ceph pools • shared, volatile & context disks exported in a shared FS.
LVM	<ul style="list-style-type: none"> • fs_lvm, images exported in a shared FS but dumped to a LV
Raw Devices	<ul style="list-style-type: none"> • dev, images are existing block devices in the nodes
iSCSI libvirt	<ul style="list-style-type: none"> • iscsi, images are iSCSI targets

8.1.2 How Should I Read This Chapter

Before reading this chapter make sure you have read the *Open Cloud Host* chapter. After that, proceed to the specific section for the Datastores you may be interested in.

After reading this chapter you should read the *Open Cloud Networking* chapter.

8.1.3 Hypervisor Compatibility

This chapter applies only to KVM.

Follow the *vCenter Storage* section for a similar guide for vCenter.

8.2 Filesystem Datastore

The Filesystem Datastore lets you store VM images in a file form. The use of file-based disk images presents several benefits over device backed disks (e.g. easily backup images, or use of shared FS) although it may be less performing in some cases.

Usually it is a good idea to have multiple filesystem datastores to:

- Balancing I/O operations between storage servers
- Use different datastores for different cluster hosts
- Apply different transfer modes to different images
- Different SLA policies (e.g. backup) can be applied to different VM types or users
- Easily add new storage to the cloud

The Filesystem datastore can be used with three different transfer modes, described below:

- **shared**, images are exported in a shared filesystem
- **ssh**, images are copied using the ssh protocol
- **qcow2**, like *shared* but specialized for the qcow2 format

8.2.1 Datastore Layout

Images are saved into the corresponding datastore directory (`/var/lib/one/datastores/<DATASTORE ID>`). Also, for each running virtual machine there is a directory (named after the VM ID) in the corresponding System Datastore. These directories contain the VM disks and additional files, e.g. checkpoint or snapshots.

For example, a system with an Image Datastore (1) with three images and 3 Virtual Machines (VM 0 and 2 running, and VM 7 stopped) running from System Datastore 0 would present the following layout:

```

/var/lib/one/datastores
|-- 0/
|   |-- 0/
|   |   |-- disk.0
|   |   |-- disk.1
|   |   |-- 2/
|   |       |-- disk.0
|   |       |-- 7/
|   |           |-- checkpoint
|   |           |-- disk.0
|-- 1
|   |-- 05a38ae85311b9dbb4eb15a2010f11ce
|   |-- 2bbec245b382fd833be35b0b0683ed09
|   |-- d0e0df1fb8cfa88311ea54dfbcfc4b0c

```

Note: The canonical path for `/var/lib/one/datastores` can be changed in `oned.conf` with the `DATASTORE_LOCATION` configuration attribute

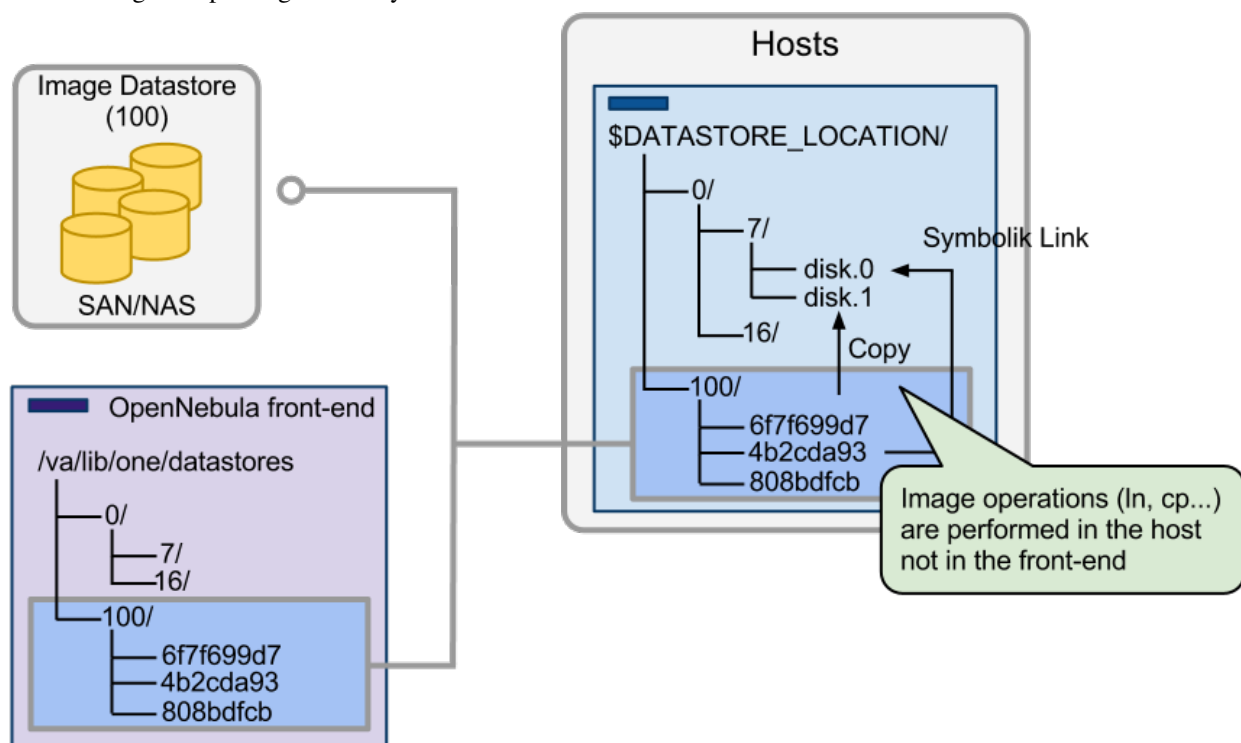
Shared & Qcow2 Transfer Modes

The shared transfer driver assumes that the datastore is mounted in all the hosts of the cluster. Typically this is achieved through a distributed FS like NFS, GlusterFS or Lustre.

When a VM is created, its disks (the `disk.i` files) are copied or linked in the corresponding directory of the system datastore. These file operations are always performed remotely on the target host.

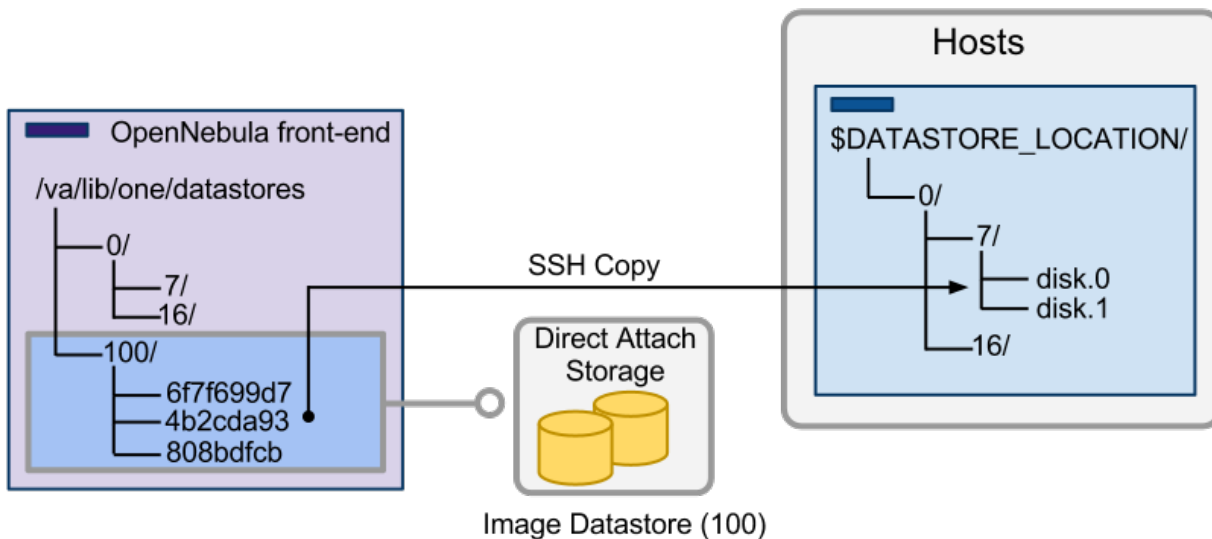
This transfer mode usually reduces VM deployment times and **enables live-migration**, but it can also become a bottleneck in your infrastructure and degrade your Virtual Machines performance if the virtualized services perform disk-intensive workloads. Usually this limitation may be overcome by:

- Using different file-system servers for the images datastores, so the actual I/O bandwidth is balanced
- Using an ssh System Datastore instead, the images are copied locally to each host
- Tuning or improving the file-system servers



SSH Transfer Mode

In this case the System Datastore is distributed among the hosts. The ssh transfer driver uses the hosts' local storage to place the images of running Virtual Machines. All the operations are then performed locally but images have to be copied always to the hosts, which in turn can be a very resource demanding operation. Also this driver prevents the use of live-migrations between hosts.



8.2.2 Frontend Setup

The Frontend needs to prepare the storage area for:

- The Image Datastores, to store the images.
- The System Datastores, will hold temporary disks and files for VMs stopped and undeployed.

Shared & Qcow2 Transfer Modes

Simply mount the **Image** Datastore directory in the front-end in `/var/lib/one/datastores/<datastore_id>`. Note that if all the datastores are of the same type you can mount the whole `/var/lib/one/datastores` directory.

Warning: The frontend only needs to mount the Image Datastores and **not** the System Datastores.

Note: NFS volumes mount tips. The following options are recommended to mount a NFS shares: `soft, intr, rsize=32768, wsize=32768`. With the documented configuration of libvirt/kvm the image files are accessed as `oneadmin` user. In case the files must be read by `root` the option `no_root_squash` must be added.

SSH Transfer Mode

Simply make sure that there is enough space under `/var/lib/one/datastores` to store Images and the disks of the stopped and undeployed virtual machines. Note that `/var/lib/one/datastores` **can be mounted from any NAS/SAN server in your network**.

8.2.3 Node Setup

Shared & Qcow2 Transfer Modes

The configuration is the same as for the Frontend above, simply mount in each node the datastore directories in `/var/lib/one/datastores/<datastore_id>`.

SSH Transfer Mode

Just make sure that there is enough space under `/var/lib/one/datastores` to store the disks of running VMs on that host.

8.2.4 OpenNebula Configuration

Once the Filesystem storage is setup, the OpenNebula configuration comprises two steps:

- Create a System Datastore
- Create an Image Datastore

Create a System Datastore

To create a new System Datastore you need to specify its type as system datastore and transfer mode:

Attribute	Description
NAME	The name of the datastore
TYPE	SYSTEM_DS
TM_MAD	shared for shared transfer mode qcow2 for qcow2 transfer mode ssh for ssh transfer mode

This can be done either in Sunstone or through the CLI, for example to create a System Datastore using the shared mode simply:

```
$ cat systemds.txt
NAME      = nfs_system
TM_MAD    = shared
TYPE      = SYSTEM_DS

$ onedatastore create systemds.txt
ID: 101
```

Create an Image Datastore

In the same way, to create an Image Datastore you need to set:

Attribute	Description
NAME	The name of the datastore
DS_MAD	fs
TM_MAD	shared for shared transfer mode qcow2 for qcow2 transfer mode ssh for ssh transfer mode

For example, the following illustrates the creation of a filesystem datastore using the shared transfer drivers.

```
$ cat ds.conf
NAME = nfs_images
DS_MAD = fs
TM_MAD = shared

$ onedatastore create ds.conf
ID: 100
```

Also note that there are additional attributes that can be set, check the datastore template attributes.

Warning: Be sure to use the same TM_MAD for both the System and Image datastore.

Additional Configuration

The qcow2 drivers are a specialization of the shared drivers to work with the qcow2 format for disk images. Images are created and through the `qemu-img` command using the original image as backing file. Custom options can be sent to `qemu-img clone` action through the variable `QCOW2_OPTIONS` in `/var/lib/one/remotes/tm/tmrc`.

8.3 Ceph Datastore

The Ceph datastore driver provides OpenNebula users with the possibility of using Ceph block devices as their Virtual Images.

Warning: This driver requires that the OpenNebula nodes using the Ceph driver must be Ceph clients of a running Ceph cluster. More information in [Ceph documentation](#).

8.3.1 Datastore Layout

Images and virtual machine disks are stored in the same Ceph pool. Each Image is named `one-<IMAGE ID>` in the pool. Virtual machines will use these rbd volumes for its disks if the Images are persistent, otherwise new snapshots are created in the form `one-<IMAGE ID>-<VM ID>-<DISK ID>`.

For example, consider a system using an Image and System Datastore backed by a Ceph pool named `one`. The pool with one Image (ID 0) and two Virtual Machines 14 and 15 using this Image as virtual disk 0 would be similar to:

```
$ rbd ls -l -p one --id libvirt
NAME          SIZE PARENT          FMT PROT LOCK
one-0         10240M          2
one-0@snap    10240M          2 yes
one-0-14-0    10240M one/one-0@snap    2
one-0-15-0    10240M one/one-0@snap    2
```

Note: In this case context disk and auxiliar files (deployment description and checkpoints) are stored locally in the nodes.

8.3.2 Ceph Cluster Setup

This guide assumes that you already have a functional Ceph cluster in place. Additionally you need to:

- Create a pool for the OpenNebula datastores. Write down the name of the pool to include it in the datastore definitions.

```
$ ceph osd pool create one 128
$ ceph osd lspools
0 data,1 metadata,2 rbd,6 one,
```

- Define a Ceph user to access the datastore pool, this user will be also used by libvirt to access the disk images. Also, get a copy of the key of this user to distribute it later to the OpenNebula nodes. For example, create a user libvirt:

```
$ ceph auth get-or-create client.libvirt mon 'allow r' osd \
    'allow class-read object_prefix rbd_children, allow rwx pool=one'
$ ceph auth get-key client.libvirt | tee client.libvirt.key
$ ceph auth get client.libvirt -o ceph.client.libvirt.keyring
```

- Although RDB format 1 is supported it is strongly recommended to use Format 2. Check that `ceph.conf` includes:

```
[global]
rbd_default_format = 2
```

- Pick a set of client nodes of the cluster to act as storage bridges. These nodes will be used to import images into the Ceph Cluster from OpenNebula. These nodes must have `qemu-img` command installed.

Note: For production environments it is recommended to **not co-allocate** ceph services (monitor, osds) with OpenNebula nodes or front-end

8.3.3 Frontend Setup

The Frontend does not need any specific Ceph setup, it will access the Ceph cluster through the storage bridges.

8.3.4 Node Setup

In order to use the Ceph cluster the nodes needs to be configured as follows:

- The ceph client tools must be available in the node
- The `mon` daemon must be defined in the `ceph.conf` for all the nodes, so `hostname` and `port` doesn't need to be specified explicitly in any Ceph command.
- Copy the Ceph user keyring (`ceph.client.libvirt.keyring`) to the nodes under `/etc/ceph`, and the user key (`client.libvirt.key`) to the `oneadmin` home.

```
$ scp ceph.client.libvirt.keyring root@node:/etc/ceph
$ scp client.libvirt.key oneadmin@node:
```

- Generate a secret for the Ceph user and copy it to the nodes under oneadmin home. Write down the UUID for later use.

```
$ UUID=`uuidgen`; echo $UUID
c7bdeabf-5f2a-4094-9413-58c6a9590980

$ cat > secret.xml <<EOF
<secret ephemeral='no' private='no'>
  <uuid>$UUID</uuid>
  <usage type='ceph'>
    <name>client.libvirt secret</name>
  </usage>
</secret>
EOF

$ scp secret.xml oneadmin@node:
```

- Define the a libvirt secret and remove key files in the nodes:

```
$ virsh -c qemu:///system secret-define secret.xml

$ virsh -c qemu:///system secret-set-value --secret $UUID --base64 $(cat client.
→libvirt.key)

$ rm client.libvirt.key
```

- The oneadmin account needs to access the Ceph Cluster using the libvirt Ceph user defined above. This requires access to the ceph user keyring. Test that Ceph client is properly configured in the node.

```
$ ssh oneadmin@node

$ rbd ls -p one --id libvirt
```

You can read more information about this in the Ceph guide [Using libvirt with Ceph](#).

- Ancillary virtual machine files like context disks, deployment and checkpoint files are created at the nodes under `/var/lib/one/datastores/`, make sure that enough storage for these files is provisioned in the nodes.

8.3.5 OpenNebula Configuration

To use your Ceph cluster with OpenNebula you need to define a System and Image datastores. Both datastores will share the same configuration parameters and Ceph pool.

Note: You may add additional Image and System Datastores pointing to other pools with different allocation/replication policies in Ceph.

Each Image/System Datastore pair needs to define the same following attributes:

Attribute	Description	Mandatory
POOL_NAME	The Ceph pool name	YES
CEPH_USER	The Ceph user name, used by libvirt and rbd commands.	YES
TM_MAD	ceph	YES
CEPH_CONF	Non default ceph configuration file if needed.	NO
RBD_FORMAT	By default RBD Format 2 will be used.	NO

Create a System Datastore

Create a System Datastore in Sunstone or through the CLI, for example:

```
$ cat systemds.txt
NAME      = ceph_system
TM_MAD    = ceph
TYPE      = SYSTEM_DS

POOL_NAME = one
CEPH_USER = libvirt

$ onedatastore create systemds.txt
ID: 101
```

Note: Ceph can also work with a System Datastore of type Filesystem in a shared transfer mode, as described *in the Filesystem Datastore section*. In that case volatile and swap disks are created as plain files in the System Datastore. Note that apart from the Ceph Cluster you need to setup a shared FS.

Create an Image Datastore

Apart from the previous attributes, that need to be the same as the associated System Datastore, the following can be set for an Image Datastore:

Attribute	Description	Mandatory
DS_MAD	ceph	YES
DISK_TYPE	RBD	YES
BRIDGE_LIST	List of storage bridges to access the Ceph cluster	YES
CEPH_HOST	Space-separated list of Ceph monitors. Example: host1 host2:port2 host3 host4:port4.	YES
CEPH_SECRET	The UUID of the libvirt secret.	YES
STAGING_DIR	Default path for image operations in the bridges	NO

An example of datastore:

```
> cat ds.conf
NAME = "cephds"
DS_MAD = ceph
TM_MAD = ceph

DISK_TYPE = RBD

POOL_NAME = one
CEPH_HOST = host1 host2:port2
CEPH_USER = libvirt
CEPH_SECRET = "6f88b54b-5dae-41fe-a43e-b2763f601cfc"

BRIDGE_LIST = cephfrontend

> onedatastore create ds.conf
ID: 101
```

Additional Configuration

Default values for the Ceph drivers can be set in `/var/lib/one/remotes/datastore/ceph/ceph.conf`:

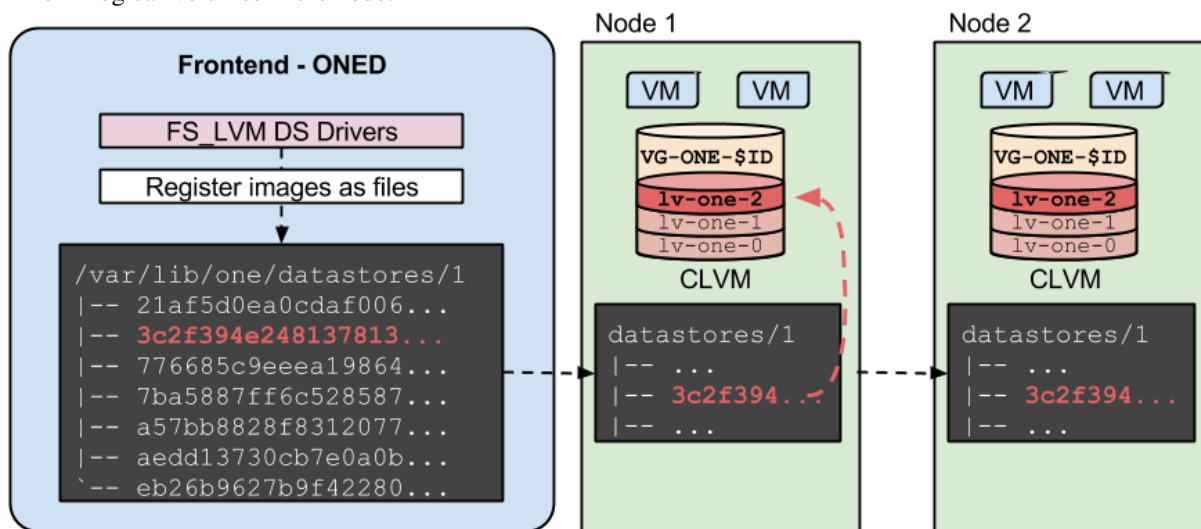
- `POOL_NAME`: Default volume group
- `STAGING_DIR`: Default path for image operations in the storage bridges
- `RBD_FORMAT`: Default format for RBD volumes.

8.4 LVM Datastore

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 it may increase I/O performance.

8.4.1 Datastore Layout

Images are stored as regular files (under the usual path: `/var/lib/one/datastores/<id>`) in the Image Datastore, but they will be dumped into a Logical Volumes (LV) upon virtual machine creation. The virtual machines will run from Logical Volumes in the node.



This is the recommended driver to be used when a high-end SAN is available. The same LUN can be exported to all the hosts, Virtual Machines will be able to run directly from the SAN.

Note: The LVM datastore does **not** need CLVM configured in your cluster. The drivers refresh LVM meta-data each time an image is needed in another host.

For example, consider a system with two Virtual Machines (9 and 10) using a disk, running in a LVM Datastore, with ID 0. The nodes have configured a shared LUN and created a volume group named `vg-one-0`, the layout of the datastore would be:

```
# lvs
LV          VG          Attr          LSize Pool Origin Data%  Meta%  Move
lv-one-10-0 vg-one-0  -wi-----  2.20g
lv-one-9-0  vg-one-0  -wi-----  2.20g
```

8.4.2 Frontend Setup

No additional configuration is needed.

8.4.3 Node Setup

Nodes needs to meet the following requirements:

- LVM2 must be available in the Hosts.
- `lvm2-lvmetad` must be disabled. Set this parameter in `/etc/lvm/lvm.conf`: `use_lvmetad = 0`, and disable the `lvm2-lvmetad.service` if running.
- `oneadmin` needs to belong to the `disk` group.
- All the nodes needs to have access to the same LUNs.
- A LVM VG needs to be created in the shared LUNs for each datastore following name: `vg-one-<system_ds_id>`. This just need to be done in one node.
- Virtual Machine disks are symbolic links to the block devices. However, additional VM files like checkpoints or deployment files are stored under `/var/lib/one/datastores/<id>`. Be sure that enough local space is present.

8.4.4 OpenNebula Configuration

Once the storage is setup, the OpenNebula configuration comprises two steps:

- Create a System Datastore
- Create an Image Datastore

Create a System Datastore

LVM System Datastores needs to be created with the following values:

Attribute	Description
NAME	The name of the Datastore
TM_MAD	<code>fs_lvm</code>
TYPE	<code>SYSTEM_DS</code>

For example:

```
> cat ds.conf
NAME    = lvm_system
TM_MAD  = fs_lvm
TYPE    = SYSTEM_DS

> onedatastore create ds.conf
ID: 100
```

Create an Image Datastore

To create an Image Datastore you just need to define the name, and set the following:

Attribute	Description
NAME	The name of the datastore
TYPE	IMAGE_DS
DS_MAD	fs
TM_MAD	fs_lvm
DISK_TYPE	BLOCK

For example, the following examples illustrates the creation of an LVM datastore using a configuration file. In this case we will use the host `host01` as one of our OpenNebula LVM-enabled hosts.

```
> cat ds.conf
NAME = production
DS_MAD = fs
TM_MAD = fs_lvm
DISK_TYPE = "BLOCK"
TYPE = IMAGE_DS
SAFE_DIRS="/var/tmp /tmp"

> onedatastore create ds.conf
ID: 101
```

8.5 Raw Device Mapping (RDM) Datastore

The RDM Datastore is an Image Datastore that enables raw access to node block devices.

Warning: The datastore should only be usable by the administrators. Letting users create images in this datastore will cause security problems. For example, register an image `/dev/sda` and reading the host filesystem.

8.5.1 Datastore Layout

The RDM Datastore is used to register already existent block devices in the nodes. The devices should be already setup and available and, VMs using these devices must be fixed to run in the nodes ready for them. Additional virtual machine files, like deployment files or volatile disks are created as regular files.

8.5.2 Frontend Setup

No additional setup is required.

8.5.3 Node Setup

The devices you want to attach to a VM should be accessible by the hypervisor. As KVM usually runs as `oneadmin`, make sure this user is in a group with access to the disk (like `disk`) and has read write permissions for the group.

8.5.4 OpenNebula Configuration

Once the storage is setup, the OpenNebula configuration comprises two steps:

- Create a System Datastore
- Create an Image Datastore

Create a System Datastore

The RDM Datastore can work with the following System Datastores:

- Filesystem, shared transfer mode
- Filesystem, ssh transfer mode

Please refer to the *Filesystem Datastore section* for more details. Note that the System Datastore is only used for volatile disks and context devices.

Create an Image Datastore

To create an Image Datastore you just need to define the name, and set the following:

Attribute	Description
NAME	The name of the datastore
TYPE	IMAGE_DS
DS_MAD	dev
TM_MAD	dev
DISK_TYPE	BLOCK

An example of datastore:

```
> cat rdm.conf
NAME      = rdm_datastore
TYPE      = "IMAGE_DS"
DS_MAD    = "dev"
TM_MAD    = "dev"
DISK_TYPE = "BLOCK"

> onedatastore create rdm.conf
ID: 101
```

8.5.5 Datastore Usage

New images can be added as any other image specifying the path. If you are using the CLI do not use the shorthand parameters as the CLI check if the file exists and the device most provably won't exist in the frontend. As an example here is an image template to add a node disk `/dev/sdb`:

```
NAME=scsi_device
PATH=/dev/sdb
PERSISTENT=YES
```

Note: As this datastore does is just a container for existing devices images does not take any size from it. All devices registered will render size of 0 and the overall devices datastore will show up with 1MB of available space

8.6 iSCSI - Libvirt Datastore

This datastore is used to register already existing iSCSI volume available to the hypervisor nodes

Warning: The datastore should only be usable by the administrators. Letting users create images in this datastore can cause security problems.

8.6.1 Frontend Setup

No additional configuration is needed

8.6.2 Node Setup

The nodes need to meet the following requirements: * The devices you want to attach to a VM should be accessible by the hypervisor. * Qemu needs to be compiled with Libiscsi support.

iSCSI CHAP Authentication

In order to use CHAP authentication, you will need to create a libvirt secret in **all** the hypervisors. Follow this [Libvirt Secret XML format](#) guide to register the secret. Take this into consideration:

- `incominguser` field on the iSCSI authentication file should match the Datastore's `ISCSI_USER` parameter.
- `<target>` field in the secret XML document will contain the `ISCSI_USAGE` parameter.
- Do this in all the hypervisors.

8.6.3 OpenNebula Configuration

Once the storage is setup, the OpenNebula configuration comprises two steps:

- Create a System Datastore
- Create an Image Datastore

Create a System Datastore

The RDM Datastore can work with the following System Datastores:

- Filesystem, shared transfer mode
- Filesystem, ssh transfer mode

Please refer to the [Filesystem Datastore section](#) for more details. Note that the System Datastore is only used for volatile disks and context devices.

Create an Image Datastore

To create an Image Datastore you just need to define the name, and set the following:

Attribute	Description
NAME	The name of the datastore
TYPE	IMAGE_DS
DS_MAD	iscsi
TM_MAD	iscsi
DISK_TYPE	ISCSI
ISCSI_HOST	iSCSI Host. Example: host or host:port.

If you need to use CHAP authentication (optional) add the following attributes to the datastore:

Attribute	Description
ISCSI_USAGE	Usage of the secret with the CHAP Auth string.
ISCSI_USER	user the iSCSI CHAP authentication.

An example of datastore:

```
> cat iscsi.ds
NAME = iscsi

DISK_TYPE = "ISCSI"

DS_MAD = "iscsi"
TM_MAD = "iscsi"

ISCSI_HOST = "the_iscsi_host"
ISCSI_USER = "the_iscsi_user"
ISCSI_USAGE = "the_iscsi_usage"

> onedatastore create iscsi.ds
ID: 101
```

Warning: Images created in this datastore should be persistent. Making the images non persistent allows more than one VM use this device and will probably cause problems and data corruption.

8.6.4 Datastore Usage

New images can be added as any other image specifying the path. If you are using the CLI **do not use the shorthand parameters** as the CLI check if the file exists and the device most provably won't exist in the frontend.

As an example here is an image template to add a node disk `iqn.1992-01.com.example:storage:diskarrays-sn-a8675309`.

```
NAME = iscsi_device
PATH = iqn.1992-01.com.example:storage:diskarrays-sn-a8675309
PERSISTENT = YES
```

Warning: As this datastore does is just a container for existing devices images does not take any size from it. All devices registered will render size of 0 and the overall devices datastore will show up with 1MB of available space

Note: You may override the any of the following: `ISCSI_HOST`, `ISCSI_USER`, `ISCSI_USAGE` and `ISCSI_IQN` parameters in the image template. These overridden parameters will come into effect for new Virtual Machines.

Here is an example of an iSCSI LUN template that uses the iSCSI transfer manager.

```
oneadmin@onedv:~/exampletemplates$ more iscsiimage.tpl
NAME=iscsi_device_with_lun
PATH=iqn.2014.01.192.168.50.61:test:7cd2ccl1e/0
ISCSI_HOST=192.168.50.61
PERSISTENT=YES
```

Note the explicit “/0” at the end of the IQN target path. This is the iSCSI LUN ID.

8.7 The Kernels & Files Datastore

The Files Datastore lets you store plain files to be used as VM kernels, ramdisks or context files. The Files Datastore does not expose any special storage mechanism but a simple and secure way to use files within VM templates. There is a Files Datastore (datastore ID: 2) ready to be used in OpenNebula.

8.7.1 Requirements

There are no special requirements or software dependencies to use the Files Datastore. The recommended drivers make use of standard filesystem utils (`cp`, `ln`, `mv`, `tar`, `mkfs`...) that should be installed in your system.

8.7.2 Configuration

Most of the configuration considerations used for disk images datastores do apply to the Files Datastore (e.g. driver setup, cluster assignment, datastore management...).

The specific attributes for this datastore driver are listed in the following table, you will also need to complete with the common datastore attributes:

Attribute	Description
<code>TYPE</code>	Use <code>FILE_DS</code> to setup a Files datastore
<code>DS_MAD</code>	The DS type, use <code>fs</code> to use the file-based drivers
<code>TM_MAD</code>	Transfer drivers for the datastore, use <code>ssh</code> to transfer the files

For example, the following illustrates the creation of File Datastore.

```
> cat kernels_ds.conf
NAME = kernels
DS_MAD = fs
TM_MAD = ssh
TYPE = FILE_DS
SAFE_DIRS = /var/tmp/files

> onedatastore create kernels_ds.conf
ID: 100

> onedatastore list
  ID NAME                CLUSTER  IMAGES TYPE DS      TM
```


0 system	-	0 sys	-	dummy
1 default	-	0 img	dummy	dummy
2 files	-	0 fil	fs	ssh
100 kernels	-	0 fil	fs	ssh

The DS and TM MAD can be changed later using the `onedatastore update` command. You can check more details of the datastore by issuing the `onedatastore show` command.

8.7.3 Host Configuration

The recommended `ssh` driver for the File Datastore does not need any special configuration for the hosts. Just make sure that there is enough space under `$DATASTORE_LOCATION` to hold the VM files in the front-end and hosts.

For more details *refer to the [Filesystem Datastore guide](#)*, as the same configuration guidelines applies.

OPEN CLOUD NETWORKING SETUP

9.1 Overview

When a new Virtual Machine is launched, OpenNebula will connect its network interfaces (defined by NIC attribute) to hypervisor physical devices as defined in the Virtual Network. This will allow the VM to have access to different networks, public or private.

OpenNebula supports four different networking modes:

- *Bridged*. The Virtual Machine is directly attached to an existing bridge in the hypervisor. This mode can be configured to use security groups and network isolation.
- *VLAN*. Virtual Networks are implemented through 802.1Q VLAN tagging.
- *VXLAN*. Virtual Networks implements VLANs using the VXLAN protocol that relies on a UDP encapsulation and IP multicast.
- *Open vSwitch*. Similar to the VLAN mode but using an openvswitch instead of a Linux bridge.

When you create a new network you will need to add the attribute `VN_MAD` to the template, specifying which of the above networking modes you want to use.

Note: Security Groups are not supported by the Open vSwitch mode.

9.1.1 How Should I Read This Chapter

Before reading this chapter make sure you have read the *Open Cloud Storage* chapter.

Start by reading the common *Node Setup* section to learn how to configure your hosts, and then proceed to the specific section for the networking mode that you are interested in.

After reading this chapter you can complete your OpenNebula installation by optionally enabling an *External Authentication* or configuring *Sunstone*. Otherwise you are ready to Operate your Cloud.

9.1.2 Hypervisor Compatibility

This chapter applies only to KVM.

9.2 Node Setup

Todo

- Architect
- KVM

This guide includes specific node setup steps to enable each network mode. You **only need** to apply the corresponding section to the select mode.

9.2.1 Bridged Networking Mode

Requirements

- The OpenNebula node packages has been installed, see [the KVM node installation section](#) for more details.
- By default, network isolation is provided through `ebtables`, this package needs to be installed in the nodes.

Configuration

- Create a linux bridge for each network that would be expose to Virtual Machines. Use the same name in all the nodes.
- Add the physical network interface to the bridge.

For example, a node with two networks one for public IP addresses (attached to `eth0`) and another one for private traffic (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
```

Note: It is recommended that this configuration is made persistent. Please refer to the network configuration guide of your system to do so.

9.2.2 VLAN Networking Mode

Requirements

- The OpenNebula node packages has been installed, see [the KVM node installation section](#) for more details.
- The `8021q` module must be loaded in the kernel.
- A network switch capable of forwarding VLAN tagged traffic. The physical switch ports should be VLAN trunks.

Configuration

No additional configuration is needed.

9.2.3 VXLAN Networking Mode

Requirements

- The OpenNebula node packages has been installed, see *the KVM node installation section* for more details.
- The node must run a Linux kernel (>3.7.0) that natively supports the VXLAN protocol and the associated iproute2 package.
- When all the nodes are connected to the same broadcasting domain be sure that the multicast traffic is not filtered by any iptable rule in the nodes. Note that if the multicast traffic needs to traverse routers a multicast protocol like IGMP needs to be configured in your network.

Configuration

No additional configuration is needed.

9.2.4 Open vSwitch Networking Mode

Requirements

- The OpenNebula node packages has been installed, see *the KVM node installation section* for more details.
- You need to install Open vSwitch on each node. Please refer to the Open vSwitch documentation to do so.

For example, a node that forwards Virtual Networks traffic through the `enp0s8` network interface should create an openvswitch like:

```
# ovs-vsctl show
c61ba96f-fc11-4db9-9636-408e763f529e
    Bridge "ovsbr0"
        Port "ovsbr0"
            Interface "ovsbr0"
                type: internal
        Port "enp0s8"
            Interface "enp0s8"
```

Configuration

- Create a openvswitch for each network that would be expose to Virtual Machines. Use the same name in all the nodes.
- Add the physical network interface to the openvswitch.

Note: It is recommended that this configuration is made persistent. Please refer to the network configuration guide of your system to do so.

9.3 Bridged Networking

This guide describes how to deploy Bridged networks. In this mode virtual machine traffic is directly bridged through an existing Linux bridge in the nodes. Bridged networks can operate on three different modes depending on the

additional traffic filtering made by OpenNebula:

- **Dummy**, no filtering is made
- **Security Group**, iptables rules are installed to implement security groups rules.
- **eatables VLAN**, same as above plus additional eatables rules to isolate (L2) each Virtual Networks.

9.3.1 Considerations & Limitations

The following needs to be considered regarding traffic isolation:

- In the **Dummy** and **Security Group** modes you can add tagged network interfaces to achieve network isolation. This is the **recommended** deployment strategy in production environments in this mode.
- The **eatables VLAN** mode is targeted to small environments without proper hardware support to implement VLANS. Note that it is limited to /24 networks, and that IP addresses cannot overlap between Virtual Networks. This mode is only recommended for testing purposes.

9.3.2 OpenNebula Configuration

No specific configuration is required for bridged networking

9.3.3 Defining a Bridged Network

To create a 802.1Q network include the following information:

Attribute	Value	Mandatory
VN_MAD	<ul style="list-style-type: none"> • <code>dummy</code> for the Dummy Bridged mode • <code>fw</code> for Bridged with Security Groups • <code>eatables</code> for Bridged with eatables isolation 	YES
BRIDGE	Name of the linux bridge in the nodes	YES

The following example defines Bridged network using the Security Groups mode:

```
NAME      = "bridged_net"
VN_MAD    = "fw"
BRIDGE    = vbr1
...
```

9.3.4 eatables VLAN Mode: default rules

This section lists the eatables rules that are created, in case you need to debug your setup

```
# Drop packets that don't match the network's MAC Address
-s ! <mac_address>/ff:ff:ff:ff:ff:ff:0 -o <tap_device> -j DROP
# Prevent MAC spoofing
-s ! <mac_address> -i <tap_device> -j DROP
```

9.4 802.1Q VLAN Networks

This guide describes how to enable Network isolation provided through host-managed VLANs. This driver will create a bridge for each OpenNebula Virtual Network and attach an VLAN tagged network interface to the bridge. This mechanism is compliant with IEEE 802.1Q.

The VLAN id will be the same for every interface in a given network, automatically computed by OpenNebula. It may also be forced by specifying an `VLAN_ID` parameter in the Virtual Network template.

9.4.1 OpenNebula Configuration

The `VLAN_ID` is calculated according to this configuration option of `oned.conf`:

```
# VLAN_IDS: VLAN ID pool for the automatic VLAN_ID assigment. This pool
# is for 802.1Q networks (Open vSwitch and 802.1Q drivers). The driver
# will try first to allocate VLAN_IDS[START] + VNET_ID
#   start: First VLAN_ID to use
#   reserved: Comma separated list of VLAN_IDs

VLAN_IDS = [
    START    = "2",
    RESERVED = "0, 1, 4095"
]
```

By modifying that parameter you can reserve some VLANs so they aren't assigned to a Virtual Network. You can also define the first `VLAN_ID`. When a new isolated network is created, OpenNebula will find a free `VLAN_ID` from the VLAN pool. This pool is global, and it's also shared with the *Open vSwitch* network mode.

9.4.2 Defining a 802.1Q Network

To create a 802.1Q network include the following information:

Attribute	Value	Mandatory
VN_MAD	802.1Q	YES
PHYDEV	Name of the physical network device that will be attached to the bridge.	YES
BRIDGE	Name of the linux bridge, defaults to <code>onebr<net_id></code> or <code>onebr.<vlan_id></code>	NO
VLAN_ID	The VLAN ID, will be generated if not defined	NO
MTU	The MTU for the tagged interface and bridge	NO

The following example defines a 802.1Q network

```
NAME      = "hmnet"
VN_MAD    = "802.1Q"
PHYDEV    = "eth0"
VLAN_ID   = 50           # optional
BRIDGE    = "brhm"      # optional
```

In this scenario, the driver will check for the existence of the `brhm` bridge. If it doesn't exist it will be created. `eth0` will be tagged (`eth0.50`) and attached to `brhm` (unless it's already attached).

9.5 VXLAN Networks

This guide describes how to enable Network isolation provided through the VXLAN encapsulation protocol. This driver will create a bridge for each OpenNebula Virtual Network and attach a VXLAN tagged network interface to the bridge.

The VLAN id will be the same for every interface in a given network, calculated automatically by OpenNebula. It may also be forced by specifying an `VLAN_ID` parameter in the Virtual Network template.

Additionally each VLAN has associated a multicast address to encapsulate L2 broadcast and multicast traffic. This address is assigned by default to the 239.0.0.0/8 range as defined by RFC 2365 (Administratively Scoped IP Multicast). In particular the multicast address is obtained by adding the `VLAN_ID` to the 239.0.0.0/8 base address.

9.5.1 Considerations & Limitations

This driver works with the default UDP server port 8472.

VXLAN traffic is forwarded to a physical device, this device can be set to be a VLAN tagged interface, but in that case you must make sure that the tagged interface is manually created first in all the hosts.

9.5.2 OpenNebula Configuration

It is possible to specify the start VLAN ID by configuring `/etc/one/oned.conf`:

```
# VXLAN_IDS: Automatic VXLAN Network ID (VNI) assignment. This is used
# for vxlan networks.
#     start: First VNI to use

VXLAN_IDS = [
    START = "2"
]
```

The following configuration attributes can be adjusted in `/var/lib/one/remotes/vnm/OpenNebulaNetwork.conf`:

Parameter	Description
<code>vxlan_mc</code>	Base multicast address for each VLAN. The multicast address is <code>vxlan_mc + vxlan_id</code>
<code>vxlan_ttl</code>	Time To Live (TTL) should be > 1 in routed multicast networks (IGMP)

9.5.3 Defining a VXLAN Network

To create a VXLAN network include the following information:

Attribute	Value	Mandatory
VN_MAD	<code>vxlan</code>	YES
PHYDEV	Name of the physical network device that will be attached to the bridge.	YES
BRIDGE	Name of the linux bridge, defaults to <code>onebr<net_id></code> or <code>onebr.<vlan_id></code>	NO
VLAN_ID	The VLAN ID, will be generated if not defined	NO
MTU	The MTU for the tagged interface and bridge	NO

The following example defines a VXLAN network

```
NAME      = "vxlan_net"
VN_MAD    = "vxlan"
PHYDEV    = "eth0"
```

```
VLAN_ID = 50          # optional
BRIDGE  = "vxlan50" # optional
...
```

In this scenario, the driver will check for the existence of the `vxlan50` bridge. If it doesn't exist it will be created. `eth0` will be tagged (`eth0.50`) and attached to `vxlan50` (unless it's already attached). Note that `eth0` can be a 802.1Q tagged interface if you want to isolate the OpenNebula VXLAN traffic.

9.6 Open vSwitch Networks

This guide describes how to use the [Open vSwitch](#) network drives. They provide network isolation using VLANs by tagging ports and basic network filtering using OpenFlow. Other traffic attributes that may be configured through Open vSwitch are not modified.

The VLAN id will be the same for every interface in a given network, calculated automatically by OpenNebula. It may also be forced by specifying an `VLAN_ID` parameter in the Virtual Network template.

Warning: This driver is not compatible with Security Groups.

9.6.1 OpenNebula Configuration

The `VLAN_ID` is calculated according to this configuration option of `oned.conf`:

```
# VLAN_IDS: VLAN ID pool for the automatic VLAN_ID assigment. This pool
# is for 802.1Q networks (Open vSwitch and 802.1Q drivers). The driver
# will try first to allocate VLAN_IDS[START] + VNET_ID
#   start: First VLAN_ID to use
#   reserved: Comma separated list of VLAN_IDS

VLAN_IDS = [
    START    = "2",
    RESERVED = "0, 1, 4095"
]
```

By modifying that parameter you can reserve some VLANs so they aren't assigned to a Virtual Network. You can also define the first `VLAN_ID`. When a new isolated network is created, OpenNebula will find a free `VLAN_ID` from the VLAN pool. This pool is global, and it's also shared with the [802.1Q VLAN](#) network mode.

The following configuration attributes can be adjusted in `/var/lib/one/remotes/vnm/OpenNebulaNetwork.conf`:

Parameter	Description
<code>arp_cache_poisoning</code>	Enable ARP Cache Poisoning Prevention Rules.

Note: Remember to run `onehost sync` to deploy the file to all the nodes.

9.6.2 Defining an Open vSwitch Network

To create a VXLAN network include the following information:

Attribute	Value	Mandatory
VN_MAD	ovswitch	YES
BRIDGE	Name of the Open vSwitch switch to use	YES
VLAN_ID	The VLAN ID, will be generated if not defined	NO

The following example defines a VXLAN network

```
NAME      = "ovswitch_net"
VN_MAD    = "ovswitch"
BRIDGE    = vbr1
VLAN_ID   = 50 # optional
...
```

Multiple VLANs (VLAN trunking)

VLAN trunking is also supported by adding the following tag to the NIC element in the VM template or to the virtual network template:

- `VLAN_TAGGED_ID`: Specify a range of VLANs to tag, for example: 1, 10, 30, 32.

9.6.3 OpenFlow Rules

This section lists the default openflow rules installed in the open vswitch.

Mac-spoofing

These rules prevent any traffic to come out of the port the MAC address has changed.

```
in_port=<PORT>,dl_src=<MAC>,priority=40000,actions=normal
in_port=<PORT>,priority=39000,actions=normal
```

IP hijacking

These rules prevent any traffic to come out of the port for IPv4 IP's not configured for a VM

```
in_port=<PORT>,arp,dl_src=<MAC>,priority=45000,actions=drop
in_port=<PORT>,arp,dl_src=<MAC>,nw_src=<IP>,priority=46000,actions=normal
```

Black ports (one rule per port)

```
tcp,dl_dst=<MAC>,tp_dst=<PORT>,actions=drop
```

ICMP Drop

```
icmp,dl_dst=<MAC>,actions=drop
```

REFERENCES

10.1 Overview

This Chapter covers references that apply to the configuration of OpenNebula to interact smoothly and efficiently with other datacenter components, as well as information on where are the log files, the database tool, and all the configuration parameters of the OpenNebula core daemon.

10.1.1 How Should I Read This Chapter

The *oned.conf* file is the main OpenNebula configuration file, and it is essential to tweak the performance and behavior of your OpenNebula installation. The section *Large Deployments* contains more helpful pointers to tune the OpenNebula performance.

Read *Logging and Debugging* for a complete reference on how to use log files, and how to adjust the verbosity and log subsystem.

For database maintenance operations, use the *command line tool onedb*. It can be used to get information from an OpenNebula database, upgrade it, or fix inconsistency problems.

10.1.2 Hypervisor Compatibility

This chapter applies to all the hypervisors.

10.2 ONED Configuration

The OpenNebula daemon `oned` manages the cluster nodes, virtual networks, virtual machines, users, groups and storage datastores. The configuration file for the daemon is called `oned.conf` and it is placed inside the `/etc/one` directory. In this reference document we describe all the format and options that can be specified in `oned.conf`.

10.2.1 Daemon Configuration Attributes

- `MANAGER_TIMER` : Time in seconds the core uses to evaluate periodical functions. `MONITORING_INTERVAL` cannot have a smaller value than `MANAGER_TIMER`.
- `MONITORING_INTERVAL` : Time in seconds between each monitorization.
- `MONITORING_THREADS` : Max. number of threads used to process monitor messages
- `HOST_PER_INTERVAL`: Number of hosts monitored in each interval.

- `HOST_MONITORING_EXPIRATION_TIME`: Time, in seconds, to expire monitoring information. Use 0 to disable HOST monitoring recording.
- `VM_INDIVIDUAL_MONITORING`: VM monitoring information is obtained along with the host information. For some custom monitor drivers you may need activate the individual VM monitoring process.
- `VM_PER_INTERVAL`: Number of VMs monitored in each interval.
- `VM_MONITORING_EXPIRATION_TIME`: Time, in seconds, to expire monitoring information. Use 0 to disable VM monitoring recording.
- `SCRIPTS_REMOTE_DIR`: Remote path to store the monitoring and VM management script.
- `PORT` : Port where oned will listen for xml-rpc calls.
- `LISTEN_ADDRESS`: Host IP to listen on for xmlrpc calls (default: all IPs).
- `DB` : Vector of configuration attributes for the database back-end.
 - `backend` : Set to `sqlite` or `mysql`. Please visit the [MySQL configuration guide](#) for more information.
 - `server` (MySQL only): Host name or an IP address for the MySQL server.
 - `user` (MySQL only): MySQL user's login ID.
 - `passwd` (MySQL only): MySQL user's password.
 - `db_name` (MySQL only): MySQL database name.
- `VNC_PORTS` : VNC port pool for automatic VNC port assignment, if possible the port will be set to `START + VMID`. Refer to the VM template reference for further information:
 - `start`: first port to assign
 - `reserved`: comma separated list of reserved ports
- `VM_SUBMIT_ON_HOLD` : Forces VMs to be created on hold state instead of pending. Values: YES or NO.
- `LOG` : Configure the logging system
 - `SYSTEM` : Can be either `file` (default), `syslog` or `std`
 - `DEBUG_LEVEL` : Sets the level of verbosity of the log messages. Possible values are:

DEBUG_LEVEL	Meaning
0	ERROR
1	WARNING
2	INFO
3	DEBUG

Example of this section:

```

#*****
# Daemon configuration attributes
#*****

LOG = [
    SYSTEM      = "file",
    DEBUG_LEVEL = 3
]

#MANAGER_TIMER = 15

MONITORING_INTERVAL = 60
MONITORING_THREADS  = 50

```

```

#HOST_PER_INTERVAL          = 15
#HOST_MONITORING_EXPIRATION_TIME = 43200

#VM_INDIVIDUAL_MONITORING   = "no"
#VM_PER_INTERVAL           = 5
#VM_MONITORING_EXPIRATION_TIME = 14400

SCRIPTS_REMOTE_DIR=/var/tmp/one

PORT = 2633

LISTEN_ADDRESS = "0.0.0.0"

DB = [ BACKEND = "sqlite" ]

# Sample configuration for MySQL
# DB = [ BACKEND = "mysql",
#         SERVER = "localhost",
#         PORT   = 0,
#         USER   = "oneadmin",
#         PASSWD = "oneadmin",
#         DB_NAME = "openebula" ]

VNC_PORTS = [
    START = 5900
#     RESERVED = "6800, 6801, 9869"
]

#VM_SUBMIT_ON_HOLD = "NO"

```

10.2.2 Federation Configuration Attributes

Control the federation capabilities of oned. Operation in a federated setup requires a special DB configuration.

- **FEDERATION** : Federation attributes.
 - **MODE** : Operation mode of this oned.
 - * **STANDALONE**: not federated. This is the default operational mode
 - * **MASTER**: this oned is the master zone of the federation
 - * **SLAVE**: this oned is a slave zone
- **ZONE_ID** : The zone ID as returned by onezone command.
- **MASTER_ONED** : The xml-rpc endpoint of the master oned, e.g. <http://master.one.org:2633/RPC2>

```

#*****
# Federation configuration attributes
#*****

FEDERATION = [
    MODE = "STANDALONE",
    ZONE_ID = 0,
    MASTER_ONED = ""
]

```

10.2.3 Default Showback Cost

The following attributes define the default cost for Virtual Machines that don't have a CPU, MEMORY or DISK cost. This is used by the oneshowback calculate method.

```

#*****
# Default showback cost
#*****

DEFAULT_COST = [
    CPU_COST      = 0,
    MEMORY_COST   = 0,
    DISK_COST     = 0
]

```

10.2.4 XML-RPC Server Configuration

- **MAX_CONN**: Maximum number of simultaneous TCP connections the server will maintain
- **MAX_CONN_BACKLOG**: Maximum number of TCP connections the operating system will accept on the server's behalf without the server accepting them from the operating system
- **KEEPALIVE_TIMEOUT**: Maximum time in seconds that the server allows a connection to be open between RPCs
- **KEEPALIVE_MAX_CONN**: Maximum number of RPCs that the server will execute on a single connection
- **TIMEOUT**: Maximum time in seconds the server will wait for the client to do anything while processing an RPC. This timeout will be also used when proxy calls to the master in a federation.
- **RPC_LOG**: Create a separated log file for xml-rpc requests, in `/var/log/one/one_xmlrpc.log`.
- **MESSAGE_SIZE**: Buffer size in bytes for XML-RPC responses.
- **LOG_CALL_FORMAT**: Format string to log XML-RPC calls. Interpreted strings:
 - %i - request id
 - %m - method name
 - %u - user id
 - %U - user name
 - %l - param list
 - %p - user password
 - %g - group id
 - %G - group name
 - %a - auth token
 - %% - %

```

#*****
# XML-RPC server configuration
#*****

#MAX_CONN          = 15
#MAX_CONN_BACKLOG = 15

```

```
#KEEPALIVE_TIMEOUT = 15
#KEEPALIVE_MAX_CONN = 30
#TIMEOUT = 15
#RPC_LOG = NO
#MESSAGE_SIZE = 1073741824
#LOG_CALL_FORMAT = "Req:%i UID:%u %m invoked %l"
```

Warning: This functionality is only available when compiled with xmlrpc-c libraires >= 1.32. Currently only the packages distributed by OpenNebula are linked with this library.

10.2.5 Virtual Networks

- **NETWORK_SIZE:** Here you can define the default size for the virtual networks
- **MAC_PREFIX:** Default MAC prefix to be used to create the auto-generated MAC addresses is defined here (this can be overwritten by the Virtual Network template)
- **VLAN_IDS:** VLAN ID pool for the automatic VLAN_ID assignment. This pool is for 802.1Q networks (Open vSwitch and 802.1Q drivers). The driver will try first to allocate VLAN_IDS[START] + VNET_ID
 - start: First VLAN_ID to use
 - reserved: Comma separated list of VLAN_IDS
- **VXLAN_IDS:** Automatic VXLAN Network ID (VNI) assignment. This is used for vxlan networks.
 - start: First VNI to use
 - _____

Note: reserved is not supported by this pool

Sample configuration:

```
*****
# Physical Networks configuration
*****

NETWORK_SIZE = 254

MAC_PREFIX = "02:00"

VLAN_IDS = [
    START = "2",
    RESERVED = "0, 1, 4095"
]

VXLAN_IDS = [
    START = "2"
]

```

10.2.6 Datastores

The *Storage Subsystem* allows users to set up images, which can be operative systems or data, to be used in Virtual Machines easily. These images can be used by several Virtual Machines simultaneously, and also shared with other

users.

Here you can configure the default values for the Datastores and Image templates. You have more information about the templates syntax here.

- **DATASTORE_LOCATION**: Path for Datastores. It IS the same for all the hosts and front-end. It defaults to `/var/lib/one/datastores` (in self-contained mode defaults to `$ONE_LOCATION/var/datastores`). Each datastore has its own directory (called **BASE_PATH**) in the form: `$DATASTORE_LOCATION/<datastore_id>`. You can symlink this directory to any other path if needed. **BASE_PATH** is generated from this attribute each time oned is started.
- **DATASTORE_CAPACITY_CHECK**: Checks that there is enough capacity before creating a new image. Defaults to Yes
- **DEFAULT_IMAGE_TYPE** : Default value for **TYPE** field when it is omitted in a template. Values accepted are:
 - **OS**: Image file holding an operating system
 - **CDROM**: Image file holding a CDROM
 - **DATABLOCK**: Image file holding a datablock, created as an empty block
- **DEFAULT_DEVICE_PREFIX** : Default value for **DEV_PREFIX** field when it is omitted in a template. The missing **DEV_PREFIX** attribute is filled when Images are created, so changing this prefix won't affect existing Images. It can be set to:

Prefix	Device type
hd	IDE
sd	SCSI
vd	KVM virtual disk

- **DEFAULT_CDROM_DEVICE_PREFIX**: Same as above but for CDROM devices.

More information on the image repository can be found in the [Managing Virtual Machine Images](#) guide.

Sample configuration:

```
#*****
# Image Repository Configuration
#*****
#DATASTORE_LOCATION = /var/lib/one/datastores

DATASTORE_CAPACITY_CHECK = "yes"

DEFAULT_IMAGE_TYPE      = "OS"
DEFAULT_DEVICE_PREFIX   = "hd"

DEFAULT_CDROM_DEVICE_PREFIX = "hd"
```

10.2.7 Information Collector

This driver CANNOT BE ASSIGNED TO A HOST, and needs to be used with KVM drivers. Options that can be set:

- **-a**: Address to bind the collectd socket (default 0.0.0.0)
- **-p**: UDP port to listen for monitor information (default 4124)
- **-f**: Interval in seconds to flush collected information (default 5)
- **-t**: Number of threads for the server (default 50)

- **-i**: Time in seconds of the monitorization push cycle. This parameter must be smaller than MONITORING_INTERVAL, otherwise push monitorization will not be effective.

Sample configuration:

```
IM_MAD = [
    name       = "collectd",
    executable = "collectd",
    arguments  = "-p 4124 -f 5 -t 50 -i 20" ]
```

10.2.8 Information Drivers

The information drivers are used to gather information from the cluster nodes, and they depend on the virtualizer you are using. You can define more than one information manager but make sure it has different names. To define it, the following needs to be set:

- **name**: name for this information driver.
- **executable**: path of the information driver executable, can be an absolute path or relative to `/usr/lib/one/mads/`
- **arguments**: for the driver executable, usually a probe configuration file, can be an absolute path or relative to `/etc/one/`.

For more information on configuring the information and monitoring system and hints to extend it please check the information driver configuration guide.

Sample configuration:

```
#-----
#  KVM UDP-push Information Driver Manager Configuration
#  -r number of retries when monitoring a host
#  -t number of threads, i.e. number of hosts monitored at the same time
#-----
IM_MAD = [
    NAME           = "kvm",
    SUNSTONE_NAME  = "KVM",
    EXECUTABLE     = "one_im_ssh",
    ARGUMENTS     = "-r 3 -t 15 kvm" ]
#-----
```

10.2.9 Virtualization Drivers

The virtualization drivers are used to create, control and monitor VMs on the hosts. You can define more than one virtualization driver (e.g. you have different virtualizers in several hosts) but make sure they have different names. To define it, the following needs to be set:

- **name**: name of the virtualization driver.
- **executable**: path of the virtualization driver executable, can be an absolute path or relative to `/usr/lib/one/mads/`
- **arguments**: for the driver executable
- **type**: driver type, supported drivers: xen, kvm or xml
- **default**: default values and configuration parameters for the driver, can be an absolute path or relative to `/etc/one/`

- **keep_snapshots**: do not remove snapshots on power on/off cycles and live migrations if the hypervisor supports that.
- **imported_vms_actions** : comma-separated list of actions supported for imported vms. The available actions are:
 - migrate
 - live-migrate
 - terminate
 - terminate-hard
 - undeploy
 - undeploy-hard
 - hold
 - release
 - stop
 - suspend
 - resume
 - delete
 - delete-recreate
 - reboot
 - reboot-hard
 - resched
 - unresched
 - poweroff
 - poweroff-hard
 - disk-attach
 - disk-detach
 - nic-attach
 - nic-detach
 - snap-create
 - snap-delete

For more information on configuring and setting up the Virtual Machine Manager Driver please check the section that suits you:

- *KVM Driver*
- *vCenter Driver*

Sample configuration:

```
#-----  
# Virtualization Driver Configuration  
#-----
```

```

VM_MAD = [
    NAME           = "kvm",
    SUNSTONE_NAME  = "KVM",
    EXECUTABLE     = "one_vmm_exec",
    ARGUMENTS      = "-t 15 -r 0 kvm",
    DEFAULT        = "vmm_exec/vmm_exec_kvm.conf",
    TYPE           = "kvm",
    KEEP_SNAPSHOTS = "no",
    IMPORTED_VMS_ACTIONS = "terminate, terminate-hard, hold, release, suspend,
        resume, delete, reboot, reboot-hard, resched, unresched, disk-attach,
        disk-detach, nic-attach, nic-detach, snap-create, snap-delete"
]

```

10.2.10 Transfer Driver

The transfer drivers are used to transfer, clone, remove and create VM images. The default TM_MAD driver includes plugins for all supported storage modes. You may need to modify the TM_MAD to add custom plugins.

- **executable:** path of the transfer driver executable, can be an absolute path or relative to `/usr/lib/one/mads/`
- **arguments:** for the driver executable:
 - **-t:** number of threads, i.e. number of transfers made at the same time
 - **-d:** list of transfer drivers separated by commas, if not defined all the drivers available will be enabled

For more information on configuring different storage alternatives *please check the storage configuration guide*.

Sample configuration:

```

#-----
# Transfer Manager Driver Configuration
#-----

TM_MAD = [
    EXECUTABLE = "one_tm",
    ARGUMENTS  = "-t 15 -d dummy,lvm,shared,fs_lvm,qcow2,ssh,ceph,dev,vcenter,iscsi_
↳ libvirt"
]

```

The configuration for each driver is defined in the TM_MAD_CONF section. These values are used when creating a new datastore and should not be modified since they define the datastore behavior.

- **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*.
- **ds_migrate**: set if system datastore migrations are allowed for this TM. Only useful for system datastore TMs.

Sample configuration:

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

TM_MAD_CONF = [
  name       = "shared",
  ln_target  = "NONE",
  clone_target = "SYSTEM",
  shared     = "yes",
  ds_migrate = "yes"
]
```

10.2.11 Datastore Driver

The Datastore Driver defines a set of scripts to manage the storage backend.

- **executable**: path of the transfer driver executable, can be an absolute path or relative to `/usr/lib/one/mads/`
- **arguments**: for the driver executable
 - **-t** number of threads, i.e. number of repo operations at the same time
 - **-d** datastore mads separated by commas
 - **-s** system datastore tm drivers, used to monitor shared system ds

Sample configuration:

```
DATASTORE_MAD = [
  EXECUTABLE = "one_datastore",
  ARGUMENTS  = "-t 15 -d dummy,fs,lvm,ceph,dev,iscsi_libvirt,vcenter -s_
↪shared,ssh,ceph,fs_lvm"
]
```

For more information on this Driver and how to customize it, please visit *its reference guide*.

10.2.12 Marketplace Driver Configuration

Drivers to manage different marketplaces, specialized for the storage back-end

- **executable**: path of the transfer driver executable, can be an absolute path or relative to `/usr/lib/one/mads/`
- **arguments** : for the driver executable
 - **-t** number of threads, i.e. number of repo operations at the same time
 - **-m** marketplace mads separated by commas

Sample configuration:

```
MARKET_MAD = [
    EXECUTABLE = "one_market",
    ARGUMENTS = "-t 15 -m http,s3,one"
]
```

10.2.13 Hook System

Hooks in OpenNebula are programs (usually scripts) which execution is triggered by a change in state in Virtual Machines or Hosts. The hooks can be executed either locally or remotely in the node where the VM or Host is running. To configure the Hook System the following needs to be set in the OpenNebula configuration file:

- **executable**: path of the hook driver executable, can be an absolute path or relative to `/usr/lib/one/mads/`
- **arguments** : for the driver executable, can be an absolute path or relative to `/etc/one/`

Sample configuration:

```
HM_MAD = [
    executable = "one_hm" ]
```

Virtual Machine Hooks (VM_HOOK) defined by:

- **name**: for the hook, useful to track the hook (OPTIONAL).
- **on**: when the hook should be executed,
 - **CREATE**, when the VM is created (`onevm create`)
 - **PROLOG**, when the VM is in the prolog state
 - **RUNNING**, after the VM is successfully booted
 - **UNKNOWN**, when the VM is in the unknown state
 - **SHUTDOWN**, after the VM is shutdown
 - **STOP**, after the VM is stopped (including VM image transfers)
 - **DONE**, after the VM is deleted or shutdown
 - **CUSTOM**, user defined specific STATE and LCM_STATE combination of states to trigger the hook
- **command**: path can be absolute or relative to `/usr/share/one/hooks`
- **arguments**: for the hook. You can access to VM information with \$
 - **\$ID**, the ID of the virtual machine
 - **\$TEMPLATE**, the VM template in xml and base64 encoded multiple
 - **PREV_STATE**, the previous STATE of the Virtual Machine
 - **PREV_LCM_STATE**, the previous LCM STATE of the Virtual Machine
- **remote**: values,
 - **YES**, The hook is executed in the host where the VM was allocated
 - **NO**, The hook is executed in the OpenNebula server (default)

Host Hooks (HOST_HOOK) defined by:

- **name:** for the hook, useful to track the hook (OPTIONAL)
- **on:** when the hook should be executed,
 - **CREATE**, when the Host is created (onehost create)
 - **ERROR**, when the Host enters the error state
 - **DISABLE**, when the Host is disabled
- **command:** path can be absolute or relative to /usr/share/one/hooks
- **arguments:** for the hook. You can use the following Host information:
 - **\$ID**, the ID of the host
 - **\$TEMPLATE**, the Host template in xml and base64 encoded
- **remote:** values,
 - **YES**, The hook is executed in the host
 - **NO**, The hook is executed in the OpenNebula server (default)

Sample configuration:

```
VM_HOOK = [
  name      = "advanced_hook",
  on        = "CUSTOM",
  state     = "ACTIVE",
  lcm_state = "BOOT_UNKNOWN",
  command   = "log.rb",
  arguments = "$ID $PREV_STATE $PREV_LCM_STATE" ]
```

10.2.14 Auth Manager Configuration

- **AUTH_MAD:** The *driver* that will be used to authenticate and authorize OpenNebula requests. If not defined OpenNebula will use the built-in auth policies
 - **executable:** path of the auth driver executable, can be an absolute path or relative to /usr/lib/one/mads/
 - **authn:** list of authentication modules separated by commas, if not defined all the modules available will be enabled
 - **authz:** list of authentication modules separated by commas
- **SESSION_EXPIRATION_TIME:** Time in seconds to keep an authenticated token as valid. During this time, the driver is not used. Use 0 to disable session caching
- **ENABLE_OTHER_PERMISSIONS:** Whether or not to enable the permissions for 'other'. Users in the oneadmin group will still be able to change these permissions. Values: YES or NO
- **DEFAULT_UMASK:** Similar to Unix umask, sets the default resources permissions. Its format must be 3 octal digits. For example a umask of 137 will set the new object's permissions to 640 um- u-- ---

Sample configuration:

```
AUTH_MAD = [
  executable = "one_auth_mad",
  authn      = "ssh,x509,ldap,server_cipher,server_x509"
]
```

```
SESSION_EXPIRATION_TIME = 900

#ENABLE_OTHER_PERMISSIONS = "YES"

DEFAULT_UMASK = 177
```

The DEFAULT_AUTH can be used to point to the desired default authentication driver, for example ldap:

```
DEFAULT_AUTH = "ldap"
```

10.2.15 Restricted Attributes Configuration

Users outside the oneadmin group won't be able to instantiate templates created by users outside the "oneadmin" group that include the attributes restricted by:

- **VM_RESTRICTED_ATTR**: Virtual Machine attribute to be restricted for users outside the "oneadmin" group
- **IMAGE_RESTRICTED_ATTR**: Image attribute to be restricted for users outside the "oneadmin" group
- **VNET_RESTRICTED_ATTR**: Virtual Network attribute to be restricted for users outside the "oneadmin" group when updating a reservation. These attributes are not considered for regular VNET creation.

If the VM template has been created by admins in the "oneadmin" group, then users outside the "oneadmin" group **can** instantiate these templates.

Sample configuration:

```
VM_RESTRICTED_ATTR = "CONTEXT/FILES"
VM_RESTRICTED_ATTR = "NIC/MAC"
VM_RESTRICTED_ATTR = "NIC/VLAN_ID"
VM_RESTRICTED_ATTR = "NIC/BRIDGE"
VM_RESTRICTED_ATTR = "NIC_DEFAULT/MAC"
VM_RESTRICTED_ATTR = "NIC_DEFAULT/VLAN_ID"
VM_RESTRICTED_ATTR = "NIC_DEFAULT/BRIDGE"
VM_RESTRICTED_ATTR = "DISK/TOTAL_BYTES_SEC"
VM_RESTRICTED_ATTR = "DISK/READ_BYTES_SEC"
VM_RESTRICTED_ATTR = "DISK/WRITE_BYTES_SEC"
VM_RESTRICTED_ATTR = "DISK/TOTAL_IOPS_SEC"
VM_RESTRICTED_ATTR = "DISK/READ_IOPS_SEC"
VM_RESTRICTED_ATTR = "DISK/WRITE_IOPS_SEC"
#VM_RESTRICTED_ATTR = "DISK/SIZE"
VM_RESTRICTED_ATTR = "DISK/ORIGINAL_SIZE"
VM_RESTRICTED_ATTR = "CPU_COST"
VM_RESTRICTED_ATTR = "MEMORY_COST"
VM_RESTRICTED_ATTR = "DISK_COST"
VM_RESTRICTED_ATTR = "PCI"
VM_RESTRICTED_ATTR = "USER_INPUTS"

#VM_RESTRICTED_ATTR = "RANK"
#VM_RESTRICTED_ATTR = "SCHED_RANK"
#VM_RESTRICTED_ATTR = "REQUIREMENTS"
#VM_RESTRICTED_ATTR = "SCHED_REQUIREMENTS"

IMAGE_RESTRICTED_ATTR = "SOURCE"

VNET_RESTRICTED_ATTR = "VN_MAD"
VNET_RESTRICTED_ATTR = "PHYDEV"
```

```
VNET_RESTRICTED_ATTR = "VLAN_ID"
VNET_RESTRICTED_ATTR = "BRIDGE"

VNET_RESTRICTED_ATTR = "AR/VN_MAD"
VNET_RESTRICTED_ATTR = "AR/PHYDEV"
VNET_RESTRICTED_ATTR = "AR/VLAN_ID"
VNET_RESTRICTED_ATTR = "AR/BRIDGE"
```

OpenNebula evaluates these attributes:

- on VM template instantiate (onetemplate instantiate)
- on VM create (onevm create)
- on VM attach nic (onevm nic-attach) (for example to forbid users to use NIC/MAC)

10.2.16 Inherited Attributes Configuration

The following attributes will be copied from the resource template to the instantiated VMs. More than one attribute can be defined.

- INHERIT_IMAGE_ATTR: Attribute to be copied from the Image template to each VM/DISK.
- INHERIT_DATASTORE_ATTR: Attribute to be copied from the Datastore template to each VM/DISK.
- INHERIT_VNET_ATTR: Attribute to be copied from the Network template to each VM/NIC.

Sample configuration:

```
#INHERIT_IMAGE_ATTR      = "EXAMPLE"
#INHERIT_IMAGE_ATTR      = "SECOND_EXAMPLE"
#INHERIT_DATASTORE_ATTR  = "COLOR"
#INHERIT_VNET_ATTR       = "BANDWIDTH_THROTTLING"

INHERIT_DATASTORE_ATTR  = "CEPH_HOST"
INHERIT_DATASTORE_ATTR  = "CEPH_SECRET"
INHERIT_DATASTORE_ATTR  = "CEPH_USER"
INHERIT_DATASTORE_ATTR  = "CEPH_CONF"
INHERIT_DATASTORE_ATTR  = "POOL_NAME"

INHERIT_DATASTORE_ATTR  = "ISCSI_USER"
INHERIT_DATASTORE_ATTR  = "ISCSI_USAGE"
INHERIT_DATASTORE_ATTR  = "ISCSI_HOST"

INHERIT_IMAGE_ATTR      = "ISCSI_USER"
INHERIT_IMAGE_ATTR      = "ISCSI_USAGE"
INHERIT_IMAGE_ATTR      = "ISCSI_HOST"
INHERIT_IMAGE_ATTR      = "ISCSI_IQN"

INHERIT_DATASTORE_ATTR  = "GLUSTER_HOST"
INHERIT_DATASTORE_ATTR  = "GLUSTER_VOLUME"

INHERIT_DATASTORE_ATTR  = "DISK_TYPE"
INHERIT_DATASTORE_ATTR  = "ADAPTER_TYPE"

INHERIT_IMAGE_ATTR      = "DISK_TYPE"
INHERIT_IMAGE_ATTR      = "ADAPTER_TYPE"

INHERIT_VNET_ATTR       = "VLAN_TAGGED_ID"
```

```

INHERIT_VNET_ATTR = "FILTER_IP_SPOOFING"
INHERIT_VNET_ATTR = "FILTER_MAC_SPOOFING"
INHERIT_VNET_ATTR = "MTU"

```

10.2.17 OneGate Configuration

- **ONEGATE_ENDPOINT**: Endpoint where OneGate will be listening. Optional.

Sample configuration:

```
ONEGATE_ENDPOINT = "http://192.168.0.5:5030"
```

10.3 Logging & Debugging

OpenNebula provides logs for many resources. It supports three logging systems: file based logging systems, syslog logging and logging to standard error stream.

In the case of file based logging, OpenNebula keeps separate log files for each active component, all of them stored in `/var/log/one`. To help users and administrators find and solve problems, they can also access some of the error messages from the CLI or the *Sunstone GUI*.

With syslog or standard error the logging strategy is almost identical, except that the logging message change slightly their format following syslog logging conventions, and resource information.

10.3.1 Configure the Logging System

The Logging system can be changed in `/etc/one/oned.conf`, specifically under the LOG section. Two parameters can be changed: `SYSTEM`, which is 'syslog', 'file' (default) or 'std', and the `DEBUG_LEVEL` is the logging verbosity.

For the scheduler the logging system can be changed in the exact same way. In this case the configuration is in `/etc/one/sched.conf`.

10.3.2 Log Resources

There are different log resources corresponding to different OpenNebula components:

- **ONE Daemon**: The core component of OpenNebula dumps all its logging information onto `/var/log/one/oned.log`. Its verbosity is regulated by `DEBUG_LEVEL` in `/etc/one/oned.conf`. By default the one start up scripts will backup the last oned.log file using the current time, e.g. `oned.log.20121011151807`. Alternatively, this resource can be logged to the syslog.
- **Scheduler**: All the scheduler information is collected into the `/var/log/one/sched.log` file. This resource can also be logged to the syslog.
- **Virtual Machines**: The information specific of the VM will be dumped in the log file `/var/log/one/<vmid>.log`. All VMs controlled by OpenNebula have their folder, `/var/lib/one/vms/<VID>`, or to the `syslog/stderr` if enabled. You can find the following information in it:
 - **Deployment description files** : Stored in `deployment.<EXECUTION>`, where `<EXECUTION>` is the sequence number in the execution history of the VM (deployment.0 for the first host, deployment.1 for the second and so on).

- **Transfer description files** : Stored in `transfer.<EXECUTION>.<OPERATION>`, where `<EXECUTION>` is the sequence number in the execution history of the VM, `<OPERATION>` is the stage where the script was used, e.g. `transfer.0.prolog`, `transfer.0.epilog`, or `transfer.1.cleanup`.
- **Drivers**: Each driver can have activated its **ONE_MAD_DEBUG** variable in their **RC** files. If so, error information will be dumped to `/var/log/one/name-of-the-driver-executable.log`; log information of the drivers is in `oned.log`.

10.3.3 Logging Format

The structure of an OpenNebula message for a file based logging system is the following:

```
date [Z<zone_id>][module][log_level]: message body
```

In the case of syslog it follows the standard:

```
date hostname process[pid]: [Z<zone_id>][module][log_level]: message
```

Where the `zone_id` is the ID of the zone in the federation, 0 for single zone set ups, `module` is any of the internal OpenNebula components: `VMM`, `ReM`, `TM`, etc. And the `log_level` is a single character indicating the log level: `I` for info, `D` for debug, etc.

For the syslog, OpenNebula will also log the Virtual Machine events like this:

```
date hostname process[pid]: [VM id][Z<zone_id>][module][log_level]: message
```

And similarly for the `stderr` logging, for `oned` and VM events the format are:

```
date [Z<zone_id>][module][log_level]: message
date [VM id][Z<zone_id>][module][log_level]: message
```

10.3.4 Virtual Machine Errors

Virtual Machine errors can be checked by the owner or an administrator using the `onevm show` output:

```
$ onevm show 0
VIRTUAL MACHINE 0 INFORMATION
ID                : 0
NAME              : one-0
USER              : oneadmin
GROUP             : oneadmin
STATE             : ACTIVE
LCM_STATE         : PROLOG_FAILED
START TIME        : 07/19 17:44:20
END TIME          : 07/19 17:44:31
DEPLOY ID        : -

VIRTUAL MACHINE MONITORING
NET_TX            : 0
NET_RX            : 0
USED MEMORY       : 0
USED CPU          : 0

VIRTUAL MACHINE TEMPLATE
CONTEXT=[
  FILES=/tmp/some_file,
```

```

TARGET=hdb ]
CPU=0.1
ERROR=[
  MESSAGE="Error excuting image transfer script: Error copying /tmp/some_file to /var/
↳lib/one/0/images/isofiles",
  TIMESTAMP="Tue Jul 19 17:44:31 2011" ]
MEMORY=64
NAME=one-0
VMID=0

VIRTUAL MACHINE HISTORY
SEQ      HOSTNAME ACTION          START      TIME      PTIME
  0      host01   none    07/19 17:44:31 00 00:00:00 00 00:00:00

```

Here the error tells that it could not copy a file, most probably it does not exist.

Alternatively you can also check the log files for the VM at `/var/log/one/<vmid>.log`.

Note: Check the Virtual Machines High Availability Guide, to learn how to recover a VM in fail state.

10.3.5 Host Errors

Host errors can be checked executing the `onehost show` command:

```

$ onehost show 1
HOST 1 INFORMATION
ID                : 1
NAME              : host01
STATE             : ERROR
IM_MAD           : im_kvm
VM_MAD           : vmm_kvm
TM_MAD           : tm_shared

HOST SHARES
MAX MEM          : 0
USED MEM (REAL)  : 0
USED MEM (ALLOCATED) : 0
MAX CPU         : 0
USED CPU (REAL)  : 0
USED CPU (ALLOCATED) : 0
RUNNING VMS     : 0

MONITORING INFORMATION
ERROR=[
  MESSAGE="Error monitoring host 1 : MONITOR FAILURE 1 Could not update remotes",
  TIMESTAMP="Tue Jul 19 17:17:22 2011" ]

```

The error message appears in the ERROR value of the monitoring. To get more information you can check `/var/log/one/oned.log`. For example for this error we get in the log file:

```

Tue Jul 19 17:17:22 2011 [InM][I]: Monitoring host host01 (1)
Tue Jul 19 17:17:22 2011 [InM][I]: Command execution fail: scp -r /var/lib/one/
↳remotes/. host01:/var/tmp/one
Tue Jul 19 17:17:22 2011 [InM][I]: ssh: Could not resolve hostname host01: nodename_
↳nor servname provided, or not known

```

```
Tue Jul 19 17:17:22 2011 [InM][I]: lost connection
Tue Jul 19 17:17:22 2011 [InM][I]: ExitCode: 1
Tue Jul 19 17:17:22 2011 [InM][E]: Error monitoring host 1 : MONITOR FAILURE 1 Could_
↪not update remotes
```

From the execution output we notice that the host name is not know, probably a mistake naming the host.

10.4 Onedb Tool

This section describes the `onedb` CLI tool. It can be used to get information from an OpenNebula database, upgrade it, or fix inconsistency problems.

10.4.1 Connection Parameters

The command `onedb` can connect to any SQLite or MySQL database. Visit the `onedb` man page for a complete reference. These are two examples for the default databases:

```
$ onedb <command> -v --sqlite /var/lib/one/one.db
$ onedb <command> -v -S localhost -u oneadmin -p oneadmin -d opennebula
```

10.4.2 onedb fsck

Checks the consistency of the DB, and fixes the problems found. For example, if the machine where OpenNebula is running crashes, or loses connectivity with the database, you may have a wrong number of VMs running in a Host, or incorrect usage quotas for some users.

```
$ onedb fsck --sqlite /var/lib/one/one.db
Sqlite database backup stored in /var/lib/one/one.db.bck
Use 'onedb restore' or copy the file back to restore the DB.

Host 0 RUNNING_VMS has 12 is 11
Host 0 CPU_USAGE has 1200 is 1100
Host 0 MEM_USAGE has 1572864 is 1441792
Image 0 RUNNING_VMS has 6 is 5
User 2 quotas: CPU_USED has 12 is 11.0
User 2 quotas: MEMORY_USED has 1536 is 1408
User 2 quotas: VMS_USED has 12 is 11
User 2 quotas: Image 0 RVMS has 6 is 5
Group 1 quotas: CPU_USED has 12 is 11.0
Group 1 quotas: MEMORY_USED has 1536 is 1408
Group 1 quotas: VMS_USED has 12 is 11
Group 1 quotas: Image 0 RVMS has 6 is 5

Total errors found: 12
```

If `onedb fsck` shows the following error message:

```
[UNREPAIRED] History record for VM <<vid>> seq # <<seq>> is not closed (etime = 0)
```

This is due to [bug #4000](#). It means that when using accounting or showback, the `etime` (end-time) of that history record is not set, and the VM is considered as still running when it should not. To fix this problem, locate the time when the VM was shut down in the logs and then execute this patch to edit the times manually:

```

$ onedb patch -v --sqlite /var/lib/one/one.db /usr/lib/one/ruby/onedb/patches/history_
↳times.rb
Version read:
Shared tables 4.11.80 : OpenNebula 5.0.1 daemon bootstrap
Local tables 4.13.85 : OpenNebula 5.0.1 daemon bootstrap

Sqlite database backup stored in /var/lib/one/one.db_2015-10-13_12:40:2.bck
Use 'onedb restore' or copy the file back to restore the DB.

> Running patch /usr/lib/one/ruby/onedb/patches/history_times.rb
This tool will allow you to edit the timestamps of VM history records, used to
↳calculate accounting and showback.
VM ID: 1
History sequence number: 0

STIME  Start time      : 2015-10-08 15:24:06 UTC
PSTIME Prolog start time : 2015-10-08 15:24:06 UTC
PETIME Prolog end time  : 2015-10-08 15:24:29 UTC
RSTIME Running start time : 2015-10-08 15:24:29 UTC
RETIME Running end time  : 2015-10-08 15:42:35 UTC
ESTIME Epilog start time : 2015-10-08 15:42:35 UTC
EETIME Epilog end time   : 2015-10-08 15:42:36 UTC
ETIME  End time        : 2015-10-08 15:42:36 UTC

To set new values:
empty to use current value; <YYYY-MM-DD HH:MM:SS> in UTC; or 0 to leave unset (open
↳history record).
STIME  Start time      : 2015-10-08 15:24:06 UTC
New value      :

ETIME  End time        : 2015-10-08 15:42:36 UTC
New value      :

The history record # 0 for VM 1 will be updated with these new values:
STIME  Start time      : 2015-10-08 15:24:06 UTC
PSTIME Prolog start time : 2015-10-08 15:24:06 UTC
PETIME Prolog end time  : 2015-10-08 15:24:29 UTC
RSTIME Running start time : 2015-10-08 15:24:29 UTC
RETIME Running end time  : 2015-10-08 15:42:35 UTC
ESTIME Epilog start time : 2015-10-08 15:42:35 UTC
EETIME Epilog end time   : 2015-10-08 15:42:36 UTC
ETIME  End time        : 2015-10-08 15:42:36 UTC

Confirm to write to the database [Y/n]: y
> Done

> Total time: 27.79s

```

10.4.3 onedb version

Prints the current DB version.

```

$ onedb version --sqlite /var/lib/one/one.db
3.8.0

```

Use the `-v` flag to see the complete version and comment.

```
$ onedb version -v --sqlite /var/lib/one/one.db
Version: 3.8.0
Timestamp: 10/19 16:04:17
Comment: Database migrated from 3.7.80 to 3.8.0 (OpenNebula 3.8.0) by onedb command.
```

If the MySQL database password contains special 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');
```

10.4.4 onedb history

Each time the DB is upgraded, the process is logged. You can use the `history` command to retrieve the upgrade history.

```
$ onedb history -S localhost -u oneadmin -p oneadmin -d opennebula
Version: 3.0.0
Timestamp: 10/07 12:40:49
Comment: OpenNebula 3.0.0 daemon bootstrap

...

Version: 3.7.80
Timestamp: 10/08 17:36:15
Comment: Database migrated from 3.6.0 to 3.7.80 (OpenNebula 3.7.80) by onedb_
→command.

Version: 3.8.0
Timestamp: 10/19 16:04:17
Comment: Database migrated from 3.7.80 to 3.8.0 (OpenNebula 3.8.0) by onedb command.
```

10.4.5 onedb upgrade

The upgrade process is fully documented in the [Upgrading from Previous Versions](#) guide.

10.4.6 onedb backup

Dumps the OpenNebula DB to a file.

```
$ onedb backup --sqlite /var/lib/one/one.db /tmp/my_backup.db
Sqlite database backup stored in /tmp/my_backup.db
Use 'onedb restore' or copy the file back to restore the DB.
```

10.4.7 onedb restore

Restores the DB from a backup file. Please note that this tool will only restore backups generated from the same backend, i.e. you cannot backup a SQLite database and then try to populate a MySQL one.

10.4.8 onedb sqlite2mysql

This command migrates from a sqlite database to a mysql database. The procedure to follow is:

- Stop OpenNebula
- Change the DB directive in `/etc/one/oned.conf` to use MySQL instead of SQLite
- Bootstrap the MySQL Database: `oned -i`
- Migrate the Database: `onedb sqlite2mysql -s <SQLITE_PATH> -u <MYSQL_USER> -p <MYSQL_PASS> -d <MYSQL_DB>`
- Start OpenNebula

10.5 Large Deployments

10.5.1 Monitoring

In KVM environments, OpenNebula supports two native monitoring systems: `ssh-pull` and `udp-push`. The former one, `ssh-pull` is the default monitoring system for OpenNebula ≤ 4.2 , however from OpenNebula 4.4 onwards, the default monitoring system is the `udp-push` system. 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. Read more in the *Monitoring guide*.

For vCenter environments, OpenNebula uses the VI API offered by vCenter to monitor the state of the hypervisor and all the Virtual Machines running in all the imported vCenter clusters. The driver is optimized to cache common VM information.

In both environments, our scalability testing achieves the monitoring of tens of thousands of VMs in a few minutes.

10.5.2 Core Tuning

OpenNebula keeps the monitorization history for a defined time in a database table. These values are then used to draw the plots in Sunstone.

These monitorization entries can take quite a bit of storage in your database. The amount of storage used will depend on the size of your cloud, and the following configuration attributes in `oned.conf`:

- `MONITORING_INTERVAL`: Time in seconds between each monitorization. Default: 60.
- `collectd IM_MAD -i` argument (KVM only): Time in seconds of the monitorization push cycle. Default: 20.
- `HOST_MONITORING_EXPIRATION_TIME`: Time, in seconds, to expire monitoring information. Default: 12h.
- `VM_MONITORING_EXPIRATION_TIME`: Time, in seconds, to expire monitoring information. Default: 4h.

If you don't use Sunstone, you may want to disable the monitoring history, setting both expiration times to 0.

Each monitoring entry will be around 2 KB for each Host, and 4 KB for each VM. To give you an idea of how much database storage you will need to prepare, here are some examples:

Monitoring interval	Host expiration	# Hosts	Storage
20s	12h	200	850 MB
20s	24h	1000	8.2 GB

Monitoring interval	VM expiration	# VMs	Storage
20s	4h	2000	1.8 GB
20s	24h	10000	7 GB

10.5.3 API Tuning

For large deployments with lots of xmlrpc calls the default values for the xmlrpc server are too conservative. The values you can modify and its meaning are explained in *oned.conf* and the *xmlrpc-c* library documentation. From our experience these values improve the server behavior with a high amount of client calls:

```
MAX_CONN = 240
MAX_CONN_BACKLOG = 480
```

The core is able to paginate some pool answers. This makes the memory consumption decrease and in some cases the parsing faster. By default the pagination value is 2000 objects but can be changed using the environment variable `ONE_POOL_PAGE_SIZE`. It should be bigger than 2. For example, to list VMs with a page size of 5000 we can use:

```
$ ONE_POOL_PAGE_SIZE=5000 onevm list
```

To disable pagination we can use a non numeric value:

```
$ ONE_POOL_PAGE_SIZE=disabled onevm list
```

This environment variable can be also used for Sunstone.

10.5.4 Driver Tuning

OpenNebula drivers have by default 15 threads. This is the maximum number of actions a driver can perform at the same time, the next actions will be queued. You can make this value in *oned.conf*, the driver parameter is `-t`.

10.5.5 Database Tuning

For non test installations use MySQL database. sqlite is too slow for more than a couple hosts and a few VMs.

10.5.6 Sunstone Tuning

Please refer to guide about *Configuring Sunstone for Large Deployments*.