OpenNebula.org

OpenNebula 4.4 Administration Guide

OpenNebula Project

December 17, 2013

CONTENTS

CHAPTER

ONE

HOSTS AND CLUSTERS

1.1 Hosts & Clusters Overview

A **Host** is a server that has the ability to run Virtual Machines and that is connected to OpenNebula's Frontend server. OpenNebula can work with Hosts with a heterogeneous configuration, i.e. you can connect Hosts to the same OpenNebula with different hypervisors or Linux distributions as long as these requirements are fulfilled:

- Every Host need to have a `oneadmin` account.
- OpenNebula's Frontend and all the Hosts need to be able to resolve, either by DNS or by `/etc/hosts` the names of all the other Hosts and Frontend.
- The `oneadmin` account in any Host or the Frontend should be able to ssh passwordlessly to any other Host or Frontend. This is achieved either by sharing the `$HOME` of `oneadmin` accross all the servers with NFS or by manually copying the `~/.ssh` directory.
- It needs to have a hypervisor supported by OpenNebula installed and properly configured. The correct way to achieve this is to follow the specific guide for each hypervisor.
- **ruby** >= 1.8.7

Clusters are pools of hosts that share datastores and virtual networks. Clusters are used for load balancing, high availability, and high performance computing.

1.1.1 Overview of Components

There are three components regarding Hosts:

- **Host Management**: Host management is achieved through the `onehost` CLI command or through the Sunstone GUI. You can read about Host Management in more detail in the *Managing Hosts* guide.
- **Host Monitorization**: In order to keep track of the available resources in the Hosts, OpenNebula launches a Host Monitoring driver, called IM (Information Driver), which gathers all the required information and submits it to the Core. The default IM driver executes `ssh` commands in the host, but other mechanism are possible. There is further information on this topic in the *Monitoring Subsystem* guide.
- **Cluster Management**: Hosts can be grouped in Clusters. These Clusters are managed with the `onecluster` CLI command, or through the Sunstone GUI. You can read about Cluster Management in more detail in the *Managing Clusters* guide..

1.2 Managing Hosts

In order to use your existing physical nodes, you have to add them to the system as OpenNebula hosts. You need the following information:

- *Hostname* of the host or IP
- *Information Driver* to be used to monitor the host, e.g. `kvm`. These should match the Virtualization Drivers installed and more info about them can be found at the *Virtualization Subsystem guide*.
- *Virtualization Driver* to boot, stop, resume or migrate VMs in the host, e.g. `kvm`. Information about these drivers can be found in *its guide*.
- *Networking Driver* to isolate virtual networks and apply firewalling rules, e.g. `802.1Q`. Information about these drivers can be found in *its guide*.
- *Cluster* where to place this host. The Cluster assignment is optional, you can read more about it in the *Managing Clusters* guide.

Warning: Before adding a host check that you can ssh to it without being prompt for a password

1.2.1 onehost Command

The following sections show the basics of the `onehost` command with simple usage examples. A complete reference for these commands can be found *here*.

This command enables Host management. Actions offered are:

- `create`: Creates a new Host
- `delete`: Deletes the given Host
- `enable`: Enables the given Host
- `disable`: Disables the given Host
- `update`: Update the template contents.
- `sync`: Synchronizes probes in all the hosts.
- `list`: Lists Hosts in the pool
- `show`: Shows information for the given Host
- `top`: Lists Hosts continuously
- `flush`: Disables the host and reschedules all the running VMs it.

Create and Delete

Hosts, also known as physical nodes, are the serves managed by OpenNebula responsible for Virtual Machine execution. To use these hosts in OpenNebula you need to register them so they are monitored and well-known to the scheduler.

Creating a host:

```
$ onehost create host01 --im dummy --vm dummy --net dummy
ID: 0
```

The parameters are:

- `--im/-i`: Information Manager driver. Valid options: `kvm`, `xen`, `vmware`, `ec2`, `ganglia`, `dummy`.
- `--vm/-v`: Virtual Machine Manager driver. Valid options: `kvm`, `xen`, `vmware`, `ec2`, `dummy`.
- `--net/-n`: Network manager driver. Valid options: `802.1Q`,`dummy`,`ebtables`,`fw`,`ovswitch`,`vmware`.

To remove a host, just like with other OpenNebula commands, you can either specify it by ID or by name. The following commands are equivalent:

```
$ onehost delete host01
$ onehost delete 0
```

Show, List and Top

To display information about a single host the `show` command is used:

```
$ onehost show 0
HOST 0 INFORMATION
ID                    : 0
NAME                  : host01
CLUSTER               : -
STATE                 : MONITORED
IM_MAD                : dummy
VM_MAD                : dummy
VN_MAD                : dummy
LAST MONITORING TIME  : 07/06 17:40:41

HOST SHARES
TOTAL MEM             : 16G
USED MEM (REAL)       : 857.9M
USED MEM (ALLOCATED)  : 0K
TOTAL CPU             : 800
USED CPU (REAL)       : 299
USED CPU (ALLOCATED)  : 0
RUNNING VMS           : 0

MONITORING INFORMATION
CPUSPEED="2.2GHz"
FREECPU="501"
FREEMEMORY="15898723"
HOSTNAME="host01"
HYPERVISOR="dummy"
TOTALCPU="800"
TOTALMEMORY="16777216"
USEDCPU="299"
USEDMEMORY="878493"
```

We can instead display this information in XML format with the `-x` parameter:

```
$ onehost show -x 0
<HOST>
  <ID>0</ID>
  <NAME>host01</NAME>
  <STATE>2</STATE>
  <IM_MAD>dummy</IM_MAD>
  <VM_MAD>dummy</VM_MAD>
  <VN_MAD>dummy</VN_MAD>
  <LAST_MON_TIME>1341589306</LAST_MON_TIME>
  <CLUSTER_ID>-1</CLUSTER_ID>
```

```
  <CLUSTER/>
  <HOST_SHARE>
    <DISK_USAGE>0</DISK_USAGE>
    <MEM_USAGE>0</MEM_USAGE>
    <CPU_USAGE>0</CPU_USAGE>
    <MAX_DISK>0</MAX_DISK>
    <MAX_MEM>16777216</MAX_MEM>
    <MAX_CPU>800</MAX_CPU>
    <FREE_DISK>0</FREE_DISK>
    <FREE_MEM>12852921</FREE_MEM>
    <FREE_CPU>735</FREE_CPU>
    <USED_DISK>0</USED_DISK>
    <USED_MEM>3924295</USED_MEM>
    <USED_CPU>65</USED_CPU>
    <RUNNING_VMS>0</RUNNING_VMS>
  </HOST_SHARE>
  <TEMPLATE>
    <CPUSPEED><![CDATA[2.2GHz]]></CPUSPEED>
    <FREECPU><![CDATA[735]]></FREECPU>
    <FREEMEMORY><![CDATA[12852921]]></FREEMEMORY>
    <HOSTNAME><![CDATA[host01]]></HOSTNAME>
    <HYPERVISOR><![CDATA[dummy]]></HYPERVISOR>
    <TOTALCPU><![CDATA[800]]></TOTALCPU>
    <TOTALMEMORY><![CDATA[16777216]]></TOTALMEMORY>
    <USEDCPU><![CDATA[65]]></USEDCPU>
    <USEDMEMORY><![CDATA[3924295]]></USEDMEMORY>
  </TEMPLATE>
</HOST>
```

To see a list of all the hosts:

```
$ onehost list
  ID NAME            CLUSTER   RVM TCPU FCPU ACPU    TMEM    FMEM    AMEM STAT
   0 host01          -           0  800  198  800     16G   10.9G     16G on
   1 host02          -           0  800  677  800     16G    3.7G     16G on
```

It can also be displayed in XML format using `-x`:

```
$ onehost list -x
<HOST_POOL>
  <HOST>
    ...
  </HOST>
  ...
</HOST_POOL>
```

The `top` command is similar to the `list` command, except that the output is refreshed until the user presses `CTRL-C`.

Enable, Disable and Flush

The `disable` command disables a host, which means that no further monitorization is performed on this host and no Virtual Machines are deployed in it. It won't however affect the running VMs in the host.

```
$ onehost disable 0
```

To re-enable the host use the `enable` command:

```
$ onehost enable 0
```

The `flush` command will mark all the running VMs in the specified host as to be rescheduled, which means that they will be migrated to another server with enough capacity. At the same time, the specified host will be disabled, so no more Virtual Machines are deployed in it. This command is useful to clean a host of running VMs.

```
$ onehost list
  ID NAME            CLUSTER   RVM TCPU FCPU ACPU    TMEM    FMEM    AMEM STAT
   0 host01          -           3  800   96  500     16G   11.1G   14.5G on
   1 host02          -           0  800  640  800     16G    8.5G     16G on
   2 host03          -           3  800  721  500     16G    8.6G   14.5G on
$ onevm list
    ID USER     GROUP    NAME            STAT UCPU    UMEM HOST             TIME
     0 oneadmin oneadmin vm01            runn   54  102.4M host03        0d 00h01
     1 oneadmin oneadmin vm02            runn   91  276.5M host02        0d 00h01
     2 oneadmin oneadmin vm03            runn   13  174.1M host01        0d 00h01
     3 oneadmin oneadmin vm04            runn   72  204.8M host03        0d 00h00
     4 oneadmin oneadmin vm05            runn   49  112.6M host02        0d 00h00
     5 oneadmin oneadmin vm06            runn   87  414.7M host01        0d 00h00
$ onehost flush host02
$ onehost list
  ID NAME            CLUSTER   RVM TCPU FCPU ACPU    TMEM    FMEM    AMEM STAT
   0 host01          -           3  800  264  500     16G    3.5G   14.5G on
   1 host02          -           0  800  153  800     16G    3.7G     16G off
   2 host03          -           3  800  645  500     16G   10.3G   14.5G on
$ onevm list
    ID USER     GROUP    NAME            STAT UCPU    UMEM HOST             TIME
     0 oneadmin oneadmin vm01            runn   95  179.2M host03        0d 00h01
     1 oneadmin oneadmin vm02            runn   27  261.1M host03        0d 00h01
     2 oneadmin oneadmin vm03            runn   70    343M host01        0d 00h01
     3 oneadmin oneadmin vm04            runn    9  133.1M host03        0d 00h01
     4 oneadmin oneadmin vm05            runn   87  281.6M host01        0d 00h01
     5 oneadmin oneadmin vm06            runn   61  291.8M host01        0d 00h01
```

Update

It's sometimes useful to store information in the host's template. To do so, the `update` command is used.

An example use case is to add the following line to the host's template:

```
TYPE="production"
```

Which can be used at a later time for scheduling purposes by adding the following section in a VM template:

```
SCHED_REQUIREMENTS="TYPE=\"production\""
```

That will restrict the Virtual Machine to be deployed in `TYPE=production` hosts.

Sync

When OpenNebula monitors a host, it copies a certain amount of files to `/var/tmp/one`. When the administrator changes these files, they can be copied again to the hosts with the `sync` command. When executed this command will copy the probes to the nodes and will return the prompt after it has finished telling which nodes it could not update.

To keep track of the probes version there's a new file in `/var/lib/one/remotes/VERSION`. By default this holds the OpenNebula version (ex. '4.4.0'). This version can be seen in he hosts with a `onehost show <host>`:

```
$ onehost show 0
HOST 0 INFORMATION
ID                    : 0
[...]
MONITORING INFORMATION
VERSION="4.4.0"
[...]
```

The command `onehost sync` only updates the hosts with `VERSION` lower than the one in the file `/var/lib/one/remotes/VERSION`. In case you modify the probes this `VERSION` file should be modified with a greater value, for example `4.4.0.01`.

In case you want to force upgrade, that is, no `VERSION` checking you can do that adding `-force` option:

```
$ onehost sync --force
```

You can also select which hosts you want to upgrade naming them or selecting a cluster:

```
$ onehost sync host01,host02,host03
$ onehost sync -c myCluster
```

`onehost sync` command can alternatively use `rsync` as the method of upgrade. To do this you need to have installed `rsync` command in the frontend and the nodes. This method is faster that the standard one and also has the benefit of deleting remote files no longer existing in the frontend. To use it add the parameter `-rsync`:

```
$ onehost sync --rsync
```

1.2.2 Host Life-cycle

Short state	State	Meaning
`init`	`INIT`	Initial state for enabled hosts.
`update`	`MONITORING_MONITORED`	Monitoring a healthy Host.
`on`	`MONITORED`	The host has been successfully monitored.
`err`	`ERROR`	An error occurred while monitoring the host. See the Host information with `onehost show` for an error message.
`off`	`DISABLED`	The host is disabled, and won't be monitored. The scheduler ignores Hosts in this state.
`retry`	`MONITORING_ERROR`	Monitoring a host in error state.

1.2.3 Scheduler Policies

You can define global Scheduler Policies for all VMs in the sched.conf file, follow the *Scheduler Guide* for more information. Additionally, users can require their virtual machines to be deployed in a host that meets certain constrains. These constrains can be defined using any attribute reported by `onehost show`, like the architecture (ARCH).

The attributes and values for a host are inserted by the monitoring probes that run from time to time on the nodes to get information. The administrator can add custom attributes either *creating a probe in the host*, or updating the host information with: `onehost update <HOST_ID>`. Calling this command will fire up an editor (the one specified in the `EDITOR` environment variable) and you will be able to add, delete or modify some of those values.

```
$ onehost show 3
[...]
MONITORING INFORMATION
CPUSPEED=2.2GHz
```

```
FREECPU=800
FREEMEMORY=16777216
HOSTNAME=ursa06
HYPERVISOR=dummy
TOTALCPU=800
TOTALMEMORY=16777216
USEDCPU=0
USEDMEMORY=0

$ onehost update 3

[in editor, add CUSTOM_ATTRIBUTE=VALUE]

$onehost show 3
[...]
MONITORING INFORMATION
CPUSPEED=2.2GHz
FREECPU=800
FREEMEMORY=16777216
HOSTNAME=ursa06
HYPERVISOR=dummy
TOTALCPU=800
TOTALMEMORY=16777216
USEDCPU=0
USEDMEMORY=0
CUSTOM_ATTRIBUTE=VALUE
```

This feature is useful when we want to separate a series of hosts or marking some special features of different hosts. These values can then be used for scheduling the same as the ones added by the monitoring probes, as a *placement requirement*:

```
SCHED_REQUIREMENTS = "CUSTOM_ATTRIBUTE = \"SOME_VALUE\""
```

1.2.4 A Sample Session

Hosts can be added to the system anytime with the `onehost` command. You can add the hosts to be used by OpenNebula like this:

```
$ onehost create host01 --im kvm --vm kvm --net dummy
$ onehost create host02 --im kvm --vm kvm --net dummy
```

The status of the hosts can be checked with the `onehost list` command:

```
$ onehost list
  ID NAME          CLUSTER     RVM   TCPU   FCPU   ACPU   TMEM   FMEM   AMEM STAT
   0 host01        -             7    400    290    400   3.7G   2.2G   3.7G   on
   1 host02        -             2    400    294    400   3.7G   2.2G   3.7G   on
   2 host03        -             0    400    312    400   3.7G   2.2G   3.7G  off
```

And specific information about a host with `show`:

```
$ onehost show host01
HOST 0 INFORMATION
ID                    : 0
NAME                  : host01
CLUSTER               : -
STATE                 : MONITORED
IM_MAD                : kvm
```

```
VM_MAD                : kvm
VN_MAD                : dummy
LAST MONITORING TIME  : 1332756227

HOST SHARES
MAX MEM               : 3921416
USED MEM (REAL)       : 1596540
USED MEM (ALLOCATED)  : 0
MAX CPU               : 400
USED CPU (REAL)       : 74
USED CPU (ALLOCATED)  : 0
RUNNING VMS           : 7

MONITORING INFORMATION
ARCH=x86_64
CPUSPEED=2393
FREECPU=326.0
FREEMEMORY=2324876
HOSTNAME=rama
HYPERVISOR=kvm
MODELNAME="Intel(R) Core(TM) i5 CPU M 450 @ 2.40GHz"
NETRX=0
NETTX=0
TOTALCPU=400
TOTALMEMORY=3921416
USEDCPU=74.0
USEDMEMORY=1596540
```

If you want not to use a given host you can temporarily disable it:

```
$ onehost disable host01
```

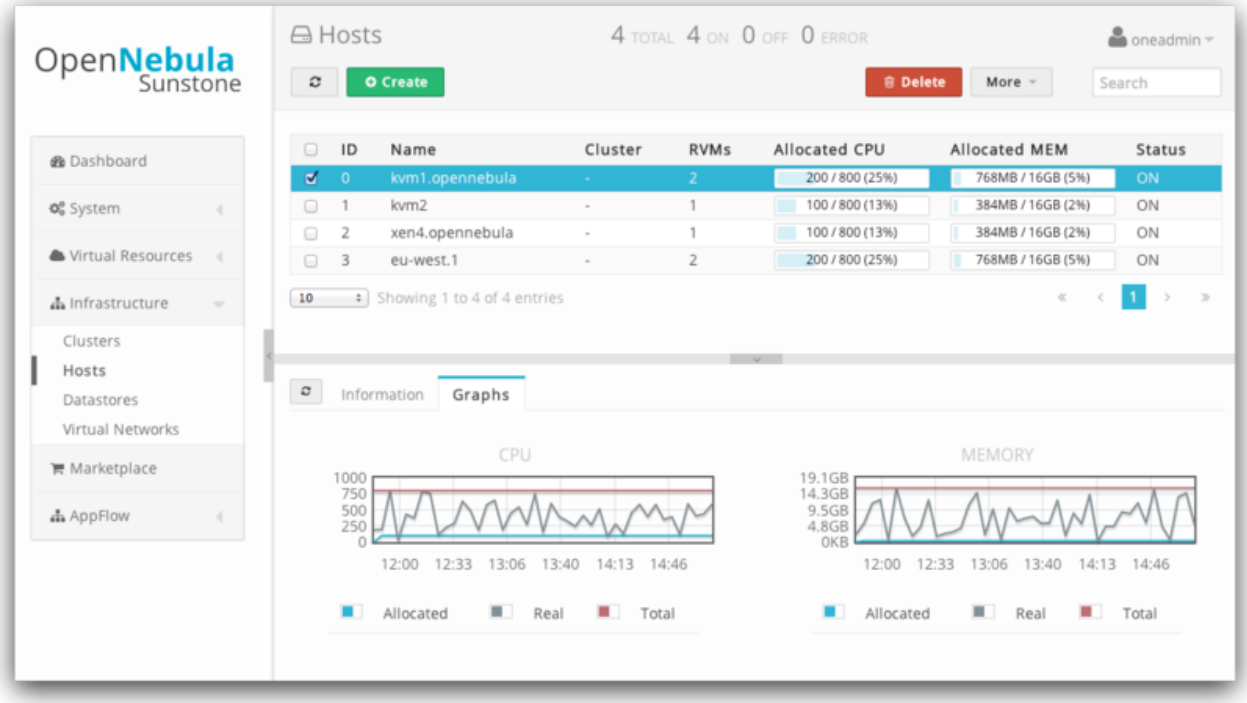
A disabled host should be listed with `STAT off` by `onehost list`. You can also remove a host permanently with:

```
$ onehost delete host01
```

Warning: Detailed information of the `onehost` utility can be found *in the Command Line Reference*

1.2.5 Using Sunstone to Manage Hosts

You can also manage your hosts using *Sunstone*. Select the Host tab, and there, you will be able to create, enable, disable, delete and see information about your hosts in a user friendly way.



1.3 Managing Clusters

A Cluster is a group of *Hosts*. Clusters can have associated *Datastores* and *Virtual Networks*, this is how the administrator sets which Hosts have the underlying requirements for each Datastore and Virtual Network configured.

1.3.1 Cluster Management

Clusters are managed with the *''onecluster" command*. To create new Clusters, use `onecluster create <name>`. Existing Clusters can be inspected with the `onecluster list` and `show` commands.

```
$ onecluster list
  ID NAME            HOSTS NETS  DATASTORES

$ onecluster create production
ID: 100

$ onecluster list
  ID NAME            HOSTS NETS  DATASTORES
 100 production      0     0     0

$ onecluster show production
CLUSTER 100 INFORMATION
ID             : 100
NAME           : production

HOSTS

VNETS

DATASTORES
```

Add Hosts to Clusters

Hosts can be created directly in a Cluster, using the `-cluster` option of `onehost create`, or be added at any moment using the command `onecluster addhost`. Hosts can be in only one Cluster at a time.

To delete a Host from a Cluster, the command `onecluster delhost` must be used. When a Host is removed from a Cluster, it is seen as part of the Cluster 'none', more about this below.

In the following example, we will add Host 0 to the Cluster we created before. You will notice that the `onecluster show` command will list the Host ID 0 as part of the Cluster.

```
$ onehost list
  ID NAME          CLUSTER     RVM   TCPU   FCPU   ACPU   TMEM   FMEM   AMEM STAT
   0 host01        -             7    400    290    400   3.7G   2.2G   3.7G   on

$ onecluster addhost production host01

$ onehost list
  ID NAME          CLUSTER     RVM   TCPU   FCPU   ACPU   TMEM   FMEM   AMEM STAT
   0 host01        producti      7    400    290    400   3.7G   2.2G   3.7G   on

$ onecluster show production
CLUSTER 100 INFORMATION
ID             : 100
NAME           : production

HOSTS
0

VNETS

DATASTORES
```

Add Resources to Clusters

Datastores and Virtual Networks can be added to one Cluster. This means that any Host in that Cluster is properly configured to run VMs using Images from the Datastores, or is using leases from the Virtual Networks.

For instance, if you have several Hosts configured to use *Open vSwitch networks*, you would group them in the same Cluster. The *Scheduler* will know that VMs using these resources can be deployed in any of the Hosts of the Cluster.

These operations can be done with the `onecluster addvnet/delvnet` and `adddatastore/deldatastore`:

```
$ onecluster addvnet production priv-ovswitch

$ onecluster adddatastore production iscsi

$ onecluster list
  ID NAME            HOSTS NETS  DATASTORES
 100 production      1     1     1

$ onecluster show 100
CLUSTER 100 INFORMATION
ID             : 100
NAME           : production

CLUSTER TEMPLATE
```

```
HOSTS
0

VNETS
1

DATASTORES
100
```

The System Datastore for a Cluster

You can associate an specific System DS to a cluster to improve its performance (e.g. balance VM I/O between different servers) or to use different system DS types (e.g. shared and ssh).

To use a specific System DS with your cluster, instead of the default one, just create it (with TYPE=SYSTEM_DS in its template), and associate it just like any other datastore (onecluster adddatastore). Check the *System DS guide for more information*.

Cluster Properties

Each cluster includes a generic template where cluster configuration properties or attributes can be defined. The following list of attributes are recognized by OpenNebula:

Attribute	Description
DATASTORE_LOCATION	*Default* path for datastores in the cluster hosts. It **is** the same for all the hosts in the cluster. Note that DATASTORE_LOCATION is only for the cluster hosts and not for the front-end. It defaults to /var/lib/one/datastores

You can easily update this values with the `onecluster` command:

```
$ onecluster update production

-----8<----- editor session ------8<------

DATASTORE_LOCATION="/mnt/nas/datastores"
~
~
~
-----8----- editor session ------>8------

$oneluster show production
CLUSTER 100 INFORMATION
ID               : 100
NAME             : production
SYSTEM DATASTORE : 100

CLUSTER TEMPLATE
DATASTORE_LOCATION="/mnt/nas/datastores"

HOSTS
0

VNETS
1
```

```
DATASTORES
100
```

You can add as many variables as you want, following the standard template syntax. These variables will be used for now only for informational purposes.

1.3.2 The Default Cluster 'None'

Hosts, Datastores and Virtual Networks can be grouped into clusters, but this is optional. By default, these resources are created outside of any Cluster, what can be seen as a special Cluster named 'none' in Sunstone. In the CLI, this Cluster name is shown as '-'.

Virtual Machines using resources from Datastores or Virtual Networks in the Cluster 'none' can be deployed in any Host, which must be properly configured.

Hosts in the Cluster 'none' will only run VMs using resources without a Cluster.

1.3.3 Scheduling and Clusters

Automatic Requirements

When a Virtual Machine uses resources (Images or Virtual Networks) from a Cluster, OpenNebula adds the following *requirement* to the template:

```
$ onevm show 0
[...]
AUTOMATIC_REQUIREMENTS="CLUSTER_ID = 100"
```

Because of this, if you try to use resources from more than one Cluster, the Virtual Machine creation will fail with a message similar to this one:

```
$ onetemplate instantiate 0
[TemplateInstantiate] Error allocating a new virtual machine. Incompatible cluster IDs.
DISK [0]: IMAGE [0] from DATASTORE [1] requires CLUSTER [101]
NIC [0]: NETWORK [1] requires CLUSTER [100]
```

Manual Requirements and Rank

The placement attributes *SCHED_REQUIREMENTS and SCHED_RANK* can use attributes from the Cluster template. Let's say you have the following scenario:

```
$ onehost list
  ID NAME            CLUSTER   RVM      ALLOCATED_CPU      ALLOCATED_MEM STAT
   1 host01          cluster_a   0       0 / 200 (0%)     0K / 3.6G (0%) on
   2 host02          cluster_a   0       0 / 200 (0%)     0K / 3.6G (0%) on
   3 host03          cluster_b   0       0 / 200 (0%)     0K / 3.6G (0%) on

$ onecluster show cluster_a
CLUSTER TEMPLATE
QOS="GOLD"

$ onecluster show cluster_b
CLUSTER TEMPLATE
QOS="SILVER"
```

You can use these expressions:

```
SCHED_REQUIREMENTS = "QOS = GOLD"

SCHED_REQUIREMENTS = "QOS != GOLD & HYPERVISOR = kvm"
```

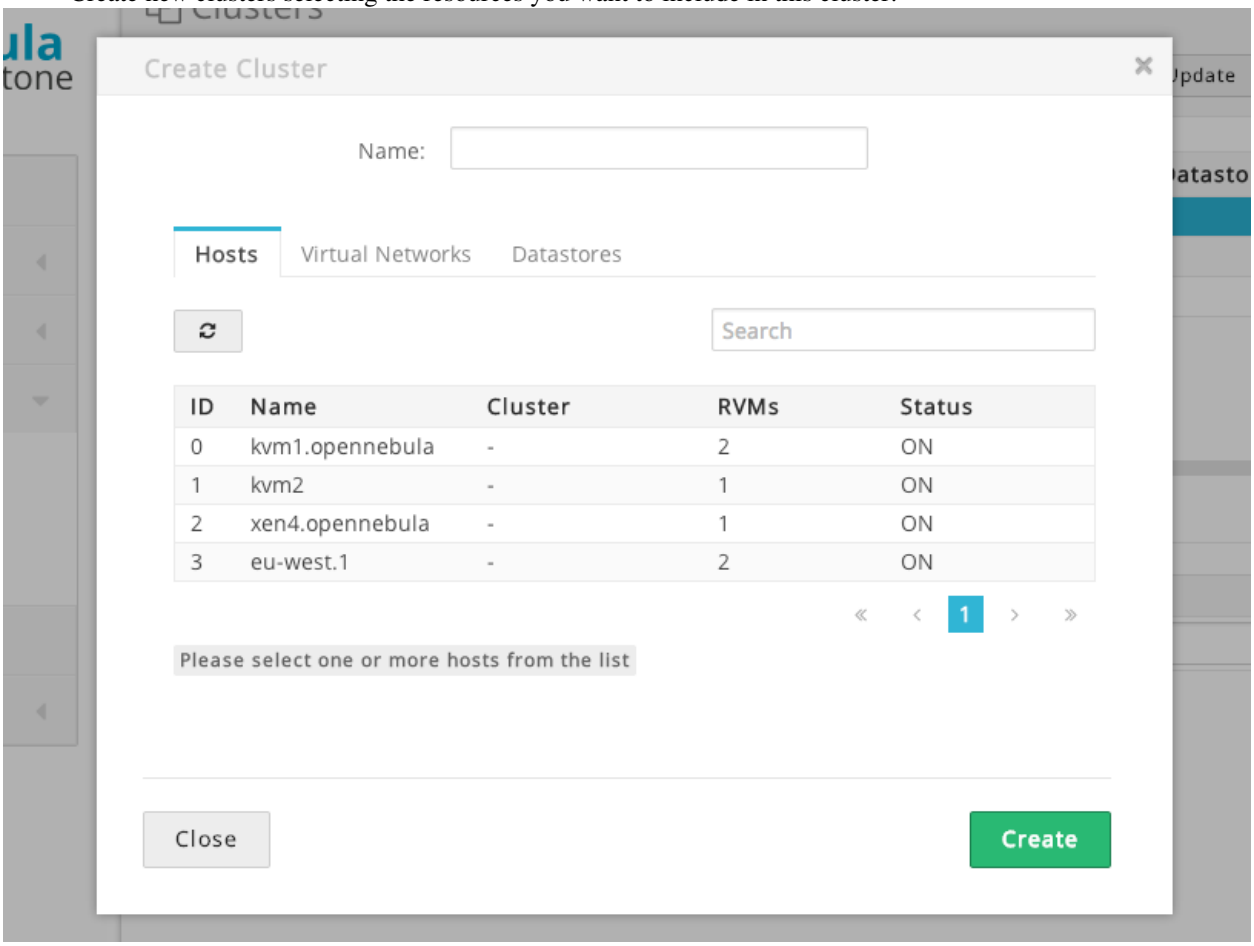
1.3.4 System Storage

The system datastore holds files for running VMs. Each cluster can use a different system datastore, read more in *the system datastore guide*.

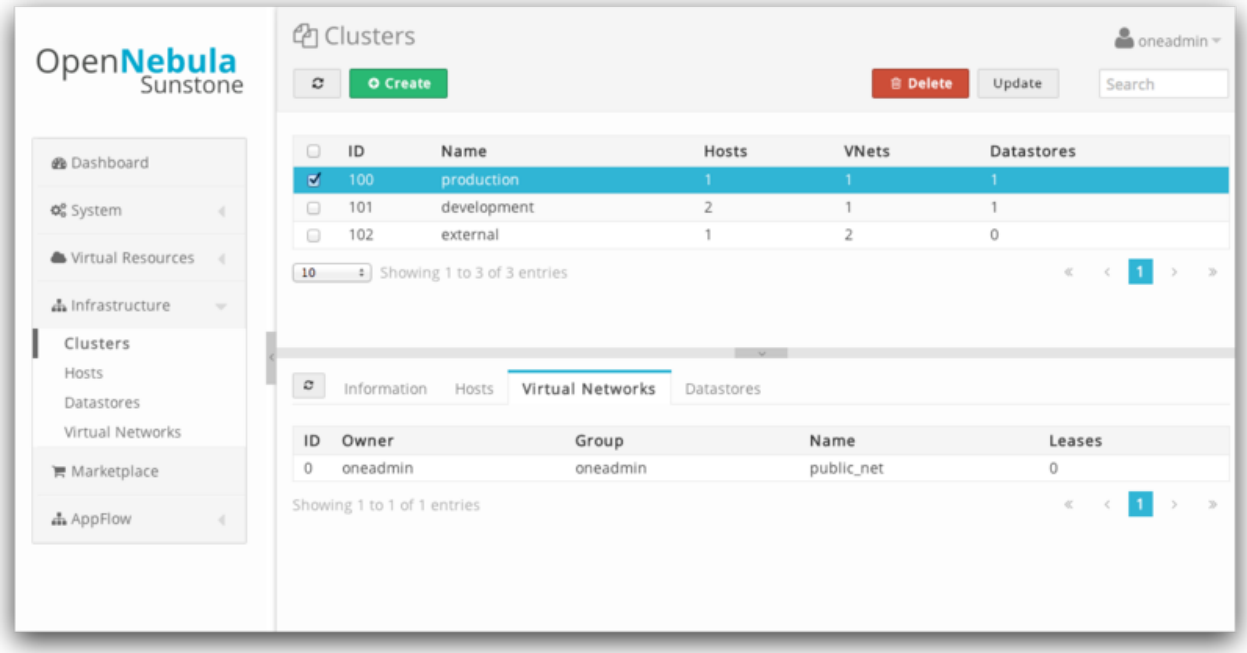
1.3.5 Managing Clusters in Sunstone

The *Sunstone UI interface* offers an easy way to manage clusters and the resources whithin them. You will find the cluster submenu under the infraestructure menu. From there, you will be able to:

- Create new clusters selecting the resources you want to include in this cluster:



- See the list of current clusters, from which you can update the template of existing ones, or delete them.


OpenNebula
Sunstone
Clusters
oneadmin
Create
Delete
Update
Search
Dashboard
System
Virtual Resources
Infrastructure
Clusters
Hosts
Datastores
Virtual Networks
Marketplace
AppFlow
ID
Name
Hosts
VNets
Datastores
100
production
1
1
1
101
development
2
1
1
102
external
1
2
0
10
Showing 1 to 3 of 3 entries
1
Information
Hosts
Virtual Networks
Datastores
ID
Owner
Group
Name
Leases
0
oneadmin
oneadmin
public_net
0
Showing 1 to 1 of 1 entries
1

CHAPTER

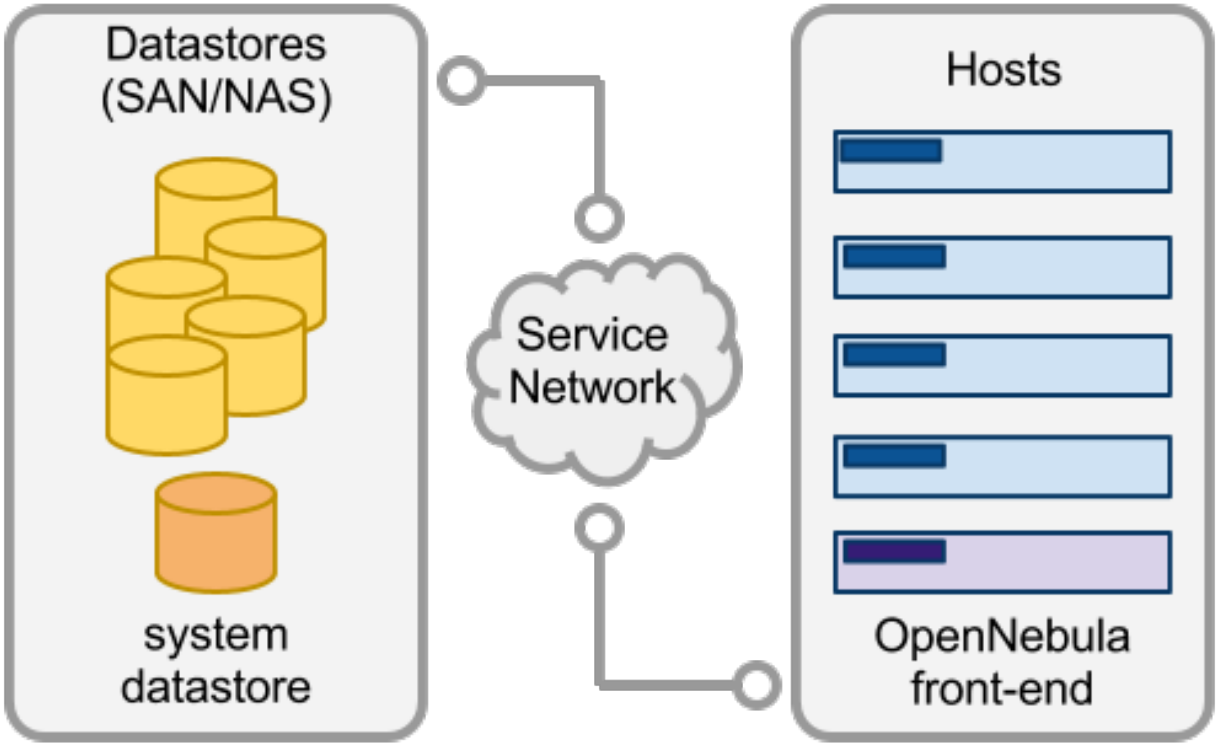
TWO

STORAGE

2.1 Storage Overview

A Datastore is any storage medium used to store disk images for VMs, previous versions of OpenNebula refer to this concept as Image Repository. Typically, a datastore will be backed by SAN/NAS servers.

An OpenNebula installation can have multiple datastores of several types to store disk images. OpenNebula also uses a special datastore, the *system datastore*, to hold images of running VMs.



2.1.1 What Datastore Types Are Available?

OpenNebula is shipped with 3 different datastore classes:

- *System*, 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.

- **Images**, stores the disk images repository. Disk images are moved, or cloned to/from the System datastore when the VMs are deployed or shutdown; or when disks are attached or snapshoted.
- *Files*, This is a special datastore used to store plain files and not disk images. The plain files can be used as kernels, ramdisks or context files.

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

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

As usual in OpenNebula the system has been architected to be highly modular, so you can easily adapt the base types to your deployment.

2.1.2 How Are the Images Transferred to the Hosts?

The Disk images registered in a datastore are transferred to the hosts by the transfer manager (TM) drivers. These drivers are specialized pieces of software that perform low-level storage operations.

The transfer mechanism is defined for each datastore. In this way a single host can simultaneously access multiple datastores that uses different transfer drivers. Note that the hosts must be configured to properly access each data-store type (e.g. mount FS shares).

OpenNebula includes 6 different ways to distribute datastore images to the hosts:

- **shared**, the datastore is exported in a shared filesystem to the hosts.
- **ssh**, datastore images are copied to the remote hosts using the ssh protocol
- **vmfs**, image copies are done using the vmkfstools (VMware filesystem tools)
- **qcow**, a driver specialized to handle qemu-qcow format and take advantage of its snapshoting capabilities
- **ceph**, a driver that delegates to libvirt/KVM the management of Ceph RBDs.
- **lvm**, images are stored as LVs in a cLVM volume.

2.1.3 Planning your Storage

You can take advantage of the multiple datastore features of OpenNebula to better scale the storage for your VMs, in particular:

- Balancing I/O operations between storage servers
- Different VM types or users can use datastores with different performance features
- Different SLA policies (e.g. backup) can be applied to different VM types or users
- Easily add new storage to the cloud

There are some limitations and features depending on the transfer mechanism you choose for your system and image datastores (check each datastore guide for more information). The following table summarizes the valid combinations of Datastore and transfer drivers:

Datastore	shared	ssh	qcow2	vmfs	ceph	lvm	fs_lvm
System	x	x		x			
File-System	x	x	x				x
vmfs				x			
ceph					x		
lvm						x	

2.1.4 Tuning and Extending

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

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

- /var/lib/one/remotes/datastore/`<DS_DRIVER>`
- /var/lib/one/remotes/tm/`<TM_DRIVER>`

2.2 The System Datastore

The system datastore is a special Datastore class that holds images for running VMs. As opposed to the regular images datastores you cannot register new images into a system datastore.

2.2.1 Types of System Datastore

For each running VM in the datastore there is a directory containing the disk images and additional configuration files. For example, the structure of the system datastore 0 with 3 VMs (VM 0 and 2 running, and VM 7 stopped) could be:

```
datastores
|-- 0/
|   |-- 0/
|   |   |-- disk.0
|   |   `-- disk.1
|   |-- 2/
|   |   `-- disk.0
|   `-- 7/
|       |-- checkpoint
|       `-- disk.0
```

There are three system datastore types, based on the TM_MAD driver used:

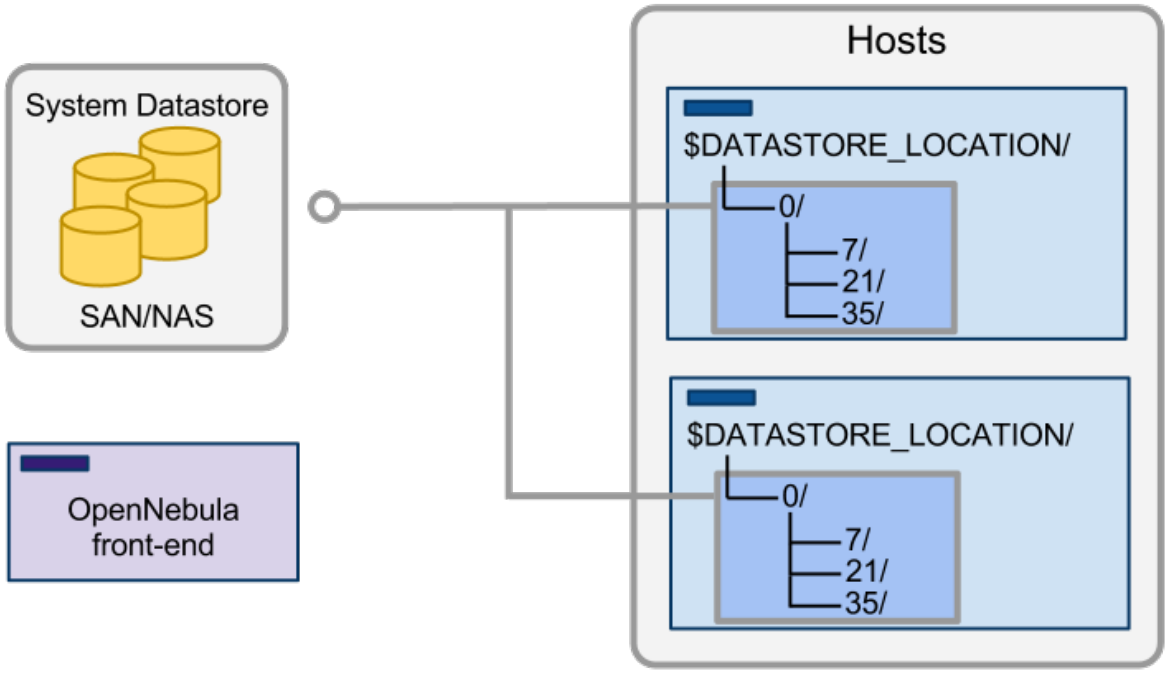
- `shared`, the storage area for the system datastore is a shared directory across the hosts.
- `vmfs`, a specialized version of the shared one to use the vmfs file system. The infrastructure notes explained here for 'shared' apply to vmfs. Then please follow to the specific *VMFS storage guide here*.
- `ssh`, uses a local storage area from each host for the system datastore

The Shared System Datastore

The shared transfer driver requires the hosts to share the system datastore directory (it does not need to be shared with the front-end). Typically these storage areas are shared using a distributed FS like NFS, GlusterFS, Lustre, etc.

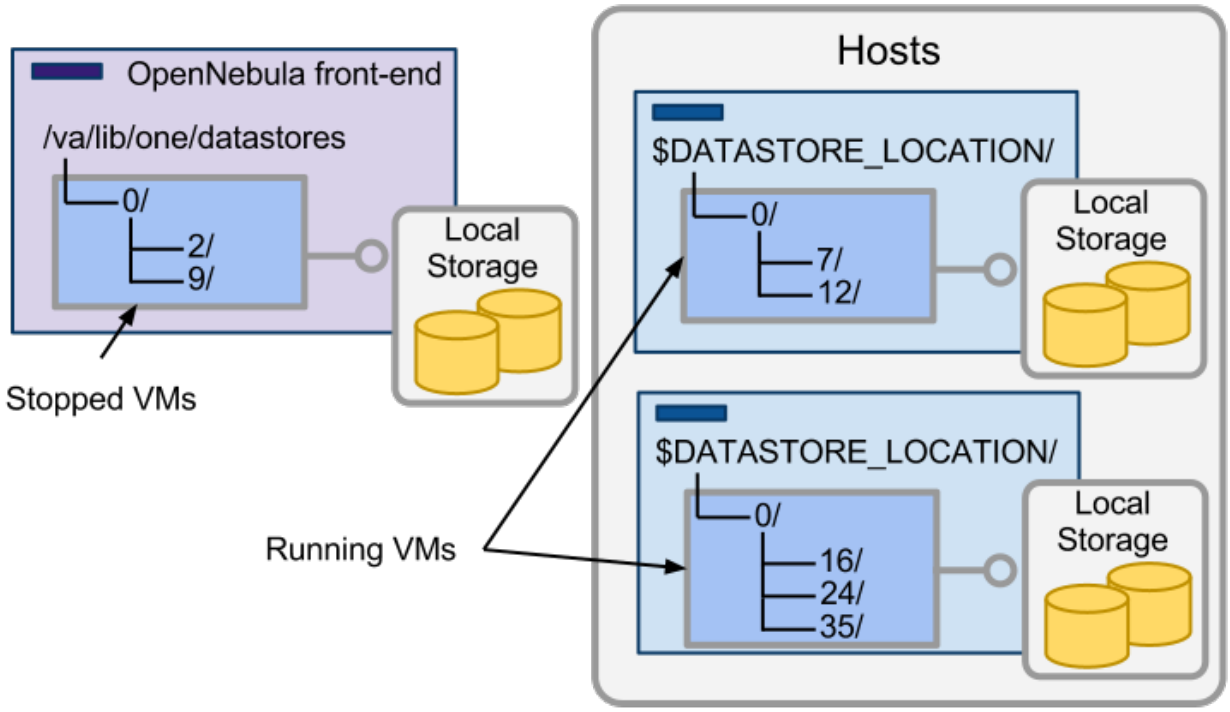
A shared system datastore usually reduces VM deployment times and **enables live-migration**, but it can also become a bottleneck in your infrastructure and degrade your VMs performance if the virtualized services perform disk-intensive workloads. Usually this limitation may be overcome by:

- Using different filesystem servers for the images datastores, so the actual I/O bandwith is balanced
- Using an ssh system datastore instead, the images are copied locally to each host
- Tuning or improving the filesystem servers



The SSH System Datastore

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 VMs (as opposed to a shared FS in the shared driver). 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.



The System and Image Datastores

OpenNebula will automatically transfer VM disk images to/from the system datastore when a VM is booted or shutdown. The actual transfer operations and the space taken from the system datastore depends on both the image configuration (persistent vs non-persistent) as well as the drivers used by the images datastore. The following table summarizes the actions performed by each transfer manager driver type.

Image Type	shared	ssh	qcow2	vmfs	ceph	lvm	shared lvm
Persistent	link	copy	link	link	link	link	lv copy
Non-persistent	copy	copy	snapshot	cp	rdb copy+	lv copy+	lv copy
Volatile	new	new	new	new	new	new	new

In the table above:

- **link** is the equivalent to a symbolic link operation that will not take any significant amount of storage from the system datastore
- **copy, rbd copy and lv copy**, are copy operations as in regular cp file operations, that may involve creation of special devices like a logical volume. This will take the same size as the original image.
- **snapshot**, qcow2 snapshot operation.
- **new**, a new image file is created on the system datastore of the specified size.

Important Note, operations with +, are performed on the original image datastore; an so those operations take storage from the image datastore and not from the system one.

Once the disk images are transferred from the image datastore to the system datastore using the operations described above, the system datastore (and its drivers) is responsible for managing the images, mainly to:

- Move the images across hosts, e.g. when the VM is stopped or migrated
- Delete any copy from the hosts when the VM is shutdown

2.2.2 Configuration Overview

You need to configure one or more system datastores for each of your *clusters*. In this way you can better plan the storage requirements, in terms of total capacity assigned, performance requirements and load balancing across system datastores. Note that hosts not assigned to a cluster can still use system datastores that are neither assigned to a cluster.

To configure the system datastores for your OpenNebula cloud you need to:

- Create as many system datastores as needed (you can add more later if you need them)
- Assign the system datastores to a given cluster
- Configure the cluster hosts to access the system datastores

2.2.3 Step 1. Create a New System Datastore

To create a new system datastore you need to specify its type as system either in Sunstone (system) or through the CLI (adding TYPE = SYSTEM_DS to the datastore template). And you need to select the system datastore drivers, as discussed above: `shared`, `vmfs` and `ssh`.

For example to create a system datastore using the shared drivers simply:

```
$ cat system.ds
NAME    = nfs_ds
TM_MAD  = shared
TYPE    = SYSTEM_DS

$ onedatastore create system.ds
ID: 100
```

2.2.4 Step 2. Assign the System Datastores

Hosts can only use use a system datastore if they are in the same cluster, so once created you need to add the system datastores to the cluster. You can **add more than one system datastore** to a cluster, the actual system DS used to deploy the VM will be selected based on storage scheduling policies, see below.

Warning: Host not associated to a cluster will also use system datastores not associated to a cluster. If you are not using clusters you can skip this section.

To associate this system datastore to the cluster, add it:

```
$ onecluster adddatastore production_cluster nfs_ds
```

As we'll see shortly, hosts need to be configured to access the systems datastore through a well-known location, that defaults to `/var/lib/one/datastores`. You can also override this setting for the hosts of a cluster using the `DATASTORE_LOCATION` attribute. It can be changed with the `onecluster update` command.

```
$ onecluster update production_cluster
#Edit the file to read as:
DATASTORE_LOCATION=/path/to/datastores/
```

Warning: DATASTORE_LOCATION defines the path to access the datastores in the hosts. It can be defined for each cluster, or if not defined for the cluster the default in oned.conf will be used.

Warning: When needed, the front-end will access the datastores at `/var/lib/one/datastores`, this path cannot be changed, you can link each datastore directory to a suitable location

2.2.5 Step 3. Configure the Hosts

The specific configuration for the hosts depends on the system datastore type (shared or ssh). Before continuing check that SSH is configured to enable oneadmin passwordless access in every host.

Configure the Hosts for the Shared System Datastore

A NAS has to be configured to export a directory to the hosts, this directory will be used as the storage area for the system datastore. Each host has to mount this directory under `$DATASTORE_LOCATION/<ds_id>`. In small installations the front-end can be also used to export the system datastore directory to the hosts. Although this deployment is not recommended for medium-large size deployments.

Warning: It is not needed to mount the system datastore in the OpenNebula front-end as `/var/lib/one/datastores/<ds_id>`

Configure the Hosts for the SSH System Datastore

There is no special configuration needed to take place to use the ssh drivers for the system datastore. Just be sure that there is enough space under `$DATASTORE_LOCATION` to hold the images of the VMs that will run in each particular host.

Also be sure that there is space in the frontend under `/var/lib/one/datastores/<ds_id>` to hold the images of the stopped or undeployed VMs.

2.2.6 Multiple System Datastore Setups

In order to distribute efficiently the I/O of the VMs across different disks, LUNs or several storage backends, OpenNebula is able to define multiple system datastores per cluster. Scheduling algorithms take into account disk requirements of a particular VM, so OpenNebula is able to pick the best execution host based on capacity and storage metrics.

Admin Perspective

For an admin, it means that she would be able to decide which storage policy to apply for the whole cloud she is administering, that will then be used to chose which system datastore is more suitable for a certain VM.

When more than one system datastore is added to a cluster, all of them can be taken into account by the scheduler to place VMs into.

System scheduling policies are defined in `/etc/one/sched.conf`. These are the defaults the scheduler would use if the VM template doesn't state otherwise. The possibilities are described here:

- **Packing**. Tries to optimize storage usage by selecting the datastore with less free space.
- **Striping**. Tries to optimize I/O by distributing the VMs across datastores.
- **Custom**. Based on any of the attributes present in the datastore template.

To activate for instance the Stripping storage policy, `/etc/one/sched.conf` must contain:

```
DEFAULT_DS_SCHED = [
   policy = 1
]
```

Warning: Any host belonging to a given cluster **must** be able to access any system or image datastore defined in that cluster.

User Perspective

For a user, OpenNebula's ability to handle multiples datastore means that she would be able to require for its VMs to be run on a system datastore backed by a fast storage cabin, or run on the host with a datastore with the most free space available. This choice is obviously limited to the underlying hardware and the administrator configuration.

This control can be exerted within the VM template, with two attributes:

Attribute	Description	Examples
SCHED_DS_REQUIREMENTS	Boolean expression that rules out entries from the pool of datastores suitable to run this VM.	**SCHED_DS_REQUIREMENTS="ID=100"** SCHED_DS_REQUIREMENTS="NAME=Gc SCHED_DS_REQUIREMENTS=FREE_MB > 250000)
SCHED_DS_RANK	States which attribute will be used to sort the suitable datastores for this VM. Basically, it defines which datastores are more suitable than others.	**SCHED_DS_RANK= FREE_MB** SCHED_DS_RANK=- FREE_MB

Warning: Admins and user with admins rights can force the deployment to a certain datastore, using 'onevm deploy' command.

2.2.7 Tuning and Extending

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

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

- /var/lib/one/remotes/datastore/<DS_DRIVER>
- /var/lib/one/remotes/tm/<TM_DRIVER>

2.3 The Filesystem Datastore

The Filesystem datastore lets you store VM images in a file form. The datastore is format agnostic, so you can store any file-type depending on the target hypervisor. The use of file-based disk images presents several benefits over deviced backed disks (e.g. easily backup images, or use of shared FS) although it may less performing in some cases.

Usually it is a good idea to have multiple filesystem datastores to:

- Group images of the same type, so you can have a qcow datastore for KVM hosts and a raw one for Xen
- Balance I/O operations, as the datastores can be in different servers

- Use different datastores for different cluster hosts
- Apply different QoS policies to different images

2.3.1 Requirements

There are no special requirements or software dependencies to use the filesystem datastore. The drivers make use of standard filesystem utils (cp, ln, mv, tar, mkfs...) that should be installed in your system.

2.3.2 Configuration

Configuring the System Datastore

Filesystem datastores can work with a system datastore that uses either the shared or the SSH transfer drivers, note that:

- Shared drivers for the **system** datastore enables live-migrations, but it could demand a high-performance SAN.
- SSH drivers for the **system** datastore may increase deployment/shutdown times but all the operations are performed locally, so improving performance in general.

See more details on the *System Datastore Guide*

Configuring the FileSystem Datastores

The first step to create a filesystem datastore is to set up a template file for it. In the following table you can see the valid configuration attributes for a filesystem datastore. The datastore type is set by its drivers, in this case be sure to add `DS_MAD=fs`.

The other important attribute to configure the datastore is the transfer drivers. These drivers determine how the images are accessed in the hosts. The Filesystem datastore can use shared, ssh and qcow2. See below for more details.

Attribute	Description	Values
`NAME`	The name of the datastore	N/A
`DS_MAD`	The DS type, use `fs` for the Filesystem datastore	ceph, dummy, fs, iscsi, lvm, vmfs
`TM_MAD`	Transfer drivers for the datastore: `shared`, `ssh` or `qcow2`, see below	ceph, dummy, iscsi, lvm, qcow2, shared, ssh, vmfs
`RESTRICTED_DIRS`	Paths that can not be used to register images. A space separated list of paths.	N/A
`SAFE_DIRS`	If you need to un-block a directory under one of the RESTRICTED_DIRS. A space separated list of paths.	N/A
`NO_DECOMPRESS`	Do not try to untar or decompress the file to be registered. Useful for specialized Transfer Managers. Use value `yes` to disable decompression.	yes
`LIMIT_TRANSFER_BW`	Specify the maximum transfer rate in bytes/second when downloading images from a http/https URL. Suffixes K, M or G can be used.	N/A
`DATASTORE_CAPACITY_CHECK`	If `yes`, the available capacity of the datastore is checked before creating a new image	yes
`BASE_PATH`	Base path to build the path of the Datastore Images. This path is used to store the images when they are created in the datastore	N/A

Note: The `RESTRICTED_DIRS` directive will prevent users registering important files as VM images and accessing them thourgh their VMs. OpenNebula will automatically add its configuration directories: /var/lib/one, /etc/one and oneadmin's home. If users try to register an image from a restricted directory, they will get the following error message: `Not allowed to copy image file`.

For example, the following illustrates the creation of a filesystem datastore using the shared transfer drivers.

```
> cat ds.conf
NAME = production
DS_MAD = fs
TM_MAD = shared

> onedatastore create ds.conf
ID: 100

> onedatastore list
  ID NAME            CLUSTER  IMAGES TYPE   TM
   0 system          none     0      fs     shared
   1 default         none     3      fs     shared
 100 production      none     0      fs     shared
```

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.

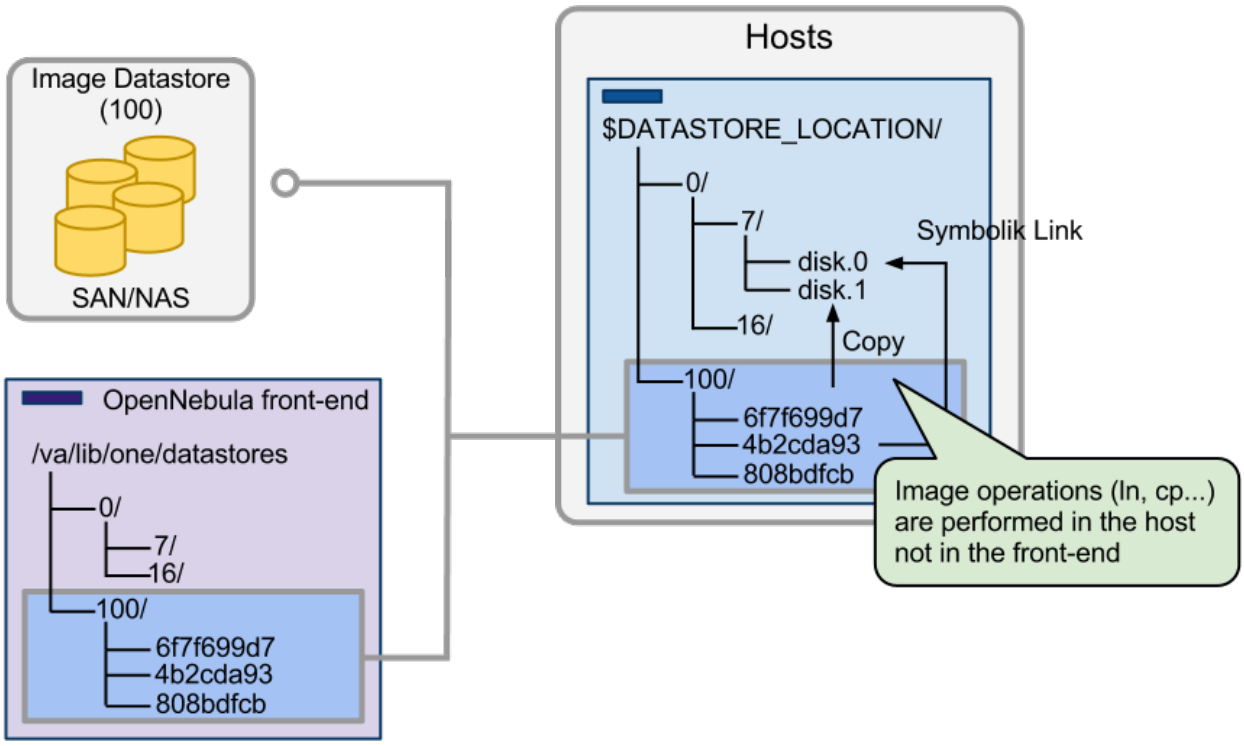
Finally, you have to prepare the storage for the datastore and configure the hosts to access it. This depends on the transfer mechanism you have chosen for your datastore.

After creating a new datastore the LN_TARGET and CLONE_TARGET parameters will be added to the template. These values should not be changed since they define the datastore behaviour. The default values for these parameters are defined in *oned.conf* for each driver.

Warning: Note that datastores are not associated to any cluster by default, and their are supposed to be accessible by every single host. If you need to configure datastores for just a subset of the hosts take a look to the *Cluster guide*.

2.3.3 Using the Shared Transfer Driver

The shared transfer driver assumes that the datastore is mounted in all the hosts of the cluster. 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.



Persistent & Non Persistent Images

If the VM uses a persistent image, a symbolic link to the datastore is created in the corresponding directory of the system datastore. Non-persistent images are copied instead. For persistent images, this allows an immediate deployment, and no extra time is needed to save the disk back to the datastore when the VM is shut down.

On the other hand, the original file is used directly, and if for some reason the VM fails and the image data is corrupted or lost, there is no way to cancel the persistence.

Finally images created using the 'onevm disk-snapshot' command will be moved to the datastore only after the VM is successfully shut down. This means that the VM has to be shutdown using the 'onevm shutdown' command, and not 'onevm delete'. Suspending or stopping a running VM won't copy the disk file to the datastore either.

Host Configuration

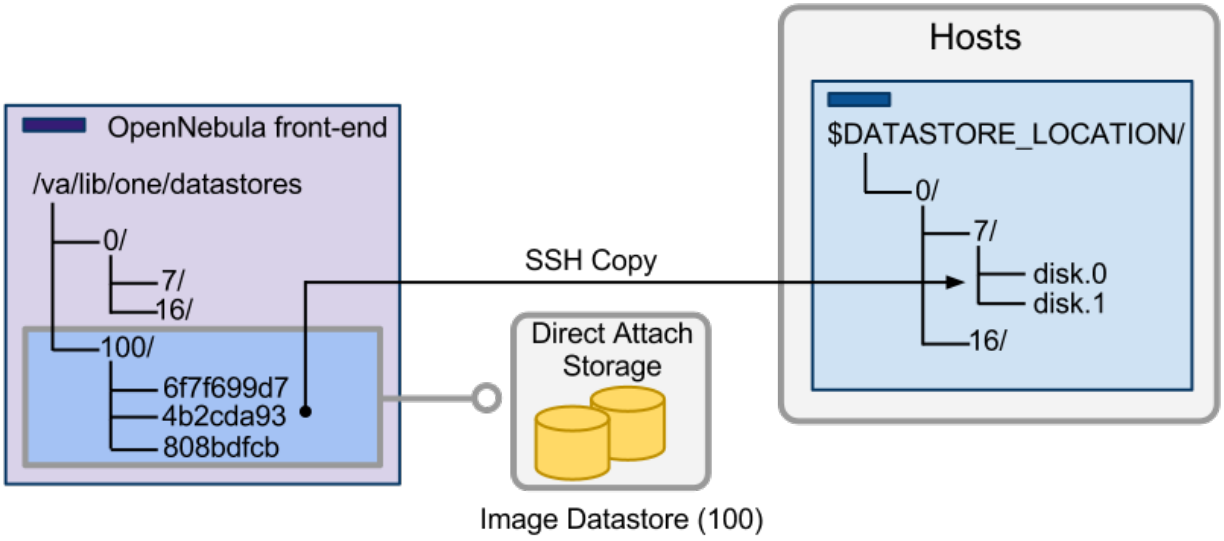
Each host has to mount the datastore under `$DATASTORE_LOCATION/<datastore_id>`. You also have to mount the datastore in the front-end in `/var/lib/one/datastores/<datastore_id>`.

Warning: DATASTORE_LOCATION defines the path to access the datastores in the hosts. It can be defined for each cluster, or if not defined for the cluster the default in oned.conf will be used.

Warning: When needed, the front-end will access the datastores using BASE_PATH (defaults to `/var/lib/one/datastores`). You can set the BASE_PATH for the datastore at creation time.

2.3.4 Using the SSH Transfer Driver

In this case the datastore is only directly accessed by the front-end. VM images are copied from/to the datastore using the SSH protocol. This may impose high VM deployment times depending on your infrastructure network connectivity.



Persistent & Non Persistent Images

In either case (persistent and non-persistent) images are always copied from the datastore to the corresponding directory of the system datastore in the target host.

If an image is persistent (or for the matter of fact, created with the 'onevm disk-snapshot' command), it is transferred back to the Datastore only after the VM is successfully shut down. This means that the VM has to be shut down using the 'onevm shutdown' command, and not 'onevm delete'. Note that no modification to the image registered in the datastore occurs till that moment. Suspending or stopping a running VM won't copy/modify the disk file registered in the datastore either.

Host Configuration

There is no special configuration for the hosts in this case. Just make sure that there is enough space under `$DATASTORE_LOCATION` to hold the images of the VMs running in that host.

2.3.5 Using the qcow2 Transfer driver

The qcow2 drivers are a specialization of the shared drivers to work with the qcow2 format for disk images. The same features/restrictions and configuration applies so be sure to read the shared driver section.

The following list details the differences:

- Persistent images are created with the `qemu-img` command using the original image as backing file
- When an image has to be copied back to the datastore the `qemu-img convert` command is used instead of a direct copy

2.3.6 Tuning and Extending

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

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

- /var/lib/one/remotes/datastore/`<DS_DRIVER>`
- /var/lib/one/remotes/tm/`<TM_DRIVER>`

2.4 The VMFS Datastore

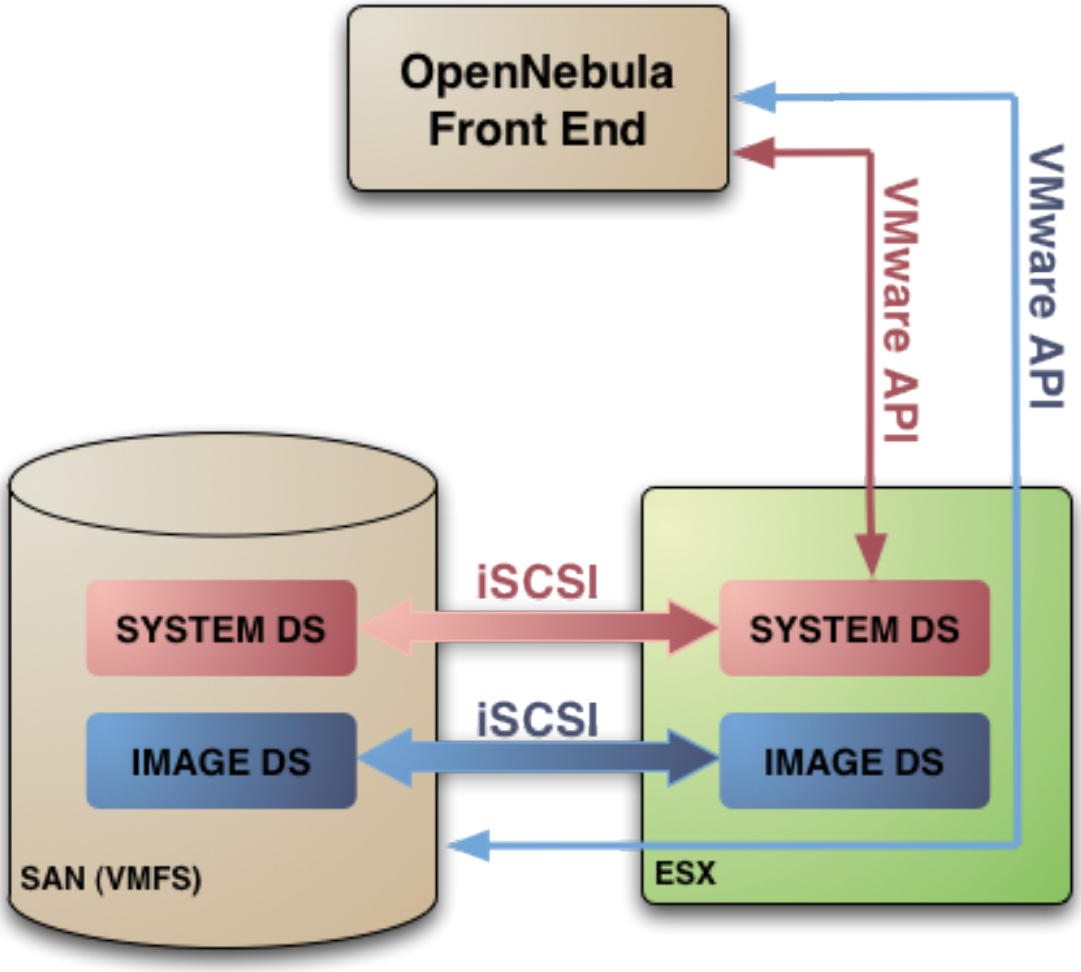
In order to use VMware hypervisors in your OpenNebula cloud you will need to use **VMFS Datastores**. To configure them, it is important to keep in mind that there are (at least) two datastores to define, the `system datastore` (where the running VMs and their images reside, only need transfer manager drivers) and the `images datastore` (where the images are stored, needs both datastore and transfer manager drivers).

2.4.1 Requirements

- In order to use the VMFS datastore, the ESX servers need to have the SSH access configured for the oneadmin account.
- If the VMFS volumes are exported through a SAN, it should be accesible and configured so the ESX server can mount the iSCSI export.

2.4.2 Description

This storage model implies that all the volumes involved in the image staging are purely VMFS volumes, taking full advantage of the VMware filesystem (VM image locking and improved performance).



2.4.3 Infrastructure Configuration

- The OpenNebula front-end doesn't need to mount any datastore.
- The ESX servers needs to present or mount (as iSCSI, NFS or local storage) both the `system` datastore and the `image` datastore (naming them with just the <datastore-id>, for instance `0` for the `system` datastore and `1` for the `image` datastore).

Warning: The system datastore can be other than the default one (`0`). In this case, the ESX will need to mount the datastore *with the same id as the datastores has in OpenNebula*. More details in the *System Datastore Guide*.

2.4.4 OpenNebula Configuration

The datastore location on ESX hypervisors is "/vmfs/volumes". There are two choices:

- In homogeneous clouds (all the hosts are ESX) set the following in /etc/one/oned.conf:

```
DATASTORE_LOCATION=/vmfs/volumes
```

- In heterogeneous clouds (mix of ESX and other hypervisor hosts) put all the ESX hosts in clusters with the following attribute in their template (e.g. onecluster update):

```
DATASTORE_LOCATION=/vmfs/volumes
```

Warning: You need also to set the BASE_PATH attribute in the template when the Datastore is created.

Datastore Configuration

The `system` and `images` datastores needs to be configured with the following drivers:

Datastore	DS Drivers	TM Drivers
System	•	vmfs
Images	vmfs	vmfs

System Datastore

vmfs drivers: the `system` datastore needs to be updated in OpenNebula (`onedatastore update <ds_id>`) to set the TM_MAD drivers to `vmfs`. There is no need to configure datastore drivers for the system datastore.

OpenNebula expects the system datastore to have the ID=0, but a system datastore with different ID can be defined per cluster. See the *system datastore guide* for more details.

Images Datastore

The `image` datastore needs to be updated to use **vmfs** drivers for the datastore drivers, and **vmfs** drivers for the transfer manager drivers. The default datastore can be updated as:

```
$ onedatastore update 1
DS_MAD=vmfs
TM_MAD=vmfs
BRIDGE_LIST=<space-separated list of ESXi host>
```

Apart from DS_MAD, TM_MAD and BRIDGE_LIST; the following attributes can be set:

Attribute	Description
`NAME`	The name of the datastore
`DS_MAD`	The DS type, use `vmware` or `vmfs`
`TM_MAD`	Transfer drivers for the datastore: `shared`, `ssh` or `vmfs`, see below
`RESTRICTED_DIRS`	Paths that can not be used to register images. A space separated list of paths.
`SAFE_DIRS`	If you need to un-block a directory under one of the RESTRICTED_DIRS. A space separated list of paths.
`UMASK`	Default mask for the files created in the datastore. Defaults to `0007`
`BRIDGE_LIST`	Space separated list of ESX servers that are going to be used as proxies to stage images into the datastore (`vmfs` datastores only)
`DS_TMP_DIR`	Path in the OpenNebula front-end to be used as a buffer to stage in files in `vmfs` datastores. Defaults to the value in `/var/lib/one/remotes/datastore/vmfs/vmfs.conf`.
`NO_DECOMPRESS`	Do not try to untar or decompress the file to be registered. Useful for specialized Transfer Managers
`DATASTORE_CAPACITY_CHECK`	If `yes`, the available capacity of the datastore is checked before creating a new image
`BASE_PATH`	This variable must be set to /vmfs/volumes for VMFS datastores.

Warning: `RESTRICTED_DIRS` will prevent users registering important files as VM images and accessing them through their VMs. OpenNebula will automatically add its configuration directories: /var/lib/one, /etc/one and oneadmin's home. If users try to register an image from a restricted directory, they will get the following error message: "Not allowed to copy image file".

After creating a new datastore the LN_TARGET and CLONE_TARGET parameters will be added to the template. These values should not be changed since they define the datastore behaviour. The default values for these parameters are defined in *oned.conf* for each driver.

Driver Configuration

Transfer Manager Drivers

These drivers trigger the events remotely through an ssh channel. The **vmfs** drivers are a specialization of the shared drivers to work with the VMware vmdk filesystem tools using the `vmkfstool` command. This comes with a number of advantages, like FS locking, easier VMDK cloning, format management, etc.

Datastore Drivers

The **vmfs** datastore drivers allows the use of the VMware VM filesystem, which handles VM file locks and also boosts I/O performance.

- To correctly configure a `vmfs` datastore set of drivers there is the need to chose the ESX bridges, i.e., the ESX serves that are going to be used as proxies to stage images into the `vmfs` datastore. A list of bridges **must** be defined with the `BRIDGE_LIST` attribute of the datastore template (see the table below). The drivers will pick one ESX server from that list in a round robin fashion.
- The `vmfs` datastore needs to use the front-end as a buffer for the image staging in some cases, this buffer can be set in the `DS_TMP_DIR` attribute.

2.4.5 Tuning and Extending

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

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

- /var/lib/one/remotes/datastore/`<DS_DRIVER>`
- /var/lib/one/remotes/tm/`<TM_DRIVER>`

2.5 LVM Drivers

The LVM datastore driver provides OpenNebula with the possibility of using LVM volumes instead of plain files to hold the Virtual Images. This reduces the overhead of having a file-system in place and thus increases performance.

2.5.1 Overview

OpenNebula ships with two sets of LVM drivers:

- **FS LVM**, file based VM disk images with Logical Volumes (LV), using the `fs_lvm` drivers

- **Block LVM**, pure Logical Volume (LV), using the `lvm` drivers

In both cases Virtual Machine will run from Logical Volumes in the host, and they both require cLVM in order to provide live-migration.

However there are some differences, in particular the way non active images are stored, and the name of the Volume Group where they are executed.

This is a brief description of both drivers:

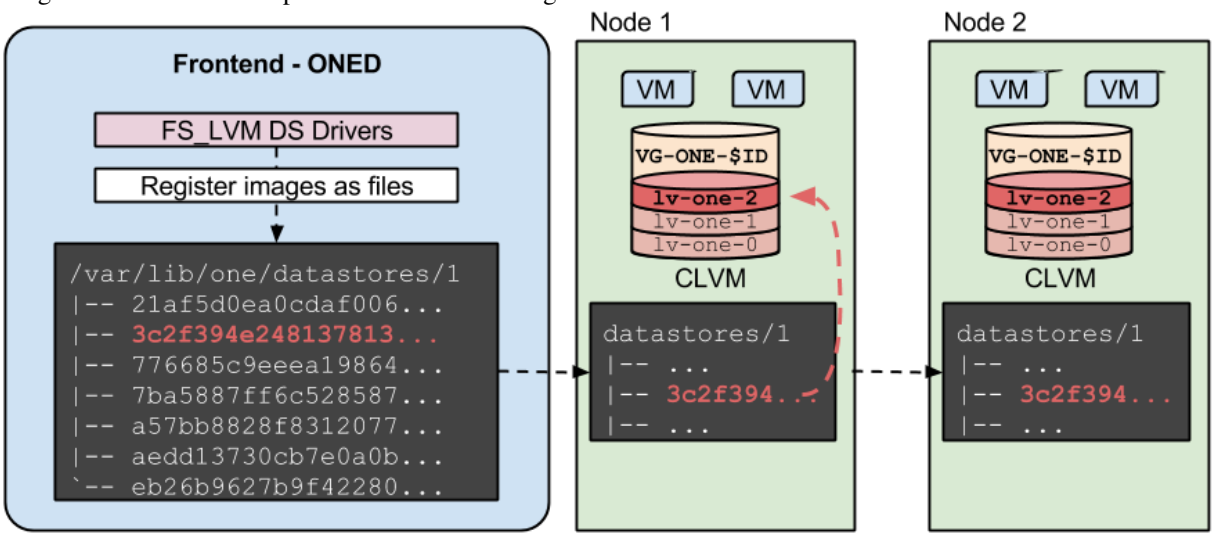
2.5.2 FS LVM

In a FS LVM datastore using the fs_lvm drivers (the now recommended LVM drivers), images are registered as files in a shared FS volume, under the usual path: `/var/lib/one/datastores/<id>`.

This directory needs to be accessible in the worker nodes, using NFS or any other shared/distributed file-system.

When a Virtual Machine is instantiated OpenNebula will dynamically select the system datastore. Let's assume for instance the selected datastore is `104`. The virtual disk image will be copied from the stored image file under the `datastores` directory and dumped into a LV under the Volume Group: `vg-one-104`. It follows that each node **must** have a cluster-aware LVM Volume Group for every possible system datastore it may execute.

This set of drivers brings precisely the advantage of dynamic selection of the system datastore, allowing therefore more granular control of the performance of the storage backend.

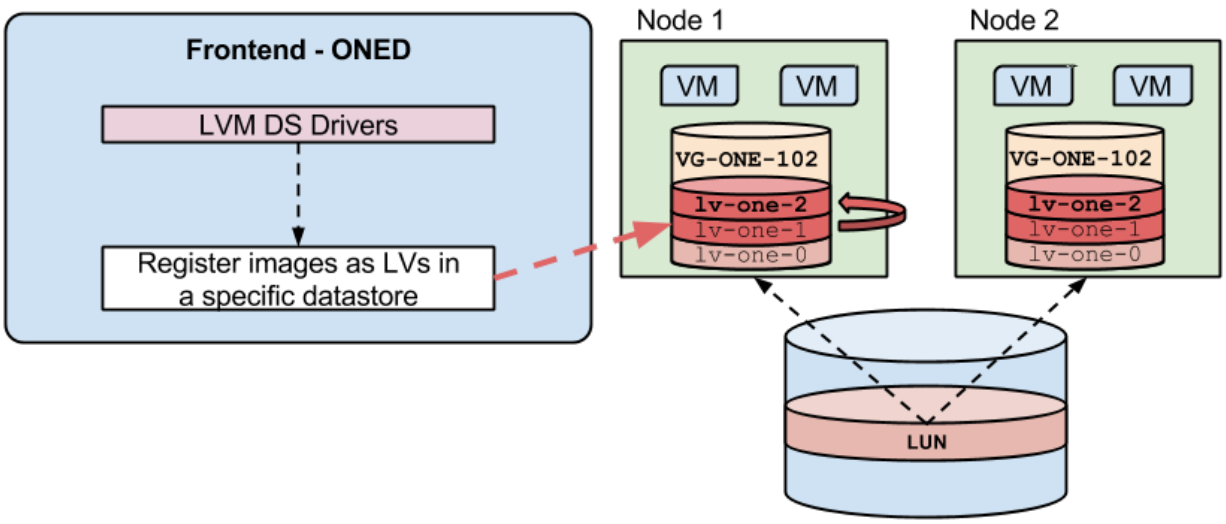


Read more

2.5.3 Block LVM

The Block LVM datastore use the `lvm` drivers with the classical approach to using LVM in OpenNebula.

When a new datastore that uses this set of drivers is created, it requires the VG_NAME parameter, which will tie the images to that Volume Group. Images will be registered directly as Logical Volumes in that Volume Group (as opposed to being registered as files in the frontend), and when they are instantiated the new cloned Logical Volume will also be created in that very same Volume Group.



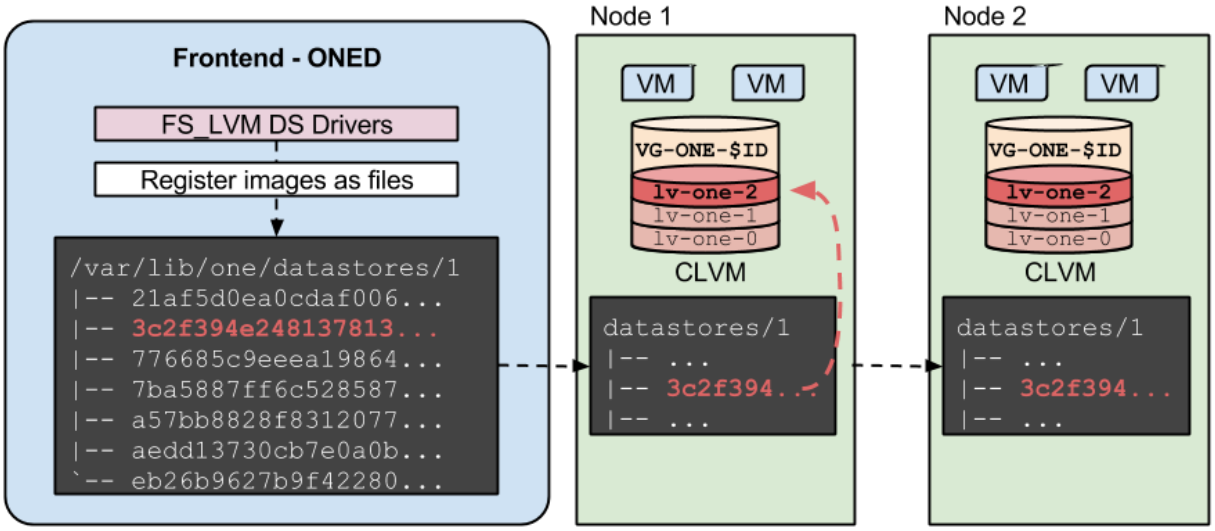
Read more

2.6 The FS LVM Datastore

2.6.1 Overview

The FS LVM datastore driver provides OpenNebula with the possibility of using LVM volumes instead of plain files to hold the Virtual Images.

It is assumed that the OpenNebula hosts using this datastore will be configured with CLVM, therefore modifying the OpenNebula Volume Group in one host will reflect in the others.



2.6.2 Requirements

OpenNebula Front-end

- Password-less ssh access to an OpenNebula LVM-enabled host.

OpenNebula LVM Hosts

LVM must be available in the Hosts. The `oneadmin` user should be able to execute several LVM related commands with sudo passwordlessly.

- Password-less sudo permission for: `lvremove`, `lvcreate`, `lvs`, `vgdisplay` and `dd`.
- LVM2
- `oneadmin` needs to belong to the `disk` group (for KVM).

2.6.3 Configuration

Configuring the System Datastore

To use LVM drivers, the system datastore **must** be `shared`. This sytem datastore will hold only the symbolic links to the block devices, so it will not take much space. See more details on the *System Datastore Guide*

It will also be used to hold context images and Disks created on the fly, they will be created as regular files.

It is worth noting that running virtual disk images will be created in Volume Groups that are hardcoded to be `vg-one-<system_ds_id>`. Therefore the nodes **must** have those Volume Groups pre-created and available for **all** possible system datastores.

Configuring LVM Datastores

The first step to create a LVM datastore is to set up a template file for it. In the following table you can see the supported configuration attributes. The datastore type is set by its drivers, in this case be sure to add `DS_MAD=fs` and `TM_MAD=fs_lvm` for the transfer mechanism, see below.

Attribute	Description
`NAME`	The name of the datastore
`DS_MAD`	Must be `fs`
`TM_MAD`	Must be `fs_lvm`
`DISK_TYPE`	Must be `block`
`RESTRICTED_DIRS`	Paths that can not be used to register images. A space separated list of paths.
`SAFE_DIRS`	If you need to un-block a directory under one of the RESTRICTED_DIRS. A space separated list of paths.
`BRIDGE_LIST`	**Mandatory** space separated list of LVM frontends.
`NO_DECOMPRESS`	Do not try to untar or decompress the file to be registered. Useful for specialized Transfer Managers
`LIMIT_TRANSFER_BW`	Specify the maximum transfer rate in bytes/second when downloading images from a http/https URL. Suffixes K, M or G can be used.
`DATASTORE_CAPACITY_CHECK`	If "yes", the available capacity of the datastore is checked before creating a new image

Note: The `RESTRICTED_DIRS` directive will prevent users registering important files as VM images and accessing them through their VMs. OpenNebula will automatically add its configuration directories: /var/lib/one, /etc/one and

oneadmin's home. If users try to register an image from a restricted directory, they will get the following error message: **Not allowed to copy image file**.

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

> onedatastore create ds.conf
ID: 100

> onedatastore list
  ID NAME            CLUSTER  IMAGES TYPE   TM
   0 system          none     0      fs     shared
   1 default         none     3      fs     shared
 100 production      none     0      fs     fs_lvm
```

Note: Datastores are not associated to any cluster by default, and they are supposed to be accessible by every single host. If you need to configure datastores for just a subset of the hosts take a look to the *Cluster guide*.

After creating a new datastore the LN_TARGET and CLONE_TARGET parameters will be added to the template. These values should not be changed since they define the datastore behaviour. The default values for these parameters are defined in *oned.conf* for each driver.

Host Configuration

The hosts must have LVM2 and **must** have a Volume-Group for every possible system-datastore that can run in the host. CLVM must also be installed and active accross all the hosts that use this datastore.

It's also required to have password-less sudo permission for: `lvremove`, `lvcreate`, `lvs` and `dd`.

2.6.4 Tuning & Extending

System administrators and integrators are encouraged to modify these drivers in order to integrate them with their datacenter:

Under `/var/lib/one/remotes/`:

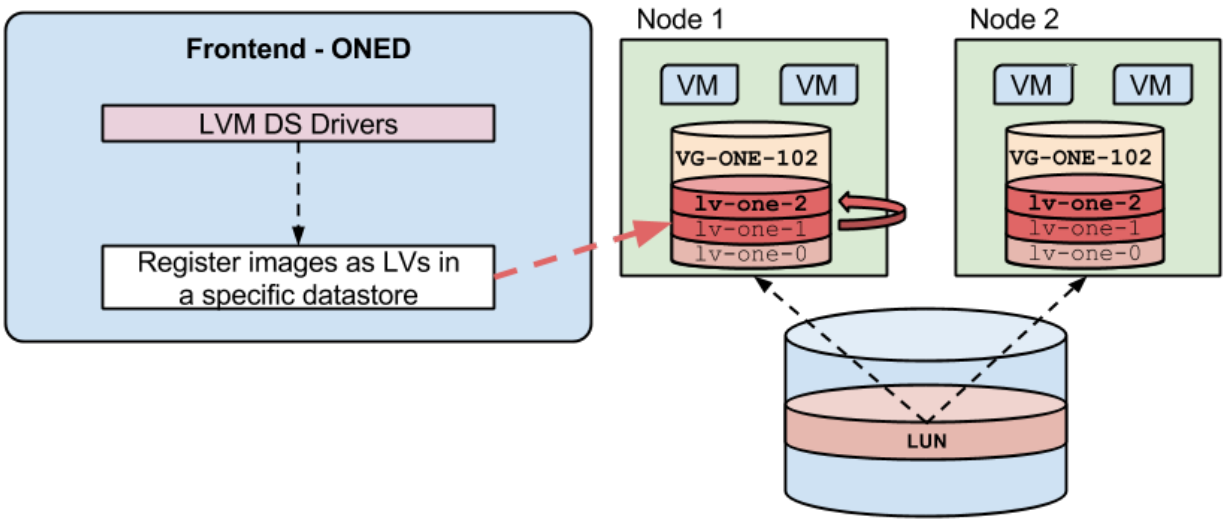
- **tm/fs_lvm/ln**: Links to the LVM logical volume.
- **tm/fs_lvm/clone**: Clones the image by creating a snapshot.
- **tm/fs_lvm/mvds**: Saves the image in a new LV for SAVE_AS.
- **tm/fs_lvm/cpds**: Saves the image in a new LV for SAVE_AS while VM is running.

2.7 The Block LVM Datastore

2.7.1 Overview

The Block LVM datastore driver provides OpenNebula with the possibility of using LVM volumes instead of plain files to hold the Virtual Images.

It is assumed that the OpenNebula hosts using this datastore will be configured with CLVM, therefore modifying the OpenNebula Volume Group in one host will reflect in the others. There is a special list of hosts (BRIDGE_LIST) which belong to the LVM cluster, that will be the ones OpenNebula uses to speak to when doing LVM operations.



2.7.2 Requirements

OpenNebula Front-end

- Password-less ssh access to an OpenNebula LVM-enabled host.

OpenNebula LVM Hosts

LVM must be available in the Hosts. The `oneadmin` user should be able to execute several LVM related commands with sudo passwordlessly.

- Password-less sudo permission for: `lvremove`, `lvcreate`, `lvs`, `vgdisplay` and `dd`.
- LVM2
- `oneadmin` needs to belong to the `disk` group (for KVM).

2.7.3 Configuration

Configuring the System Datastore

To use LVM drivers, the system datastore will work both with `shared` or as `ssh`. This sytem datastore will hold only the symbolic links to the block devices, so it will not take much space. See more details on the *System Datastore Guide*

It will also be used to hold context images and Disks created on the fly, they will be created as regular files.

Configuring Block LVM Datastores

The first step to create a LVM datastore is to set up a template file for it. In the following table you can see the supported configuration attributes. The datastore type is set by its drivers, in this case be sure to add `DS_MAD=lvm` and `TM_MAD=lvm` for the transfer mechanism, see below.

Attribute	Description
`NAME`	The name of the datastore
`DS_MAD`	Must be `lvm`
`TM_MAD`	Must be `lvm`
`DISK_TYPE`	Must be `block`
`BRIDGE_LIST`	List of LVM server hosts (any of them since it assumes they have CLVM).
`VG_NAME`	The LVM volume group name. Defaults to `vg-one`
`BRIDGE_LIST`	**Mandatory** space separated list of LVM frontends.
`RESTRICTED_DIRS`	Paths that can not be used to register images. A space separated list of paths.
`SAFE_DIRS`	If you need to un-block a directory under one of the RESTRICTED_DIRS. A space separated list of paths.
`NO_DECOMPRESS`	Do not try to untar or decompress the file to be registered. Useful for specialized Transfer Managers
`LIMIT_TRANSFER_BW`	Specify the maximum transfer rate in bytes/second when downloading images from a http/https URL. Suffixes K, M or G can be used.
`DATASTORE_CAPACITY_CHECK`	If "yes", the available capacity of the datastore is checked before creating a new image

Warning: `RESTRICTED_DIRS` will prevent users registering important files as VM images and accessing them through their VMs. OpenNebula will automatically add its configuration directories: /var/lib/one, /etc/one and oneadmin's home. If users try to register an image from a restricted directory, they will get the following error message: `Not allowed to copy image file`.

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 = lvm
TM_MAD = lvm
VG_NAME = vg-one
HOST = host01

> onedatastore create ds.conf
ID: 100

> onedatastore list
  ID NAME            CLUSTER  IMAGES TYPE   TM
   0 system          none     0      fs     shared
   1 default         none     3      fs     shared
 100 production      none     0      lvm  shared
```

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.

Warning: Note that datastores are not associated to any cluster by default, and they are supposed to be accessible by every single host. If you need to configure datastores for just a subset of the hosts take a look to the *Cluster guide*.

After creating a new datastore the LN_TARGET and CLONE_TARGET parameters will be added to the template. These values should not be changed since they define the datastore behaviour. The default values for these parameters are defined in *oned.conf* for each driver.

Host Configuration

The hosts must have LVM2 and have the Volume-Group used in the `VG_NAME` attributed of the datastore template. CLVM must also be installed and active accross all the hosts that use this datastore.

It's also required to have password-less sudo permission for: `lvremove`, `lvcreate`, `lvs` and `dd`.

2.7.4 Tuning & Extending

System administrators and integrators are encouraged to modify these drivers in order to integrate them with their datacenter:

Under `/var/lib/one/remotes/`:

- **datastore/lvm/lvm.conf**: Default values for LVM parameters
  - HOST: Default LVM target host
  - VG_NAME: Default volume group
- **datastore/lvm/cp**: Registers a new image. Creates a new logical volume in LVM.
- **datastore/lvm/mkfs**: Makes a new empty image. Creates a new logical volume in LVM.
- **datastore/lvm/rm**: Removes the LVM logical volume.
- **tm/lvm/ln**: Links to the LVM logical volume.
- **tm/lvm/clone**: Clones the image by creating a snapshot.
- **tm/lvm/mvds**: Saves the image in a new LV for SAVE_AS.

2.8 The Ceph Datastore

The Ceph datastore driver provides OpenNebula users with the possibility of using Ceph block devices as their Virtual Images.

Warning: This driver **only** works with libvirt/KVM drivers. Xen is not (yet) supported.

Warning: This driver requires that the OpenNebula nodes using the Ceph driver must be part of a running Ceph cluster. More information in Ceph documentation.

Warning: The hypervisor nodes need to be part of a working Ceph cluster and the Libvirt and QEMU packages need to be recent enough to have support for Ceph. For Ubuntu systems this is available out of the box, however for CentOS systems you will need to manually install this version of qemu-kvm.

2.8.1 Requirements

Ceph Cluster Configuration

The hosts where Ceph datastores based images will be deployed must be part of a running Ceph cluster. To do so refer to the Ceph documentation.

The Ceph cluster must be configured in such a way that no specific authentication is required, which means that for `cephx` authentication the keyring must be in the expected path so that `rbd` and `ceph` commands work without specifying explicitely the keyring's location.

Also the `mon` daemon must be defined in the ceph.conf for all the nodes, so `hostname` and `port` doesn't need to be specified explicitely in any Ceph command.

Additionally each OpenNebula datastore is backed by a ceph pool, these pools must be created and configured in the Ceph cluster. The name of the pool by default is `one` but can be changed on a per-datastore basis (see below).

OpenNebula Ceph Frontend

This driver requires the system administrator to specify one or several Ceph frontends (which need to be nodes in the Ceph cluster) where many of the datastores storage actions will take place. For instance, when creating an image, OpenNebula will choose one of the listed Ceph frontends (using a round-robin algorithm) and transfer the image to that node and run `qemu-img convert -O rbd`. These nodes need to be specified in the `BRIDGE_LIST` section.

Note that this Ceph frontend can be any node in the OpenNebula setup: the OpenNebula frontend, any worker node, or a specific node (recommended).

Ceph Nodes

All the nodes listed in the `BRIDGE_LIST` variable must have`qemu-img` installed.

OpenNebula Hosts

There are no specific requirements for the host, besides being libvirt/kvm nodes, since xen is not (yet) supported for the Ceph drivers.

2.8.2 Configuration

Configuring the System Datastore

To use ceph drivers, the system datastore will work both with `shared` or as `ssh`. This sytem datastore will hold only the symbolic links to the block devices, so it will not take much space. See more details on the *System Datastore Guide*

It will also be used to hold context images and Disks created on the fly, they will be created as regular files.

Configuring Ceph Datastores

The first step to create a Ceph datastore is to set up a template file for it. In the following table you can see the supported configuration attributes. The datastore type is set by its drivers, in this case be sure to add `DS_MAD=ceph` and `TM_MAD=ceph` for the transfer mechanism, see below.

Attribute	Description
NAME	The name of the datastore
DS_MAD	The DS type, use ceph for the Ceph datastore
TM_MAD	Transfer drivers for the datastore, use ceph, see below
DISK_TYPE	The type **must** be RBD
BRIDGE_LIST	**Mandatory** space separated list of Ceph servers that are going to be used as frontends.
POOL_NAME	The OpenNebula Ceph pool name. Defaults to one. **This pool must exist before using the drivers**.
STAGING_DIR	Default path for image operations in the OpenNebula Ceph frontend.
RESTRICTED_DIRS	Paths that can not be used to register images. A space separated list of paths.
SAFE_DIRS	If you need to un-block a directory under one of the RESTRICTED_DIRS. A space separated list of paths.
NO_DECOMPRESS	Do not try to untar or decompress the file to be registered. Useful for specialized Transfer Managers
LIMIT_TRANSFER_BW	Specify the maximum transfer rate in bytes/second when downloading images from a http/https URL. Suffixes K, M or G can be used.
DATASTORE_CAPACITY_CHECK	If yes, the available capacity of the datastore is checked before creating a new image
CEPH_HOST	Space-separated list of Ceph monitors. Example: host1 host2:port2 host3 host4:port4 (if no port is specified, the default one is chosen). **Required for Libvirt 1.x when cephx is enabled** .
CEPH_SECRET	A generated UUID for a LibVirt secret (to hold the CephX authentication key in Libvirt on each hypervisor). This should be generated when creating the Ceph datastore in OpenNebula. **Required for Libvirt 1.x when cephx is enabled** .

Warning: This will prevent users registering important files as VM images and accessing them through their VMs. OpenNebula will automatically add its configuration directories: /var/lib/one, /etc/one and oneadmin's home. If users try to register an image from a restricted directory, they will get the following error message: "Not allowed to copy image file".

For example, the following examples illustrates the creation of an Ceph datastore using a configuration file. In this case we will use the host cephfrontend as one the OpenNebula Ceph frontend The one pool must already exist, if it doesn't create it with:

```
> ceph osd pool create one 128

> ceph osd lspools
0 data,1 metadata,2 rbd,6 one,
```

An example of datastore:

```
> cat ds.conf
NAME = "cephds"
DS_MAD = ceph
TM_MAD = ceph

# the following line *must* be preset
DISK_TYPE = RBD

POOL_NAME = one
BRIDGE_LIST = cephfrontend

> onedatastore create ds.conf
ID: 101

> onedatastore list
  ID NAME            CLUSTER  IMAGES TYPE   TM
```

```
  0 system          none     0      fs     shared
  1 default         none     3      fs     shared
100 cephds          none     0      ceph   ceph
```

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.

Warning: Note that datastores are not associated to any cluster by default, and they are supposed to be accessible by every single host. If you need to configure datastores for just a subset of the hosts take a look to the *Cluster guide*.

After creating a new datastore the LN_TARGET and CLONE_TARGET parameters will be added to the template. These values should not be changed since they define the datastore behaviour. The default values for these parameters are defined in *oned.conf* for each driver.

2.8.3 Ceph Authentication (Cephx)

If Cephx is enabled, there are some special considerations the OpenNebula administrator must take into account.

Create a Ceph user for the OpenNebula hosts. We will use the name `client.libvirt`, but any other name is fine. Create the user in Ceph and grant it rwx permissions on the `one` pool:

```
ceph auth get-or-create client.libvirt mon 'allow r' osd 'allow class-read object_prefix rbd_childre
```

Extract the `client.libvirt` key, save it to a file named `client.libvirt.key`and distribute it to all the KVM hosts:

```
sudo ceph auth list
# save client.libvirt's key to client.libvirt.key
```

Generate a UUID, for example running `uuigden` (the generated uuid will referenced as `%UUID%` from now onwards).

Create a file named `secret.xml` (using the genereated `%UUID%` and distribute it to all the KVM hosts:

```
cat > secret.xml <<EOF
<secret ephemeral='no' private='no'>
  <uuid>%UUID%</uuid>
  <usage type='ceph'>
         <name>client.libvirt secret</name>
  </usage>
</secret>
EOF
```

The following commands must be executed in all the KVM hosts as oneadmin (assuming the `secret.xml` and `client.libvirt.key` files have been distributed to the hosts):

```
# Replace %UUID% with the value generated in the previous step
virsh secret-set-value --secret %UUID% --base64 $(cat client.libvirt.key)
```

Finally, the Ceph datastore must be updated to add the following values:

```
CEPH_USER="libvirt"
CEPH_SECRET="%UUID%"
CEPH_HOST="<list of ceph mon hosts, see table above>"
```

You can read more information about this in the Ceph guide Using libvirt with Ceph.

2.8.4 Using the Ceph Transfer Driver

The workflow for Ceph images is similar to the other datastores, which means that a user will create an image inside the Ceph datastores by providing a path to the image file locally available in the OpenNebula frontend, or to an http url, and the driver will convert it to a Ceph block device.

All the usual operations are avalaible: oneimage create, oneimage delete, oneimage clone, oneimage persistent, oneimage nonpersistent, onevm disk-snapshot, etc...

2.8.5 Tuning & Extending

File Location

System administrators and integrators are encouraged to modify these drivers in order to integrate them with their datacenter:

Under `/var/lib/one/remotes/`:

- **datastore/ceph/ceph.conf**: Default values for ceph parameters
  - HOST: Default OpenNebula Ceph frontend
  - POOL_NAME: Default volume group
  - STAGING_DIR: Default path for image operations in the OpenNebula Ceph frontend.
- **datastore/ceph/cp**: Registers a new image. Creates a new logical volume in ceph.
- **datastore/ceph/mkfs**: Makes a new empty image. Creates a new logical volume in ceph.
- **datastore/ceph/rm**: Removes the ceph logical volume.
- **tm/ceph/ln**: Does nothing since it's handled by libvirt.
- **tm/ceph/clone**: Copies the image to a new image.
- **tm/ceph/mvds**: Saves the image in a Ceph block device for SAVE_AS.
- **tm/ceph/delete**: Removes a non-persistent image from the Virtual Machine directory if it hasn't been subject to a `disk-snapshot` operation.

Using SSH System Datastore

Another option would be to manually patch the post and pre-migrate scripts for the **ssh** system datastore to `scp` the files residing in the system datastore before the live-migration. Read more.

2.9 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.

2.9.1 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.

2.9.2 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...). However, given the special nature of the Files Datastore most of these attributes can be fixed as summarized in the following table:

Attribute	Description
NAME	The name of the datastore
TYPE	Use FILE_DS to setup a Files datastore
DS_MAD	The DS type, use fs to use the file-based drivers
TM_MAD	Transfer drivers for the datastore, use ssh to transfer the files
RESTRICTED_DIRS	Paths that can not be used to register images. A space separated list of paths.
SAFE_DIRS	If you need to un-block a directory under one of the RESTRICTED_DIRS. A space separated list of paths.
LIMIT_TRANSFER_BW	Specify the maximum transfer rate in bytes/second when downloading images from a http/https URL. Suffixes K, M or G can be used.
DATASTORE_CAPACITY_CHECK	If "yes", the available capacity of the datastore is checked before creating a new image

Warning: This will prevent users registering important files as VM images and accessing them thourgh their VMs. OpenNebula will automatically add its configuration directories: /var/lib/one, /etc/one and oneadmin's home. If users try to register an image from a restricted directory, they will get the following error message: "Not allowed to copy image file".

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.

2.9.3 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.

CHAPTER

THREE

VIRTUALIZATION

3.1 Virtualization Overview

The Virtualization Subsystem is the component in charge of talking with the hypervisor installed in the hosts and taking the actions needed for each step in the VM lifecycle.

Configuration options and specific information for each hypervisor can be found in these guides:

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

3.1.1 Common Configuration Options

Drivers 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
-l <actions>	actions executed locally

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

3.1.2 Hypervisor Configuration

A feature supported by both KVM and Xen Hypervisor drivers is selecting the timeout for VM Shutdown. 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 both `/var/lib/one/remotes/vmm/xen/xenrc` and `/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
```

3.2 Xen Driver

The XEN hypervisor offers a powerful, efficient and secure feature set for virtualization of x86, IA64, PowerPC and other CPU architectures. It delivers both paravirtualization and full virtualization. This guide describes the use of Xen with OpenNebula, please refer to the Xen specific documentation for further information on the setup of the Xen hypervisor itself.

3.2.1 Requirements

The Hosts must have a working installation of Xen that includes a Xen aware kernel running in Dom0 and the Xen utilities.

3.2.2 Considerations & Limitations

- Xen HVM currently only supports 4 IDE devices, for more disk devices you should better use SCSI. You have to take this into account when adding disks. See the *Virtual Machine Template documentation* for an explanation on how OpenNebula assigns disk targets.
- OpenNebula manages kernel and initrd files. You are encouraged to register them in the *files datastore*.
- To modify the default disk driver to one that works with your Xen version you can change the files `/etc/one/vmm_exec/vmm_exec_xen*.conf` and `/var/lib/one/remotes/vmm/xen*/xenrc`. Make sure that you have `blktap2` modules loaded to use `tap2:tapdisk:aio`:

```
export IMAGE_PREFIX="tap2:tapdisk:aio"

DISK   = [ driver = "tap2:tapdisk:aio:" ]
```

- If target device is not supported by the linux kernel you will be able to attach disks but not detach them. It is recomended to attach `xvd` devices for xen paravirtualized hosts.

3.2.3 Configuration

Xen Configuration

In each Host you must perform the following steps to get the driver running:

- The remote hosts must have the xend daemon running (`/etc/init.d/xend`) and a `XEN` aware kernel running in Dom0
- The `<oneadmin>` user may need to execute Xen commands using root privileges. This can be done by adding this two lines to the `sudoers` file of the hosts so `<oneadmin>` user can execute Xen commands as root (change paths to suit your installation):

```
%xen    ALL=(ALL) NOPASSWD: /usr/sbin/xm *
%xen    ALL=(ALL) NOPASSWD: /usr/sbin/xentop *
```

- You may also want to configure network for the virtual machines. OpenNebula assumes that the VMs have network access through standard bridging, please refer to the Xen documentation to configure the network for your site.
- Some distributions have **requiretty** option enabled in the `sudoers` file. It must be disabled to so ONE can execute commands using sudo. The line to remove or comment out (by placing a # at the beginning of the line) is this one:

```
#Defaults requiretty
```

OpenNebula Configuration

OpenNebula needs to know if it is going to use the XEN Driver. There are two sets of Xen VMM drivers, one for Xen version 3.x and other for 4.x, you will have to uncomment the version you will need. To achieve this for Xen version 4.x, uncomment these drivers in */etc/one/oned.conf*:

```
IM_MAD = [
    name       = "xen",
    executable = "one_im_ssh",
    arguments  = "xen" ]

VM_MAD = [
    name       = "xen",
    executable = "one_vmm_exec",
    arguments  = "xen4",
    default    = "vmm_exec/vmm_exec_xen4.conf",
    type       = "xen" ]
```

`xen4` drivers are meant to be used with Xen >=4.2 with `xl`/`xenlight` interface. You will need to stop `xend` daemon for this. For Xen 3.x and Xen 4.x with xm (with `xend` daemon) you will need to use `xen3` drivers.

Warning: When using ‘‘xen3‘ drivers for Xen 4.x you should change the configuration file `/var/lib/oneremotes/vmm/xen3/xenrc` and uncomment the `XM_CREDITS` line.‘

3.2.4 Usage

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

XEN Specific Attributes

DISK

- **driver**, This attribute defines the Xen backend for disk images, possible values are `file:`, `tap:aio:`... Note the trailing `:`.

NIC

- **model**, This attribute defines the type of the vif. This corresponds to the type attribute of a vif, possible values are `ioemu`, `netfront`...
- **ip**, This attribute defines the ip of the vif and can be used to set antispoofing rules. For example if you want to use antispoofing with network-bridge, you will have to add this line to `/etc/xen/xend-config.sxp`:

```
(network-script 'network-bridge antispoofing=yes')
```

OS

- **bootloader**, You can use this attribute to point to your `pygrub` loader. This way you wont need to specify the kernel/initrd and it will use the internal one. Make sure the kernel inside is domU compatible if using paravirtualization.
- When no `kernel`/`initrd` or `bootloader` attributes are set then a HVM machine is created.

CONTEXT

- **driver**, for the CONTEXT device, e.g. 'file:', 'phy:'...

Additional Attributes

The **raw** attribute offers the end user the possibility of passing by attributes not known by OpenNebula to Xen. Basically, everything placed here will be written ad literally into the Xen deployment file.

```
RAW = [ type="xen", data="on_crash=destroy" ]
```

3.2.5 Tuning & Extending

The driver consists of the following files:

- `/usr/lib/one/mads/one_vmm_exec` : generic VMM driver.
- `/var/lib/one/remotes/vmm/xen` : commands executed to perform actions.

And the following driver configuration files:

- `/etc/one/vmm_exec/vmm_exec_xen3/4.conf` : This file is home for default values for domain definitions (in other words, OpenNebula templates). Let's go for a more concrete and VM related example. If the user wants to set a default value for KERNEL for all of their XEN domain definitions, simply edit the `vmm_exec_xen.conf` file and set a

```
OS = [ kernel="/vmlinuz" ]
```

into it. Now, when defining a ONE template to be sent to a XEN resource, the user has the choice of "forgetting" to set the **KERNEL** parameter, in which case it will default to /vmlinuz.

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

- `/var/lib/one/remotes/vmm/xen/xenrc` : This file contains environment variables for the driver. You may need to tune the values for `XM_PATH`, if `/usr/sbin/xm` do not live in their default locations in the remote hosts. This file can also hold 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
```

Parameter	Description
IMAGE_PREFIX	This will be used as the default handler for disk hot plug
SHUTDOWN_TIMEOUT	Seconds to wait after shutdown until timeout
FORCE_DESTROY	Force VM cancellation after shutdown timeout

See the *Virtual Machine drivers reference* for more information.

3.2.6 Credit Scheduler

Xen comes with a credit scheduler. The credit scheduler is a proportional fair share CPU scheduler built from the ground up to be work conserving on SMP hosts. This attribute sets a 16 bit value that will represent the amount of sharing this VM will have respect to the others living in the same host. This value is set into the driver configuration file, is not intended to be defined per domain.

Xen drivers come preconfigured to use this credit scheduler and uses the scale "1 OpenNebula CPU" = "256 xen scheduler credits". A VM created with CPU=2.0 will have 512 xen scheduler credits. If you need to change this scaling parameter it can be configured in `/etc/one/vmm_exec/vmm_exec_xen[3/4].conf`. The variable name is called `CREDIT`.

3.3 KVM Driver

KVM (Kernel-based Virtual Machine) is a complete virtualization technique for Linux. It offers full virtualization, where each Virtual Machine interacts with its own virtualized hardware. This guide describes the use of the KVM virtualizer with OpenNebula, please refer to KVM specific documentation for further information on the setup of the KVM hypervisor itself.

3.3.1 Requirements

The hosts must have a working installation of KVM, that usually requires:

- CPU with VT extensions
- libvirt >= 0.4.0
- kvm kernel modules (kvm.ko, kvm-{intel,amd}.ko). Available from kernel 2.6.20 onwards.
- the qemu user-land tools

3.3.2 Considerations & Limitations

- KVM currently only supports 4 IDE devices, for more disk devices you should better use SCSI or virtio. You have to take this into account when adding disks. See the *Virtual Machine Template documentation* for an explanation on how OpenNebula assigns disk targets.
- By default live migrations are started from the host the VM is currently running. If this is a problem in your setup you can activate local live migration adding `-l migrate=migrate_local` to `vmm_mad` arguments.
- If you get error messages similar to `error: cannot close file: Bad file descriptor` upgrade libvirt version. Version 0.8.7 has a bug related to file closing operations.

3.3.3 Configuration

KVM Configuration

OpenNebula uses the libvirt interface to interact with KVM, so the following steps are required in the hosts to get the KVM driver running:

- Qemu should be configured to not change file ownership. Modify `/etc/libvirt/qemu.conf` to include `dynamic_ownership = 0`. To be able to use the images copied by OpenNebula, change also the user and group under which the libvirtd is run to "oneadmin":

```
$ grep -vE '^($|#)' /etc/libvirt/qemu.conf
user = "oneadmin"
group = "oneadmin"
dynamic_ownership = 0
```

Warning: Note that oneadmin's group may be other than oneadmin. Some distributions adds oneadmin to the cloud group. Use group = "cloud" above in that case.

- The remote hosts must have the libvirt daemon running.
- The user with access to these remotes hosts on behalf of OpenNebula (typically `<oneadmin>`) has to pertain to the <libvirtd> and <kvm> groups in order to use the deaemon and be able to launch VMs.

Warning: If apparmor is active (by default in Ubuntu it is), you should add /var/lib/one to the end of /etc/apparmor.d/libvirt-qemu

```
owner /var/lib/one/** rw,
```

Warning: If your distro is using PolicyKit you can use this recipe by Jan Horacek to add the require privileges to `oneadmin` user:

```
# content of file: /etc/polkit-1/localauthority/50-local.d/50-org.libvirt.unix.manage-opennebula.pkla
[Allow oneadmin user to manage virtual machines]
Identity=unix-user:oneadmin
Action=org.libvirt.unix.manage
#Action=org.libvirt.unix.monitor
ResultAny=yes
ResultInactive=yes
ResultActive=yes
```

OpenNebula uses libvirt's migration capabilities. More precisely, it uses the TCP protocol offered by libvirt. In order to configure the physical hosts, the following files have to be modified:

- `/etc/libvirt/libvirtd.conf` : Uncomment "listen_tcp = 1". Security configuration is left to the admin's choice, file is full of useful comments to achieve a correct configuration. As a tip, if you don't want to use TLS for connections set `listen_tls = 0`.
- Add the listen option to libvirt init script:
  - `/etc/default/libvirt-bin` : add **-l** option to `libvirtd_opts`
  - For RHEL based distributions, edit this file instead: `/etc/sysconfig/libvirtd` : uncomment `LIBVIRTD_ARGS="--listen"`

OpenNebula Configuration

OpenNebula needs to know if it is going to use the KVM Driver. To achieve this, uncomment these drivers in */etc/one/oned.conf*:

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

VM_MAD = [
    name       = "kvm",
    executable = "one_vmm_exec",
```

```
    arguments  = "-t 15 -r 0 kvm",
    default    = "vmm_exec/vmm_exec_kvm.conf",
    type       = "kvm" ]
```

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 frontend**:

- 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. wih 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;
}

# /etc/cgrules.conf

*:libvirtd       memory          virt/
```

- After configuring the hosts start/restart the cgroups service.
- (Optional) If you have limited the amount of memory for VMs, you may want to set `HYPERVISOR_MEM` parameter in `/etc/one/sched.conf`

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
```

Udev Rules

When creating VMs as a regular user, `/dev/kvm` needs to be chowned to the `oneadmin` user. For that to be persistent you have to apply the following UDEV rule:

```
# cat /etc/udev/rules.d/60-qemu-kvm.rules
KERNEL=="kvm", GROUP="oneadmin", MODE="0660"
```

3.3.4 Usage

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.

Default Attributes

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. You may need to adjust it to your ditsro
- os, the attraibutes: kernel, initrd, boot, root, kernel_cmd, 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.

For example:

```
OS   = [
  KERNEL = /vmlinuz,
  BOOT   = hd,
  ARCH   = "x86_64"]

DISK = [ driver = "raw" , cache = "none"]

NIC  = [ filter = "clean-traffic", model = "virtio" ]

RAW  = "<devices><serial type=\"pty\"><source path=\"/dev/pts/5\"/><target port=\"0\"/></serial><cons
```

KVM Specific Attributes

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`

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 `model=virtio`

FEATURES

- **pae**: Physical address extension mode allows 32-bit guests to address more than 4 GB of memory:
- **acpi**: useful for power management, for example, with KVM guests it is required for graceful shutdown to work.

Format and valid values:

```
FEATURES=[
    pae={yes|no},
    acpi={yes|no} ]
```

Default values for this features can be set in the driver configuration file so they don't need to be specified for every VM.

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\"/></seri
```

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.

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

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 flavour, 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
```

3.3.5 Tuning & Extending

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 libvirtd
QEMU_PROTOCOL	Protocol used for live migrations
SHUTDOWN_TIMEOUT	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).

See the *Virtual Machine drivers reference* for more information.

3.4 VMware Drivers

The **VMware Drivers** enable the management of an OpenNebula cloud based on VMware ESX and/or VMware Server hypervisors. They use `libvirt` and direct API calls using `RbVmomi` to invoke the Virtual Infrastructure SOAP API exposed by the VMware hypervisors, and feature a simple configuration process that will leverage the stability, performance and feature set of any existing VMware based OpenNebula cloud.

3.4.1 Requirements

In order to use the **VMware Drivers**, some software dependencies have to be met:

- **libvirt**: At the OpenNebula front-end, libvirt is used to access the VMware hypervisors , so it needs to be installed with ESX support. We recommend version 0.8.3 or higher, which enables interaction with the vCenter VMware product, required to use vMotion. This will be installed by the OpenNebula package.
- **rbvmomi**: Also at the OpenNebula front-end, the rbvmomi gem needs to be installed. This will be installed by the OpenNebula package or the install_gems script.
- **ESX, VMware Server**: At least one VMware hypervisor needs to be installed. Further configuration for the `DATASTORE` is needed, and it is explained in the TM part of the Configuration section.

Optional Requirements. To enable some OpenNebula features you may need:

- **vMotion**: VMware's vMotion capabilities allows to perform live migration of a Virtual Machine between two ESX hosts, allowing for load balancing between cloud worker nodes without downtime in the migrated virtual machine. In order to use this capability, the following requisites have to be meet:
  - **Shared storage** between the source and target host, mounted in both hosts as the same DATASTORE (we are going to assume it is called "images" in the rest of this document)
  - **vCenter Server** installed and configured, details in the Installation Guide for ESX and vCenter.
  - A **datacenter** created in the vCenter server that includes all ESX hosts between which Virtual Machines want to be live migrated (we are going to assume it is called "onecenter" in the rest of this document).
  - A **user** created in vCenter with the same username and password than the ones in the ESX hosts, with administrator permissions.

Warning: Please note that the libvirt version shipped with some linux distribution does not include ESX support. In these cases it may be needed to recompile the libvirt package with the –with-esx option.

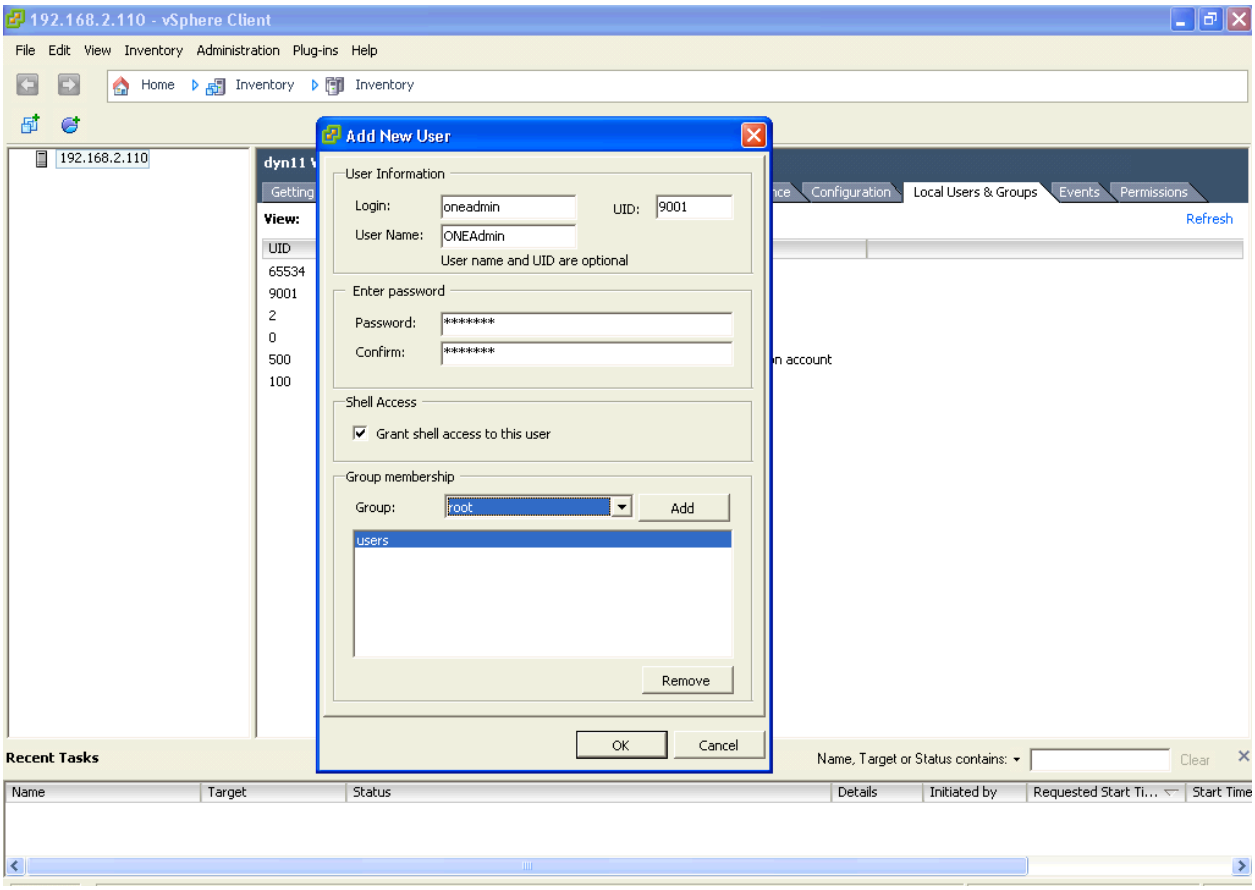
3.4.2 Considerations & Limitations

- Only one vCenter can be used for livemigration.
- Datablock images and volatile disk images will be always created without format, and thus have to be formatted by the guest.
- In order to use the **attach/detach** functionality, the original VM must have at least one SCSI disk, and the disk to be attached/detached must be placed on a SCSI bus (ie, "sd" as DEV_PREFIX).
- The ESX hosts need to be properly licensed, with write access to the exported API (as the Evaluation license does).

3.4.3 VMware Configuration

Users & Groups

The creation of a user in the VMware hypervisor is recommended. Go to the Users & Group tab in the VI Client, and create a new user (for instance, "oneadmin") with the same UID and username as the oneadmin user executing OpenNebula in the front-end. Please remember to give full permissions to this user (Permissions tab).



Warning: After registering a datastore, make sure that the "oneadmin" user can write in said datastore (this is not needed if the "root" user is used to access the ESX). In case "oneadmin" cannot write in "/vmfs/volumes/<ds_id>", then permissions need to be adjusted. This can be done in various ways, the recommended one being:

- Add "oneadmin" to the "root" group using the Users & Group tab in the VI Client
- `$ chmod g+w /vmfs/volumes/<ds_id>` in the ESX host

SSH Access

SSH access from the front-end to the ESX hosts is required (or, at least, they need it to unlock all the functionality of OpenNebula). to ensure so, please remember to click the "Grant shell access to this user" checkbox when creating the `oneadmin` user.

The access via SSH needs to be passwordless. Please follow the next steps to configure the ESX node:

- login to the esx host (ssh <esx-host>)

```
 $ su -
 $ mkdir /etc/ssh/keys-oneadmin
 $ chmod 755 /etc/ssh/keys-oneadmin
 $ su - oneadmin
 $ vi  /etc/ssh/keys-oneadmin/authorized_keys
<paste here the contents of the oneadmin's front-end account public key (FE -> $HOME/.ssh/id_{rsa,dsa
 $ chmod 600 /etc/ssh/keys-oneadmin/authorized_keys
```

More information on passwordless ssh connections here.

Tools Setup

- In order to enable all the functionality of the drivers, several short steps remain:

```
$ su
$ chmod +s /sbin/vmkfstools
```

- In order to use the **attach/detach functionality** for VM disks, some extra configuration steps are needed in the ESX hosts. For ESX > 5.0

```
$ su
$ chmod +s /bin/vim-cmd
```

- In order to use the *dynamic network mode* for VM disks, some extra configuration steps are needed in the ESX hosts. For ESX > 5.0

```
$ su
$ chmod +s /sbin/esxcfg-vswitch
```

Persistency

Persistency of the ESX filesystem has to be handled with care. Most of ESX 5 files reside in a in-memory filesystem, meaning faster access and also non persistency across reboots, which can be inconvenient at the time of managing a ESX farm for a OpenNebula cloud.

Here is a recipe to make the configuration needed for OpenNebula persistent across reboots. The changes need to be done as root.

```
# vi /etc/rc.local
   ## Add this at the bottom of the file

mkdir /etc/ssh/keys-oneadmin
cat > /etc/ssh/ssh-oneadmin/authorized_keys << _SSH_HEYS_
ssh-rsa <really long string with oneadmin's ssh public key>
_SSH_KEYS_
chmod 600 /etc/ssh/keys-oneadmin/authorized_keys
chmod +s /sbin/vmkfstools /bin/vim-cmd
chmod 755 /etc/ssh/keys-oneadmin
chown oneadmin /etc/ssh/keys-oneadmin/authorized_keys

# /sbin/auto-backup.sh
```

This information was based on this blog post.

Storage

There are additional configuration steps regarding storage. Please refer to the *VMware Storage Model guide for more details*.

Networking

Networking can be used in two different modes: **pre-defined** (to use pre-defined port groups) or **dynamic** (to dynamically create port groups and VLAN tagging). Please refer to the *VMware Networking guide for more details*.

VNC

In order to access running VMs through VNC, the ESX host needs to be configured beforehand, basically to allow VNC inbound connections via their firewall. To do so, please follow this guide.

3.4.4 OpenNebula Configuration

OpenNebula Daemon

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

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

#-------------------------------------------------------------------------------
#  VMware Information Driver Manager Configuration
#-------------------------------------------------------------------------------
IM_MAD = [
      name       = "vmware",
      executable = "one_im_sh",
      arguments  = "-c -t 15 -r 0 vmware" ]
#-------------------------------------------------------------------------------

SCRIPTS_REMOTE_DIR=/tmp/one
```

VMware Drivers

The configuration attributes for the VMware drivers are set in the `/etc/one/vmwarerc` file. In particular the following values can be set:

SCHEDULER OPTIONS	DESCRIPTION
:libvirt_uri	used to connect to VMware through libvirt. When using VMware Server, the connection string set under LIBVIRT_URI needs to have its prefix changed from *esx* to *gsx*
:username	username to access the VMware hypervisor
:password	password to access the VMware hypervisor
:datacenter	(only for vMotion) name of the datacenter where the hosts have been registered.
:vcenter	(only for vMotion) name or IP of the vCenter that manage the ESX hosts

Example of the configuration file:

```
:libvirt_uri: "esx://@HOST@/?no_verify=1&auto_answer=1"
:username: "oneadmin"
:password: "mypass"
:datacenter: "ha-datacenter"
:vcenter: "London-DC"
```

Warning: Please be aware that the above rc file, in stark contrast with other rc files in OpenNebula, uses yaml syntax, therefore please input the values between quotes.

VMware Physical Hosts

The physical hosts containing the VMware hypervisors need to be added with the appropriate **VMware Drivers**. If the box running the VMware hypervisor is called, for instance, **esx-host**, the host would need to be registered with the following command (dynamic netwotk mode):

```
$ onehost create esx-host -i vmware -v vmware -n vmware
```

or for pre-defined networking

```
$ onehost create esx-host -i vmware -v vmware -n dummy
```

3.4.5 Usage

Images

To register an existing VMware disk in an OpenNebula image catalog you need to:

- Place all the .vmdk files that conform a disk (they can be easily spotted, there is a main <name-of-the-image>.vmdk file, and various <name-of-the-image-sXXX.vmdk flat files) in the same directory, with no more files than these.
- Afterwards, an image template needs to be written, using the the absolut path to the directory as the PATH value. For example:

```
NAME = MyVMwareDisk
PATH =/absolute/path/to/disk/folder
TYPE = OS
```

Warning: To register a .iso file with type CDROM there is no need to create a folder, just point with PATH to he absolute path of the .iso file.

Warning: In order to register a VMware disk through Sunstone, create a zip compressed tarball (.tar.gz) and upload that (it will be automatically decompressed in the datastore). Please note that the tarball is only of the folder with the .vmdk files inside, no extra directories can be contained in that folder.

Once registered the image can be used as any other image in the OpenNebula system as described in the *Virtual Machine Images guide*.

Datablocks & Volatile Disks

Datablock images and volatile disks will appear as a raw devices on the guest, which will then need to be formatted. The FORMAT attribute is compulsory, possible values (more info on this here) are:

- **vmdk_thin**
- **vmdk_zeroedthick**
- **vmdk_eagerzeroedthick**

Virtual Machines

The following attributes can be used for VMware Virtual Machines:

- GuestOS: This parameter can be used in the OS section of the VM template. The os-identifier can be one of this list.

```
OS=[GUESTOS=<os-identifier]
```

- PCIBridge: This parameter can be used in the FEATURES section of the VM template. The <bridge-number> is the number of PCI Bridges that will be available in the VM (that is, 0 means no PCI Bridges, 1 means PCI Bridge with ID = 0 present, 2 means PCI Bridges with ID = 0,1 present, and so on).

```
FEATURES=[PCIBRIDGE=<bridge-number>]
```

3.4.6 Custom VMX Attributes

You can add metadata straight to the .vmx file using RAW/DATA_VMX. This comes in handy to specify for example a specific guestOS type, more info *here*.

Following the two last sections, if we want a VM of guestOS type "Windows 7 server 64bit", with disks plugged into a LSI SAS SCSI bus, we can use a template like:

```
NAME = myVMwareVM

CPU    = 1
MEMORY = 256

DISK = [IMAGE_ID="7"]
NIC  = [NETWORK="public"]

RAW=[
  DATA="<devices><controller type='scsi' index='0' model='lsisas1068'/></devices>",
  DATA_VMX="pciBridge0.present = \"TRUE\"\npciBridge4.present = \"TRUE\"\npciBridge4.virtualDev = \"p
  TYPE="vmware" ]
```

3.4.7 Tuning & Extending

The **VMware Drivers** consists of three drivers, with their corresponding files:

- **VMM Driver**
  - `/var/lib/one/remotes/vmm/vmware` : commands executed to perform actions.
- **IM Driver**
  - `/var/lib/one/remotes/im/vmware.d` : vmware IM probes.
- **TM Driver**
  - `/usr/lib/one/tm_commands` : commands executed to perform transfer actions.

And the following driver configuration files:

- **VMM Driver**
  - `/etc/one/vmm_exec/vmm_exec_vmware.conf` : This file is home for default values for domain definitions (in other words, OpenNebula templates). For example, if the user wants to set a default value for **CPU** requirements for all of their VMware domain definitions, simply edit the `/etc/one/vmm_exec/vmm_exec_vmware.conf` file and set a

```
CPU=0.6
```

into it. Now, when defining a template to be sent to a VMware resource, the user has the choice of "forgetting" to set the **CPU** requirement, in which case it will default to 0.6.

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

- **TM Driver**
  - `/etc/one/tm_vmware/tm_vmware.conf` : This files contains the scripts tied to the different actions that the TM driver can deliver. You can here deactivate functionality like the DELETE action (this can be accomplished using the dummy tm driver, dummy/tm_dummy.sh) or change the default behavior.

More generic information about drivers:

- *Virtual Machine Manager drivers reference*
- *Transfer Manager driver reference*

CHAPTER

FOUR

NETWORKING

4.1 Networking Overview

Before diving into Network configuration in OpenNebula make sure that you've followed the steps described in the *Networking section of the Installation guide*.

When a new Virtual Machine is launched, OpenNebula will connect its network interfaces (defined in the NIC section of the template) to the bridge or physical device specified in the *Virtual Network definition*. This will allow the VM to have access to different networks, public or private.

The OpenNebula administrator must take into account that although this is a powerful setup, it should be complemented with mechanisms to restrict network access only to the expected Virtual Machines, to avoid situations in which an OpenNebula user interacts with another user's VM. This functionality is provided through Virtual Network Manager drivers. The OpenNebula administrator may associate one of the following drivers to each Host, when the hosts are created with the *onehost command*:

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

Note that some of these drivers also create the bridging device in the hosts.

The administrator must take into account the following matrix that shows the compatibility of the hypervisors with each networking driver:

	Firewall	Open vSwitch	802.1Q	ebtables	VMware
KVM	Yes	Yes	Yes	Yes	No
Xen	Yes	Yes	Yes	Yes	No
VMware	No	No	No	No	Yes

The Virtual Network isolation is enabled with any of the 801.1Q, ebtables, vmware or ovswitch drivers. These drivers also enable the firewalling rules to allow a regular OpenNebula user to filter TCP, UDP or ICMP traffic.

OpenNebula also comes with a *Virtual Router appliance* that provides networking services like DHCP, DNS, etc.

4.1.1 Tuning & Extending

Customization of the Drivers

The network is dynamically configured in three diferent steps:

- **Pre**: Right before the hypervisor launches the VM.
- **Post**: Right after the hypervisor launches the VM.
- **Clean**: Right after the hypervisor shuts down the VM.

Each driver execute different actions (or even none at all) in these phases depending on the underlying switching fabric. Note that, if either `Pre` or `Post` fail, the VM will be shut down and will be placed in a `FAIL` state.

You can easily customize the behavior of the driver for your infrastructure by modifying the files in located in `/var/lib/one/remotes/vnm`. Each driver has its own folder that contains at least three programs `pre`, `post` and `clean`. These programs are executed to perform the steps described above.

Fixing Default Paths

The default paths for the binaries/executables used during the network configuration may change depending on the distro. OpenNebula ships with the most common paths, however these may be wrong for your particular distro. In that case, please fix the proper paths in the `COMMANDS` hash of `/var/lib/one/remotes/vnm/OpenNebulaNetwork.rb`:

```
COMMANDS = {
  :ebtables => "sudo /sbin/ebtables",
  :iptables => "sudo /sbin/iptables",
  :brctl    => "sudo /sbin/brctl",
  :ip       => "sudo /sbin/ip",
  :vconfig  => "sudo /sbin/vconfig",
  :virsh    => "virsh -c qemu:///system",
  :xm       => "sudo /usr/sbin/xm",
  :ovs_vsctl=> "sudo /usr/local/bin/ovs-vsctl",
  :lsmod    => "/sbin/lsmod"
}
```

4.2 802.1Q VLAN

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, calculated by adding a constant to the network id. It may also be forced by specifying an VLAN_ID parameter in the *Virtual Network template*.

4.2.1 Requirements

A network switch capable of forwarding VLAN tagged traffic. The physical switch ports should be VLAN trunks.

4.2.2 Considerations & Limitations

This driver requires some previous work on the network components, namely the switches, to enable VLAN trunking in the network interfaces connected to the OpenNebula hosts. If this is not activated the VLAN tags will not get trough and the network will behave erratically.

In OpenNebula 3.0, this functionality was provided through a hook, and it wasn't effective after a migration. Since OpenNebula 3.2 this limitation does not apply.

4.2.3 Configuration

Hosts Configuration

- The `sudoers` file must be configured so `oneadmin` can execute `vconfig`, `brctl` and `ip` in the hosts.
- The package `vconfig` must be installed in the hosts.
- Hosts must have the module `8021q` loaded.

To enable VLAN (802.1Q) support in the kernel, one must load the 8021q module:

```
$ modprobe 8021q
```

If the module is not available, please refer to your distribution's documentation on how to install it. This module, along with the `vconfig` binary which is also required by the script, is generally supplied by the `vlan` package.

OpenNebula Configuration

To enable this driver, use **802.1Q** as the Virtual Network Manager driver parameter when the hosts are created with the *onehost command*:

```
$ onehost create host01 -i kvm -v kvm -n 802.1Q
```

Driver Actions

Action	Description
Pre	Creates a VLAN tagged interface in the Host and a attaches it to a dynamically created bridge.
Post	N/A
Clean	It doesn't do anything. The VLAN tagged interface and bridge are kept in the Host to speed up future VMs

4.2.4 Usage

The driver will be automatically applied to every Virtual Machine deployed in the Host. However, this driver requires a special configuration in the *Virtual Network template*: only the virtual networks with the attribute `VLAN` set to `YES` will be isolated. The attribute PHYDEV must be also defined, with the name of the physical network device that will be attached to the bridge. The BRIDGE attribute is not mandatory, if it isn't defined, OpenNebula will generate one automatically.

```
NAME    = "hmnet"
TYPE    = "fixed"

PHYDEV  = "eth0"
VLAN    = "YES"
VLAN_ID = 50        # optional
```

```
BRIDGE  = "brhm"    # optional

LEASES = ...
```

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.<vlan_id>`) and attached to `brhm` (unless it's already attached).

Warning: Any user with Network creation/modification permissions may force a custom vlan id with the `VLAN_ID` parameter in the network template. In that scenario, any user may be able to connect to another network with the same network id. Techniques to avoid this are explained under the Tuning & Extending section.

4.2.5 Tuning & Extending

Warning: Remember that any change in the `/var/lib/one/remotes` directory won't be effective in the Hosts until you execute, as oneadmin:

```
oneadmin@frontend $ onehost sync
```

Calculating VLAN ID

The vlan id is calculated by adding the network id to a constant defined in `/var/lib/one/remotes/vnm/OpenNebulaNetwork.rb`. You can customize that value to your own needs:

```
CONF = {
    :start_vlan => 2
}
```

Restricting Manually the VLAN ID

You can either restrict permissions on Network creation with *ACL rules*, or you can entirely disable the possibility to redefine the VLAN_ID by modifying the source code of `/var/lib/one/remotes/vnm/802.1Q/HostManaged.rb`. Change these lines:

```
if nic[:vlan_id]
    vlan = nic[:vlan_id]
else
    vlan = CONF[:start_vlan] + nic[:network_id].to_i
end
```

with this one:

```
vlan = CONF[:start_vlan] + nic[:network_id].to_i
```

4.3 Ebtables

This guide describes how to enable Network isolation provided through ebtables rules applied on the bridges. This method will only permit isolation with a mask of 255.255.255.0.

4.3.1 Requirements

This hook requires `ebtables` to be available in all the OpenNebula Hosts.

4.3.2 Considerations & Limitations

Although this is the most easily usable driver, since it doesn't require any special hardware or any software configuration, it lacks the ability of sharing IPs amongst different VNETs, that is, if an VNET is using leases of 192.168.0.0/24, another VNET can't be using IPs in the same network.

4.3.3 Configuration

Hosts Configuration

- The package `ebtables` must be installed in the hosts.
- The `sudoers` file must be configured so `oneadmin` can execute `ebtables` in the hosts.

OpenNebula Configuration

To enable this driver, use **ebtables** as the Virtual Network Manager driver parameter when the hosts are created with the *onehost command*:

```
$ onehost create host01 -i kvm -v kvm -n ebtables
```

Driver Actions

Action	Description
Pre	N/A
Post	Creates EBTABLES rules in the Host where the VM has been placed.
Clean	Removes the EBTABLES rules created during the `Post` action.

4.3.4 Usage

The driver will be automatically applied to every Virtual Machine deployed in the Host. Only the virtual networks with the attribute `VLAN` set to `YES` will be isolated. There are no other special attributes required.

```
NAME    = "ebtables_net"
TYPE    = "fixed"
BRIDGE  = vbr1

VLAN    = "YES"

LEASES = ...
```

4.3.5 Tuning & Extending

EBTABLES Rules

This section lists the EBTABLES rules that are created:

```
# Drop packets that don't match the network's MAC Address
-s ! <mac_address>/ff:ff:ff:ff:ff:0 -o <tap_device> -j DROP
# Prevent MAC spoofing
-s ! <mac_address> -i <tap_device> -j DROP
```

4.4 Open vSwitch

This guide describes how to use the Open vSwitch network drives. They provide two indepent functionalities that can be used together: network isolation using VLANs, and network filtering using OpenFlow. Each Virtual Network interface will receive a VLAN tag enabling network isolation. 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 by adding a constant to the network id. It may also be forced by specifying an VLAN_ID parameter in the *Virtual Network template*.

The network filtering functionality is very similar to the *Firewall* drivers, with a few limitations discussed below.

4.4.1 Requirements

This driver requires Open vSwitch to be installed on each OpenNebula Host. Follow the resources specified in *hosts_configuration* to install it.

4.4.2 Considerations & Limitations

Integrating OpenNebula with Open vSwitch brings a long list of benefits to OpenNebula, read Open vSwitch Features to get a hold on these features.

This guide will address the usage of VLAN tagging and OpenFlow filtering of OpenNebula Virtual Machines. On top of that any other Open vSwitch feature may be used, but that's outside of the scope of this guide.

ovswitch and ovswitch_brcompat

OpenNebula ships with two sets of drivers that provide the same functionality: **ovswitch** and **ovsvswitch_brcompat**. The following list details the differences between both drivers:

- `ovswitch`: Recommended for `kvm` hosts. Only works with `kvm`. Doesn't require the Open vSwitch compatibility layer for Linux bridging.
- `ovswitch_brcompat`: Works with `kvm` and `xen`. This is the only set that currently works with `xen`. Not recommended for `kvm`. Requires Open vSwitch compatibility layer for Linux bridging.

4.4.3 Configuration

Hosts Configuration

- You need to install Open vSwitch on each OpenNebula Host. Please refer to the Open vSwitch documentation to do so.
- If using `ovswitch_brcompat` it is also necessary to install the Open vSwitch compatibility layer for Linux bridging.
- The `sudoers` file must be configured so `oneadmin` can execute `ovs_vsctl` in the hosts.

OpenNebula Configuration

To enable this driver, use **ovswitch** or **ovswitch_brcompat** as the Virtual Network Manager driver parameter when the hosts are created with the *onehost command*:

```
# for kvm hosts
$ onehost create host01 -i kvm -v kvm -n ovswitch

# for xen hosts
$ onehost create host02 -i xen -v xen -n ovswitch_brcompat
```

Driver Actions

Action	Description
Pre	N/A
Post	Performs the appropriate Open vSwitch commands to tag the virtual tap interface.
Clean	It doesn't do anything. The virtual tap interfaces will be automatically discarded when the VM is shut down.

4.4.4 Usage

Network Isolation

The driver will be automatically applied to every Virtual Machine deployed in the Host. Only the virtual networks with the attribute `VLAN` set to `YES` will be isolated. There are no other special attributes required.

```
NAME    = "ovswitch_net"
TYPE    = "fixed"
BRIDGE  = vbr1

VLAN    = "YES"
VLAN_ID = 50        # optional

LEASES = ...
```

Warning: Any user with Network creation/modification permissions may force a custom vlan id with the `VLAN_ID` parameter in the network template. In that scenario, any user may be able to connect to another network with the same network id. Techniques to avoid this are explained under the Tuning & Extending section.

Network Filtering

The first rule that is always applied when using the Open vSwitch drivers is the MAC-spoofing rule, that prevents any traffic coming out of the VM if the user changes the MAC address.

The firewall directives must be placed in the *network section* of the Virtual Machine template. These are the possible attributes:

- `BLACK_PORTS_TCP = iptables_range`: Doesn't permit access to the VM through the specified ports in the TCP protocol. Superseded by WHITE_PORTS_TCP if defined.
- `BLACK_PORTS_UDP = iptables_range`: Doesn't permit access to the VM through the specified ports in the UDP protocol. Superseded by WHITE_PORTS_UDP if defined.
- `ICMP = drop`: Blocks ICMP connections to the VM. By default it's set to accept.

`iptables_range`: a list of ports separated by commas, e.g.: `80,8080`. Currently no ranges are supporteg, e.g.: `5900:6000` is **not** supported.

Example:

```
NIC = [ NETWORK_ID = 3, BLACK_PORTS_TCP = "80, 22", ICMP = drop ]
```

Note that WHITE_PORTS_TCP and BLACK_PORTS_TCP are mutually exclusive. In the event where they're both defined the more restrictive will prevail i.e. WHITE_PORTS_TCP. The same happens with WHITE_PORTS_UDP and BLACK_PORTS_UDP.

4.4.5 Tuning & Extending

Warning: Remember that any change in the `/var/lib/one/remotes` directory won't be effective in the Hosts until you execute, as oneadmin:

```
oneadmin@frontend $ onehost sync
```

This way in the next monitoring cycle the updated files will be copied again to the Hosts.

Calculating VLAN ID

The vlan id is calculated by adding the network id to a constant defined in `/var/lib/one/remotes/vnm/OpenNebulaNetwork.rb`. You can customize that value to your own needs:

```
CONF = {
    :start_vlan => 2
}
```

Restricting Manually the VLAN ID

You can either restrict permissions on Network creation with *ACL rules*, or you can entirely disable the possibility to redefine the VLAN_ID by modifying the source code of `/var/lib/one/remotes/vnm/ovswitch/OpenvSwitch.rb`. Change these lines:

```
if nic[:vlan_id]
    vlan = nic[:vlan_id]
else
    vlan = CONF[:start_vlan] + nic[:network_id].to_i
end
```

with this one:

```
vlan = CONF[:start_vlan] + nic[:network_id].to_i
```

OpenFlow Rules

To modify these rules you have to edit: `/var/lib/one/remotes/vnm/ovswitch/OpenvSwitch.rb`.

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
```

Black ports (one rule per port)

```
tcp,dl_dst=<MAC>,tp_dst=<PORT>,actions=drop
```

ICMP Drop

```
icmp,dl_dst=<MAC>,actions=drop
```

4.5 VMware Networking

This guide describes how to use the VMware network driver in OpenNebula. This driver optionally provides network isolation through VLAN tagging. The VLAN id will be the same for every interface in a given network, calculated by adding a constant to the network id. It may also be forced by specifying an VLAN_ID parameter in the *Virtual Network template*.

4.5.1 Requirements

In order to use the dynamic network mode for VM disks, some extra configuration steps are needed in the ESX hosts.

```
$ su
$ chmod +s /sbin/esxcfg-vswitch
```

4.5.2 Considerations & Limitations

It should be noted that the drivers will not create/delete/manage VMware virtual switches, these should be created before-hand by VMware administrators.

Since the dynamic driver will however create VMware port groups, it should be noted that there's a default limit of 56 port groups per switch. Administrators should be aware of these limitations.

4.5.3 Configuration

The vSphere hosts can work in two different networking modes, namely:

- **pre-defined**: The VMWare administrator has set up the network for each vSphere host, defining the vSwitch'es and port groups for the VMs. This mode is associated with the `dummy` network driver. To configure this mode use `dummy` as the Virtual Network Manager driver parameter when the hosts are created:

```
$ onehost create host01 -i vmware -v vmware -n dummy
```

- **dynamic**: In this mode OpenNebula will create on-the-fly port groups (with an optional VLAN_ID) for your VMs. The VMWare administrator has to set up only the vSwitch to be used by OpenNebula. To enable this driver, use `vmware` as the VNM driver for the hosts:

```
$ onehost create host02 -i vmware -v vmware -n vmware
```

Warning: Dynamic and pre-defined networking modes can be mixed in a datacenter. Just use the desired mode for each host.

4.5.4 Usage

Using the Pre-defined Network Mode

In this mode there the VMware admin has created one or more port groups in the ESX hosts to bridge the VMs. The port group has to be specified for each Virtual Network in its template through the `BRIDGE` attribute (*check the Virtual Network usage guide for more info*).

The NICs of the VM in this Virtual Network will be attached to the specified port group in the vSphere host. For example:

```
NAME    = "pre-defined_vmware_net"
TYPE    = "fixed"
BRIDGE  = "VM Network"  # This is the port group

LEASES = ...
```

Using the Dynamic Network Mode

In this mode the driver will dynamically create a port-group with name `one-pg-<network_id>` in the specified vSwitch of the target host. In this scenario the vSwitch is specified by the `BRIDGE` attribute of the Virtual Network template.

Additionally the port groups can be tagged with a vlan_id. You can set VLAN="YES" in the Virtual Network template to automatically tag the port groups in each ESX host. Optionally the tag can be specified through the VLAN_ID attribute. For example:

```
NAME    = "dynamic_vmware_net"
TYPE    = "fixed"
BRIDGE  = "vSwitch0" # In this mode this is the vSwitch name

VLAN    = "YES"
VLAN_ID = 50       # optional

LEASES = ...
```

4.5.5 Tuning & Extending

The predefined mode (dummy driver) does not execute any operation in the pre, post and clean steps (see *for more details on these phases*).

The strategy of the dynamic driver is to dynamically create a VMware port group attached to a pre-existing VMware virtual switch (standard or distributed) for each Virtual Network.

Action	Description
Pre	Creates the VMware port group with name `one-pg-<network_id>`.
Post	No operation
Clean	No operation

Calculating VLAN ID

The vlan id is calculated by adding the network id to a constant defined in `/var/lib/one/remotes/vnm/OpenNebulaNetwork.rb`. The administrator may customize that value to their own needs:

```
CONF = {
    :start_vlan => 2
}
```

4.6 Configuring Firewalls for VMs

This driver installs iptables rules in the physical host executing the VM. This driver can be used to filter (and enforce) TCP and UDP ports, and to define a policy for ICMP connections, without any additional modification to the guest VMs.

4.6.1 Requirements

- The package `iptables` must be installed in the hosts.

4.6.2 Considerations & Limitations

In OpenNebula 3.0, this functionality was provided through a hook, and it wasn't effective after a migration. Since OpenNebula 3.2 this limitation does not apply.

4.6.3 Configuration

Hosts Configuration

- The `sudoers` file must be configured so `oneadmin` can execute `iptables` in the hosts.

OpenNebula Configuration

This Virtual Machine Network Manager driver can be used individually, or combined with the isolation features of either *802.1Q* or *ebtables*. However it's **not** currently supported with the *ovswitch* drivers, they provide their own filtering mechanism.

To enable firewalling without any network isolation features, use **fw** as the Virtual Network Manager driver parameter when the hosts are created with the *onehost command*:

```
$ onehost create host01 -i kvm -v kvm -n fw
```

The firewall driver is automatically enabled when any of the previously mentioned drivers are used, additional configuration is not required.

Driver Actions

Action	Description
Pre	N/A
Post	Creates appropriate IPTABLES rules in the Host where the VM has been placed.
Clean	Removes the IPTABLES rules created during the `Post` action.

4.6.4 Usage

The firewall directives must be placed in the *network section* of the Virtual Machine template. These are the possible attributes:

- `WHITE_PORTS_TCP = <iptables_range>`: Permits access to the VM only through the specified ports in the TCP protocol. Supersedes BLACK_PORTS_TCP if defined.
- `BLACK_PORTS_TCP = <iptables_range>`: Doesn't permit access to the VM through the specified ports in the TCP protocol. Superseded by WHITE_PORTS_TCP if defined.
- `WHITE_PORTS_UDP = <iptables_range>`: Permits access to the VM only through the specified ports in the UDP protocol. Supersedes BLACK_PORTS_UDP if defined.
- `BLACK_PORTS_UDP = <iptables_range>`: Doesn't permit access to the VM through the specified ports in the UDP protocol. Superseded by WHITE_PORTS_UDP if defined.
- `ICMP = drop`: Blocks ICMP connections to the VM. By default it's set to accept.

`iptables_range`: a list of ports separated by commas or ranges separated by semicolons, e.g.: `22,80,5900:6000`

Example:

```
NIC = [ NETWORK_ID = 3, WHITE_PORTS_TCP = "80, 22", ICMP = drop ]
```

Note that WHITE_PORTS_TCP and BLACK_PORTS_TCP are mutually exclusive. In the event where they're both defined the more restrictive will prevail i.e. WHITE_PORTS_TCP. The same happens with WHITE_PORTS_UDP and BLACK_PORTS_UDP.

4.6.5 Tuning & Extending

IPTABLES Rules

This section lists the IPTABLES rules that are created for each possible configuration:

TCP_WHITE_PORTS and UDP_WHITE_PORTS

```
# Create a new chain for each network interface
-A FORWARD -m physdev --physdev-out <tap_device> -j one-<vm_id>-<net_id>
# Accept already established connections
-A one-<vm_id>-<net_id> -p <protocol> -m state --state ESTABLISHED -j ACCEPT
# Accept the specified <iprange>
-A one-<vm_id>-<net_id> -p <protocol> -m multiport --dports <iprange> -j ACCEPT
# Drop everything else
-A one-<vm_id>-<net_id> -p <protocol> -j DROP
```

TCP_BLACK_PORTS and UDP_BLACK_PORTS

```
# Create a new chain for each network interface
-A FORWARD -m physdev --physdev-out <tap_device> -j one-<vm_id>-<net_id>
# Drop traffic directed to the iprange ports
-A one-<vm_id>-<net_id> -p <protocol> -m multiport --dports <iprange> -j DROP
```

ICMP DROP

```
# Create a new chain for each network interface
-A FORWARD -m physdev --physdev-out <tap_device> -j one-<vm_id>-<net_id>
# Accept already established ICMP connections
-A one-<vm_id>-<net_id> -p icmp -m state --state ESTABLISHED -j ACCEPT
```

```
# Drop new ICMP connections
-A one-<vm_id>-<net_id> -p icmp -j DROP
```

These rules will be removed once the VM is shut down or destroyed.

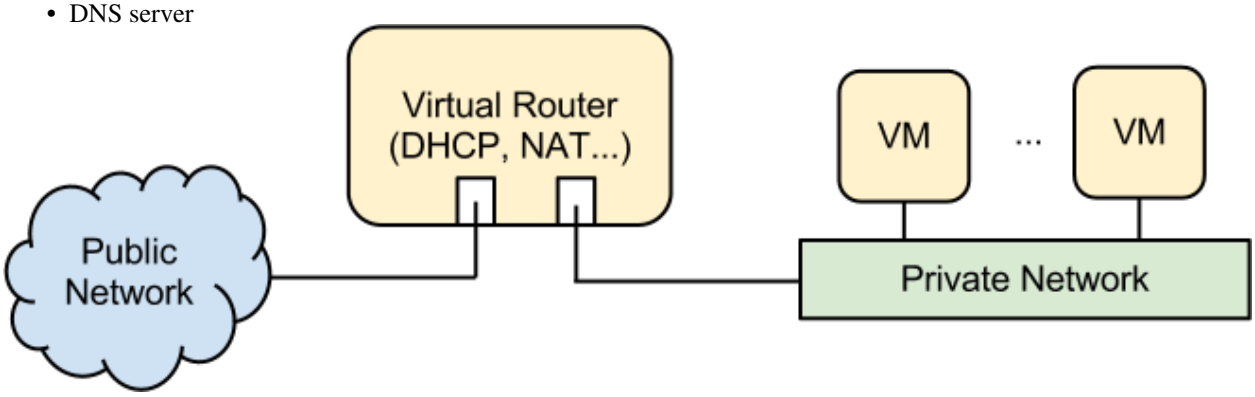
4.7 Virtual Router

This guide describes how to use the Virtual Router in OpenNebula.

4.7.1 Overview

When instantiated in a network, this appliance provides the following services for other Virtual Machines running in the same network:

- Router (masquerade)
- Port forwarding
- DHCP server
- RADVD server
- DNS server



A big advantage of using this appliance is that Virtual Machines can be run in the same network **without** being *contextualized* for OpenNebula.

This appliance is controlled via `CONTEXT`. More information in the following sections.

4.7.2 Considerations & Limitations

This is a 64-bit appliance and will run both in `KVM`, `Xen` and `VMware` environments. It will run with any network driver.

Since each virtual router will start a DHCP server and it's not recommended to have more than one DHCP server per network, it's recommend to use it along network isolation drivers if you're going to deploy two or more router instances in your environment:

- *Open vSwitch*
- *Ebtables*
- *802.1Q VLAN*

4.7.3 Configuration

The appliance is based on alpinelinux. There's only one user account: `root`. There is no default password for the root account, however, it can be specified in the `CONTEXT` section along with root's public key.

- **ROOT_PUBKEY**: If set, it will be set as root's authorized_keys.
- **ROOT_PASSWORD**: To change the root account password use this attribute. It expects the password in an encrypted format as returned by `openssl passwd -1` and encoded in base64.

4.7.4 Usage

The virtual router can be used in two ways:

DHCP or RADVD Server

Only one interface. Useful if you only want DHCP or RADVD. Of course, enabling RADVD only makes sense if the private network is IPv6.

To enable this you need to add the following context to the VM:

```
TARGET   = "hdb"
PRIVNET  = "$NETWORK[TEMPLATE, NETWORK=\"private_network_name\"]",
TEMPLATE = "$TEMPLATE"
DHCP     = "YES|NO"
RADVD    = "YES|NO"
```

If you're going to a use a netmask different to '255.255.255.0' you will have to add the following to the private network's template:

```
NETWORK_MASK = 255.255.255.254
```

Full Router

In this case, the Virtual Machine will need two network interfaces: a private and a public one. The public one will be masqueraded. In this mode you can also configure a DNS server by setting the `DNS` and optionally the `SEARCH` attribute (useful for domain searches in `/etc/resolv.conf`). This mode also includes all the attributes related to the previous section, i.e. DHCP and RADVD servers.

This is an example context for the router mode:

```
TARGET     = "hdb"
PRIVNET    = "$NETWORK[TEMPLATE, NETWORK=\"private_network\"]",
PUBNET     = "$NETWORK[TEMPLATE, NETWORK=\"public_network\"]",
TEMPLATE   = "$TEMPLATE"
DHCP       = "YES|NO"
RADVD      = "YES|NO" # Only useful for an IPv6 private network
NTP_SERVER = "10.0.10.1"
DNS        = "8.8.4.4 8.8.8.8"
SEARCH     = "local.domain"
FORWARDING = "8080:10.0.10.10:80 10.0.10.10:22"
```

DNS

This attribute expects a list of dns servers separated by spaces.

NTP_SERVER

This attribute expects the IP of the NTP server of the cluster. The DHCP server will be configured to serve the NTP parameter to its leases.

FORWARDING

This attribute expects a list of forwarding rules separated by spaces. Each rule has either 2 or 3 components separated by `:`. If only two components are specified, the first is the IP to forward the port to, and the second is the port number. If there are three components, the first is the port in the router, the second the IP to forward to, and the third the port in the forwarded Virtual Machine. Examples:

- `8080:10.0.10.10:80` This will forward the port 8080 in the router to the port 80 to the VM with IP 10.0.10.10.
- `10.0.10.10:22` This will forward the port 22 in the router to the port 22 to the VM with IP 10.0.10.10.

If the public network uses a netmask different to `255.255.255.0` or if the gateway is not the ip's network with one as the last byte: `x.y.z.1` it can be explicitely set adding the following attributes to the public network's template:

```
GATEWAY      = "192.168.1.100"
NETWORK_MASK = "255.255.254.0"
```

CHAPTER

FIVE

MONITORING

5.1 Monitoring Overview

This guide 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 VM status and capacity consumption. This information is collected by executing a set of static probes provided by OpenNebula. The output of these probes is sent to OpenNebula in two different ways: using a push or a pull paradigm. Below you can find a brief description of the two models and when to use one or the other.

5.1.1 The UDP-push Model

Warning: **Default**. This is the default IM for KVM and Xen in OpenNebula >= 4.4.

In this model, 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.

This model is highly scalable and its limit (in terms of number of VMs monitored per second) is bounded to the performance of the server running oned and the database server.

Please read the *UDP-push guide* for more information.

When to Use the UDP-push Model

This mode can be used only with Xen and KVM (VMware only supports the SSH-pull mode).

This monitoring model is adequate when:

- You are using KVM or Xen (VMware is not supported in this mode)
- Your infrastructure has a medium-to-high number of hosts (e.g. more than 50)
- You need a high responsive system
- You need a high frequently updated monitor information
- All your hosts communicate through a secure network (UDP packages are not encrypted and their origin is not verified)

5.1.2 The Pull Model

When using this mode OpenNebula periodically actively queries each host and executes the probes via `ssh`. In KVM and Xen this means establishing an ssh connection to each host and executing several scripts to retrieve this information. Note that VMware uses the VI API for this, instead of a `ssh` connection.

This mode is limited by the number of active connections that can be made concurrently, as hosts are queried sequentially.

Please read the *KVM and Xen SSH-pull guide* or the *ESX-pull guide* for more information.

When to Use the SSH-pull Model

This mode can be used with VMware, Xen and KVM.

This monitoring model is adequate when:

- Your infrastructure has a low number of hosts (e.g. 50 or less)
- You are communicating with the hosts through an insecure network
- You do not need to update the monitoring with a high frequency (e.g. for 50 hosts the monitoring period would be typically of about 5 minutes)

5.1.3 Other Monitorization Systems

OpenNebula can be easily integrated with other monitorization system. Please read the *Information Manager Driver integration guide* for more information.

5.1.4 The Monitor Metrics

The information manage by the monitoring system includes the typical performance and configuration parameters for the host and VMs, e.g. CPU or network consumption, Hostname or CPU model.

These metrics are gathered by specialized programs, called probes, that can be easily added to the system. Just write your own program, or shell script that returns the metric that you are interested in. Please read the *Information Manager Driver integration guide* for more information.

5.2 KVM and Xen SSH-pull Monitoring

KVM and Xen can be monitored with this `ssh` based monitoring system. The OpenNebula frontend starts a driver which triggers `ssh` connections to the hosts which return the monitoring information of the host and of all the virtual machines running within.

5.2.1 Requirements

- `ssh` access from the frontends to the hosts as `oneadmin` without password has to be possible.
- `ruby` is required in the hosts.
- KVM hosts: `libvirt` must be enabled.
- Xen hosts: `sudo` access to run `xl` or `xm` and `xentop` as `oneadmin`.

5.2.2 OpenNebula Configuration

Enabling the Drivers

To enable this monitoring system `/etc/one/oned.conf` must be configured with the following snippets:

KVM:

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

Xen 3.x:

```
IM_MAD = [
    name       = "xen",
    executable = "one_im_ssh",
    arguments  = "-r 0 -t 15  xen3-probes" ]
```

Xen 4.x:

```
IM_MAD = [
    name       = "xen",
    executable = "one_im_ssh",
    arguments  = "-r 0 -t 15 xen4-probes" ]
```

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 behaviour 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.
VM_PER_INTERVAL	Number of VMs monitored in each interval.

Warning: VM_PER_INTERVAL is only relevant in case of host failure when OpenNebula pro-actively monitors each VM.

The information gathered by the probes is also stored in a monitoring table. This table is used by Sunstone to draw monitoring graphics and can be queried using the OpenNebula API. The size of this table can be controlled with:

Parameter	Description
HOST_MONITORING_EXPIRATION_TIME	Time, in seconds, to expire monitoring information. Use 0 to disable HOST monitoring recording.
VM_MONITORING_EXPIRATION_TIME	Time, in seconds, to expire monitoring information. Use 0 to disable VM monitoring recording.

5.2.3 Troubleshooting

In order to test the driver, add a host to OpenNebula using **onehost**, specifying the defined IM driver:

```
$ onehost create ursa06 --im xen --vm xen --net dummy
```

Now give it time to monitor the host (this time is determined by the value of MONITORING_INTERVAL in `/etc/one/oned.conf`). After one interval, check the output of **onehost list**, it should look like the following:

```
$ onehost list
  ID NAME            CLUSTER   RVM      ALLOCATED_CPU      ALLOCATED_MEM STAT
   0 ursa06          -           0       0 / 400 (0%)     0K / 7.7G (0%) on
```

Host management information is logged to `/var/log/one/oned.log`. Correct monitoring log lines look like this:

```
Fri Nov 22 12:02:26 2013 [InM][D]: Monitoring host ursa06 (0)
Fri Nov 22 12:02:30 2013 [InM][D]: Host ursa06 (0) successfully monitored.
```

Both lines have the ID of the host being monitored.

If there are problems monitoring the host you will get an `err` state:

```
$ onehost list
  ID NAME            CLUSTER   RVM      ALLOCATED_CPU      ALLOCATED_MEM STAT
   0 ursa06          -           0       0 / 400 (0%)     0K / 7.7G (0%) err
```

The way to get the error message for the host is using `onehost show` command, specifying the host id or name:

```
$ onehost show 0
[...]
MONITORING INFORMATION
ERROR=[
  MESSAGE="Error monitoring host 0 : MONITOR FAILURE 0 Could not update remotes",
  TIMESTAMP="Nov 22 12:02:30 2013" ]
```

The log file is also useful as it will give you even more information on the error:

```
Mon Oct  3 15:26:57 2011 [InM][I]: Monitoring host ursa06 (0)
Mon Oct  3 15:26:57 2011 [InM][I]: Command execution fail: scp -r /var/lib/one/remotes/. ursa06:/var/
Mon Oct  3 15:26:57 2011 [InM][I]: ssh: Could not resolve hostname ursa06: nodename nor servname prov
Mon Oct  3 15:26:57 2011 [InM][I]: lost connection
Mon Oct  3 15:26:57 2011 [InM][I]: ExitCode: 1
Mon Oct  3 15:26:57 2011 [InM][E]: Error monitoring host 0 : MONITOR FAILURE 0 Could not update remot
```

In this case the node `ursa06` could not be found in the DNS or `/etc/hosts`.

5.2.4 Tuning & Extending

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/<hypervisor>-probes.d` for Xen and 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.

5.3 KVM and Xen UDP-push Monitoring

KVM and Xen can be monitored with this `UDP` based monitoring system.

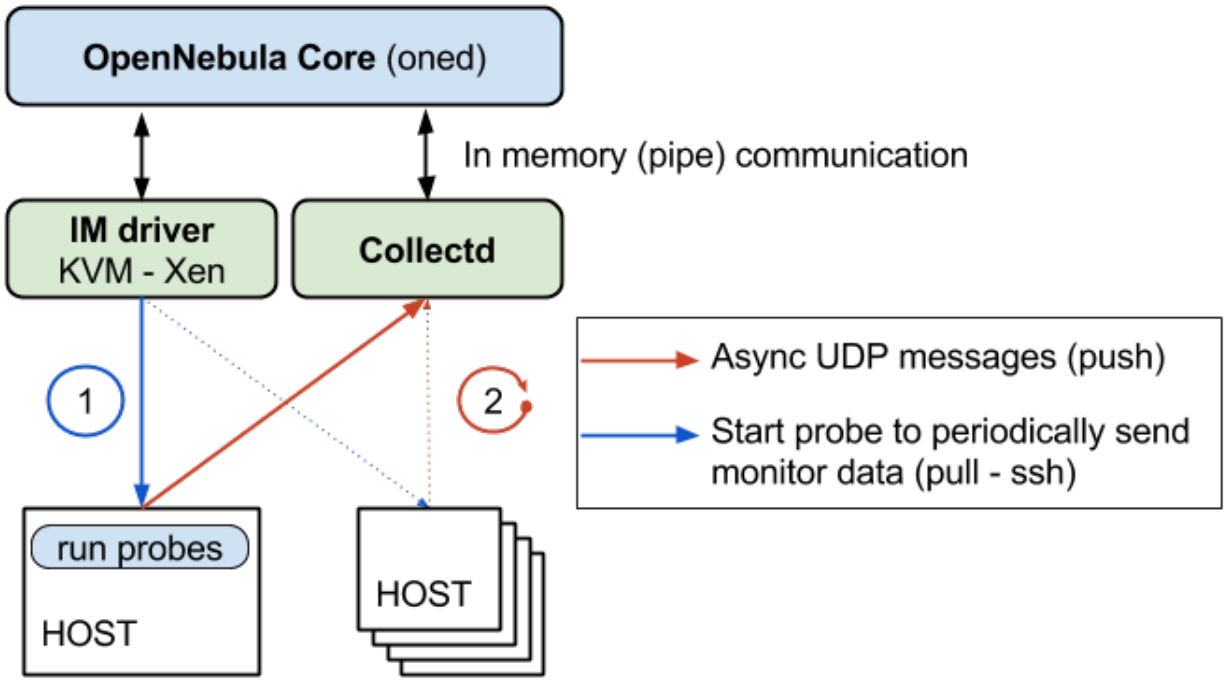
Monitorization data is sent from each host to the frontend periodically via `UDP` by an agent. This agent is started by the initial bootstrap system of the monitoring system which is performed via `ssh` like with the SSH-pull system.

5.3.1 Requirements

- `ssh` access from the frontends to the hosts as `oneadmin` without password has to be possible.
- `ruby` is required in the hosts.
- KVM hosts: `libvirt` must be enabled.
- Xen hosts: `sudo` access to run `xl` or `xm` and `xentop` as `oneadmin`.
- The firewall of the frontend (if enabled) must allow UDP packages incoming from the hosts on port 4124.

5.3.2 Overview

OpenNebula starts a `collectd` daemon running in the frontend host 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 as in the SSH-pull model and sends the collected data to the collectd daemon in the fronted 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 the hosts when it needs data.



If the agent stops in a specific host, OpenNebula will detect that no monitorization data is received from that hosts and will automatically fallback to the SSH-pull model, thus starting the agent again in the host.

5.3.3 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 and Xen:

```
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 sockect (defults 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 (defult 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:

```
IM_MAD = [
      name       = "kvm",
      executable = "one_im_ssh",
      arguments  = "-r 3 -t 15 kvm" ]
```

Xen 3:

```
IM_MAD = [
    name       = "xen",
    executable = "one_im_ssh",
    arguments  = "-r 3 -t 15 xen3" ]
```

Xen 4:

```
IM_MAD = [
    name       = "xen",
    executable = "one_im_ssh",
    arguments  = "-r 3 -t 15 xen4" ]
```

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 behaviour 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.

Warning: Note that in this case HOST_PER_INTERVAL is only relevant when bootstraping the monitor agents. Once the agents are up and running, OpenNebula does not polls the hosts.

5.3.4 Troubleshooting

Healthy Monitoring System

If the `UDP-push` model is running successfully, it means that it has not fallen back to the `SSH-pull` model. We can verify this based on the information logged in `oned.log`.

Every (approximately) `monitoring_push_cycle` of seconds OpenNebula is receiving the monitoring data of every Virtual Machine and of a host like such:

```
Mon Nov 18 22:25:00 2013 [InM][D]: Host thost001 (1) successfully monitored.
Mon Nov 18 22:25:01 2013 [VMM][D]: VM 0 successfully monitored: ...
Mon Nov 18 22:25:21 2013 [InM][D]: Host thost001 (1) successfully monitored.
Mon Nov 18 22:25:21 2013 [VMM][D]: VM 0 successfully monitored: ...
Mon Nov 18 22:25:40 2013 [InM][D]: Host thost001 (1) successfully monitored.
Mon Nov 18 22:25:41 2013 [VMM][D]: VM 0 successfully monitored: ...
```

However, if in `oned.log` a host is being monitored **actively** periodically (every `MONITORING_INTERVAL` seconds) then the `UDP-push` monitorization is **not** working correctly:

```
Mon Nov 18 22:22:30 2013 [InM][D]: Monitoring host thost087 (87)
Mon Nov 18 22:23:30 2013 [InM][D]: Monitoring host thost087 (87)
Mon Nov 18 22:24:30 2013 [InM][D]: Monitoring host thost087 (87)
```

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 of the *SSH-pull guide*.

5.3.5 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/<hypervisor>-probes.d` for Xen and 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.

5.4 VMware VI API-pull Monitor

5.4.1 Requirements

- VI API access to the ESX hosts.
- ESX hosts configured to work with OpenNebula

5.4.2 OpenNebula Configuration

In order to configure VMware you need to:

- Enable the VMware monitoring driver in `/etc/one/oned.conf` by uncommenting the following lines:

```
IM_MAD = [
      name       = "vmware",
      executable = "one_im_sh",
      arguments  = "-c -t 15 -r 0 vmware" ]
```

- Make sure that the configuration attributes for VMware drivers are set in `/etc/one/vmwarerc`, see the *VMware guide*

Monitoring Configuration Parameters

OpenNebula allows to customize the general behaviour 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.
VM_PER_INTERVAL	Number of VMs monitored in each interval.

Warning: VM_PER_INTERVAL is only relevant in case of host failure when OpenNebula pro-actively monitors each VM.

The information gathered by the probes is also stored in a monitoring table. This table is used by Sunstone to draw monitoring graphics and can be queried using the OpenNebula API. The size of this table can be controlled with:

Parameter	Description
HOST_MONITORING_EXPIRATION_TIME	Time, in seconds, to expire monitoring information. Use 0 to disable HOST monitoring recording.
VM_MONITORING_EXPIRATION_TIME	Time, in seconds, to expire monitoring information. Use 0 to disable VM monitoring recording.

5.4.3 Troubleshooting

In order to test the driver, add a host to OpenNebula using **onehost**, specifying the defined IM driver:

```
$ onehost create esx_node1 --im vmware --vm vmware --net dummy
```

Now give it time to monitor the host (this time is determined by the value of MONITORING_INTERVAL in `/etc/one/oned.conf`). After one interval, check the output of **onehost list**, it should look like the following:

```
$ onehost list
  ID NAME            CLUSTER   RVM      ALLOCATED_CPU      ALLOCATED_MEM STAT
   0 esx_node1       -           0       0 / 400 (0%)     0K / 7.7G (0%) on
```

Host management information is logged to `/var/log/one/oned.log`. Correct monitoring log lines look like this:

```
Fri Nov 22 12:02:26 2013 [InM][D]: Monitoring host esx_node1 (0)
Fri Nov 22 12:02:30 2013 [InM][D]: Host esx1_node (0) successfully monitored.
```

Both lines have the ID of the host being monitored.

If there are problems monitoring the host you will get an `err` state:

```
$ onehost list
  ID NAME            CLUSTER   RVM      ALLOCATED_CPU      ALLOCATED_MEM STAT
   0 esx_node1        -           0       0 / 400 (0%)     0K / 7.7G (0%) err
```

The way to get the error message for the host is using `onehost show` command, specifying the host id or name:

```
$ onehost show 0
[...]
MONITORING INFORMATION
ERROR=[
  MESSAGE="Error monitoring host 0 : MONITOR FAILURE 0 Could not update remotes",
  TIMESTAMP="Nov 22 12:02:30 2013" ]
```

The log file is also useful as it will give you even more information on the error.

5.4.4 Tuning & Extending

The probes are specialized programs that obtain the monitor metrics. VMware probes are obtained by querying the ESX server through the VI API. The probe is located at `/var/lib/one/remotes/im/vmware.d`.

You can easily write your own probes or modify existing ones, please see the *Information Manager Drivers* guide.

CHAPTER

SIX

USERS AND GROUPS

6.1 Users & Groups Overview

OpenNebula includes a complete user & group management system. Users in an OpenNebula installation are classified in four types:

- **Administrators**, an admin user belongs to the oneadmin group and can perform any operation
- **Regular users**, that may access most OpenNebula functionality.
- **Public users**, only basic functionality (and public interfaces) are open to public users.
- **Service users**, a service user account is used by the OpenNebula services (i.e. cloud APIs like EC2 or GUI's like Sunstone) to proxy auth requests.

The resources a user may access in OpenNebula are controlled by a permissions system that resembles the typical UNIX one. By default, only the owner of a resource (e.g. a VM or an image) can use and manage it. Users can easily share the resources by granting use or manage permissions to other users in her group or to any other user in the system.

Along with the users & groups the Auth Subsystem is responsible for the authentication and authorization of user's requests.

Any interface to OpenNebula (CLI, Sunstone, Ruby or Java OCA) communicates with the core using xml-rpc calls, that contain the user's session string, which is authenticated by the OpenNebula core comparing the username and password with the registered users.

Each operation generates an authorization request that is checked against the registered ACL rules. The core then can grant permission, or reject the request.

OpenNebula comes with a default set of ACL rules that enables a standard usage. You don't need to manage the ACL rules unless you need the level of permission customization if offers.

Please proceed to the following guides to learn more:

- *Managing Users and Groups*
- *Managing Permissions*
- *Managing ACL Rules*
- *Quota Management*

By default, the authentication and authorization is handled by the OpenNebula Core as described above. Optionally, you can delegate it to an external module, see the *External Auth Setup* guide for more information.

6.2 Managing Users and Groups

OpenNebula supports user accounts and groups. This guide shows how to manage both. To manage user rights, visit the *Managing ACL Rules* guide.

6.2.1 Users

A user in OpenNebula is defined by a username and password. You don't need to create a new Unix account in the front-end for each OpenNebula user, they are completely different concepts. OpenNebula users are authenticated using a session string included in every *operation*, which is checked by the OpenNebula core.

Each user has a unique ID, and belongs to a group.

After the installation, you will have two administrative accounts, `oneadmin` and `serveradmin`; and two default groups. You can check it using the `oneuser list` and `onegroup list` commands.

There are different user types in the OpenNebula system:

- **Administrators**, the **oneadmin** account is created **the first time** OpenNebula is started using the ONE_AUTH data. `oneadmin` has enough privileges to perform any operation on any object. Any other user in the oneadmin group has the same privileges as `oneadmin`
- **Regular user** accounts may access most of the functionality offered by OpenNebula to manage resources.
- **Public users** can only access OpenNebula through a public API (e.g. OCCI, EC2), hence they can only use a limited set of functionality and can not access the xml-rpc API directly (nor any application using it like the CLI or SunStone)
- User **serveradmin** is also created the first time OpenNebula is started. Its password is created randomly, and this account is used by the *Sunstone*, *OCCI* and *EC2* servers to interact with OpenNebula.

OpenNebula users should have the following environment variables set, you may want to place them in the .bashrc of the user's Unix account for convenience:

ONE_XMLRPC

URL where the OpenNebula daemon is listening. If it is not set, CLI tools will use the default: **http://localhost:2633/RPC2**. See the `PORT` attribute in the *Daemon configuration file* for more information.

ONE_AUTH

Needs to point to **a file containing just a single line stating ''username:password''**. If ONE_AUTH is not defined, $HOME/.one/one_auth will be used instead. If no auth file is present, OpenNebula cannot work properly, as this is needed by the core, the CLI, and the cloud components as well.

If OpenNebula was installed from sources in **self-contained mode** (this is not the default, and not recommended), these two variables must be also set. Usually, these are not needed.

ONE_LOCATION

It must point to the installation <destination_folder>.

PATH

`$ONE_LOCATION/bin`:$PATH .

For instance, a user named `regularuser` may have the following environment:

```
$ tail ~/.bashrc

ONE_XMLRPC=http://localhost:2633/RPC2
```

```
export ONE_XMLRPC

$ cat ~/.one/one_auth
regularuser:password
```

Warning: Please note that the example above is intended for a user interacting with OpenNebula from the front-end, but you can use it from any other computer. Just set the appropriate hostname and port in the ONE_XMLRPC variable.

An alternative method to specify credentials and OpenNebula endpoint is using command line parameters. Most of the commands can understand the following parameters:

`--user name`

User name used to connect to OpenNebula

`--password password`

Password to authenticate with OpenNebula

`--endpoint endpoint`

URL of OpenNebula xmlrpc frontend

If `user` is specified but not `password` the user will be prompted for the password. `endpoint` has the same meaning and get the same value as `ONE_XMLRPC`. For example:

```
$ onevm list --user my_user --endpoint http://one.frontend.com:2633/RPC2
Password:
[...]
```

Warning: You should better not use `--password` parameter in a shared machine. Process parameters can be seen by any user with the command `ps` so it is highly insecure.

Adding and Deleting Users

User accounts within the OpenNebula system are managed by `oneadmin` with the `oneuser create` and `oneuser delete` commands. This section will show you how to create the different account types supported in OpenNebula

Administrators

Administrators can be easily added to the system like this:

```
$ oneuser create otheradmin password
ID: 2

$ oneuser chgrp otheradmin oneadmin

$ oneuser list
  ID GROUP    NAME            AUTH                                     PASSWORD
   0 oneadmin oneadmin        core      5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8
   1 oneadmin serveradmin     server_c  1224ff12545a2e5dfeda4eddacdc682d719c26d5
   2 oneadmin otheradmin      core      5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8

$ oneuser show otheradmin
```

```
USER 2 INFORMATION
ID              : 2
NAME            : otheradmin
GROUP           : 0
PASSWORD        : 5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8
AUTH_DRIVER     : core
ENABLED         : Yes

USER TEMPLATE
```

Regular Users

Simply create the usets with the create command:

```
$ oneuser create regularuser password
ID: 3
```

The enabled flag can be ignored as it doesn't provide any functionality. It may be used in future releases to temporarily disable users instead of deleting them.

Public Users

Public users needs to define a special authentication method that internally relies in the core auth method. First create the public user as it was a regular one:

```
$ oneuser create publicuser password
ID: 4
```

and then change its auth method (see below for more info) to the public authentication method.

```
$ oneuser chauth publicuser public
```

Server Users

Server user accounts are used mainly as proxy authentication accounts for OpenNebula services. Any account that uses the server_cipher or server_x509 auth methods are a server user. You will never use this account directly. To create a user account just create a regular account

```
$ oneuser create serveruser password
ID: 5
```

and then change its auth method to `server_cipher` (for other auth methods please refer to the *External Auth guide*):

```
$ oneuser chauth serveruser server_cipher
```

Managing Users

User Authentication

Each user has an authentication driver, `AUTH_DRIVER`. The default driver, `core`, is a simple user-password match mechanism. Read the *External Auth guide* to improve the security of your cloud, enabling *SSH* or *x509* authentication.

User Templates

The `USER TEMPLATE` section can hold any arbitrary data. You can use the `oneuser update` command to open an editor and add, for instance, the following `DEPARTMENT` and `EMAIL` attributes:

```
$ oneuser show 2
USER 2 INFORMATION
ID              : 2
NAME            : regularuser
GROUP           : 1
PASSWORD        : 5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8
AUTH_DRIVER     : core
ENABLED         : Yes

USER TEMPLATE
DEPARTMENT=IT
EMAIL=user@company.com
```

These attributes can be later used in the *Virtual Machine Contextualization*. For example, using contextualization the user's public ssh key can be automatically installed in the VM:

```
ssh_key = "$USER[SSH_KEY]"
```

Manage your Own User

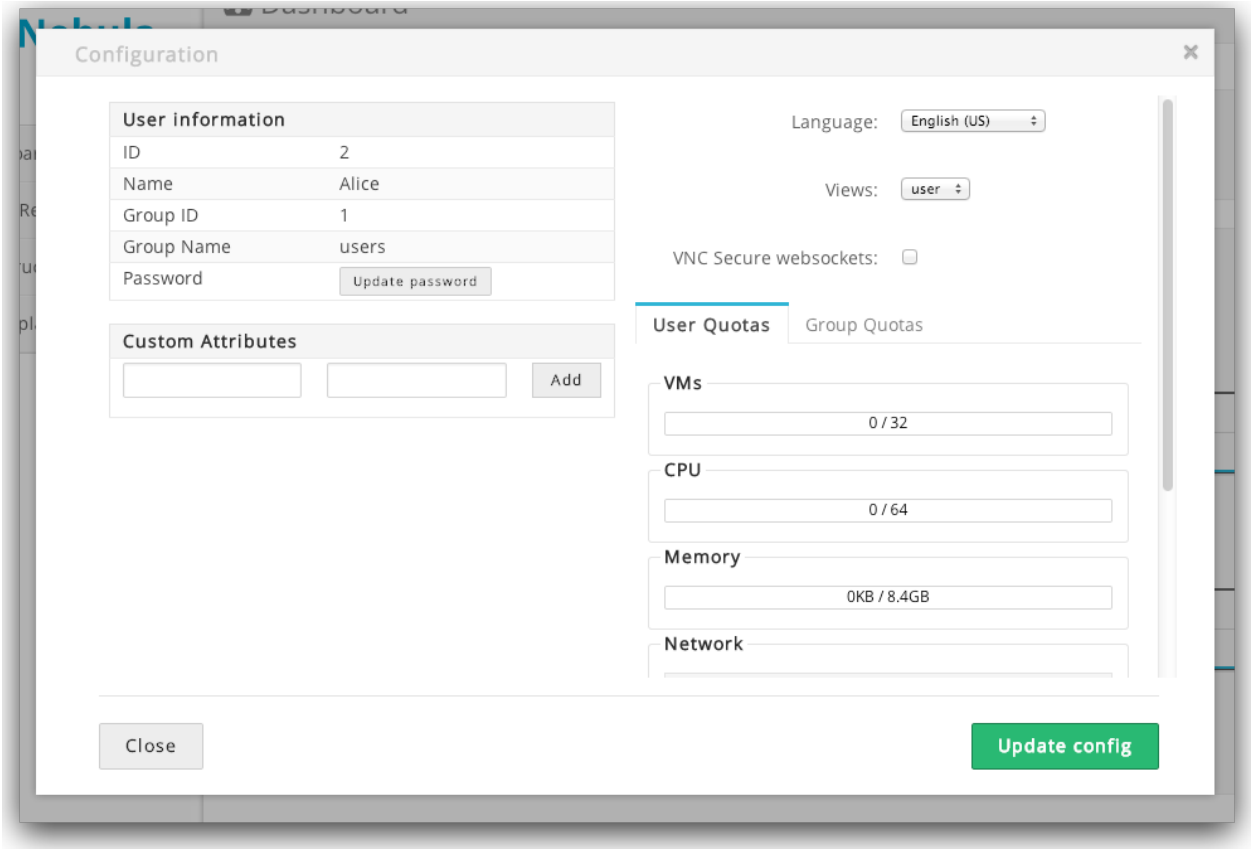
Regular users can see their account information, and change their password.

For instance, as `regularuser` you could do the following:

```
$ oneuser list
[UserPoolInfo] User [2] not authorized to perform action on user.

$ oneuser show
USER 2 INFORMATION
ID              : 2
NAME            : regularuser
GROUP           : 1
PASSWORD        : 5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8
AUTH_DRIVER     : core
ENABLED         : Yes

USER TEMPLATE
DEPARTMENT=IT
EMAIL=user@company.com

$ oneuser passwd 1 abcdpass
```

As you can see, any user can find out his ID using the `oneuser show` command without any arguments.

Regular users can retrieve their quota and user information in the settings section in the top right corner of the main



screen:

6.2.2 Groups

A group in OpenNebula makes possible to isolate users and resources. A user can see and use the *shared resources* from other users.

There are two special groups created by default. The `onedmin` group allows any user in it to perform any operation, allowing different users to act with the same privileges as the `oneadmin` user. The `users` group is the default group where new users are created.

Adding and Deleting Groups

Your can use the `onegroup` command line tool to manage groups in OpenNebula. There are two groups created by default, `oneadmin` and `users`.

To create new groups:

```
$ onegroup list
  ID NAME
   0 oneadmin
   1 users

$ onegroup create "new group"
ID: 100
ACL_ID: 2
ACL_ID: 3
```

The new group has ID 100 to differentiate the special groups to the user-defined ones.

When a new group is created, two ACL rules are also created to provide the default behaviour. You can learn more about ACL rules in *this guide*; but you don't need any further configuration to start using the new group.

Adding Users to Groups

Use the `oneuser chgrp` command to assign users to groups.

```
$ oneuser chgrp -v regularuser "new group"
USER 1: Group changed

$ onegroup show 100
GROUP 100 INFORMATION
ID              : 100
NAME            : new group

USERS
ID               NAME
1                regularuser
```

To delete a user from a group, just move it again to the default `users` group.

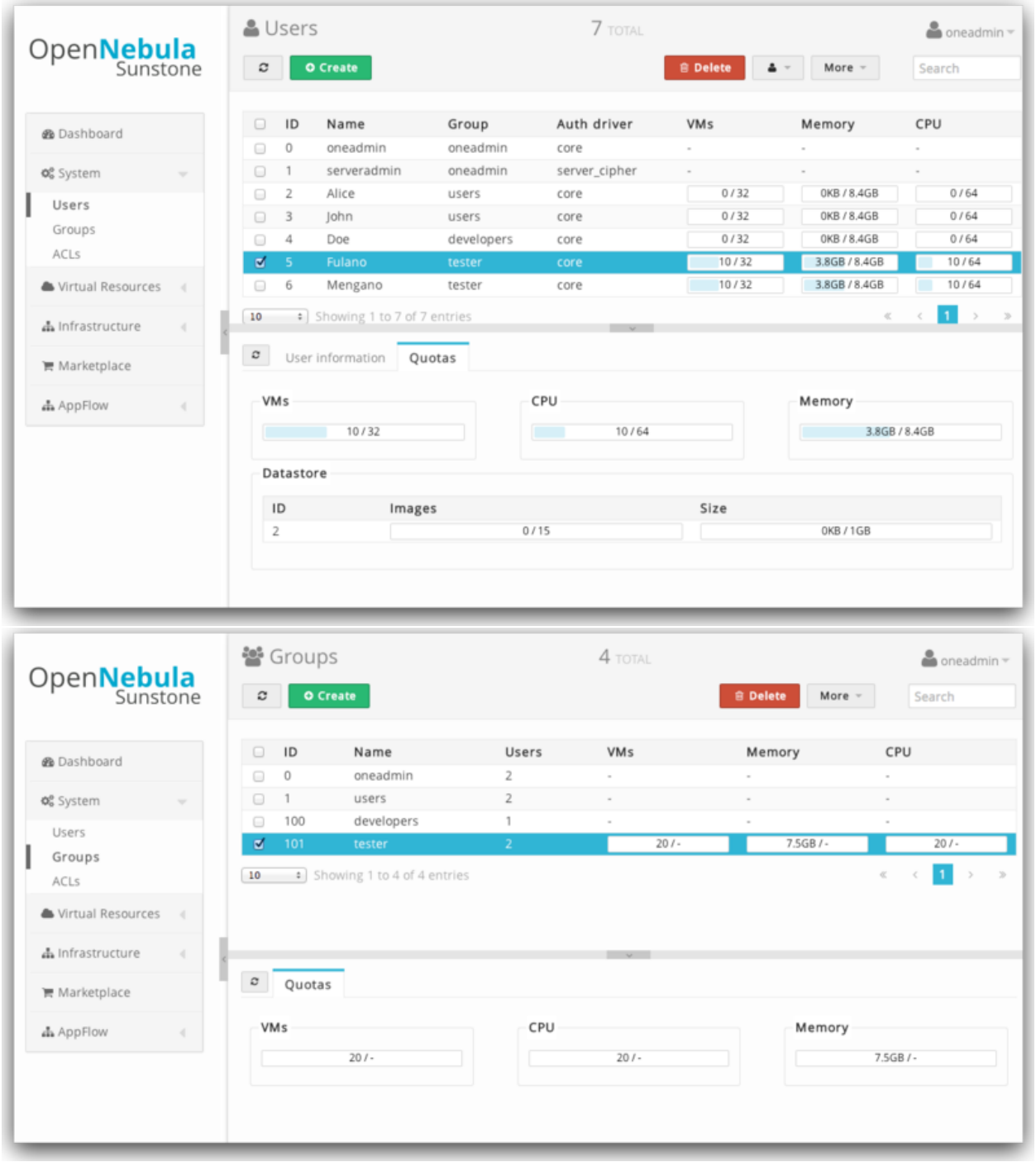
Primary and Secondary Groups

With the commands `oneuser addgroup` and `delgroup` the administrator can add or delete secondary groups. Users assigned to more than one group will see the resources from all their groups. e.g. a user in the groups testing and production will see VMs from both groups.

The group set with `chgrp` is the primary group, and resources (Images, VMs, etc) created by a user will belong to this primary group. Users can change their primary group to any of their secondary group without the intervention of an administrator, using `chgrp` again.

6.2.3 Managing Users and Groups in Sunstone

All the described functionality is available graphically using *Sunstone*:



6.3 Managing Permissions

Most OpenNebula resources have associated permissions for the **owner**, the users in her **group**, and **others**. For each one of these groups, there are three rights that can be set: **USE**, **MANAGE** and **ADMIN**. These permissions are very similar to those of UNIX file system.

The resources with associated permissions are *Templates*, *VMs*, *Images* and *Virtual Networks*. The exceptions are *Users*, *Groups* and *Hosts*.

6.3.1 Managing Permission through the CLI

This is how the permissions look in the terminal:

```
$ onetemplate show 0
TEMPLATE 0 INFORMATION
ID              : 0
NAME            : vm-example
USER            : oneuser1
GROUP           : users
REGISTER TIME   : 01/13 05:40:28

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

[...]
```

The previous output shows that for the Template 0, the owner user `oneuser1` has **USE** and **MANAGE** rights. Users in the group `users` have **USE** rights, and users that are not the owner or in the `users` group don't have any rights over this Template.

You can check what operations are allowed with each of the **USE**, **MANAGE** and **ADMIN** rights in the *xml-rpc reference documentation*. In general these rights are associated with the following operations:

- **USE**: Operations that do not modify the resource like listing it or using it (e.g. using an image or a virtual network). Typically you will grant `USE` rights to share your resources with other users of your group or with the rest of the users.
- **MANAGE**: Operations that modify the resource like stopping a virtual machine, changing the persistent attribute of an image or removing a lease from a network. Typically you will grant `MANAGE` rights to users that will manage your own resources.
- **ADMIN**: Special operations that are typically limited to administrators, like updating the data of a host or deleting an user group. Typically you will grant `ADMIN` permissions to those users with an administrator role.

Warning: By default every user can update any permission group (owner, group or other) with the exception of the admin bit. There are some scenarios where it would be advisable to limit the other set (e.g. OpenNebula Zones so users can not break the VDC limits). In these situations the `ENABLE_OTHER_PERMISSIONS` attribute can be set to `NO` in `/etc/one/oned.conf` file

Changing Permissions with chmod

The previous permissions can be updated with the chmod command. This command takes an octet as a parameter, following the octal notation of the Unix chmod command. The octet must be a three-digit base-8 number. Each digit, with a value between 0 and 7, represents the rights for the **owner**, **group** and **other**, respectively. The rights are represented by these values:

- The **USE** bit adds 4 to its total (in binary 100)
- The **MANAGE** bit adds 2 to its total (in binary 010)
- The **ADMIN** bit adds 1 to its total (in binary 001)

Let's see some examples:

```
$ onetemplate show 0
...
PERMISSIONS
OWNER           : um-
GROUP           : u--
OTHER           : ---

$ onetemplate chmod 0 664 -v
VMTEMPLATE 0: Permissions changed

$ onetemplate show 0
...
PERMISSIONS
OWNER           : um-
GROUP           : um-
OTHER           : u--

$ onetemplate chmod 0 644 -v
VMTEMPLATE 0: Permissions changed

$ onetemplate show 0
...
PERMISSIONS
OWNER           : um-
GROUP           : u--
OTHER           : u--

$ onetemplate chmod 0 607 -v
VMTEMPLATE 0: Permissions changed

$ onetemplate show 0
...
PERMISSIONS
OWNER           : um-
GROUP           : ---
OTHER           : uma
```

Setting Default Permissions with umask

The default permissions given to newly created resources can be set:

- Globally, with the **DEFAULT_UMASK** attribute in *oned.conf*
- Individually for each User, using the *oneuser umask command*.

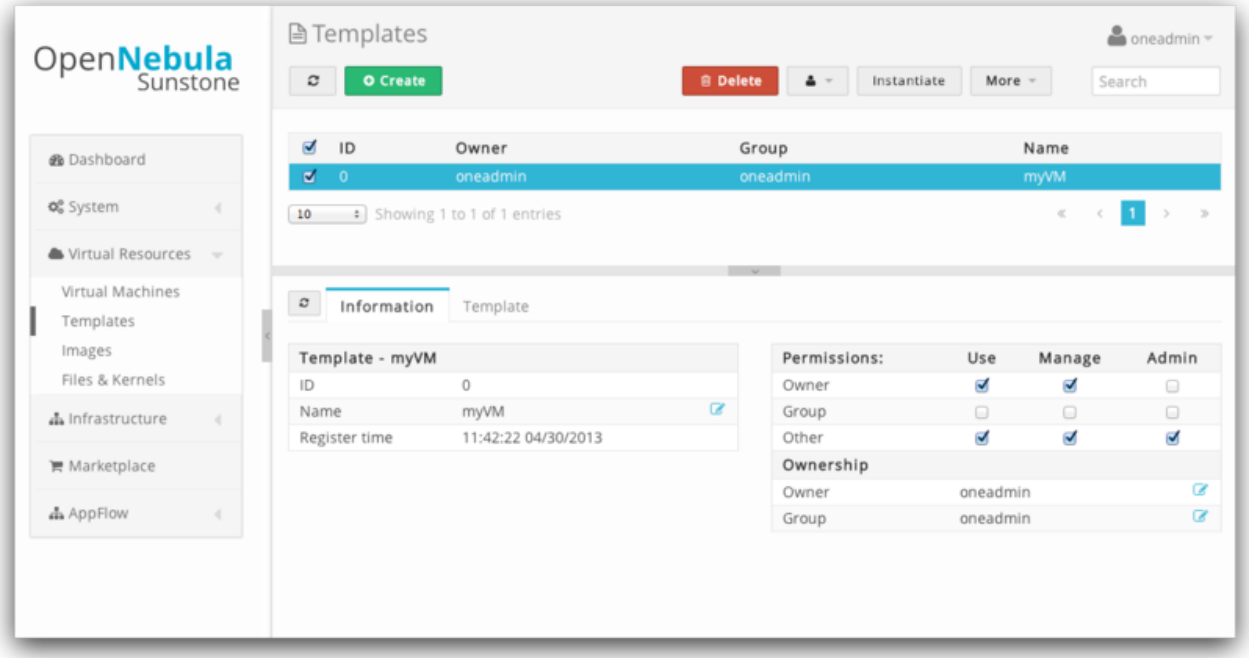
These mask attributes work in a similar way to the Unix umask command. The expected value is a three-digit base-8 number. Each digit is a mask that **disables** permissions for the **owner**, **group** and **other**, respectively.

This table shows some examples:

umask	permissions (octal)	permissions
177	600	um- --- ---
137	640	um- u-- ---
113	664	um- um- u--

6.3.2 Managing Permissions in Sunstone

Sunstone offers a convenient way to manage resources permissions. This can be done by selecting resources from a view (for example the templates view) and clicking on the `update properties` button. The update dialog lets the user conveniently set the resource's permissions.



6.4 Accounting Client

The accounting toolset visualizes and reports resource usage data, and allows their integration with chargeback and billing platforms. The toolset generates accounting reports using the information retrieved from OpenNebula.

This accounting tool addresses the accounting of the virtual resources. It includes resource consumption of the virtual machines as reported from the hypervisor.

6.4.1 Usage

`oneacct` - prints accounting information for virtual machines

```
Usage: oneacct [options]
  -s, --start TIME        Start date and time to take into account
  -e, --end TIME          End date and time
  -u, --user user         User id to filter the results
  -g, --group group       Group id to filter the results
  -H, --host hostname     Host id to filter the results
      --xpath expression  Xpath expression to filter the results. For example: oneacct --xpath 'HIST
  -j, --json              Output in json format
  -x, --xml               Output in xml format
      --split             Split the output in a table for each VM
  -h, --help              Show this message
```

The time can be written as `month/day/year hour:minute:second`, or any other similar format, e.g `month/day hour:minute`.

To integrate this tool with your billing system you can use `-j` or `-x` flags to get all the information in an easy computer readable format.

6.4.2 Accounting Output

The oneacct command shows individual Virtual Machine history records. This means that for a single VM you may get several accounting entries, one for each migration or stop/suspend action.

Each entry contains the complete information of the Virtual Machine, including the Virtual Machine monitoring information. By default, only network consumption is reported, see the Tuning & Extending section for more information.

When the results are filtered with the `-s` and/or `-e` options, all the history records that were active during that time interval are shown, but they may start or end outside that interval.

For example, if you have a VM that was running from 01/01/2012 to 05/15/2012, and you request the accounting information with this command:

```
$ oneacct -s '02/01/2012' -e '01/03/2012'
Showing active history records from Wed Feb 01 00:00:00 +0100 2012 to Tue Jan 03 00:00:00 +0100 2012

 VID HOSTNAME        REAS     START_TIME       END_TIME MEMORY CPU NET_RX NET_TX
   9 host01          none 01/01 14:03:27 05/15 16:38:05  1024K   2   1.5G    23G
```

The record shows the complete history record, and total network consumption. It will not reflect the consumption made only during the month of February.

Other important thing to pay attention to is that active history records, those with END_TIME **'-'**, refresh their monitoring information each time the VM is monitored. Once the VM is shut down, migrated or stopped, the END_TIME is set and the monitoring information stored is frozen. The final values reflect the total for accumulative attributes, like NET_RX/TX.

Sample Output

Obtaining all the available accounting information:

```
$ oneacct
# User 0 oneadmin

 VID HOSTNAME        REAS     START_TIME       END_TIME MEMORY CPU NET_RX NET_TX
   0 host02          user 06/04 14:55:49 06/04 15:05:02  1024M   1     0K     0K

# User 2 oneuser1

 VID HOSTNAME        REAS     START_TIME       END_TIME MEMORY CPU NET_RX NET_TX
   1 host01          stop 06/04 14:55:49 06/04 14:56:28  1024M   1     0K     0K
   1 host01          user 06/04 14:56:49 06/04 14:58:49  1024M   1     0K   0.6K
   1 host02          none 06/04 14:58:49              -  1024M   1     0K   0.1K
   2 host02          erro 06/04 14:57:19 06/04 15:03:27     4G   2     0K     0K
   3 host01          none 06/04 15:04:47              -     4G   2     0K   0.1K
```

The columns are:

Column	Meaning
VID	Virtual Machine ID
HOSTNAME	Host name
REASON	**VM state change reason: none**: Normal termination **erro**: The VM ended in error **stop**: Stop/resume request **user**: Migration request **canc**: Cancel request
START_TIME	Start time
END_TIME	End time
MEMORY	Assigned memory. This is the requested memory, not the monitored memory consumption
CPU	Number of CPUs. This is the requested number of Host CPU share, not the monitored cpu usage
NETRX	Data received from the network
NETTX	Data sent to the network

Obtaining the accounting information for a given user

```
$ oneacct -u 2 --split
# User 2 oneuser1

 VID HOSTNAME        REAS     START_TIME       END_TIME MEMORY CPU NET_RX NET_TX
   1 host01          stop 06/04 14:55:49 06/04 14:56:28  1024M   1     0K     0K
   1 host01          user 06/04 14:56:49 06/04 14:58:49  1024M   1     0K   0.6K
   1 host02          none 06/04 14:58:49              -  1024M   1     0K   0.1K

 VID HOSTNAME        REAS     START_TIME       END_TIME MEMORY CPU NET_RX NET_TX
   2 host02          erro 06/04 14:57:19 06/04 15:03:27     4G   2     0K     0K

 VID HOSTNAME        REAS     START_TIME       END_TIME MEMORY CPU NET_RX NET_TX
   3 host01          none 06/04 15:04:47              -     4G   2     0K   0.1K
```

Output Reference

If you execute oneacct with the `-x` option, you will get an XML output defined by the following xsd:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
  targetNamespace="http://opennebula.org/XMLSchema" xmlns="http://opennebula.org/XMLSchema">

  <xs:element name="HISTORY_RECORDS">
    <xs:complexType>
      <xs:sequence maxOccurs="1" minOccurs="1">
        <xs:element ref="HISTORY" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="HISTORY">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="OID" type="xs:integer"/>
        <xs:element name="SEQ" type="xs:integer"/>
        <xs:element name="HOSTNAME" type="xs:string"/>
        <xs:element name="HID" type="xs:integer"/>
```

```
<xs:element name="STIME" type="xs:integer"/>
<xs:element name="ETIME" type="xs:integer"/>
<xs:element name="VMMMAD" type="xs:string"/>
<xs:element name="VNMMAD" type="xs:string"/>
<xs:element name="TMMAD" type="xs:string"/>
<xs:element name="DS_ID" type="xs:integer"/>
<xs:element name="PSTIME" type="xs:integer"/>
<xs:element name="PETIME" type="xs:integer"/>
<xs:element name="RSTIME" type="xs:integer"/>
<xs:element name="RETIME" type="xs:integer"/>
<xs:element name="ESTIME" type="xs:integer"/>
<xs:element name="EETIME" type="xs:integer"/>

<!-- REASON values:
  NONE        = 0  Normal termination
  ERROR       = 1  The VM ended in error
  STOP_RESUME = 2  Stop/resume request
  USER        = 3  Migration request
  CANCEL      = 4  Cancel request
-->
<xs:element name="REASON" type="xs:integer"/>

<xs:element name="VM">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="ID" type="xs:integer"/>
      <xs:element name="UID" type="xs:integer"/>
      <xs:element name="GID" type="xs:integer"/>
      <xs:element name="UNAME" type="xs:string"/>
      <xs:element name="GNAME" type="xs:string"/>
      <xs:element name="NAME" type="xs:string"/>
      <xs:element name="PERMISSIONS" minOccurs="0" maxOccurs="1">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="OWNER_U" type="xs:integer"/>
            <xs:element name="OWNER_M" type="xs:integer"/>
            <xs:element name="OWNER_A" type="xs:integer"/>
            <xs:element name="GROUP_U" type="xs:integer"/>
            <xs:element name="GROUP_M" type="xs:integer"/>
            <xs:element name="GROUP_A" type="xs:integer"/>
            <xs:element name="OTHER_U" type="xs:integer"/>
            <xs:element name="OTHER_M" type="xs:integer"/>
            <xs:element name="OTHER_A" type="xs:integer"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="LAST_POLL" type="xs:integer"/>

      <!-- STATE values,
      see http://opennebula.org/documentation:documentation:api#actions_for_virtual_machine_r

        INIT      = 0
        PENDING   = 1
        HOLD      = 2
        ACTIVE    = 3 In this state, the Life Cycle Manager state is relevant
        STOPPED   = 4
        SUSPENDED = 5
        DONE      = 6
```

```
              FAILED    = 7
              POWEROFF  = 8
            -->
            <xs:element name="STATE" type="xs:integer"/>

            <!-- LCM_STATE values, this sub-state is relevant only when STATE is
                 ACTIVE (4)

              LCM_INIT          = 0
              PROLOG            = 1
              BOOT              = 2
              RUNNING           = 3
              MIGRATE           = 4
              SAVE_STOP         = 5
              SAVE_SUSPEND      = 6
              SAVE_MIGRATE      = 7
              PROLOG_MIGRATE    = 8
              PROLOG_RESUME     = 9
              EPILOG_STOP       = 10
              EPILOG            = 11
              SHUTDOWN          = 12
              CANCEL            = 13
              FAILURE           = 14
              CLEANUP           = 15
              UNKNOWN           = 16
              HOTPLUG           = 17
              SHUTDOWN_POWEROFF = 18
              BOOT_UNKNOWN      = 19
              BOOT_POWEROFF     = 20
              BOOT_SUSPENDED    = 21
              BOOT_STOPPED      = 22
            -->
            <xs:element name="LCM_STATE" type="xs:integer"/>
            <xs:element name="RESCHED" type="xs:integer"/>
            <xs:element name="STIME" type="xs:integer"/>
            <xs:element name="ETIME" type="xs:integer"/>
            <xs:element name="DEPLOY_ID" type="xs:string"/>

            <!-- MEMORY consumption in kilobytes -->
            <xs:element name="MEMORY" type="xs:integer"/>

            <!-- Percentage of 1 CPU consumed (two fully consumed cpu is 200) -->
            <xs:element name="CPU" type="xs:integer"/>

            <!-- NET_TX: Sent bytes to the network -->
            <xs:element name="NET_TX" type="xs:integer"/>

            <!-- NET_RX: Received bytes from the network -->
            <xs:element name="NET_RX" type="xs:integer"/>
            <xs:element name="TEMPLATE" type="xs:anyType"/>
            <xs:element name="HISTORY_RECORDS">
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
</xs:schema>
```

6.4.3 Tuning & Extending

There are two kinds of monitoring values:

- Instantaneous values: For example, `VM/CPU` or `VM/MEMORY` show the memory consumption last reported by the monitoring probes.
- Accumulative values: For example, `VM/NET_TX` and `VM/NET_TX` show the total network consumption since the history record started.

Developers interacting with OpenNebula using the Ruby bindings can use the VirtualMachinePool.accounting method to retrieve accounting information filtering and ordering by multiple parameters.

6.5 Managing ACL Rules

The ACL authorization system enables fine-tuning of the allowed operations for any user, or group of users. Each operation generates an authorization request that is checked against the registered set of ACL rules. The core then can grant permission, or reject the request.

This allows administrators to tailor the user roles according to their infrastructure needs. For instance, using ACL rules you could create a group of users that can see and use existing virtual resources, but not create any new ones. Or grant permissions to a specific user to manage Virtual Networks for some of the existing groups, but not to perform any other operation in your cloud. Some examples are provided at the end of this guide.

6.5.1 Understanding ACL Rules

Lets start with an example:

```
#5 IMAGE+NET/@103 INFO+MANAGE+DELETE
```

This rule grants the user with ID 5 the right to perform INFO, MANAGE and DELETE operations over all Images and VNets in the group with id 103.

The rule is split in three components, separated by a space:

- **User** component is composed only by an **ID definition**.
- **Resources** is composed by a list of **'+'** separated resource types, **'/'** and an **ID definition**.
- **Rights** is a list of Operations separated by the **'+'** character.

The **ID definition** for User in a rule is written as:

- `#<id>` : for individual IDs
- `@<id>` : for a group ID
- `*` : for All

The **ID definition** for a Resource has the same syntax as the ones for Users, but adding:

- `%<id>` : for cluster IDs

Some more examples:

This rule allows all users in group 105 to create new virtual resources:

```
@105 VM+NET+IMAGE+TEMPLATE/* CREATE
```

The next one allows all users in the group 106 to use the Virtual Network 47. That means that they can instantiate VM templates that use this network.

```
@106 NET/#47 USE
```

Note: Note the difference between `* NET/#47 USE"` **vs** `* NET/@47 USE`

All Users can use NETWORK with ID 47 **vs** All Users can use NETWORKS belonging to the Group whose ID is 47

The following one allows users in group 106 to deploy VMs in Hosts assigned to the cluster 100

```
@106 HOST/%100 MANAGE
```

6.5.2 Managing ACL Rules via Console

The ACL rules are managed using the *oneacl command*. The 'oneacl list' output looks like this:

```
$ oneacl list
   ID     USER RES_VHNIUTGDCO   RID OPE_UMAC
    0       @1     V-NI-T----     *     ---c
    1       @1     -H--------     *     -m--
    2       #5     --NI-T----  @104     u---
    3     @106     ---I------   #31     u---
```

The four rules shown correspond to the following ones:

```
@1      VM+NET+IMAGE+TEMPLATE/* CREATE
@1      HOST/*                  MANAGE
#5      NET+IMAGE+TEMPLATE/@104 USE
@106    IMAGE/#31               USE
```

The first two were created on bootstrap by OpenNebula, and the last two were created using oneacl:

```
$ oneacl create "#5 NET+IMAGE+TEMPLATE/@104 USE"
ID: 2

$ oneacl create "@106 IMAGE/#31 USE"
ID: 3
```

The **ID** column identifies each rule's ID. This ID is needed to delete rules, using **'oneacl delete <id>'**.

Next column is **USER**, which can be an individual user (#) or group (@) id; or all (*) users.

The **Resources** column lists the existing Resource types initials. Each rule fills the initials of the resource types it applies to.

- `V : VM`
- `H : HOST`
- `N : NET`
- `I : IMAGE`
- `U : USER`
- `T : TEMPLATE`
- `G : GROUP`

- `D : DATASTORE`
- `C : CLUSTER`
- `O : DOCUMENT`

RID stands for Resource ID, it can be an individual object (#), group (@) or cluster (%) id; or all (*) objects.

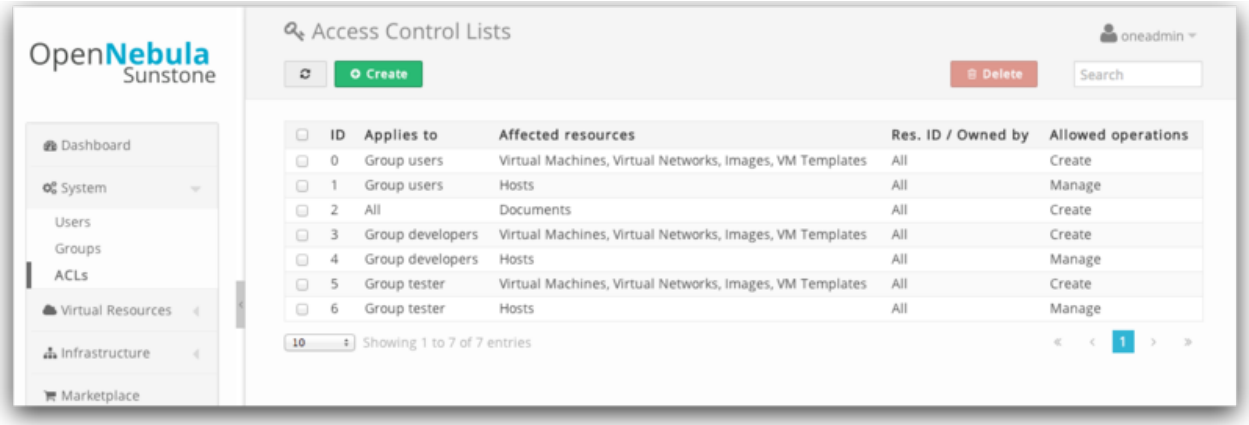
The last **Operations** column lists the allowed operations initials.

- `U : USE`
- `M : MANAGE`
- `A : ADMIN`
- `C : CREATE`

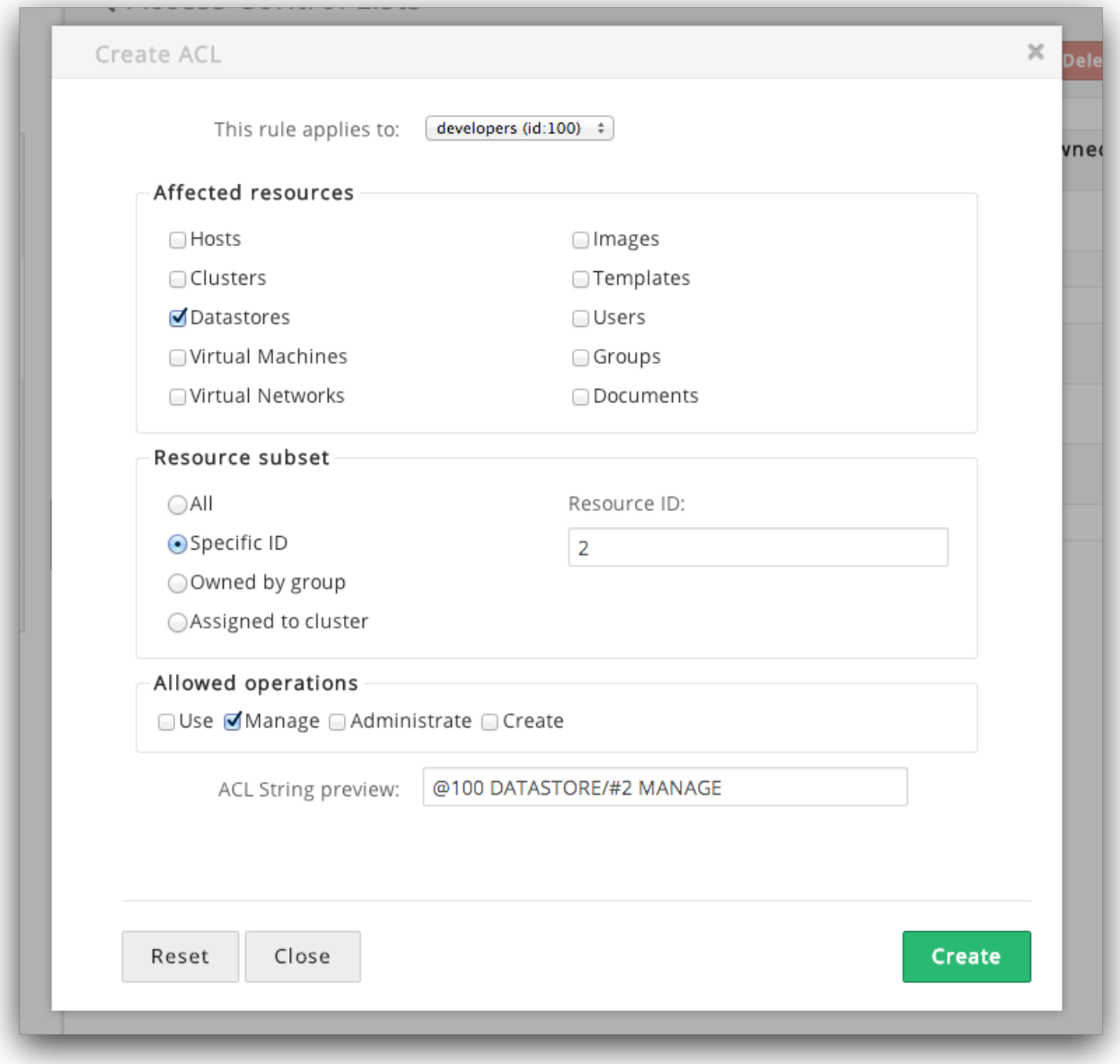
6.5.3 Managing ACLs via Sunstone

Sunstone ACL plugin offers a very intuitive and easy way of managing ACLs.

Select ACLs in the left-side menu to access a view of the current ACLs defined in OpenNebula:



This view is designed to easily undestand what the purpose of each ACL is. You can create new ACLs by clicking on the `New` button at the top. A dialog will pop up:



In the creation dialog you can easily define the resouces affected by the rule and the permissions that are granted upon them.

6.5.4 How Permission is Granted or Denied

Warning: Visit the *XML-RPC API reference documentation* for a complete list of the permissions needed by each OpenNebula command.

For the internal Authorization in OpenNebula, there is an implicit rule:

- The oneadmin user, or users in the oneadmin group are authorized to perform any operation.

If the resource is one of type `VM`, `NET`, `IMAGE` or `TEMPLATE`, the object's permissions are checked. For instance, this is an example of the oneimage show output:

```
$ oneimage show 2
IMAGE 2 INFORMATION
ID             : 2
[...]

PERMISSIONS
OWNER          : um-
GROUP          : u--
OTHER          : ---
```

The output above shows that the owner of the image has `USE` and `MANAGE` rights.

If none of the above conditions are true, then the set of ACL rules is iterated until one of the rules allows the operation.

An important concept about the ACL set is that each rule adds new permissions, and they can't restrict existing ones: if any rule grants permission, the operation is allowed.

This is important because you have to be aware of the rules that apply to a user and his group. Consider the following example: if a user **#7** is in the group **@108**, with the following existing rule:

```
@108 IMAGE/#45 USE+MANAGE
```

Then the following rule won't have any effect:

```
#7 IMAGE/#45 USE
```

6.5.5 Use Case

Let's say you have a work group where the users should be able to deploy VM instances of a predefined set of VM Templates. You also need two users that will administer those resources.

The first thing to do is create a new group, and check the automatically created ACL rules:

```
$ onegroup create restricted
ID: 100
ACL_ID: 2
ACL_ID: 3

$ oneacl list
   ID     USER RES_VHNIUTGDCO   RID OPE_UMAC
    0       @1     V-NI-T----     *      ---c
    1       @1     -H--------     *      -m--
    2     @100     V-NI-T----     *      ---c
    3     @100     -H--------     *      -m--
```

The rule #2 allows all users in this group to create new resources. We want users to be able to see only existing VM Templates and VM instances in their group:

```
$ oneacl delete 2

$ oneacl list
   ID     USER RES_VHNIUTGDCO   RID OPE_UMAC
    0       @1     V-NI-T----     *      ---c
    1       @1     -H--------     *      -m--
    3     @100     -H--------     *      -m--
```

And now we can authorize users #1 and #2 to perform any operation on the group resources:

```
$ oneacl create "#1 VM+NET+IMAGE+TEMPLATE/* USE+MANAGE+CREATE"
ID: 4

$ oneacl create "#2 VM+NET+IMAGE+TEMPLATE/* USE+MANAGE+CREATE"
ID: 5

$ oneacl list
   ID     USER RES_VHNIUTGDCO   RID OPE_UMAC
    0       @1     V-NI-T----     *     ---c
    1       @1     -H--------     *     -m--
    3     @100     -H--------     *     -m--
    4       #1     V-NI-T----     *     um-c
    5       #2     V-NI-T----     *     um-c
```

With this configuration, users #1 and #2 will manage all the resources in the group 'restricted'. Because of the implicit rules, the rest of the users can use any VM Template that they create and share using the GROUP_U bit in the chmod operation.

For example, users #1 or #2 can allow other users in their group `USE` (list, show and instantiate) the Template 8 with the chmod command:

```
$ onetemplate show 8
TEMPLATE 8 INFORMATION
[...]

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

TEMPLATE CONTENTS

$ onetemplate chmod 8 640
$ onetemplate show 8
TEMPLATE 8 INFORMATION
[...]

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

TEMPLATE CONTENTS
```

In practice, this means that regular users in the *restricted* group will be able to list and use only the resources prepared for them by the users #1 and #2.

6.6 Managing Quotas

This guide will show you how to set the usage quotas for users and groups.

6.6.1 Overview

The quota system tracks user and group usage of system resources, and allows the system administrator to set limits on the usage of these resources. Quota limits can be set for:

- **users**, to individually limit the usage made by a given user.
- **groups**, to limit the overall usage made by all the users in a given group. This can be of special interest for the OpenNebula Zones and Virtual Data Center (VDC) components.

6.6.2 Which Resource can be limited?

The quota system allows you to track and limit usage on:

- **Datastores**, to control the amount of storage capacity allocated to each user/group for each datastore.
- **Compute**, to limit the overall memory, cpu or VM instances.
- **Network**, to limit the number of IPs a user/group can get from a given network. This is specially interesting for networks with public IPs, which usually are a limited resource.
- **Images**, you can limit the how many VM instances from a given user/group are using a given image. You can take advantage of this quota when the image contains consumable resources (e.g. software licenses).

6.6.3 Defining User/Group Quotas

Usage quotas are set in a traditional template syntax (either plain text or XML). The following table explains the attributes needed to set each quota:

Datastore Quotas. Attribute name: DATASTORE

DATASTORE Attribute	Description
ID	ID of the Datastore to set the quota for
SIZE	Maximum size in MB that can be used in the datastore
IMAGE	Maximum number of images that can be created in the datastore

Compute Quotas. Attribute name: VM

VM Attribute	Description
VMS	Maximum number of VMs that can be created
MEMORY	Maximum memory in MB that can be requested by user/group VMs
CPU	Maximum CPU capacity that can be requested by user/group VMs
VOLATILE_SIZE	Maximum volatile disks size (in MB) that can be requested by user/group VMs

Network Quotas. Attribute name: NETWORK

NETWORK Attribute	Description
ID	ID of the Network to set the quota for
LEASES	Maximum IPs that can be leased from the Network

Image Quotas. Attribute name: IMAGE

IMAGE Attribute	Description
ID	ID of the Image to set the quota for
RVMS	Maximum VMs that can used this image at the same time

For each quota, there are two special limits:

- **0** means **unlimited**

- **-1** means that the **default quota** will be used

Warning: Each quota has an usage counter associated named <QUOTA_NAME>_USED. For example MEMORY_USED means the total memory used by user/group VMs, and its associated quota is MEMORY.

The following template shows a quota example for a user in plain text. It limits the overall usage in Datastore 0 to 20Gb (for an unlimited number of images); the number of VMs that can be created to 4 with a maximum memory to 2G and 5 CPUs; the number of leases from network 1 to 4; and image 1 can only be used by 3 VMs at the same time:

```
DATASTORE=[
  ID="1",
  IMAGES="0",
  SIZE="20480"
]

VM=[
  CPU="5",
  MEMORY="2048",
  VMS="4",
  VOLATILE_SIZE="-1"
]

NETWORK=[
  ID="1",
  LEASES="4"
]

IMAGE=[
  ID="1",
  RVMS="3"
]

IMAGE=[
  ID="2",
  RVMS="0"
]
```

Warning: Note that whenever a network, image, datastore or VM is used the corresponding quota counters are created for the user with an unlimited value. This allows to track the usage of each user/group even when quotas are not used.

6.6.4 Setting User/Group Quotas

User/group quotas can be easily set up either trough the command line interface or sunstone. Note that you need MANAGE permissions to set a quota of user, and ADMIN permissions to set the quota of a group. In this way, by default, only oneadmin can set quotas for a group, but if you define a group manager (as in a VDC) she can set specific usage quotas for the users on her group (so distributing resources as required). You can always change this behaviour setting the appropriate ACL rules.

To set the quota for a user, e.g. userA, just type:

```
$ oneuser quota userA
```

This will open an editor session to edit a quota template (with some tips about the syntax).

Warning: Usage metrics are included for information purposes (e.g. CPU_USED, MEMORY_USED, LEASES_USED...) you cannot modify them

Warning: You can add as many resource quotas as needed even if they have not been automatically initialized.

Similarly, you can set the quotas for group A with:

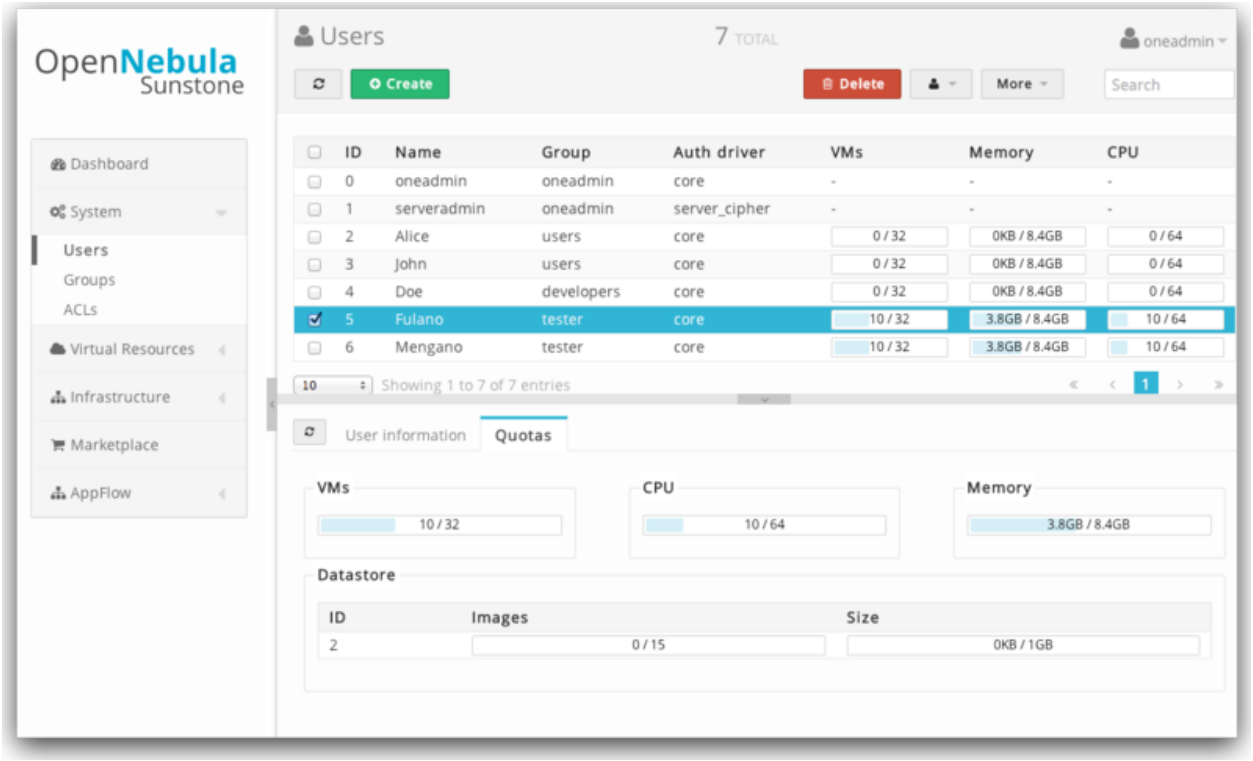
```
$ onegroup quota groupA
```

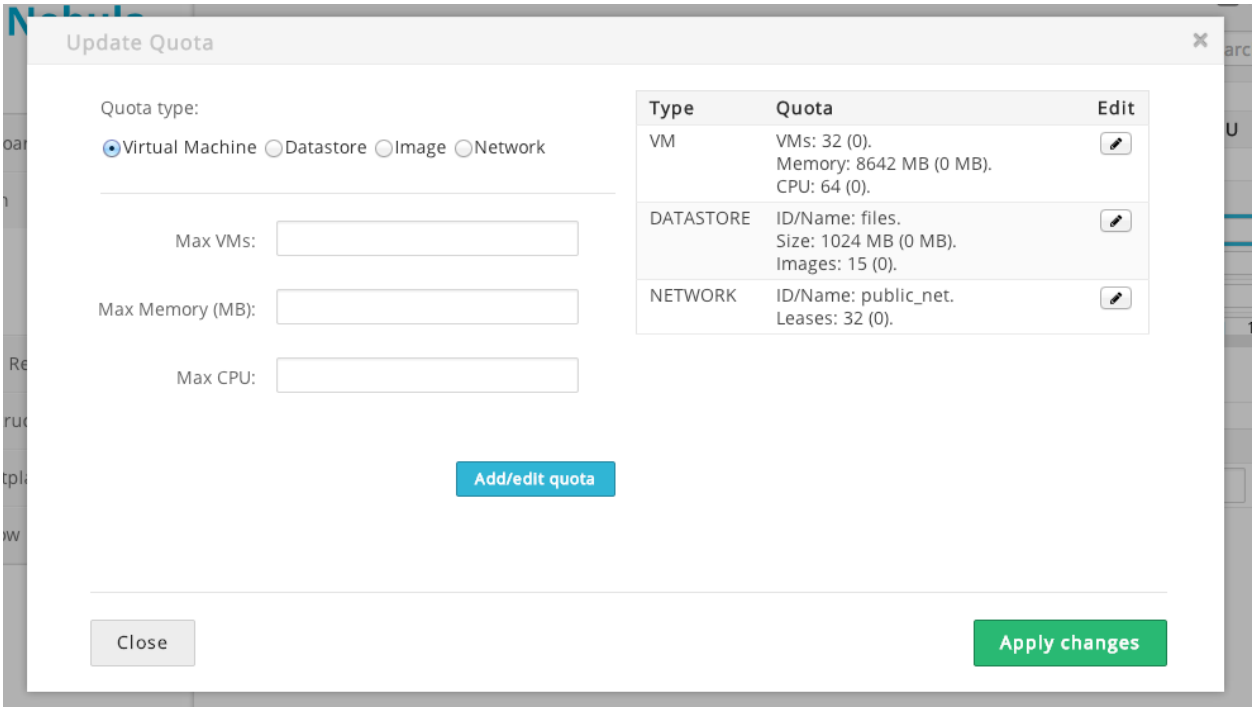
There is a `batchquota` command that allows you to set the same quotas for several users or groups:

```
$ oneuser batchquota userA,userB,35
```

```
$ onegroup batchquota 100..104
```

You can also set the user/group quotas in Sunstone through the user/group tab.





6.6.5 Setting Default Quotas

There are two default quota limit templates, one for users and another for groups. This template applies to all users/groups, unless they have an individual limit set.

Use the `oneuser/onegroup defaultquota` command.

```
$ oneuser defaultquota
```

6.6.6 Checking User/Group Quotas

Quota limits and usage for each user/group is included as part of its standard information, so it can be easily check with the usual commands. Check the following examples:

```
$ oneuser show uA
USER 2 INFORMATION
ID              : 2
NAME            : uA
GROUP           : gA
PASSWORD        : a9993e364706816aba3e25717850c26c9cd0d89d
AUTH_DRIVER     : core
ENABLED         : Yes

USER TEMPLATE


RESOURCE USAGE & QUOTAS

DATASTORE ID  IMAGES (used) IMAGES (limit)     SIZE (used)    SIZE (limit)
1                        1              0            1024               0
```

```
VMS           MEMORY (used) MEMORY (limit)     CPU (used)    CPU (limit)
0                       1024              0              1              0

NETWORK ID    LEASES (used) LEASES (limit)
1                          1              0

IMAGE ID        RVMS (used)   RVMS (limit)
1                          0              0
2                          0              0
```

And for the group:

```
$ onegroup show gA
GROUP 100 INFORMATION
ID              : 100
NAME            : gA

USERS
ID
2
3

RESOURCE USAGE & QUOTAS

DATASTORE ID  IMAGES (used) IMAGES (limit)    SIZE (used)   SIZE (limit)
1                          2              0           2048              0

VMS           MEMORY (used) MEMORY (limit)     CPU (used)    CPU (limit)
0                       2048              0              2              0

NETWORK ID    LEASES (used) LEASES (limit)
1                          1              0
2                          1              0

IMAGE ID        RVMS (used)   RVMS (limit)
1                          0              0
2                          0              0
5                          1              0
6                          1              0
```

This information is also available through Sunstone as part of the user/group information.

CHAPTER

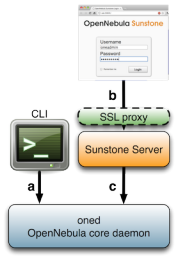
SEVEN

AUTHENTICATION

7.1 External Auth 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.

7.1.1 Authentication



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

a) CLI Authentication

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

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

b) Sunstone Authentication

By default, users with the "core" authentication driver (user/password) can login in Sunstone. You can enable users with the "x authentication driver to login using an external **SSL proxy** (e.g. Apache).

Proceed to the Sunstone documentation to configure the x509 access:

- *Sunstone Authentication Methods*

c) Servers Authentication

OpenNebula ships with three servers: *Sunstone*, *EC2* and *OCCI*. 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 frontend.

- *Cloud Servers Authentication*

7.2 SSH Auth

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.

7.2.1 Requirements

You don't need to install any additional software.

7.2.2 Considerations & Limitations

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

7.2.3 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.

7.2.4 Usage

Create New Users

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

```
newuser@frontend $ 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:

```
newuser@frontend $ oneuser key
Enter PEM pass phrase:
MIIBCAKCAQEApUO+JISjSf02rFVtDr1yar/34EoUoVETx0n+RqWNav+5wi+gHiPp3e03AfEkXzjDYi8F
voS4a4456f1OUQlQddfyPECn59OeX8Zu4DH3gp1VUuDeeE8WJWyAzdK5hg6F+RdyP1pT26mnyunZB8Xd
bll8seoIAQiOS6tlVfA8FrtwLGmdEETfttS9ukyGxw5vdTplse/fcam+r9AXBR06zjc77x+DbRFbXcgI
1XIdpVrjCFL0fdN53L0aU7kTE9VNEXRxK8sPv1Nfx+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:

```
oneadmin@frontend $ 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:

```
oneadmin@frontend $ 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:

```
oneadmin@frontend $ oneuser chauth <id|name> ssh

oneadmin@frontend $ 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, and export the new `ONE_AUTH` environment variable. The command requires the OpenNebula username, and the authentication method (`-ssh` in this case).

```
newuser@frontend $ oneuser login newuser --ssh
export ONE_AUTH=/home/newuser/.one/one_ssh

newuser@frontend $ export ONE_AUTH=/home/newuser/.one/one_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 1 hour. You can change that with the `-time` option.

7.3 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.

7.3.1 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.

7.3.2 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.

7.3.3 Configuration

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

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

```
oneadmin@frontend $ 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.

```
oneadmin@frontend $ 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

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

7.3.4 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:

```
oneadmin@frontend $ 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:

```
oneadmin@frontend $ 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:

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

- Using the user certificate subject DN:

```
oneadmin@frontend $ 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.

```
oneadmin@frontend $ 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, and export the new `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:
export ONE_AUTH=/home/user/.one/one_x509

newuser@frontend $ export ONE_AUTH=/home/user/.one/one_x509
```

The generated token has a default **expiration time** of 1 hour. You can change that with the `-time` option.

7.3.5 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"
]
```

7.3.6 Enabling x509 auth in Sunstone

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

```
:auth: x509
```

7.4 LDAP Authentication

The LDAP Authentication addon 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.

7.4.1 Prerequisites

> **Warning:** This Addon requires the **'net/ldap'** ruby library provided by the 'net-ldap' gem.

This Addon 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.

7.4.2 Considerations & Limitations

LDAP auth driver has a bug that does not let it connect to TLS LDAP instances. A patch is available in the bug issue to fix this. The fix will be applied in future releases.

7.4.3 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

    # 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'

# 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
: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

To enable ldap authentication the described parameters should be configured. OpenNebula must be also configured to enable external authentication. Uncomment these lines in /etc/one/oned.conf and add ldap and default (more on this later) as an enabled authentication method.

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

To be able to use this driver for users that are still not in the user database you must set it to the default driver. To do this go to the auth drivers directory and copy the directory ldap to default. In system-wide installations you can do this using this command:

```
$ cp -R /var/lib/one/remotes/auth/ldap /var/lib/one/remotes/auth/default
```

7.4.4 User Management

Using LDAP authentication module the administrator doesn't need to create users with oneuser command as this will be automatically done. The user should add its credentials to $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

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.

7.4.5 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 te 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.

7.4.6 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
```

CHAPTER

EIGHT

SUNSTONE GUI

8.1 OpenNebula Sunstone: The Cloud Operations Center

OpenNebula Sunstone is the OpenNebula Cloud Operations Center, a Graphical User Interface (GUI) intended for regular 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.

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

8.1.1 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 guide 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 session to a Virtual Machine. This is done by using a VNC websocket-based client (noVNC) on the client side and a VNC proxy translating and redirecting the connections on the server-side.

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.

8.1.2 Considerations & Limitations

OpenNebula Sunstone supports Firefox (> 3.5) and Chrome browsers. Internet Explorer, Opera and others are not supported and may not work well.

8.1.3 Configuration

sunstone-server.conf

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` for everyone. `127.0.0.1` by default.
:port	Port on which the server will listen. `9869` by default.
:sessions	Method of keeping user sessions. It can be `memory` or `memcache`. For server that spawn more than one process (like Passenger or Unicorn) `memcache` should be used
:memcache_host	Host where `memcached` server resides
:memcache_port	Port of `memcached` server
:memcache_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
:auth	Authentication driver for incoming requests. Possible values are `sunstone`, `opennebula` and `x509`. Check *authentication methods* for more info
:core_auth	Authentication driver to communicate with OpenNebula core. Possible values are `x509` or `cipher`. Check *cloud_auth* for more information
: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	`yes`, `no`, `only`. If enabled, the proxy will be set up with a certificate and a key to use secure websockets. If set to `only` 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)
:table_order	Default table order, resources get ordered by ID in `asc` or `desc` order.
:marketplace_username	Username credential to connect to the Marketplace.
:marketplace_password	Password to connect to the Marketplace.
:marketplace_url	Endpoint to connect to the Marketplace. If commented, a 503 `service unavailable` error will be returned to clients.
:oneflow_server	Endpoint to connect to the OneFlow server.
:routes	List of files containing custom routes to be loaded. Check *server plugins* for more info.

Warning: In order to access Sunstone from other place than `localhost` you need to set the server's public IP in the `:host` option. Otherwise it will not be reachable from the outside.

Todo

Running Sunstone Server separate host.

Starting Sunstone

To start Sunstone just issue the following command as oneadmin

```
$ sunstone-server start
```

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

To stop the Sunstone service:

```
$ sunstone-server stop
```

VNC Troubleshooting

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 latest 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.
- Make sure that you have not checked the `Secure websockets connection` in the Configuration dialog if your proxy has not been configured to support them. Connection will fail if so.
- 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 user list only when you have gone through the suggestion above and provide full sunstone logs, shown errors and any relevant information of your infraestructure (if there are Firewalls etc)

8.1.4 Tuning & Extending

For more information on how to customize and extend you Sunstone deployment use the following links:

- *Sunstone Views*, different roles different views.

- *Security & Authentication Methods*, improve security with x509 authentication and SSL
- *Advanced Deployments*, improving scalability and isolating the server

8.2 Sunstone Views

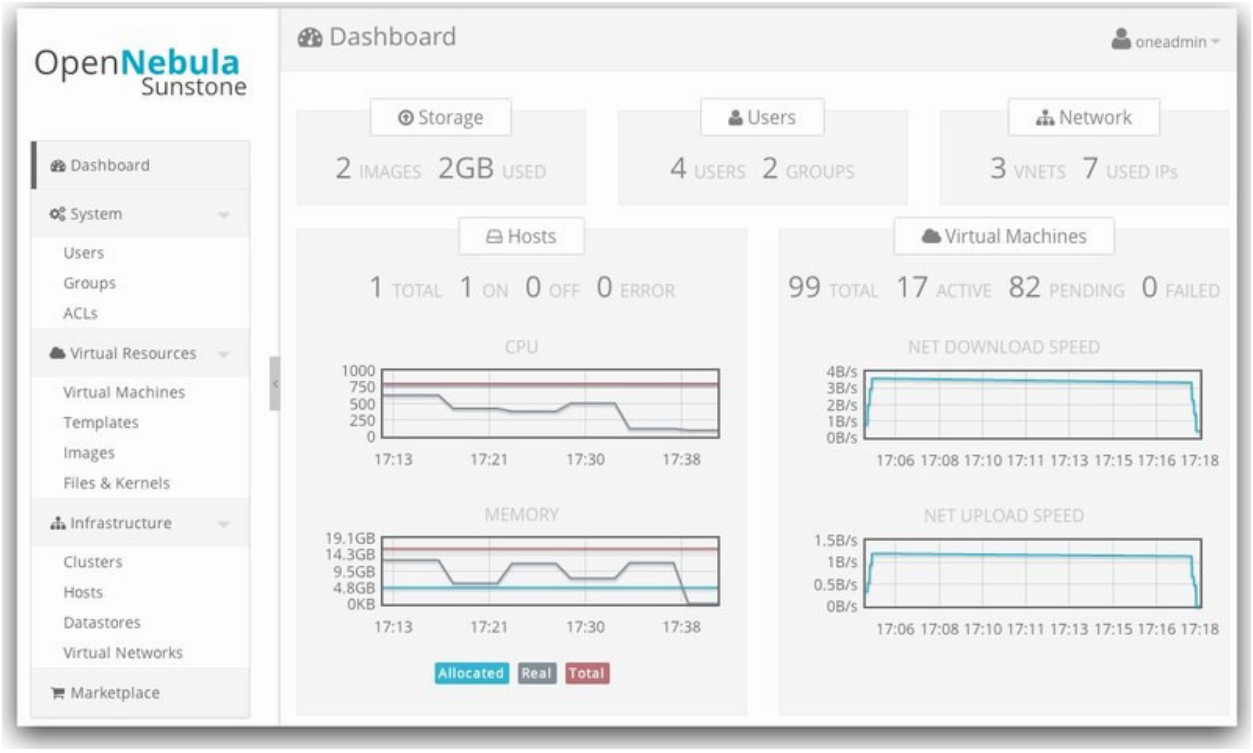
Using the new OpenNebula Sunstone Views you will be able to provide a simplified UI aimed at end-users of an OpenNebula cloud. The OpenNebula Sunstone Views are fully customizable, so you can easily enable or disable specific information tabs or action buttons. You can define multiple cloud views for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

8.2.1 Default Views

OpenNebula provides a default `admin`, `vdcadmin`, `user` and `cloud` view that implements four common 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 deafault `user` view.

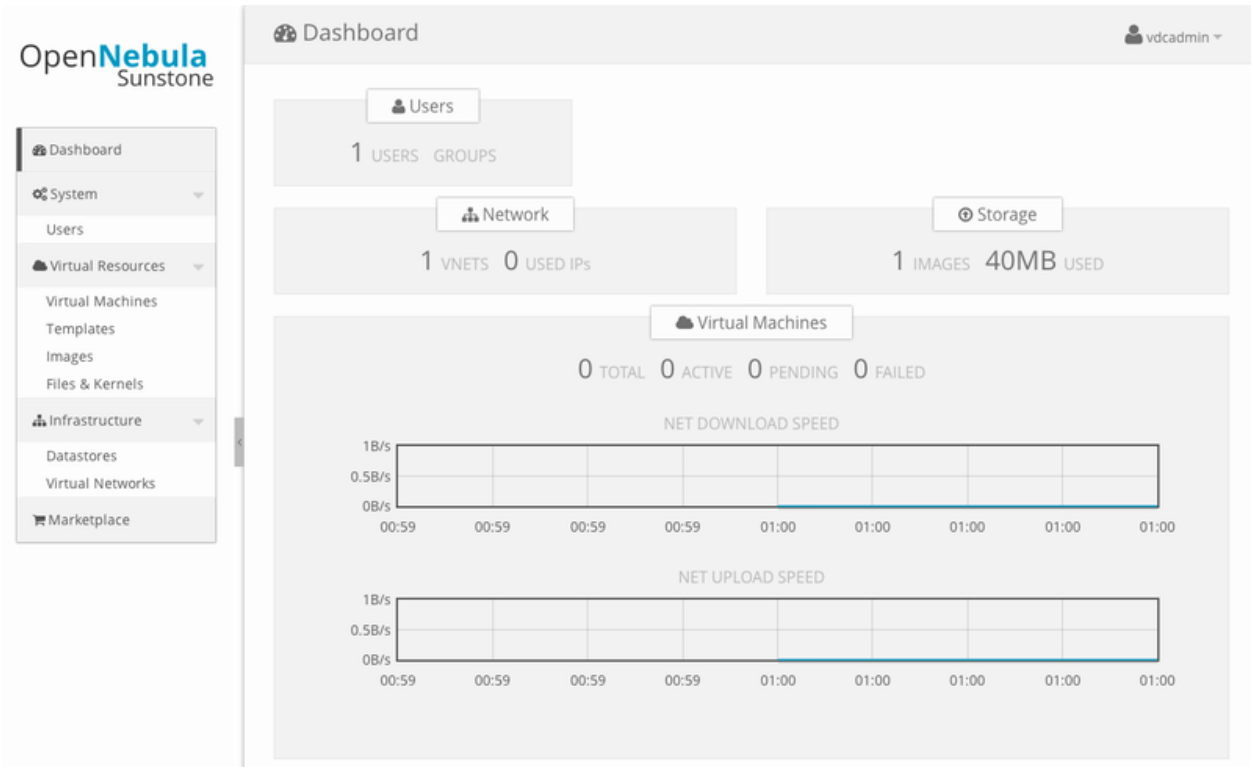
Admin View

This view provides full control of the cloud.



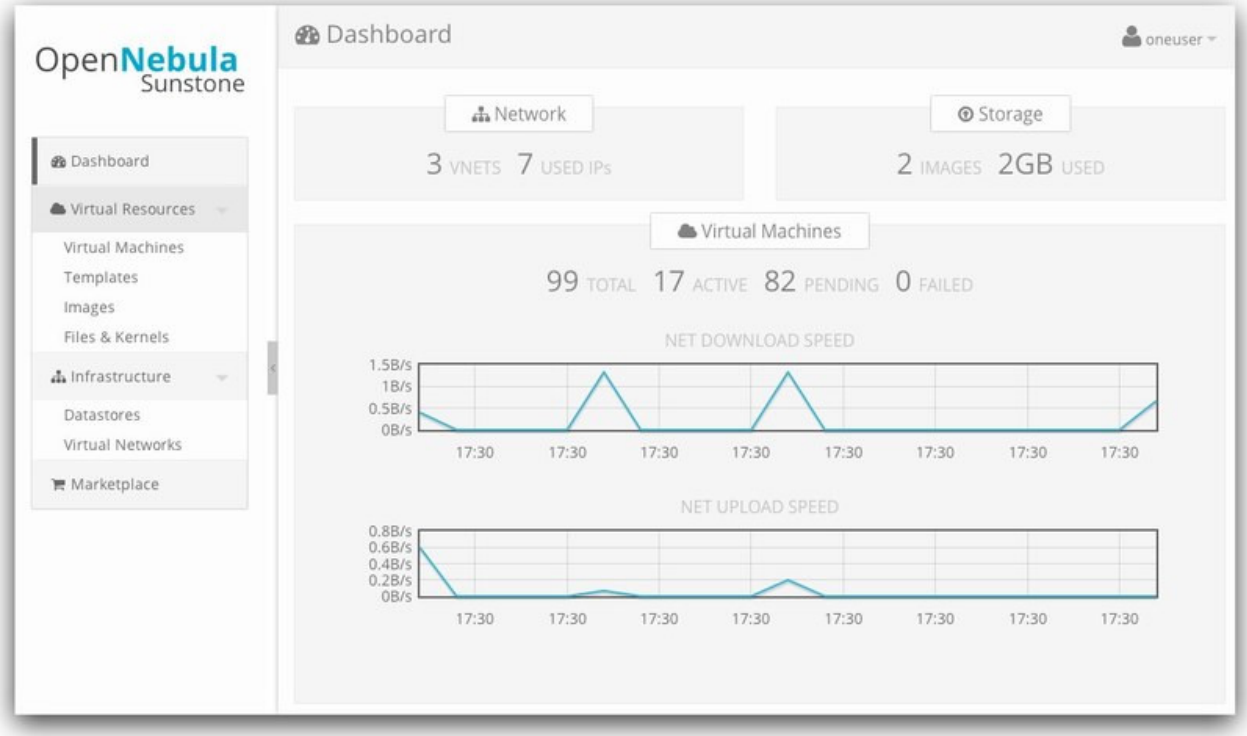
VDCAdmin View

This view provides control of all the resources belonging to a Virtual DataCenter (VDC), but with no access to resources outside that VDC. It is basically and Admin view restricted to the physical and virtual resources of the VDC, with the ability to create new users within the VDC.



User View

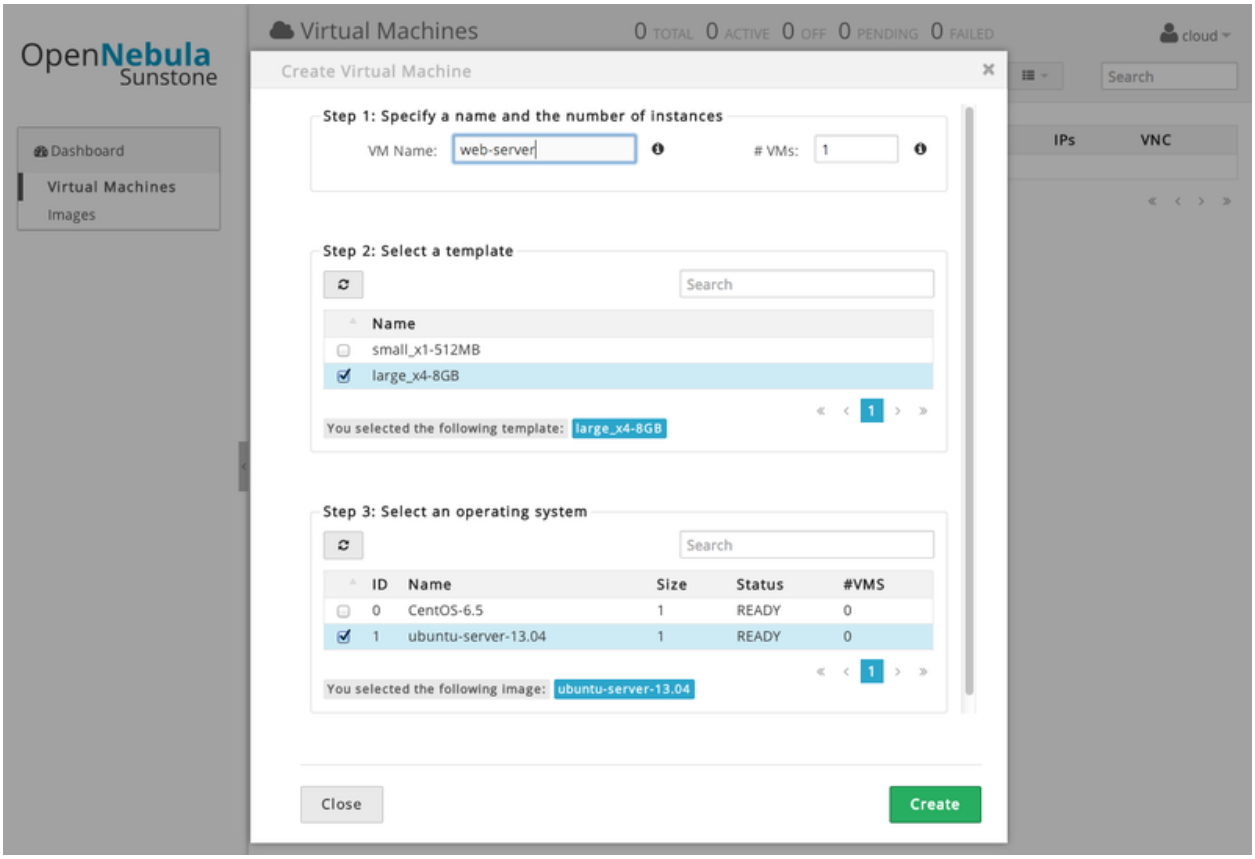
In this view users will not be able to manage nor retrieve the hosts and clusters of the cloud. They will be able to see Datastores and Virtual Networks in order to use them when creating a new Image or Virtual Machine, but they will not be able to create new ones. For more information about this view, please check the `/etc/one/sunstone-views/user.yaml` file.



Cloud View

This is a simplified view mainly intended for user that just require a portal where they can provision new virtual machines easily. They just have to select one of the available templates and the operating system that will run in this virtual machine. 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. Templates must define all the required parameters and just leave the DISK section empty, so the user can select any of the available images. New virtual machines will be created merging the information provided by the user (image, vm_name...) and the base template. Thereby, the user doesn't have to know any details of the infrastructure such as networking, storage. For more information on how to configure this scenario see *this guide*



8.2.2 Requirements

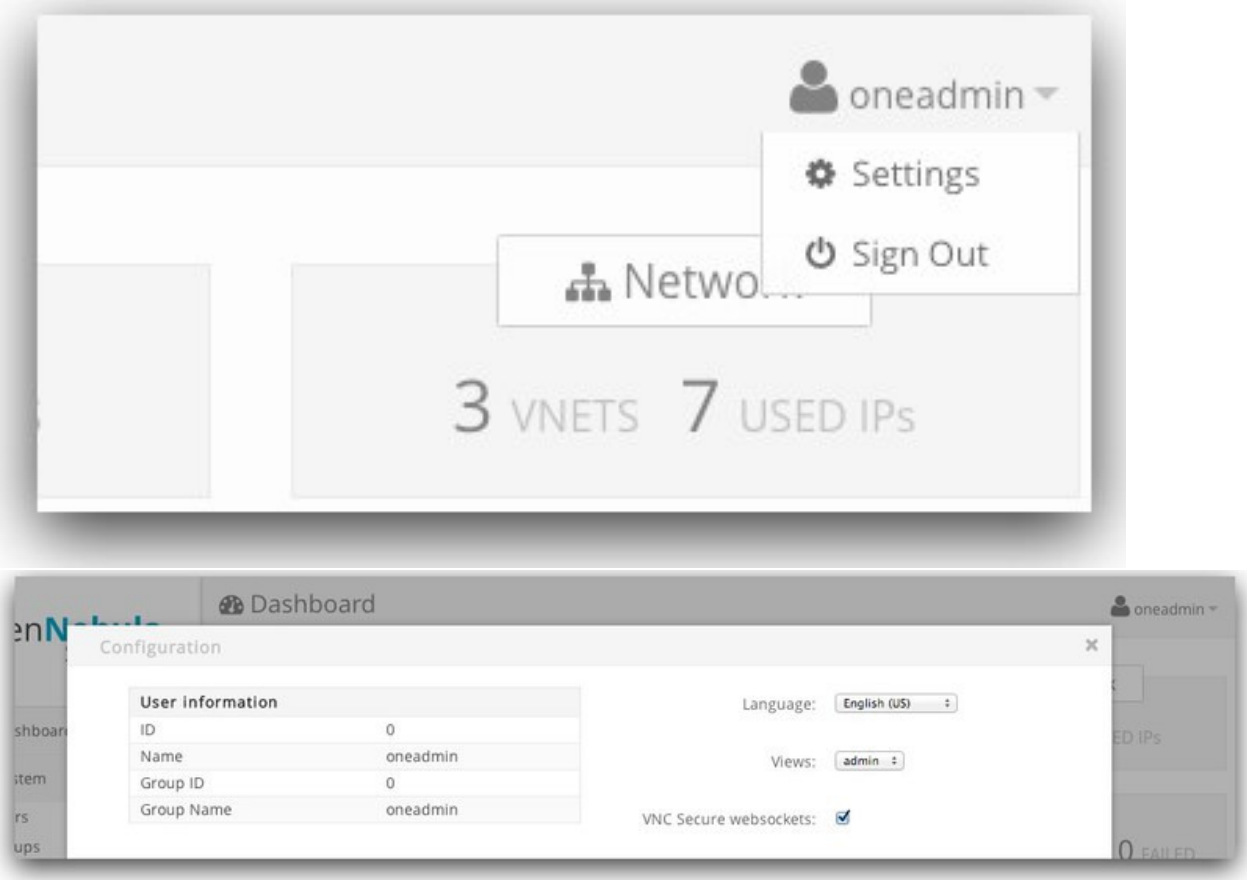
OpenNebula Sunstone Views does not require any additional service to run. You may want to review the Sunstone configuration to deploy advanced setups, to scale the access to the web interface or to use SSL security.

8.2.3 Usage

Sunstone users can configure several options from the configuration tab:

- 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.
- Views: change between the different available views for the given user/group
- 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.

This options are saved in the user template. If not defined, defaults from `sunstone-server.conf` are taken.



Changing your View

If more than one view are available for this user, she can easily change between them in the settings window, along with other settings (e.g. language).

Warning: By default users in the oneadmin group have access to all the views; users in the users group can only use the `users view`. If you want to expose the `cloud view` to a given group of users, you have to modify the `sunstone-views.yaml`. For more information check the configuring access to views section

Internationalization and Languages

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

- Transifex poject 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 user list. 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.

8.2.4 Advanced Configuration

There are three basic areas that can be tuned to adapt the default behavior to your provisioning needs:

- Define views, the set of UI components that will be enabled.
- Define the users and groups that may access to each view.
- Brand your OpenNebula Sunstone portal.

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 is the the filename without the yaml extension. The default views are defined by the user.yaml and admin.yaml files, as shown below:

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

The content of a view file specifies the tabs available in the view (note: tab is on 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, just set `clusters-tab` value to `false`:

```
enabled_tabs:
    dashboard-tab: true
    system-tab: true
    users-tab: true
    groups-tab: true
    acls-tab: true
    vresources-tab: true
    vms-tab: true
    templates-tab: true
    images-tab: true
    files-tab: true
    infra-tab: true
    clusters-tab: false
    hosts-tab: true
    datastores-tab: true
    vnets-tab: true
    marketplace-tab: true
    oneflow-dashboard: tru
    oneflow-services: true
    oneflow-templates: true
```

Each tab, can be tuned by selecting:

- The bottom 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
    table_columns:
        - 0         # Checkbox
        - 1         # ID
        - 2         # Owner
        - 3         # Group
        - 4         # Name
        - 5         # Cluster
        #- 6         # Basepath
        #- 7         # TM
        #- 8         # DS
        #- 9         # Type
    actions:
        Datastore.refresh: true
        Datastore.create_dialog: false
        Datastore.addtocluster: false
        Datastore.chown: false
        Datastore.chgrp: false
        Datastore.chmod: false
        Datastore.delete: false
```

Warning: The easiest way to create a custom view is to copy the `admin.yaml` file to the new view then harden it as needed.

Configuring Access to the Views

Once you have defined and customized the UI views for the different roles, you need to define which user groups or users may access to each view. This information is defined in the `/etc/one/sunstone-views.yaml`.

The 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.

For example the following 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 for a given user the first one is the default:

```
...
users:
    helen:
        - cloud
        - user
groups:
    cloud-users:
        - cloud
default:
    - user
```

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 (TODO deploy in a different machine link) 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
```

Branding the Sunstone Portal

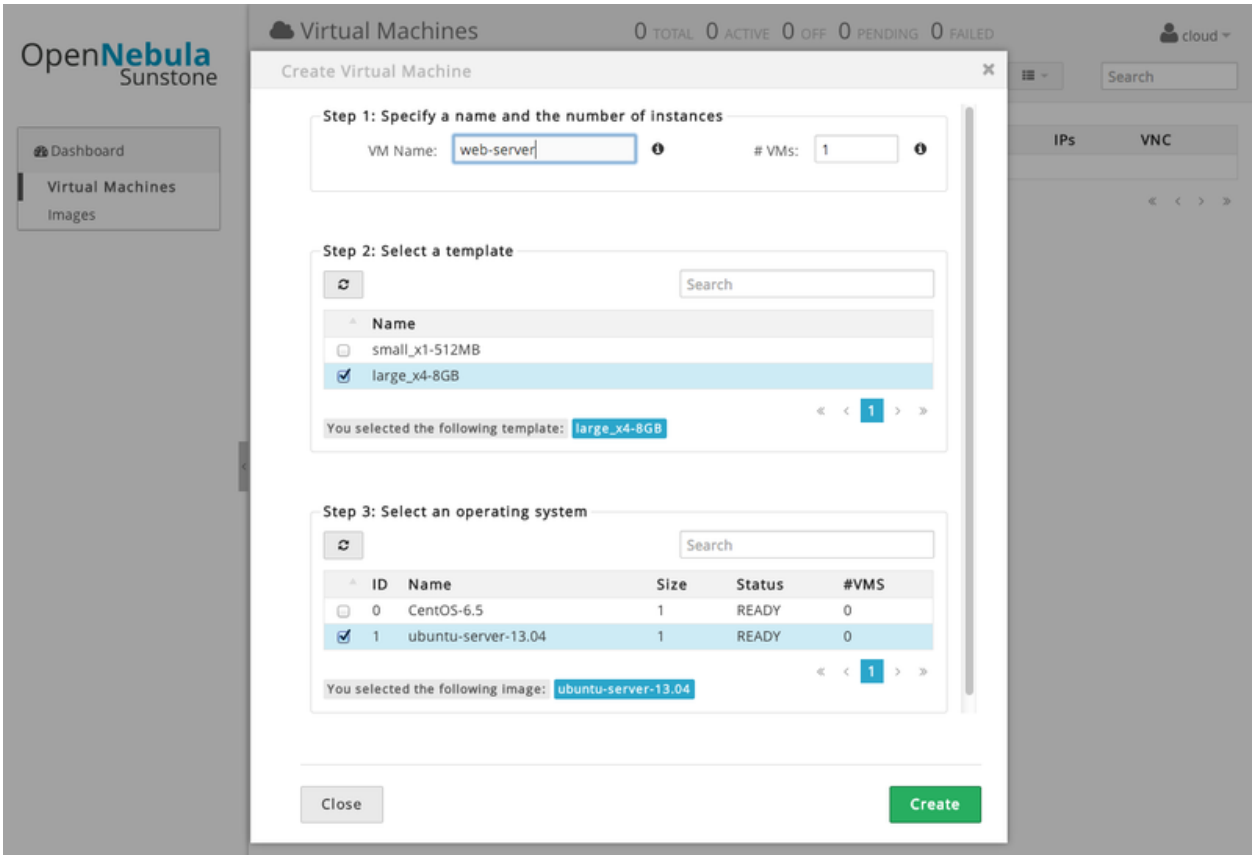
You can easily add you logos to the login and main screens by updating the `logo:` attribute as follows:

- 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 file.

8.3 Self-service Cloud View

This is a simplified view mainly intended for cloud consumers that just require a portal where they can provision new virtual machines easily. They just have to select one of the available templates and the operating system that will run in this virtual machine. For more information about the suntone views, please check the following *guide*.

In this scenario the cloud administrator must prepare a set of templates and images and make them available to the cloud users. Templates must define all the required parameters and just leave the DISK section empty, so the user can select any of the available images. New virtual machines will be created merging the information provided by the user (image, vm_name...) and the base template. Thereby, the user doesn't have to know any details of the infrastructure such as networking, storage...



8.3.1 How to Configure

These are the steps that an administrator should follow in order to prepare a self-service scenario for his users.

Create the Cloud Group

Create a new group for the users, to which you want to expose the cloud vie.

```
$ onegroup create cloud_consumers
```

Update the `/etc/one/sunstone-views.yaml` file adding the new group and the desired view (`cloud`).

```
groups:
    oneadmin:
        - admin
        - user
    cloud_consumers:
        - cloud
```

and restart sunstone-server after that.

Create new users in this group

```
$ oneuser crete new_user password
$ oneuser chgrp new_user cloud_consumer
```

You can modify the functionality that is exposed in this view, in the `/etc/one/sunstone-views/cloud.yaml` file.

Prepare a Set of Templates

Prepare a set of template that the cloud consumers will use to create new instances. These templates should define all the required parameters of the virtual machine that depends on you network, storage... but should not define the OS image of the virtual machine. The OS image will be selected by the user in the creation dialog along with the template.

Example:

```
$ cat small-x1-1GB.template
NAME   = small-x1-1GB
MEMORY = 1024
CPU    = 1

NIC = [ NETWORK = "Public" ]

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

$ cat large-x4-8GB.template
NAME   = large-x4-8GB
MEMORY = 8192
CPU    = 8

NIC = [ NETWORK = "Public" ]

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

$ onetemplate create small-x1-1GB.template
$ onetemplate create large-x4-8GB.template
```

If you want to make these template available to the users of the cloud_cosumers group, the easiest way is to move them to that group and enable the use permission for group:

```
ontemplate chgrp small-x1-1GB.template cloud_consumers
ontemplate chmod small-x1-1GB.template 640
```

You can also create the template using the Sunstone wizard

Prepare a Set of OS Images

Prepare a set of images that will be used by the cloud consumers in the templates that were created in the previous step.

```
$ oneimage create --datastore default --name Ubuntu-1204 --path /home/cloud/images/ubuntu-desktop \
  --description "Ubuntu 12.04 desktop for students."
$ oneimage create --datastore default --name CentOS-65 --path /home/cloud/images/ubuntu-desktop \
  --description "CentOS-65 desktop for developers."
```

If you want to make these available available to the users of the cloud_consumers group, the easiest way is to move them to that group and enable the use permission for group:

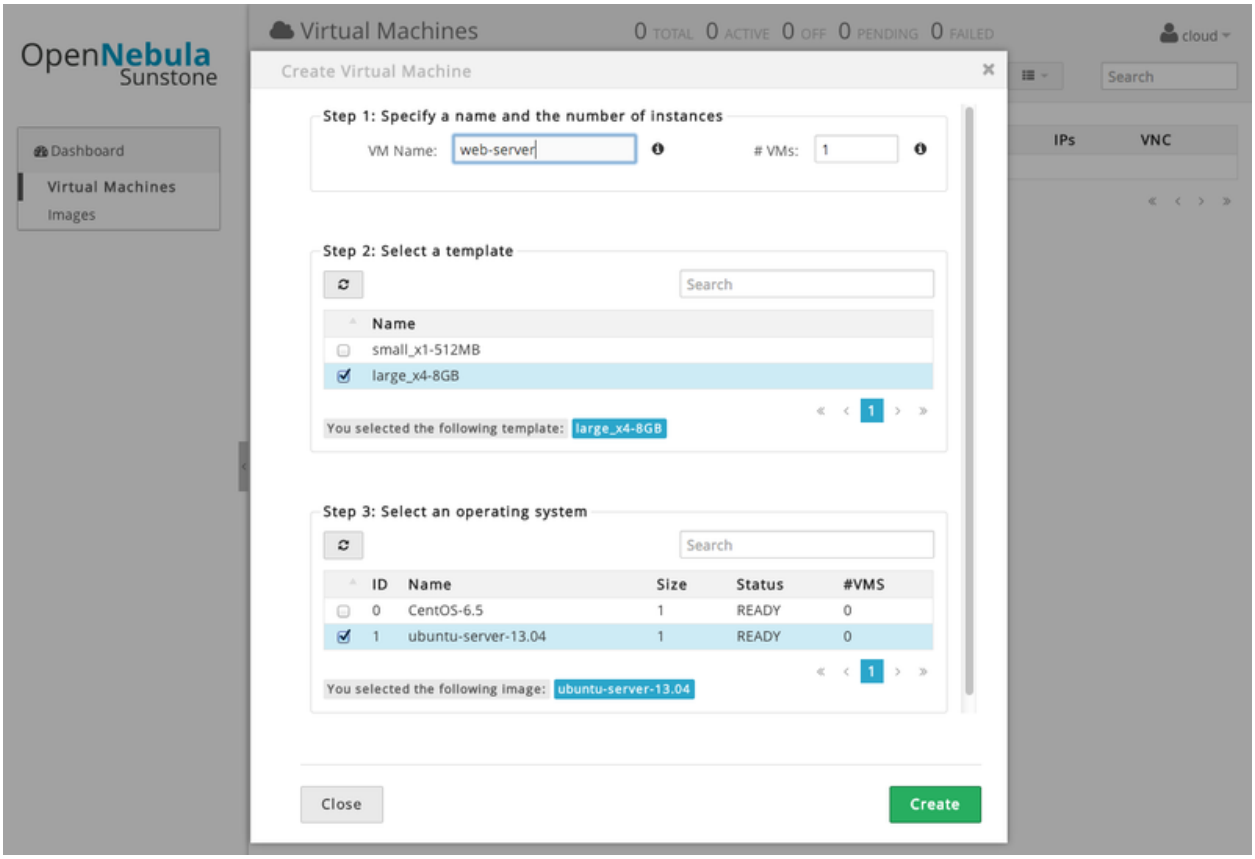
```
oneimage chgrp Ubuntu-1204 cloud_consumers
oneimage chmod CentOS-65 640
```

8.3.2 The Cloud Consumer View

End users that want to interact with Sunstone have to open a new browser and go to the url where the Sunstone server is deployed. They will find the login screen where the username and password correspond to the OpenNebula credentials.

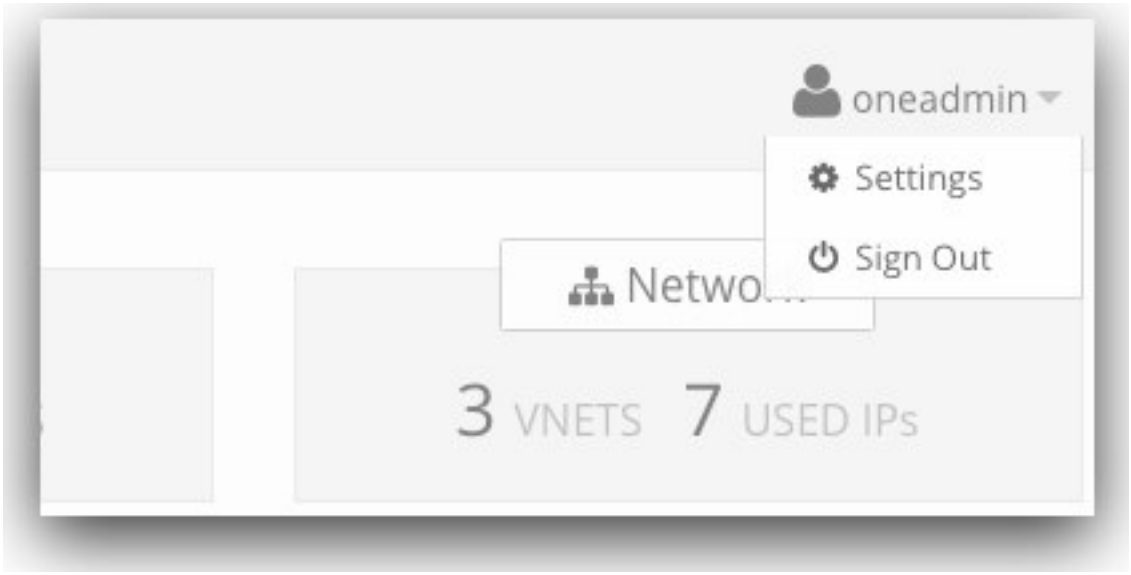
Launch a New VM in Three Steps

- Define a name and the number of instances
- Select one of the available templates
- Select one of the available OS images



Internationalization and Languages

Sunstone support multiple languages. Users can change it from the settings dialog:



8.4 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 guide you will learn how to enable other authentication methods and how to secure the Sunstone connections through SSL.

8.4.1 Authentication Methods

Authentication is two-folded:

- **Web client and Sunstone server**. Authentication is based on the credentials store in the OpenNebula database for the user. Depending on the type of this credentials the authentication method can be: basic, 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 guide*.

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.

8.4.2 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 lighttpd configured to act as an HTTP proxy to a correctly configured OpenNebula Sunstone.

Let's assume the server were the lighttpd 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 milleage may vary, but not a lot):

- Install the `ssl-cert` package

```
$ sudo apt-get install ssl-cert
```

- Generate the certificate

```
$ sudo /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

```
$ sudo cat /etc/ssl/private/ssl-cert-snakeoil.key /etc/ssl/certs/ssl-cert-snakeoil.pem > /etc/lighttp
```

Step 2: SSL HTTP Proxy (e.g. 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.

Step 3: Sunstone Configuration

Start the Sunstone server using the default values, this way the server will be listening at localhost:9869

Once the lighttpd 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.

8.5 Cloud Servers Authentication

OpenNebula ships with three servers: *Sunstone*, *EC2* and *OCCI*. 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.

8.5.1 Server Users

The *Sunstone*, *EC2* and *OCCI* 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**.

8.5.2 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`
- *OCCI*: `/etc/one/occi-server.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 -1 /var/lib/one/.one
ec2_auth
occi_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
```

8.5.3 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`
- *OCCI*: `/etc/one/occi-server.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
```

8.5.4 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 below.

CHAPTER

NINE

OTHER SUBSYSTEMS

9.1 MySQL Backend

The MySQL backend was introduced in OpenNebula 2.0 as an alternative to the Sqlite backend available in previous releases.

Either of them can be used seamlessly to the upper layers and ecosystem tools. These high level components do not need to be modified or configured.

The two backends cannot coexist, and you will have to decide which one is going to be used while planning your OpenNebula installation.

9.1.1 Building OpenNebula with MySQL Support

This section is only relevant if you are building OpenNebula from source. If you downloaded our compiled packages, you can skip to *Installation*.

Requirements

An installation of the mysql server database is required. For an Ubuntu distribution, the packages to install are:

- libmysql++-dev
- libxml2-dev

Also, you will need a working mysql server install. For Ubuntu again, you can install the mysql server with:

- mysql-server-5.1

Compilation

To compile OpenNebula from source with mysql support, you need the following option passed to the scons:

```
$ scons mysql=yes
```

Afterwards, installation proceeds normally, configuration needs to take into account the mysql server details, and for users of OpenNebula the DB backend is fully transparent.

9.1.2 Installation

First of all, you need a working MySQL server.

Of course, you can use any instance you have already deployed. OpenNebula will connect to other machines different from localhost.

You can also configure two different OpenNebula installations to use the same MySQL server. In this case, you have to make sure that they use different database names.

Configuring MySQL

In order to let OpenNebula use your existing MySQL database server, you need to add a new user and grant it privileges on one new 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 'oneadmin';
Query OK, 0 rows affected (0.00 sec)
```

Warning: Remember to choose different values, at least for the password.

Warning: GRANT ALL PRIVILEGES ON <db_name>.* TO <user> IDENTIFIED BY <passwd>

Visit the MySQL documentation to learn how to manage accounts.

Configuring OpenNebula

Before you run OpenNebula, 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  = "oneadmin",
       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

9.1.3 Using OpenNebula with MySQL

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

CHAPTER

TEN

REFERENCES

10.1 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`.

10.1.1 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_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.
- `DB` : Vector of configuration attributes for the database backend.
  - `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_BASE_PORT` : VNC ports for VMs can be automatically set to `VNC_BASE_PORT` + `VMID`. Refer to the *VM template reference* for further information.
- `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) or syslog.
- 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 = 30

MONITORING_INTERVAL = 60
MONITORING_THREADS  = 50

#HOST_PER_INTERVAL               = 15
#HOST_MONITORING_EXPIRATION_TIME = 43200

#VM_PER_INTERVAL               = 5
#VM_MONITORING_EXPIRATION_TIME = 14400

SCRIPTS_REMOTE_DIR=/var/tmp/one

PORT = 2633

DB = [ backend = "sqlite" ]

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

VNC_BASE_PORT = 5900

#VM_SUBMIT_ON_HOLD = "NO"
```

10.1.2 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

```
#*******************************************************************************
# XML-RPC server configuration
#*******************************************************************************

#MAX_CONN            = 15
#MAX_CONN_BACKLOG    = 15
#KEEPALIVE_TIMEOUT   = 15
#KEEPALIVE_MAX_CONN  = 30
#TIMEOUT             = 15
```

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.

10.1.3 Virtual Networks

- NETWORK_SIZE: Default size for virtual networks
- MAC_PREFIX: Default MAC prefix to generate virtual network MAC addresses

Sample configuration:

```
#*******************************************************************************
# Physical Networks configuration
#*******************************************************************************

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

10.1.4 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 in the hosts. It is the same for all the hosts in the cluster. DATASTORE_LOCATION **is only for the hosts and not the front-end**. It defaults to /var/lib/one/datastores (or $ONE_LOCATION/var/datastores in self-contained mode)
- DATASTORE_BASE_PATH: This is the base path for the SOURCE attribute of the images registered in a Datastore. This is a default value, that can be changed when the datastore is created.
- DATASTORE_CAPACITY_CHECK: Checks that there is enough capacity before creating a new imag. Defaults to Yes.
- DEFAULT_IMAGE_TYPE : Default value for TYPE field when it is omitted in a template. Values accepted are OS, CDROM, DATABLOCK.
- 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
xvd	XEN Virtual Disk
vd	KVM virtual disk

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_BASE_PATH = /var/lib/one/datastores

DATASTORE_CAPACITY_CHECK = "yes"

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

10.1.5 Information Collector

This driver CANNOT BE ASSIGNED TO A HOST, and needs to be used with KVM or Xen drivers Options that can be set:

- `-a`: Address to bind the collectd sockect (defults 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 (defult 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" ]
```

10.1.6 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 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",
      executable = "one_im_ssh",
      arguments  = "-r 0 -t 15 kvm" ]
#-------------------------------------------------------------------------------
```

10.1.7 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/`

For more information on configuring and setting up the virtualizer please check the guide that suits you:

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

Sample configuration:

```
#-------------------------------------------------------------------------------
# Virtualization Driver Configuration
#-------------------------------------------------------------------------------

VM_MAD = [
    name       = "kvm",
    executable = "one_vmm_ssh",
    arguments  = "-t 15 -r 0 kvm",
    default    = "vmm_ssh/vmm_ssh_kvm.conf",
    type       = "kvm" ]
```

10.1.8 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,vmfs,ceph" ]
```

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 behaviour.

- **name** : name of the transfer driver, listed in the -d option of the TM_MAD section
- **ln_target** : determines how the persistent images will be cloned when a new VM is instantiated.
  - **NONE**: The image will be linked and no more storage capacity will be used
  - **SELF**: The image will be cloned in the Images datastore
  - **SYSTEM**: The image will be cloned in the System datastore
- **clone_target** : determines how the non persistent images will be cloned when a new VM is instantiated.
  - **NONE**: The image will be linked and no more storage capacity will be used
  - **SELF**: The image will be cloned in the Images datastore
  - **SYSTEM**: The image will be cloned in the System datastore
- **shared** : determines if the storage holding the system datastore is shared among the different hosts or not. Valid values: *yes* or *no*.

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"
]
```

10.1.9 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

Sample configuration:

```
DATASTORE_MAD = [
    executable = "one_datastore",
    arguments  = "-t 15 -d dummy,fs,vmfs,lvm,ceph"
]
```

For more information on this Driver and how to customize it, please visit *its reference guide*.

10.1.10 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
  - **FAILED**, when the VM enters the failed state
  - **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      = "on_failure_recreate",
   on        = "FAILED",
   command   = "/usr/bin/env onevm delete --recreate",
   arguments = "$ID" ]

VM_HOOK = [
  name      = "advanced_hook",
  on        = "CUSTOM",
  state     = "ACTIVE",
  lcm_state = "BOOT_UNKNOWN",
  command   = "log.rb",
  arguments = "$ID $PREV_STATE $PREV_LCM_STATE" ]
```

10.1.11 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
```

10.1.12 Restricted Attributes Configuration

- **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

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 = "RANK"
#VM_RESTRICTED_ATTR = "SCHED_RANK"
#VM_RESTRICTED_ATTR = "REQUIREMENTS"
#VM_RESTRICTED_ATTR = "SCHED_REQUIREMENTS"

IMAGE_RESTRICTED_ATTR = "SOURCE"
```

10.1.13 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_VNET_ATTR       = "VLAN_TAGGED_ID"
```

10.1.14 OneGate Configuration

- **ONEGATE_ENDPOINT**: Endpoint where OneGate will be listening. Optional.

Sample configuration:

```
ONEGATE_ENDPOINT = "http://192.168.0.5:5030"
```

10.2 Scheduler

The Scheduler module is in charge of the assignment between pending Virtual Machines and known Hosts. OpenNebula's architecture defines this module as a separate process that can be started independently of `oned`. The OpenNebula scheduling framework is designed in a generic way, so it is highly modifiable and can be easily replaced by third-party developments.

10.2.1 The Match-making Scheduler

OpenNebula comes with a `match making` scheduler (*mm_sched*) that implements the *Rank Scheduling Policy*. The goal of this policy is to prioritize those resources more suitable for the VM.

The match-making algorithm works as follows:

- Each disk of a running VM consumes storage from an Image Datastore. The VMs that require more storage than there is currently available are filtered out, and will remain in the 'pending' state.
- Those hosts that do not meet the VM requirements (see the *SCHED_REQUIREMENTS attribute*) or do not have enough resources (available CPU and memory) to run the VM are filtered out.
- The same happens for System Datastores: the ones that do not meet the DS requirements (see the *SCHED_DS_REQUIREMENTS attribute*) or do not have enough free storage are filtered out.
- The *SCHED_RANK and SCHED_DS_RANK expressions* are evaluated upon the Host and Datastore list using the information gathered by the monitor drivers. Any variable reported by the monitor driver (or manually set in the Host or Datastore template) can be included in the rank expressions.
- Those resources with a higher rank are used first to allocate VMs.

This scheduler algorithm easily allows the implementation of several placement heuristics (see below) depending on the RANK expressions used.

Configuring the Scheduling Policies

The policy used to place a VM can be configured in two places:

- For each VM, as defined by the SCHED_RANK and SCHED_DS_RANK attributes in the VM template.
- Globally for all the VMs in the `sched.conf` file

Re-Scheduling Virtual Machines

When a VM is in the running state it can be rescheduled. By issuing the `onevm resched` command the VM's recheduling flag is set. In a subsequent scheduling interval, the VM will be consider for rescheduling, if:

- There is a suitable host for the VM
- The VM is not already running in it

This feature can be used by other components to trigger rescheduling action when certain conditions are met.

Scheduling VM Actions

Users can schedule one or more VM actions to be executed at a certain date and time. The *onevm command* 'schedule' option will add a new SCHED_ACTION attribute to the Virtual Machine editable template. Visit *the VM guide* for more information.

10.2.2 Configuration

The behavior of the scheduler can be tuned to adapt it to your infrastructure with the following configuration parameters defined in /etc/one/sched.conf:

- `ONED_PORT`: Port to connect to the OpenNebula daemon oned (Default: 2633)
- `SCHED_INTERVAL`: Seconds between two scheduling actions (Default: 30)
- `MAX_VM`: Maximum number of Virtual Machines scheduled in each scheduling action (Default: 5000). Use 0 to schedule all pending VMs each time.
- `MAX_DISPATCH`: Maximum number of Virtual Machines actually dispatched to a host in each scheduling action (Default: 30)
- `MAX_HOST`: Maximum number of Virtual Machines dispatched to a given host in each scheduling action (Default: 1)
- `HYPERVISOR_MEM`: Fraction of total MEMORY reserved for the hypervisor. E.g. 0.1 means that only 90% of the total MEMORY will be used
- `LIVE_RESCHEDS`: Perform live (1) or cold migrations (0) when rescheduling a VM
- `DEFAULT_SCHED`: Definition of the default scheduling algorithm.
  - `RANK`: Arithmetic expression to rank suitable **hosts** based on their attributes.
  - `POLICY`: A predefined policy, it can be set to:

POLICY	DESCRIPTION
0	**Packing**: Minimize the number of hosts in use by packing the VMs in the hosts to reduce VM fragmentation
1	**Striping**: Maximize resources available for the VMs by spreading the VMs in the hosts
2	**Load-aware**: Maximize resources available for the VMs by using those nodes with less load
3	**Custom**: Use a custom RANK
4	**Fixed**: Hosts will be ranked according to the PRIORITY attribute found in the Host or Cluster template

- `DEFAULT_DS_SCHED`: Definition of the default storage scheduling algorithm.
  - `RANK`: Arithmetic expression to rank suitable **datastores** based on their attributes.
  - `POLICY`: A predefined policy, it can be set to:

POLICY	DESCRIPTION
0	**Packing**:: Tries to optimize storage usage by selecting the DS with less free space
1	**Striping**: Tries to optimize I/O by distributing the VMs across datastores
2	**Custom**: Use a custom RANK
3	**Fixed**: Datastores will be ranked according to the PRIORITY attribute found in the Datastore template

The optimal values of the scheduler parameters depend on the hypervisor, storage subsystem and number of physical hosts. The values can be derived by finding out the max number of VMs that can be started in your set up with out getting hypervisor related errors.

Sample Configuration:

```
ONED_PORT = 2633

SCHED_INTERVAL = 30

MAX_VM       = 5000
MAX_DISPATCH = 30
MAX_HOST     = 1

LIVE_RESCHEDS  = 0

HYPERVISOR_MEM = 0.1

DEFAULT_SCHED = [
   policy = 3,
   rank   = "- (RUNNING_VMS * 50  + FREE_CPU)"
]

DEFAULT_DS_SCHED = [
   policy = 1
]
```

Pre-defined Placement Policies

The following list describes the predefined policies (`DEFAULT_SCHED`) that can be configured through the `sched.conf` file.

Packing Policy

- **Target**: Minimize the number of cluster nodes in use

- **Heuristic**: Pack the VMs in the cluster nodes to reduce VM fragmentation
- **Implementation**: Use those nodes with more VMs running first

```
RANK = RUNNING_VMS
```

Striping Policy

- **Target**: Maximize the resources available to VMs in a node
- **Heuristic**: Spread the VMs in the cluster nodes
- **Implementation**: Use those nodes with less VMs running first

```
RANK = "- RUNNING_VMS"
```

Load-aware Policy

- **Target**: Maximize the resources available to VMs in a node
- **Heuristic**: Use those nodes with less load
- **Implementation**: Use those nodes with more FREE_CPU first

```
RANK = FREE_CPU
```

Fixed Policy

- **Target**: Sort the hosts manually
- **Heuristic**: Use the PRIORITY attribute
- **Implementation**: Use those nodes with more PRIORITY first

```
RANK = PRIORITY
```

Pre-defined Storage Policies

The following list describes the predefined storage policies (`DEFAULT_DS_SCHED`) that can be configured through the `sched.conf` file.

Packing Policy

Tries to optimize storage usage by selecting the DS with less free space

- **Target**: Minimize the number of system datastores in use
- **Heuristic**: Pack the VMs in the system datastores to reduce VM fragmentation
- **Implementation**: Use those datastores with less free space first

```
RANK = "- FREE_MB"
```

Striping Policy

- **Target**: Maximize the I/O available to VMs
- **Heuristic**: Spread the VMs in the system datastores
- **Implementation**: Use those datastores with more free space first

```
RANK = "FREE_MB"
```

Fixed Policy

- **Target**: Sort the datastores manually
- **Heuristic**: Use the PRIORITY attribute
- **Implementation**: Use those datastores with more PRIORITY first

```
RANK = PRIORITY
```

10.3 Logging & Debugging

OpenNebula provides logs for many resources. It supports two logging systems: file based logging systems and syslog logging.

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 the logging strategy is almost identical, except that the logging message change slightly their format following syslog logging conventions.

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 either 'syslog' or 'file' (default), 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 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 anatomy of an OpenNebula message for a file based logging system is the following:

```
date [module][log_level]: message body
```

In the case of syslog it follows the standard:

```
date hostname process[pid]: [module][log_level]: message body
```

Where 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][module][log_level]: message body
```

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               : FAILED
LCM_STATE           : LCM_INIT
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/image
  TIMESTAMP="Tue Jul 19 17:44:31 2011" ]
MEMORY=64
NAME=one-0
VMID=0

VIRTUAL MACHINE HISTORY
 SEQ        HOSTNAME REASON           START        TIME       PTIME
   0          host01   erro  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`.

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/
Tue Jul 19 17:17:22 2011 [InM][I]: ssh: Could not resolve hostname host01: nodename nor servname prov
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 remot
```

From the execution output we notice that the host name is not know, probably a mistake naming the host.

10.4 Onedb Tool

This guide 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 looses 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
```

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 specials characters, such as `@` or `#`, the onedb command will fail to connect to it.

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

```
$ mysql -u oneadmin -p

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

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 not 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.

10.5 Datastore configuration

Datastores can be parametrized with several parameters. In the following list there are the meaning of the different parameters for all the datastores.

- **RESTRICTED_DIRS**: Paths not allowed for image importing
- **SAFE_DIRS**: Paths allowed for image importing
- **NO_DECOMPRESS**: Do not decompress images downloaded
- **LIMIT_TRANSFER_BW**: Maximum bandwidth used to download images. By default is bytes/second but you can use k, m and g for kilo/mega/gigabytes.
- **BRIDGE_LIST**: List of hosts used for image actions. Used as a roundrobin list.
- **POOL_NAME**: Name of the Ceph pool to use
- **VG_NAME**: Volume group to use
- **BASE_IQN**: iscsi base identifier
- **STAGING_DIR**: Temporary directory where images are downloaded
- **DS_TMP_DIR**: Temporary directory where images are downloaded
- **CEPH_HOST**: Space-separated list of Ceph monitors.
- **CEPH_SECRET**: A generated UUID for a LibVirt secret.
- **LIMIT_MB**: Limit, in MB, of storage that OpenNebula will use for this datastore

Not all these parameters have meaning for all the datastores. Here is the matrix of parameters accepted by each one:

Parameter	ceph	fs	iscsi	lvm	vmfs
RESTRICTED_DIRS	yes	yes	yes	yes	yes
SAFE_DIRS	yes	yes	yes	yes	yes
NO_DECOMPRESS	yes	yes	yes	yes	yes
LIMIT_TRANSFER_BW	yes	yes	yes	yes	yes
BRIDGE_LIST	•	yes	•	•	•
POOL_NAME	yes	•	•	•	•
VG_NAME	•	•	yes	yes	•
BASE_IQN	•	•	yes	•	•
STAGING_DIR	yes	•	•	•	•
DS_TMP_DIR	yes	•	•	•	•
CEPH_HOST	yes	•	•	•	•
CEPH_SECRET	yes	•	•	•	•
LIMIT_MB	yes	yes	yes	yes	yes