CLOUD DESIGN WITH OPENNEBULA

OVERVIEW

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

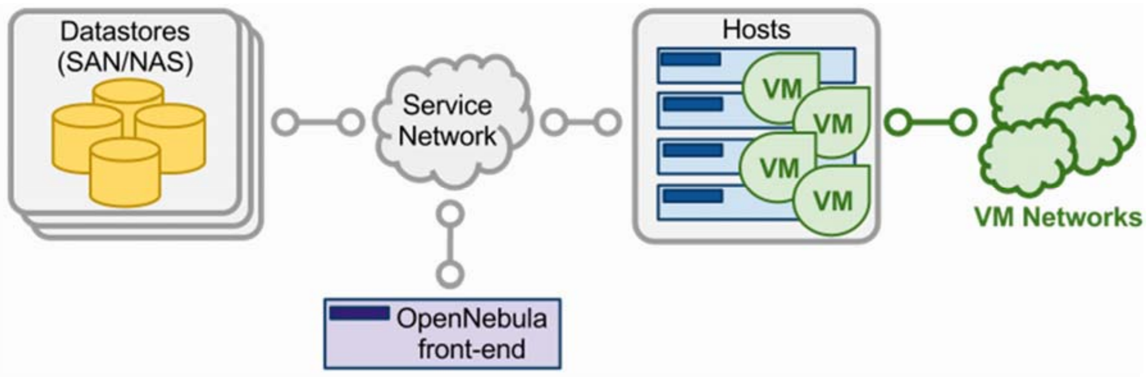
In order to get the most out of a OpenNebula Cloud, we recommend that you create a plan with the features, performance, scalability, and high availability characteristics you want in your deployment. This Chapter provides information to plan an OpenNebula cloud based on KVM or vCenter. With this information, you will be able to easily architect and dimension your deployment, as well as understand the technologies involved in the management of virtualized resources and their relationship.

OPEN CLOUD ARCHITECTURE

Enterprise cloud computing is the next step in the evolution of data center (DC) virtualization. OpenNebula is a simple but feature-rich and flexible solution to build and manage enterprise clouds and virtualized DCs, that combines existing virtualization technologies with advanced features for multi-tenancy, automatic provision and elasticity. OpenNebula follows a bottom-up approach driven by sysadmins, devops and users real needs.

ARCHITECTURAL OVERVIEW

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



A cloud architecture is defined by three components: storage, networking and virtualization. Therefore, the basic components of an OpenNebula system are:

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

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

DIMENSIONING THE CLOUD

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

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

OPENNEBULA FRONT-END

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

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

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

KVM NODES

Regarding the dimensions of the KVM virtualization nodes:

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

STORAGE

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

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

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

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

NETWORK

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

FRONT-END

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

OpenNebula services include:

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

Note

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

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

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

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

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

MONITORING

The monitoring subsystem gathers information relative to the hosts and the virtual machines, such as the host status, basic performance indicators, as well as VM status and capacity consumption. This information is collected by executing a set of static probes provided by OpenNebula. The information is sent according to the following process: each host periodically sends monitoring data to the front-end which collects it and processes it in a dedicated module. This model is highly scalable and its limit (in terms of number of VMs monitored per second) is bounded to the performance of the server running oned and the database server.

Please check the the Monitoring Section for more details.

VIRTUALIZATION HOSTS

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

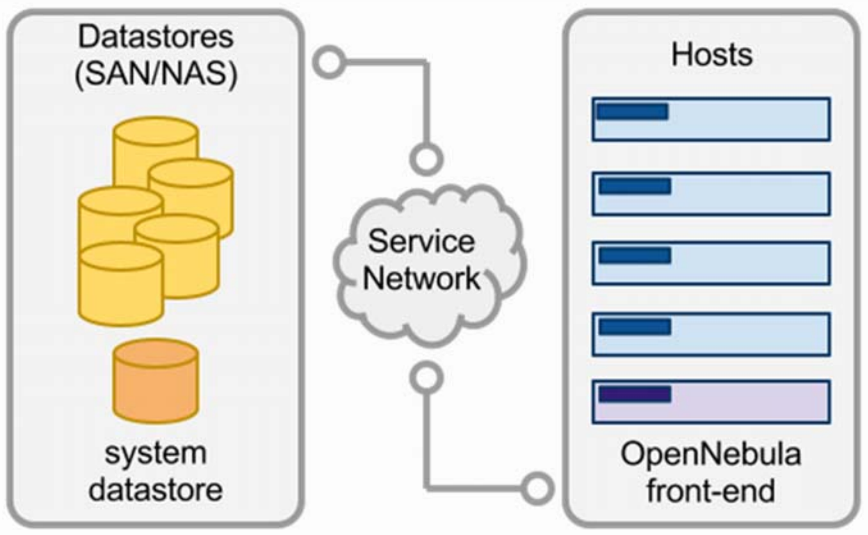
OpenNebula natively supports one open source hypervisor, the KVM hypervisor, and OpenNebula is configured by default to interact with hosts running KVM.

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

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

STORAGE

OpenNebula uses *Datastores* to store VMs' disk images. A datastore is any storage medium, typically backed by SAN/NAS servers. In general, each datastore has to be accessible through the front-end using any suitable technology NAS, SAN or direct attached storage.



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

OpenNebula is shipped with 3 different datastore classes:

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

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

- Filesystem: to store disk images in a file form. There are three types: ssh, shared and qcow.
- LVM: to use LVM volumes instead of plain files to hold the Virtual Images. This reduces the overhead of having a file-system in place and thus increases performance.
- Ceph: to store disk images using Ceph block devices.

Warning

Default: The default system and images datastores are configured to use a filesystem with the ssh transfer drivers.

Please check the Storage Chapter for more details.

NETWORKING

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

- **Service Network**: used by the OpenNebula front-end daemons to access the hosts in order to manage and monitor the hypervisors, and move image files. It is highly recommended to install a dedicated network for this purpose;
- **Instance Network**: offers network connectivity to the VMs across the different hosts. To make an effective use of your VM deployments, you will probably need to make one or more physical networks accessible to them.

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

- **dummy** (default): doesn't perform any network operation, and firewalling rules are also ignored.
- fw: firewalling rules are applied, but networking isolation is ignored.
- 802.1Q: restrict network access through VLAN tagging, which requires support by the hardware switches.
- ebtables: restrict network access through Ebtables rules. No special hardware configuration required.
- ovswitch: restrict network access with Open vSwitch Virtual Switch.
- vxlan: segment a VLAN in isolated networks using the VXLAN encapsulation protocol.

Please check the Networking Chapter to find out more information about the networking technologies supported by OpenNebula.

AUTHENTICATION

The following authentication methods are supported to access OpenNebula:

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

Warning

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

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

ADVANCED COMPONENTS

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

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