Hybrid Cloud Computing with OpenNebula 1.4

C12G Labs S.L.

Rev20100611

C12G LABS

Contents

C12G LABS

Chapter 1

Getting Started

1.1 Building a Hybrid Cloud

1.1.1 What is a Hybrid Cloud?

A Hybrid Cloud is an **extension of a Private Cloud to combine local resources with resources from remote Cloud providers**. The remote provider could be a commercial Cloud service, such as Amazon EC2 or ElasticHosts, or a partner infrastructure running a different OpenNebula instance. Such support for cloudbursting enables highly scalable hosting environments.

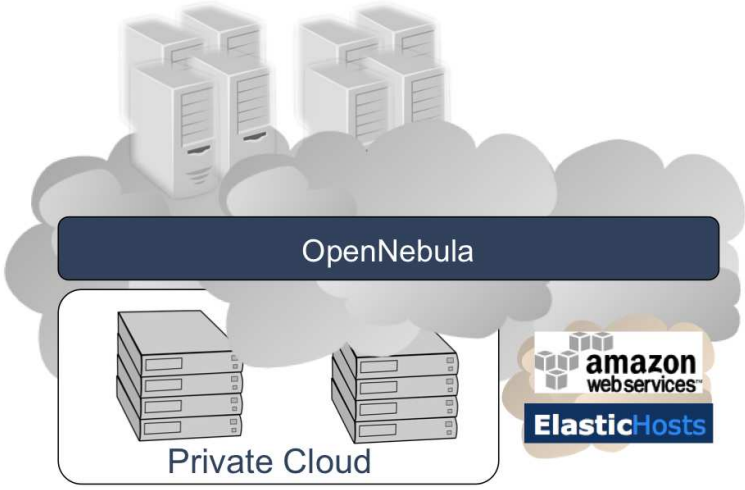

Figure 1.1:

1.1.2 The User View

An Hybrid Cloud Deployment powered by OpenNebula is **fully transparent to infrastructure users**. Users continue using the same private and public Cloud interfaces, so the federation is not performed at service or application level but at infrastructure level by OpenNebula. It is the infrastructure administrator who takes decisions about the scale out of the infrastructure according to infrastructure or business policies.

Amazon EC2 cloud appears as a new host in the OpenNebula system:

A **simple template** like the following is enough to launch Virtual Machines in Amazon EC2:

```
EC2 = [ AMI="ami-acc723c5",
        AUTHORIZED_PORTS="22" ]
```

Assuming the template above has been saved to a file called *myEC2Machine.one*, the following will **launch the Virtual Machine within Amazon EC2**:

After submission, **details about the Virtual Machine** (including hostname to access the machine) can be requested using:

1.1.3 How the System Operates

There is **no modification in the operation of OpenNebula to integrate Cloud services**. A Cloud service **is managed as any other OpenNebula host** that may provide ”infinite” capacity for the execution of VMs.

C12G LABS

Chapter 2

Configuring the Cloud Service Adaptors

2.1 Amazon EC2 Adaptor

You should take into account the following technical considerations when using the EC2 cloud with OpenNebula:

- There is no direct access to the dom0, so it cannot be monitored (we don't know where the VM is running on the EC2 cloud).
- Since EC2 is beta, you can launch simultaneously at much 20 instances. Although you can request a higher number to Amazon.
- The usual OpenNebula functionality for snapshotting, restoring, or migration is not available with EC2.
- By default OpenNebula will always launch small instances, unless otherwise specified.

Please refer to the EC2 documentation to obtain more information about Amazon instances types and image management:

- General information of instances
- Standard instances usage
- High CPU instances usage

2.1.1 EC2 Configuration

You must have a working account for AWS and signup for EC2 and S3 services, and also download and unpack the EC2 API tools provided, do some manual test to verify everything works before start configuring OpenNebula for EC2 support.

Please note that EC2 has to be installed only in the cluster front-end.

2.1.2 Driver Files

The driver consists of the following files:

- `$ONE_LOCATION/lib/mads/one_im_ec2.rb` : This file is accessed by the Information Manager to get the maximum memory and cpu constraints for EC2 allocations.
- `$ONE_LOCATION/lib/mads/one_vmm_ec2.rb`: This is the main ruby program file that invokes EC2 actions like deploy, shutdown...

- `$ONE_LOCATION/etc/im_ec2/im_ec2.conf` : In this file we define the maximum capacity that we want to allocate in EC2.

```
# Max number of instances that can be launched into EC2
SMALL_INSTANCES=5
LARGE_INSTANCES=
EXTRALARGE_INSTANCES=
```

- `$ONE_LOCATION/etc/vmm_ec2/vmm_ec2.conf` : In this file we define default configurations for the VM placed in EC2, for example the "instancetype" attribute.

```
<!--
 Default configuration attributes for the EC2 driver
 (all domains will use these values as defaults)
 Valid atributes are:
   - ec2[keypair,authorizedports,instancetype]
 Use XML syntax to specify defaults, note elements are UPCASE
 Example:
 <TEMPLATE>
   <EC2>
     <KEYPAIR>gsg-keypair</KEYPAIR>
     <AUTHORIZEDPORTS>22</AUTHORIZEDPORTS>
     <INSTANCETYPE>m1.small</INSTANCETYPE>
   </EC2>
 </TEMPLATE>
-->

<TEMPLATE>
  <EC2>
    <INSTANCETYPE>m1.small</INSTANCETYPE>
  </EC2>
</TEMPLATE>
```

- `$ONE_LOCATION/etc/vmm_ec2/vmm_ec2rc` : In this file we configure the account that will be used to launch instances on EC2, these are the environment variables needed by the EC2 API[1].

Note: If OpenNebula was installed in **system wide** mode these directories become `/usr/lib/one/mads` and `/etc/one/`, respectively. The rest of this guide refers to the `$ONE_LOCATION` paths (corresponding to **self contained** mode) and omits the equivalent **system wide** locations. More information on installation modes can be found here.

2.1.3 Configuration

OpenNebula Configuration

Two lines must be added to the `$ONE_LOCATION/etc/oned.conf` file in order to use the driver.

```
  IM_MAD = [
      name       = "im_ec2",
      executable = "one_im_ec2",
      arguments  = "im_ec2/im_ec2.conf",
      default    = "im_ec2/im_ec2.conf" ]

  VM_MAD = [
      name       = "vmm_ec2",
```

[1] Application Programming Interface

```
executable = "one_vmm_ec2",
arguments  = "<ec2_configuration_options> vmm_ec2/vmm_ec2.conf",
type       = "xml" ]
```

where *<ec2_configuration_options>* can be used to set up the EC2 environment. It can be any number of the following flags, each corresponding to one EC2 environmental variable:

FLAG	SETs
-u	EC2_URL[2]
-h	EC2_HOME
-k	EC2_PRIVATE_KEY
-c	EC2_CERT

For instance, the following line will make the driver use a specific certificate to communicate with EC2:

```
arguments  = "-c /home/user/.ec2/ec2-cert.pem vmm_ec2/vmm_ec2.conf",
```

Make sure that the default configuration file (`vmm_ec2.conf`) is passed as the **last** argument.

After configuring everything when you start ONE, you need to add the ec2 host to the host list to be able to submit virtual machines, like the following:

Driver Configuration

Additionally you must configure the location of your EC2 certificates and EC2 API[1] installation path, for this edit the file **$ONE_LOCATION/etc/mad/vmm_ec2rc** and add:

```
EC2_HOME="<path_to_your_ec2_installation>"
EC2_PRIVATE_KEY="<path_to_your_private_key>"
EC2_CERT="<path_to_your_public_cert>"
```

Also you must configure the maximum capacity that you want OpenNebula to deploy on the EC2, for this edit the file **$ONE_LOCATION/etc/im_ec2/im_ec2.conf** , in this example we say that we want at much 4 small and 1 large instances launched into EC2:

```
# Max number of instances that can be launched into EC2

SMALL_INSTANCES=4
LARGE_INSTANCES=1
EXTRALARGE_INSTANCES=
```

2.1.4 EC2 Specific Template Attributes

Mandatory Attributes

- **AMI**:the AMI name that will be launched
- **KEYPAIR**: This attribute indicates the rsa key-pair used to initiate the machines, the private keypair later will be used to execute commands like *ssh -i id_keypair* or *scp -i id_keypair*

Optional Attributes

- **ELASTICIP**: This is the elastic IP address you want to assign to the instance launched.
- **AUTHORIZED_PORTS**: this parameter is passed to the command *ec2-authorize default -p port*, and must be in the form of a number "22" or a range "22-90",
- **INSTANCETYPE**: this atribute indicates the type of instace to be launched in EC2, by default all instances will be "m1.small". Remember valid values for this are m1.small, m1.large, m1.xlarge, c1.medium, c1.xlarge.

2.1.5 Multi EC2 Site Support

From OpenNebula 1.4 onwards it is possible to define various EC2 sites to allow opennebula the managing of EC2 availability zones or even the use of various private clouds offering EC2 interfaces.

To properly configure multiple EC2 sites, you need to follow these steps:

- define one VMM driver for each EC2 site, like:

```
VM_MAD = [
    name       = "vmm_amazon_eu_west",
    executable = "one_vmm_ec2",
    arguments  = "-u https://eu-west-1.ec2.amazonaws.com vmm_ec2/vmm_ec2.conf",
    type       = "xml" ]
```

- create a host that uses the MAD defined above. The EC2 site will be incarnated in this host for OpenNebula. We will use the EC2 IM driver, the ad-hoc defined VM mad and a dummy TM (all images for EC2 must have been uploaded previously on S3):
- create a host template with an EC2 section targeting the created EC2 site. OpenNebula 1.4 introduces a new tag (**CLOUD**) in the template's EC2 section for this purpose. You can create multiple EC2 sections so with one template you can define different AMIs depending on which host it is scheduled.

```
EC2 = [ CLOUD="ec2_eu_west",
        AMI="ami-0022c769",
        AUTHORIZED_PORTS="22" ]
EC2 = [ CLOUD="ec2_eu_east",
        AMI="ami-03324cc9",
        AUTHORIZED_PORTS="22" ]
```

If you create another EC2 host called *ec2_eu_east* then you will have *ami-0022c769* launched when this VM template is sent to host *ec2_eu_west* and *ami-03324cc9* whenever the VM template is sent to host *ec2_eu_east*.

If only one EC2 site is defined, the EC2 driver will deploy all EC2 templates onto it, not paying attention to the **CLOUD** attribute.

2.1.6 Testing

You must create a template file containing the information of the AMIs you want to launch, its important to note that when deploying VMs on EC2 with OpenNebula, the template file should contain the attributes AMI and KEYPAIR used by the EC2 VMM Mad.

Additionally if you have an elastic IP address you want to use with your EC2 instances, you can specify it as an optional parameter.

```
CPU      = 0.5
MEMORY   = 128

#Xen or KVM template machine, this will be use when submitting this VM to local resources

OS       = [kernel="/vmlinuz",initrd= "/initrd.img",root="sda1" ]
DISK     = [source="/imges/apache.img",target="sda",readonly="no"]
NIC      = [bridge="eth0"]

#EC2 template machine, this will be use wen submitting this VM to EC2

EC2 = [ AMI="ami-00bafcb5",
        KEYPAIR="gsg-keypair",
        ELASTICIP="75.101.155.97",
        AUTHORIZED_PORTS="22",
        INSTANCETYPE=m1.small]

#Add this if you want to use only EC2 cloud
#REQUIREMENTS = "HOSTNAME = ec2"
```

You only can submit and control the template using the OpenNebula interface:

Now you can monitor the state of the VM with

Also you can see information (like IP address) related to the amazon instance launched via the command

You can check out the EC2-ONE Use Case to see how to scale out a computing cluster with OpenNebula and EC2.

2.2 ElasticHosts Adaptor

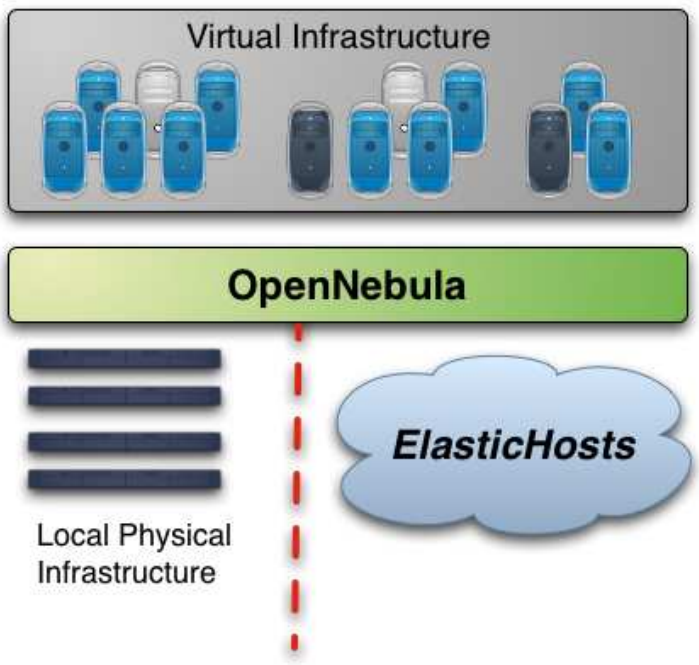

Figure 2.1:

ElasticHosts offers KVM based virtualized hosts in a cloud like fashion, i.e., à la Amazon EC2. This set of drivers speaks with the neat RESTful ElasticHosts API enabling cloudbursting if needed.

As seen in the figure, OpenNebula can be used to manage a virtual infrastructure on top of physical local infrastructure. In particular, we are using dedicated physical servers to provide the fabric for the flexible virtual infrastructure, which can be dynamically increased or decreased driven by its demand. If the local fabric is not enough to provide the demanded virtual infrastructure capacity (that is, we need to keep growing the cluster), we can spawn more nodes from a cloud provider, in this case the KVM based ElasticHosts. This guide shows how to configure OpenNebula to be able to interface this cloud provider.

2.2.1 ElasticHosts Configuration

You will need the following:

- An ElasticHosts account, with a valid EHAUTH
- An updated OpenNebula installation.
- The elastichosts script installed and accessible in your PATH

2.2.2 Driver Files

The driver consists of the following files:

- `$ONE_LOCATION/lib/mads/one_vmm_eh` : Shell script wrapper to the virtualization ElasticHosts driver. Sets the environment and other bootstrap tasks.
- `$ONE_LOCATION/lib/mads/one_vmm_eh.rb` : The actual ElasticHosts virtualization driver.

* `$ONE_LOCATION/etc/vmm_eh/vmm_ehrc` : environment setup and bootstrap instructions for the ElasticHosts virtualization driver.

- `$ONE_LOCATION/etc/vmm_eh/vmm_eh.conf` : set here default values for ElasticHosts domain definitions, useful for the ElasticHosts virtualization driver.
- `$ONE_LOCATION/lib/mads/one_im_eh` : Shell script wrapper to the ElasticHosts information driver. Sets the environment and other bootstrap tasks.
- `$ONE_LOCATION/lib/mads/one_im_eh.rb` : The actual ElasticHosts information driver.
- `$ONE_LOCATION/etc/im_eh/im_ehrc` : environment setup and bootstrap instructions for the ElasticHosts information driver
- `$ONE_LOCATION/etc/im_eh/im_eh.conf` : set here default values for ElasticHosts domain definitions, useful for the ElasticHosts information driver.

Note: If OpenNebula was installed in system wide mode these directories become `/usr/lib/one/mads` and `/etc/one`, respectively. The rest of this guide refers to the $ONE_LOCATION paths (corresponding to self contained mode) and omits the equivalent system wide locations. More information on installation modes can be found here.

2.2.3 Configuration

OpenNebula Configuration

OpenNebula needs to know if it is going to use the ElasticHosts Driver. To achieve this, two lines have to be placed within `$ONE_LOCATION/etc/oned.conf`, one for the VM driver and other for the information (IM) driver:

```
#-------------------------------------------------------------------------------
#  ElasticHost Information Driver Manager sample configuration
#-------------------------------------------------------------------------------
  IM_MAD = [
      name       = "im_eh",
```

```
      executable = "one_im_eh",
      arguments  = "im_eh/im_eh.conf",
      default    = "im_eh/im_eh.conf" ]
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
#  ElasticHost Virtualization Driver Manager sample configuration
#-------------------------------------------------------------------------------
   VM_MAD = [
      name       = "vmm_eh",
      executable = "one_vmm_eh",
      default    = "vmm_eh/vmm_eh.conf",
      type       = "xml" ]
#-------------------------------------------------------------------------------
```

- The **name** of the driver needs to be provided at the time of adding a new host to OpenNebula.
- **executable** points to the path of the driver executable file. It can be an absolute path or relative to `$ONE_LOCATION/lib/mads`.
- The **default** points to the configuration file for the driver. It can be an absolute path or relative to `$ONE_LOCATION/etc`.
- **type** identifies this driver as using XML[3] messages.

Additionally, the tm_dummy section needs to be uncommented.
Once the configuration, OpenNebula needs to be restarted. Afterwards, you need to add ElasticHosts to the host list to be able to submit virtual machines, like the following:

Driver Configuration

- The EHAUTH can be grabbed from the environment or can be set in `$ONE_LOCATION/etc/vmm_eh/vmm_ehrc`
- `$ONE_LOCATION/etc/im_eh/im_ehrc` holds the default capacity for the ElasticHosts provider. By default, this is set to a very low value, change it at your will.

```
TOTAL_MEMORY=2048
TOTAL_CPU=2
```

2.2.4 ElasticHosts Specific Template Attributes

- **CPUMHZ**: Minimum speed of the CPU requested.
- **MEMORY**: Memory in MB[4] requested for the VM.
- **SMP**: Number of virtual processors or 'auto' to calculate based on cpu.
- **DRIVE**: Comma separated list of previously uploaded drives to conform the VM.
- **NIC**: Comma separated list of network interfaces.
- **BOOT**: Which drive to boot from.

2.2.5 Testing

In order to test the ElasticHosts driver, the following template can be instantiated with appropriate values and sent to a ElasticHosts resource:

The big number after DRIVE="ide:0:0= is the uuid of a drive previously uploaded to ElasticHosts. See this for details.

[3] Extensible Markup Language
[4] Megabyte