LXCoNe. LXC + OpenNebula. Installation & Configuration Guide.

The purpose of this guide is to provide users with a step by step guide to install OpenNebula, LXC and manage LXC containers from OpenNebula.

This guide was designed for all users, to make the installing and configuration process as easy as possible.

After following this guide, users will have a working OpenNebula with graphical interface (Sunstone), at least one host and a running VMs.

Through the installation there are two separate roles: frontend and nodes. The frontend server will execute the OpenNebula services, and the nodes will be used to execute virtual machines.

> ***Note***
> We built and tested this drivers with the frontend installed on Ubuntu 14.04 (Trusty Tahr) and the nodes on Debian 8 (Jessie)

LXC linuxcontainers.org

LXC is a userspace interface for the Linux kernel containment features. Through a powerful API and simple tools, it lets Linux users easily create and manage system or application containers.

OpenNebula opennebula.org

OpenNebula is a cloud computing platform for managing heterogeneous distributed data center infrastructures. The OpenNebula platform manages a data center's virtual infrastructure to build private, public and hybrid implementations of infrastructure as a service. OpenNebula is free and open-source software, subject to the requirements of the Apache License version 2.

1. Installation in the Frontend

> ***Warning***
> *Commands prefixed by # are meant to be run as root. Commands prefixed by $ must be run as oneadmin.*

1.1. Configure *Opennebula* repositories

Add the OpenNebula repository:

```
# wget -q -O- http://downloads.opennebula.org/repo/Ubuntu/repo.key
| apt-key add -
#echo "deb http://downloads.opennebula.org/repo/4.12/Ubuntu/14.04/
stable opennebula" \ > /etc/apt/sources.list.d/opennebula.list
# apt-get update
```

1.2. Install the required packages

```
# apt-get install opennebula opennebula-sunstone
nfs-kernel-server
```

1.3 Configure and Start the services

There are two main processes that must be started, the main OpenNebula daemon: **opennebula**, and the graphical user interface: **opennebula-sunstone**.

Start OpenNebula Sunstone

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

Sunstone listens only in the **loopback** interface by default for security reasons. To change it edit **/etc/one/sunstone-server.conf** and change **:host: 127.0.0.1** to **:host: 0.0.0.0.**

Now restart Sunstone:

```
# service opennebula-sunstone restart
```

1.4 Configure NFS

Warning
Skip this section if you are using a single server for both the frontend and worker node roles.

Export /var/lib/one/ from the frontend to the worker nodes. To do so add the following to **/etc/exports** in the frontend:

```
/var/lib/one/ *(rw,sync,no_subtree_check,no_root_squash,crossmnt,
nohide)
```

Refresh the NFS exports:

```
# service nfs-kernel-server restart
```

1.5. Configure SSH Public Key

OpenNebula will need to SSH passwordlessly from any node (including the frontend) to any other node. Set public key as authorized key:

```
# su - oneadmin
$ cp ~/.ssh/id_rsa.pub ~/.ssh/authorized_keys
```

Add the following snippet to **~/.ssh/config** so it doesn't prompt to add the keys to the **known_hosts** file:

```
$ cat << EOT > ~/.ssh/config
Host *
 StrictHostKeyChecking no
 UserKnownHostsFile /dev/null
EOT
$ chmod 600 ~/.ssh/config
```

1.6. Copy the LXC drivers

Copy the LXC folder under **vmm** to the frontend on this location:
/var/lib/one/remotes/vmm.
The files located inside (**deploy**...) should be in the following path**:**
/var/lib/one/remotes/vmm/lxc/deploy
Copy the LXC folder under **im** to the frontend on this location:
/var/lib/one/remotes/im.
The files located inside (**mon_lxc.sh**...) should be in the following path:
/var/lib/one/remotes/im/lxc.d/mon_lxc.sh

Warning
Make sure this folders and files are owned by oneadmin
#chown -R oneadmin:oneadmin /var/lib/one/remotes/vmm/lxc
#chown -R oneadmin:oneadmin /var/lib/one/remotes/im/lxc.d

1.7. Modify /etc/one/oned.conf

Under Information Driver Configuration, add this:

```
#-------------------------------------------------------------
# LXC 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 = "lxc",
   executable = "one_im_ssh",
   arguments = "-c -t 1 -r 0 lxc" ]
#-------------------------------------------------------------
```

Under Virtualization Driver Configuration, add this:

```
#-------------------------------------------------------------
# LXC Virtualization Driver Manager Configuration
# -r number of retries when monitoring a host
# -t number of threads, i.e.  number of actions performed at the
same time
#-------------------------------------------------------------
VM_MAD = [ name = "lxc",
   executable = "one_vmm_exec",
   arguments = "-t 15 -r 0 lxc",
   type = "xml" ]
#-------------------------------------------------------------
```

We are adding a configuration file example, you can check it.

2. Installation in the Nodes

2.1. Configure of *Opennebula* repositories

Add the OpenNebula repository:

```
# wget -q -O- http://downloads.opennebula.org/repo/Ubuntu/repo.key
| apt-key add -
#echo "deb http://downloads.opennebula.org/repo/4.12/Ubuntu/14.04/
stable opennebula" \ > /etc/apt/sources.list.d/opennebula.list
# apt-get update
```

2.2. Install required packages

```
# apt-get install opennebula-node nfs-common bridge-utils lxc
```

Warning
We installed the host over Debian 8 (jessie). Packages for Jessie aren't in the Opennebula repositories, but you can manually install them using any package manager (dpkg, GDebi) and watching for dependencies.

2.3. Configure the network

Turn down your network interface

```
# ifdown eth0
```

Configure the new bridge in **/etc/network/interfaces**. This is my configuration

```
# This file describes the network interfaces available on your
system
# and how to activate them.  For more information, see
interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
#allow-hotplug eth0
#iface eth0 inet dhcp

auto br0
iface br0 inet static
address 10.8.91.88
netmask 255.255.255.0
gateway 10.8.91.1
bridge_ports eth0
bridge_fd 0
bridge_maxwait 0
```

Turn up the new bridge

```
# ifup br0
```

> *Note*
> **eth0** was my primary network adapter, if the name is different in your case, remember to change it in **bridge_ports** option

2.4. Configure fstab to mount /var/lib/one from the frontend

Add this line to /**etc**/**fstab**

```
192.168.1.1:/var/lib/one/ /var/lib/one/ nfs
soft,intr,rsize=8192,wsize=8192, noauto
```

Replace **192.168.1.1** with the frontend's ip address

Mount the directory

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

Now, the frontend should be able to SSH inside the host without password using **oneadmin** user.

2.5. Add oneadmin to the sudoers file, and enable it to run root commands without password.

Add the following line to **/etc/sudoers**

```
oneadmin ALL= NOPASSWD: ALL
```

2.6. Activate memory limit capability

Check if **cgroup** memory capability is available:

```
# cat /proc/cgroups | grep memory | awk '{ print $4 }'
0
```

A **0** indicates that capability is no loaded (**1** indicates the oposite).

To manage memory on containers add **cgroup** argument to **grub** to activate those functionality. Add, in **GRUB_CMDLINE_LINUX** entry of **/etc/default/grub** file,**cgroup_enable=memory** and **swapaccount=1** parameters.

```
[...]
GRUB_CMDLINE_LINUX="cgroup_enable=memory swapaccount=1"
[...]
```

Regenerate **grub** config

```
# update-grub
```

Reboot to make changes available.

3. Create LXC image

3.1. Create a raw image using LXC

```
# lxc-create -t debian -B loop --fssize=3G -n name
```

We just created a 3Gb raw image with a linux container inside.

3.2. Configure this container

3.2.1. Start the container

First, be sure to copy the **root** password at the end of **lxc-create**

```
# lxc-start -n name
```

The credentials will be **root** with the password you just copied.

3.2.2. Change the default root password

Inside the container type:

```
# passwd
```

3.3.3. Install the software you want

Openssh-server, for example.
You can also modify the template located at:
/usr/share/lxc/templates/lxc-debian
And under "download_debian()" add the packages you want to preinstall. Then, create the container (step 3.1).

4. Sunstone

4.1. Enter the sunstone interface

Log in to this address:

```
http://192.168.1.1:9869/
```

Replace 192.168.1.1 with the frontend's ip address

The credentials are located in the frontend in the following file:
/var/lib/one/.one/one_auth

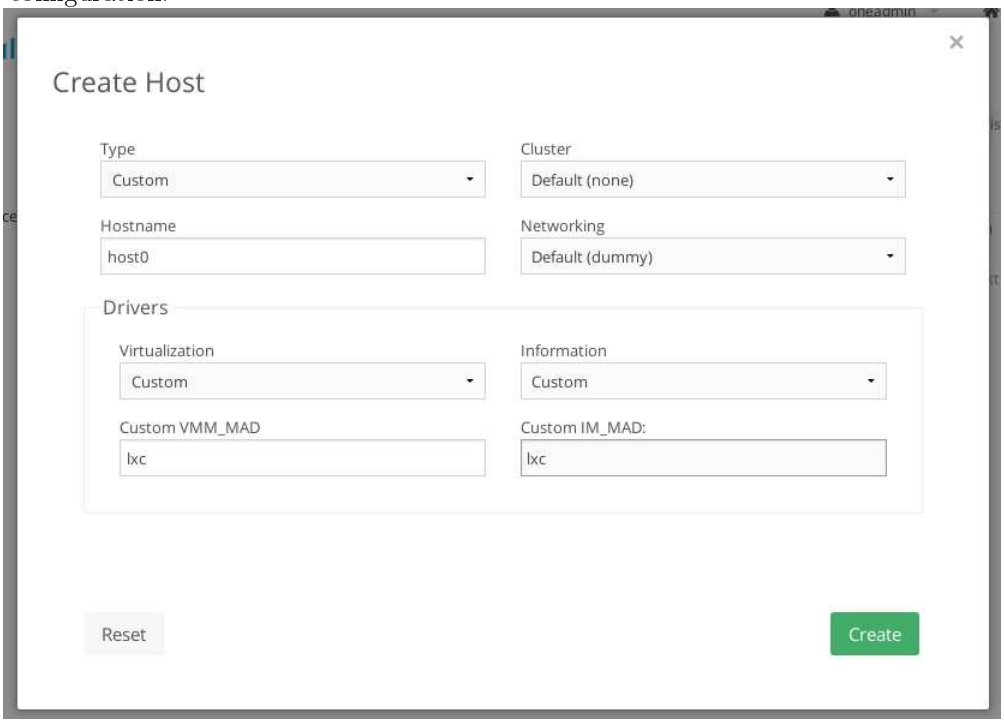
4.2. Upload the image previously created with LXC to OPENNEBULA using SUNSTONE

The raw image file will be located at:
/var/lib/lxc/name/rootdev
Being "name" the name of the container.

Warning
Until now, we are only supporting images inside the default datastore created by OPENNEBULA. Please, use this one.

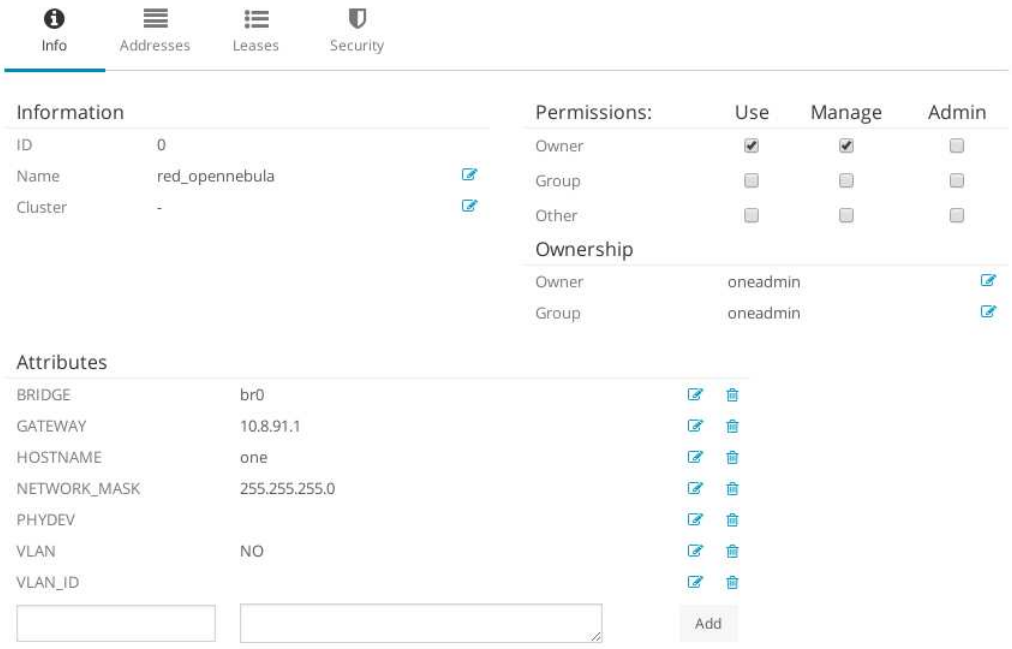
4.3. Add the host

You can add one using sunstone under **Infrastructure** –> **Hosts**. This is the configuration.



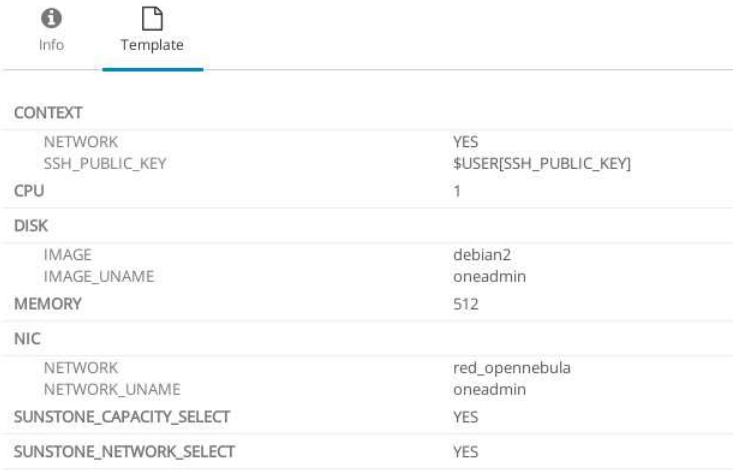
4.4. Create a virtual network

You can add one using sunstone under **Infrastructure** –> **Virtual Networks**. This is our configuration as an example.



4.5. Create a new template

You can add one using sunstone under **Virtual Resources** –> **Templates**. This is our configuration as an example.



Warning
Don't specify any VNC server, it makes no sense here right now and it can bring trouble.

4.6. Deploy a new virtual machine

Note
To check that containers really start, you can execute #lxc-monitor inside the host before deploying the virtual machine, or #lxc-ls –fancy after.

Note
The container's name after beeing deployed will be one-ID, beeing ID the ID of the virtual machine. This is not the hostname of the lxc-container, but for managing this containers locally, directly from lxc, we will use this name.

Note
If you want to locally attach a console to an lxc container, just run inside the host:

#lxc-attach -n name

Note
The container's take a little while to totally deploy after PROLOG status, 20 seconds aproximately.