OpenNebula 5.4 Installation

OpenNebula is an open-source management platform to build IaaS private, public and hybrid clouds. Installing a cloud from scratch could be a complex process, in the sense that many components and concepts are involved. The degree of familiarity with these concepts (system administration, infrastructure planning, virtualization management...) will determine the difficulty of the installation process.

➢ Front-end Installation

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

• Step 1. Add OpenNebula Repositories

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

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

Ubuntu 14.04

```
#echo "deb https://downloads.opennebula.org/repo/5.4/Ubuntu/14.04 stable
opennebula" > /etc/apt/sources.list.d/opennebula.list
```

Ubuntu 16.04

```
#echo "deb https://downloads.opennebula.org/repo/5.4/Ubuntu/16.04 stable
opennebula" > /etc/apt/sources.list.d/opennebula.list
```

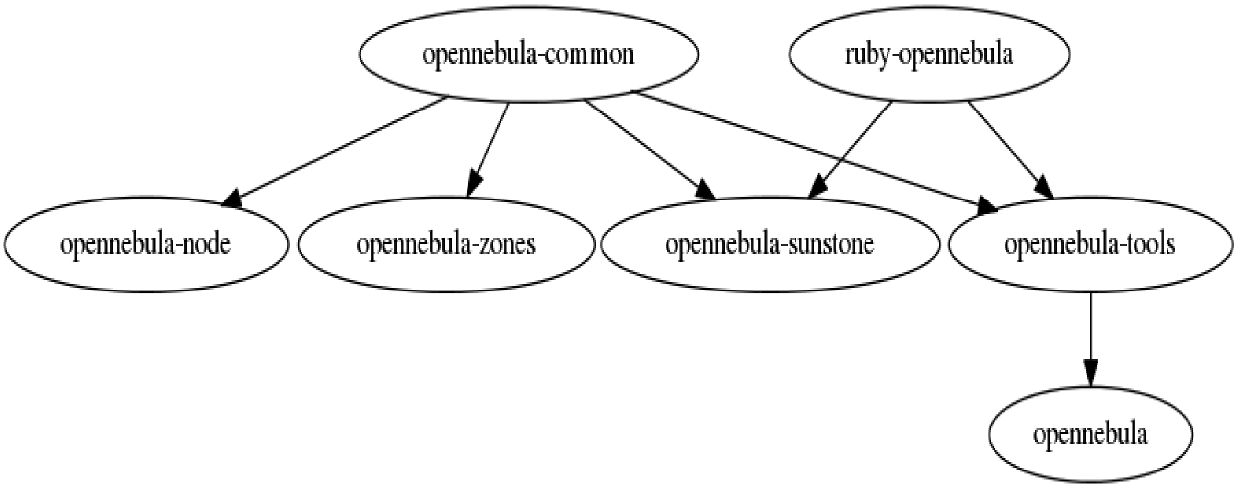
• Step 2: Installing the Software Debian/Ubuntu

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

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

Debian/Ubuntu Package Description

These are the packages available for these distributions:



- **opennebula-common**: Provides the user and common files.
- **ruby-opennebula**: Ruby API.
- **libopennebula-java**: Java API.
- **libopennebula-java-doc**: Java API Documentation.
- **opennebula-node**: Prepares a node as an opennebula-node.
- **opennebula-sunstone**: Sunstone (the GUI).
- **opennebula-tools**: Command Line interface.
- **opennebula-gate**: OneGate server that enables communication between VMs and OpenNebula.
- **opennebula-flow**: OneFlow manages services and elasticity.
- **opennebula**: OpenNebula Daemon.
- Step 3. Ruby Runtime Installation

Some OpenNebula components need Ruby libraries. OpenNebula provides a script that installs the required gems as well as some development libraries packages needed.

As root execute:

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

The previous script is prepared to detect common Linux distributions and install the required libraries. If it fails to find the packages needed in your system, manually install these packages:

- sqlite3 development library
- mysql client development library
- curl development library
- libxml2 and libxslt development libraries
- ruby development library
- gcc and g++
- make\

- **Step 3. Starting OpenNebula**

Log in as the oneadmin user follow these steps:

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

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

You are ready to start the OpenNebula daemons. You can use systemctl for Linux distributions which have adopted systemd:

```
# systemctl start opennebula
# systemctl start opennebula-sunstone
```

Or use service in older Linux systems:

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

- **Step 4. Verifying the Installation**

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

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

```
$oneuser show

USER 0 INFORMATION
ID              : 0
NAME            : oneadmin
GROUP           : oneadmin
PASSWORD        : 3bc15c8aae3e4124dd409035f32ea2fd6835efc9
AUTH_DRIVER     : core
ENABLED         : Yes

USER TEMPLATE
TOKEN_PASSWORD="ec21d27e2fe4f9ed08a396cbd47b08b8e0a4ca3c"

RESOURCE USAGE & QUOTAS
```

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

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

The OpenNebula logs are located in /var/log/one, you should have at least the files oned.log and sched.log, the core and scheduler logs. Check oned.log for any error messages, marked with [E].

Sunstone

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

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

Step 5. Node Installation

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

➢ KVM Node Installation

- **Step 1. Installing the Software on Debian/Ubuntu**

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

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

- **Step 2. Configure Passwordless SSH**

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

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

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

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

- **Step 3. Networking Configuration**

You may want to use the simplest network model that corresponds to the bridged drivers. For this driver, you will need to setup a linux bridge and include a physical device to the bridge. Later on ,when defining the network in OpenNebula,

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

you will specify the name of this bridge and OpenNebula will know that it should connect the VM to this bridge, thus giving it connectivity with the physical network device connected to the bridge. For example, a typical host with two physical networks, one for public IP addresses (attached to an eth0 NIC for example) and the other for private virtual LANs (NIC eth1 for example) should have two bridges:

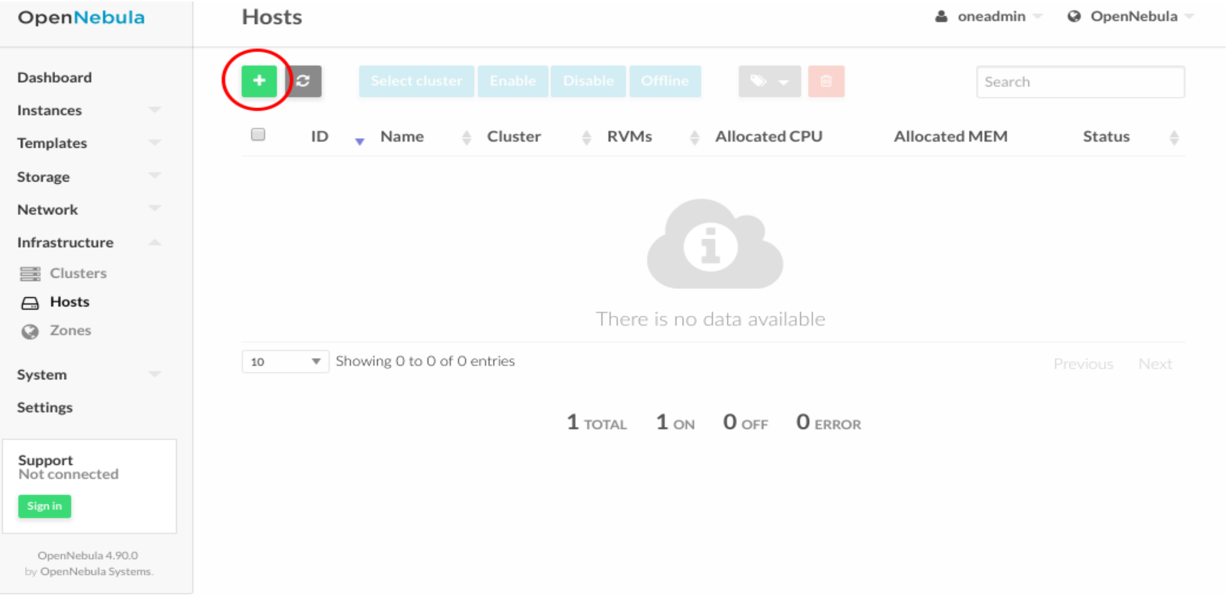
```
$brctl show
bridge name bridge id         STP enabled interfaces
br0         8000.001e682f02ac no          eth0
br1         8000.001e682f02ad no          eth1
```

- **Step 4. Adding a Host to OpenNebula**

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

Adding a Host through Sunstone

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

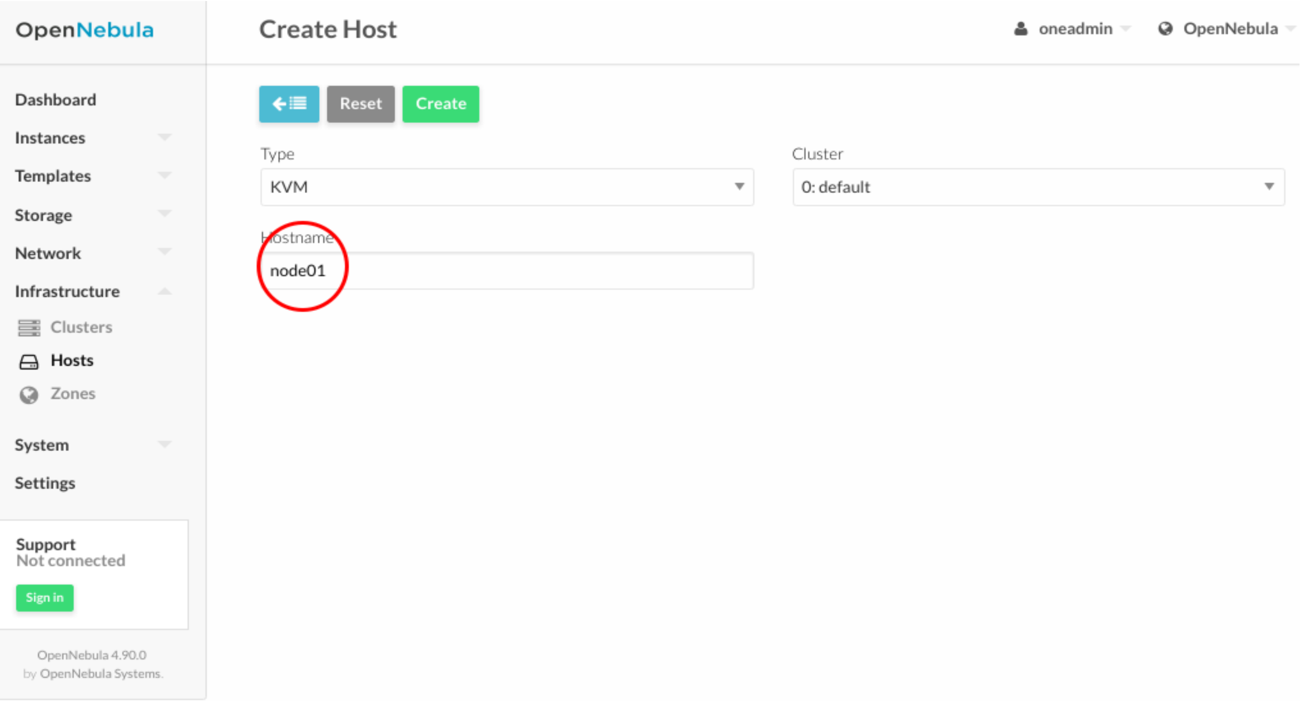


There is no data available

Showing 0 to 0 of 0 entries

1 TOTAL 1 ON 0 OFF 0 ERROR

Creating Host:



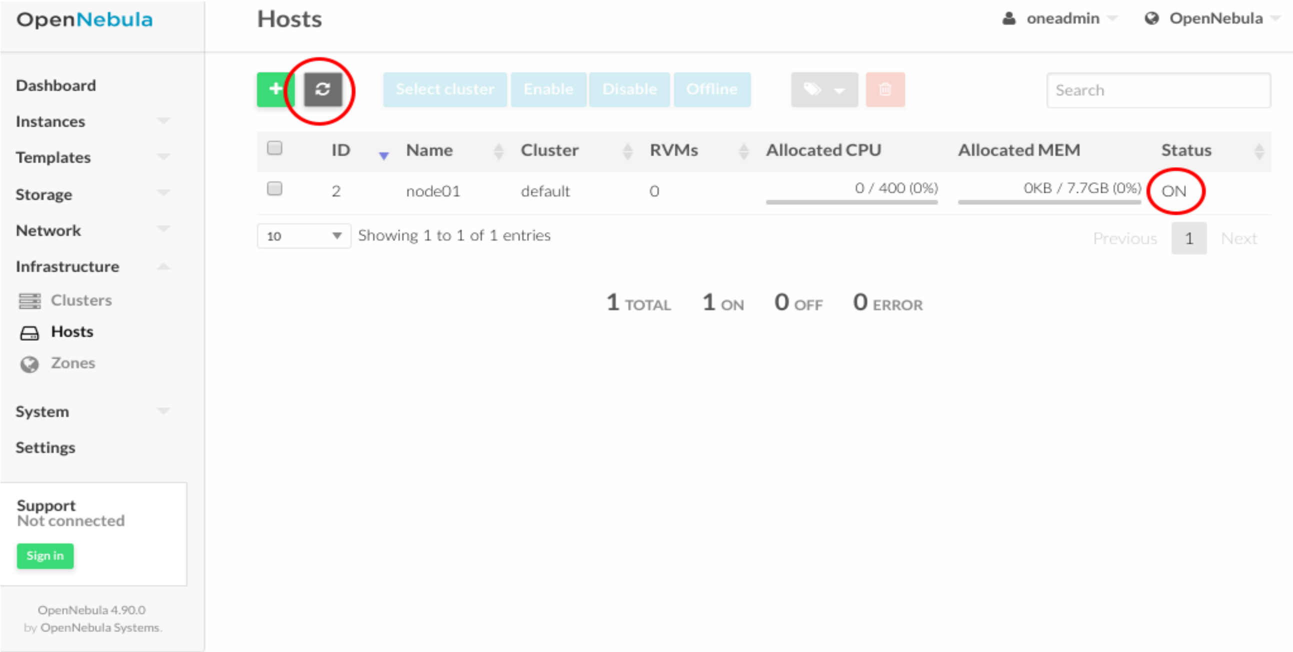
Type: KVM

Cluster: 0: default

Hostname: node01

Step 5. Check that the Hosts are on

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



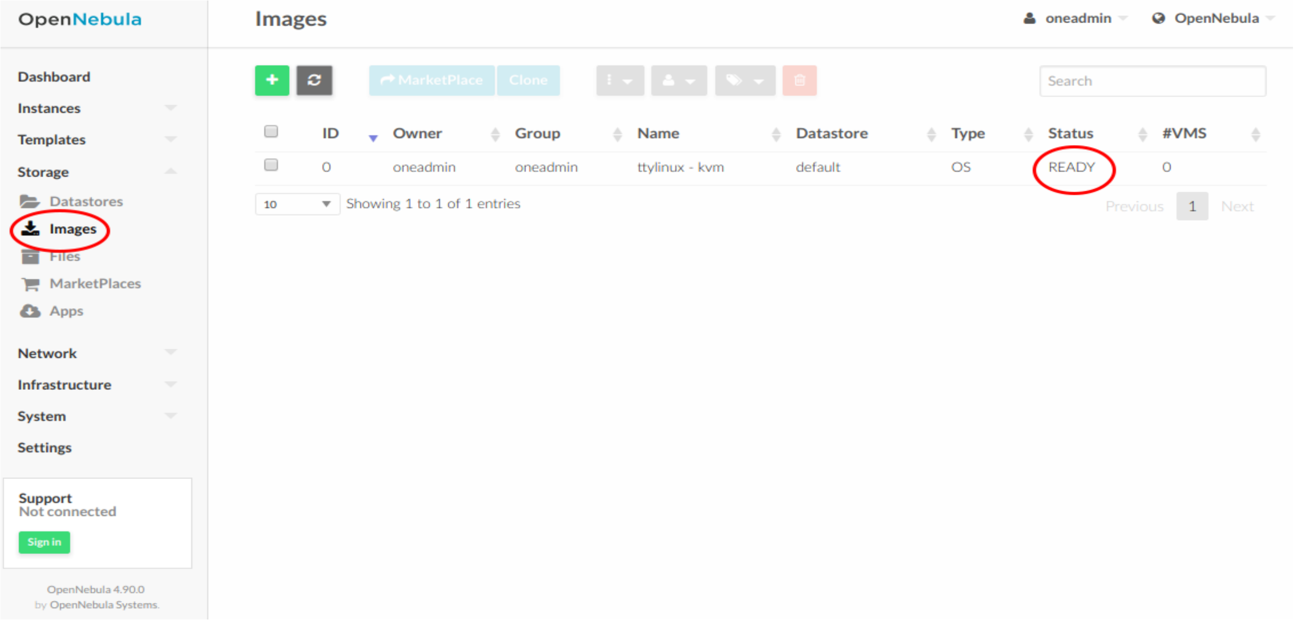
☐	ID	Name	Cluster	RVMs	Allocated CPU	Allocated MEM	Status
☐	2	node01	default	0	0 / 400 (0%)	0KB / 7.7GB (0%)	ON

Showing 1 to 1 of 1 entries

1 TOTAL 1 ON 0 OFF 0 ERROR

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

- **Step 6: Navigate now to the Storage -> Images tab and refresh until the Status of the Image switches to READY.**

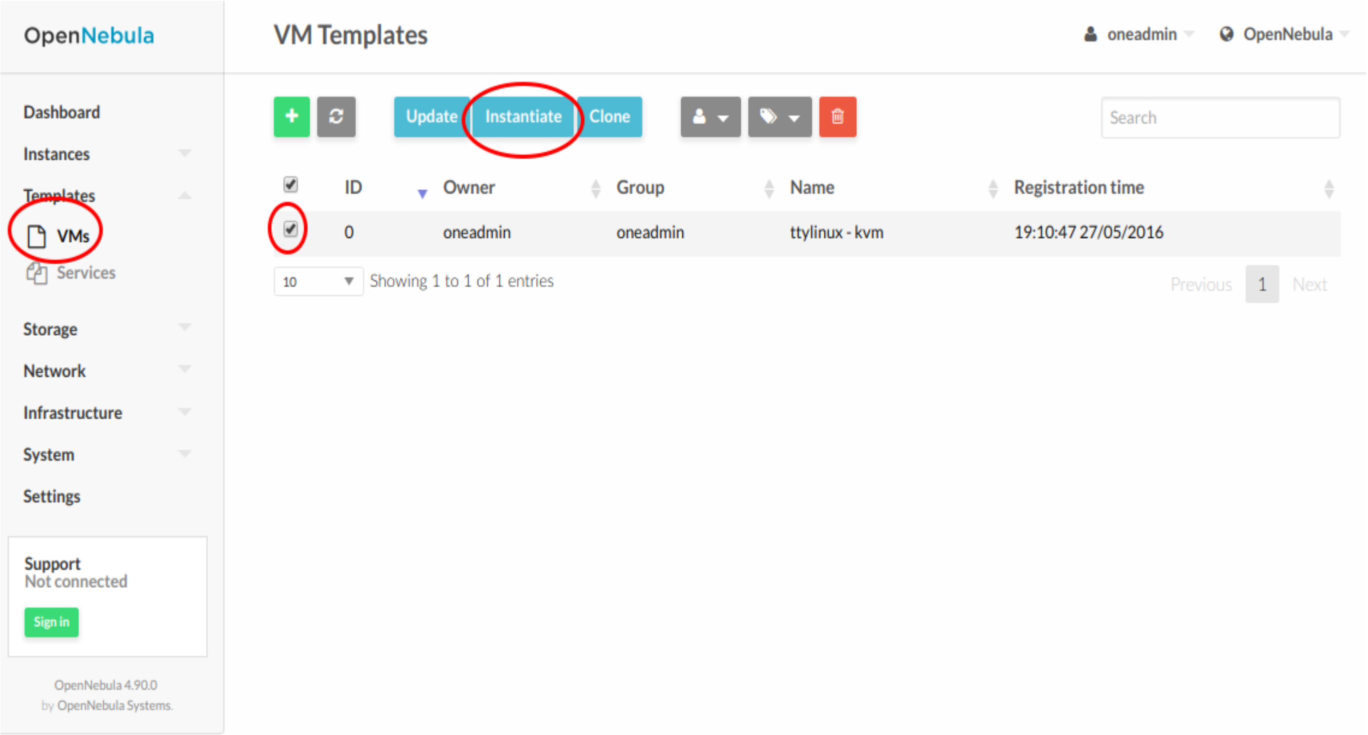


□	ID	Owner	Group	Name	Datastore	Type	Status	#VMS
□	0	oneadmin	oneadmin	ttylinux - kvm	default	OS	READY	0

10 Showing 1 to 1 of 1 entries

- **Step 7. Instantiate a VM**

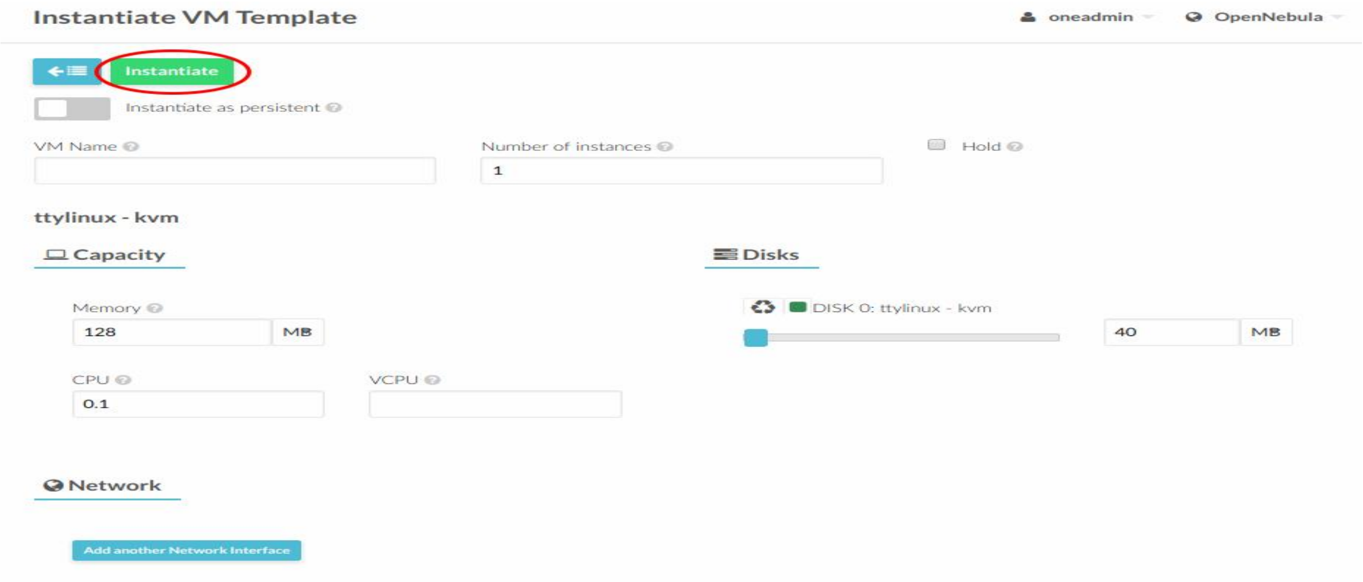
Navigate to Templates -> VMs, then select the ttylinux - kvm template that has been created and click on the “Instantiate”.



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

10 Showing 1 to 1 of 1 entries

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



VM Name:

Number of instances: 1

□ Hold

ttylinux - kvm

Memory: 128 MB

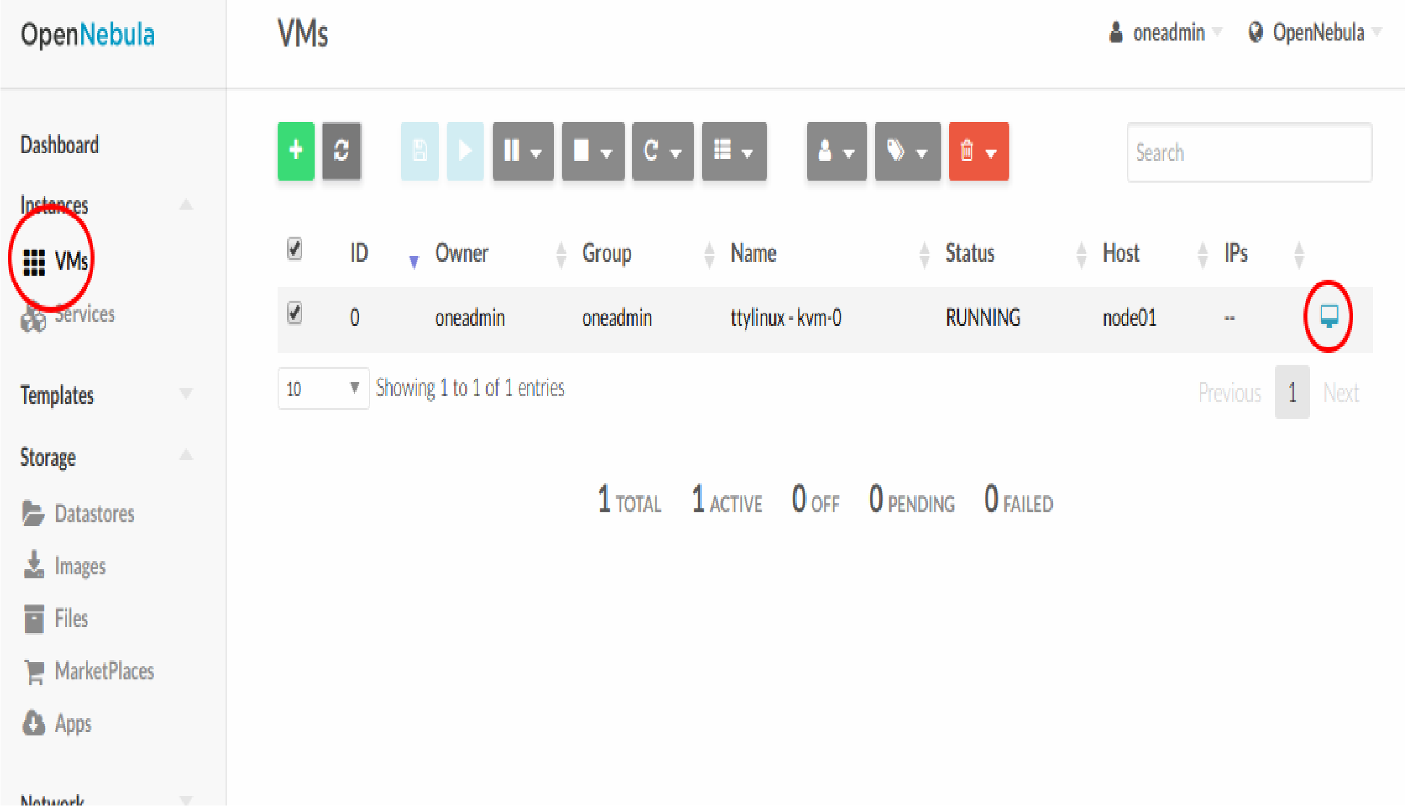
CPU: 0.1

VCPU:

DISK 0: ttylinux - kvm: 40 MB

- **Step 8. Test VM**

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



☑	ID	Owner	Group	Name	Status	Host	IPs
☑	0	oneadmin	oneadmin	ttylinux - kvm-0	RUNNING	node01	--

10 Showing 1 to 1 of 1 entries

1 TOTAL 1 ACTIVE 0 OFF 0 PENDING 0 FAILED

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