Xen Users Guide

5.5 Edition

Xen Users Guide :

5.5 Edition

Published May 08 2012

Table of Contents

List of Tables

Preface

The Xen Roll installs and configures Virtual Machines (VMs) on Rocks Clusters.

A physical frontend can configure VMs on client nodes (*VM container* appliances). A VM container is a physical machine that houses and runs VMs.

The Xen Roll also supports building virtual clusters. The frontend can be installed as a *VM server* appliance and the client nodes can be installed as VM containers. Then a virtual frontend can be installed on the VM server while virtual compute nodes can be installed on the VM containers. All network traffic is encapsulated within a unique VLAN, that is, each virtual cluster has its own VLAN.

Please visit the xen site[1] to learn more about their release and the individual software components.

Notes

1. http://xen.org

Chapter 1. Overview

Table 1-1. Summary

Name	xen
Version	5.5
Maintained By	Rocks Group
Architecture	i386, x86_64
Compatible with Rocks®	5.5

The xen roll has the following requirements of other rolls. Compatability with all known rolls is assured, and all known conflicts are listed. There is no assurance of compatiblity with third-party rolls.

Table 1-2. Compatibility

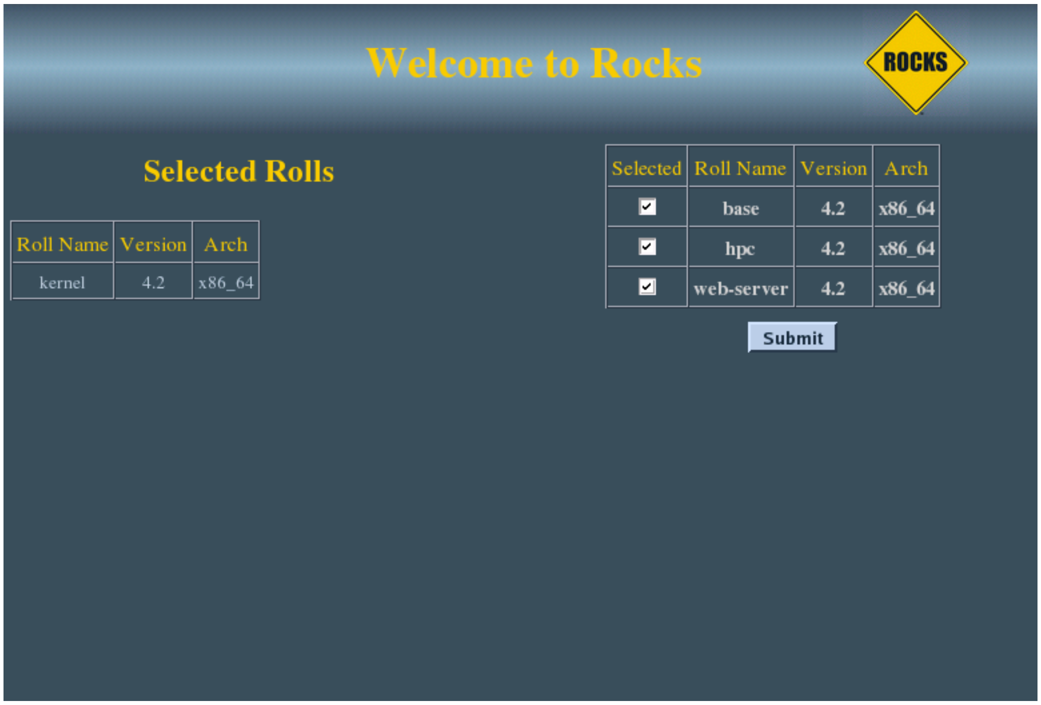
Requires	Conflicts
Base	
Kernel	
OS	

This roll has been released independent of the corresponding Rocks® release. It therefore requires the complete **OS** roll and will not function correctly if using only the **Jumbo** or incomplete set of **OS** CDROMs.

Chapter 2. Installing

2.1. On a New Server

The xen roll should be installed during the initial installation of your server (or cluster). This procedure is documented in section 1.2 of the Rocks® usersguide. You should select the xen roll from the list of available rolls when you see a screen that is similar to the one below.



Welcome to Rocks

Selected Rolls

Roll Name	Version	Arch
kernel	4.2	x86_64

Selected	Roll Name	Version	Arch
☑	base	4.2	x86_64
☑	hpc	4.2	x86_64
☑	web-server	4.2	x86_64

Chapter 3. Using the Xen Roll

3.1. Installing a VM Server

A VM Server is machine that can house virtual frontend appliances. It is required to build a VM Server if you wish to build virtual clusters.

Building a VM Server is just like building a traditional frontend, except that you *must supply the Xen Roll during the frontend installation*. Follow the procedure Install and Configure Your Frontend[1] and be sure to supply the Xen Roll.

After you build the VM Server, you'll need to install VM Containers (see the next section).

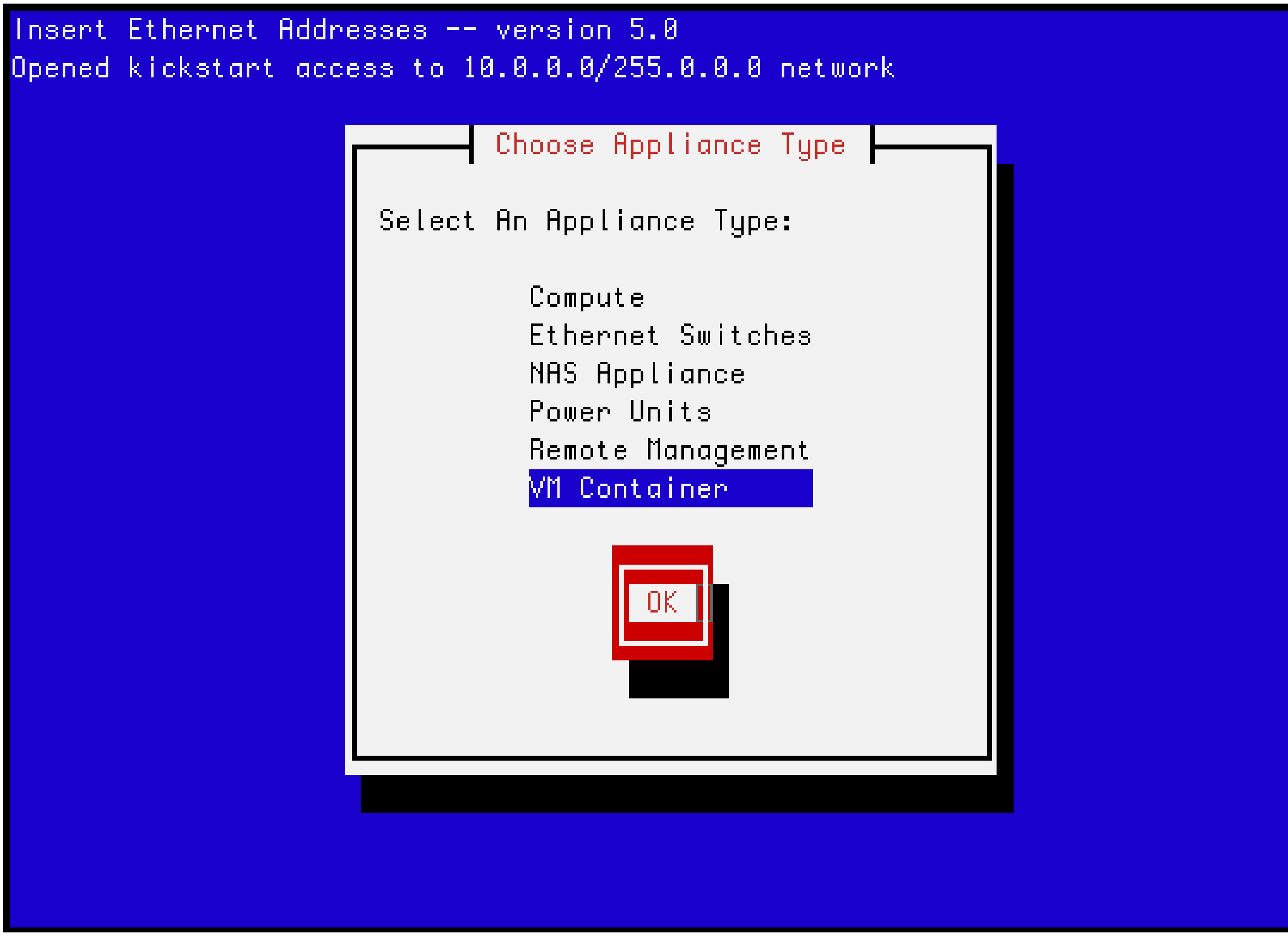
3.2. Installing VM Containers

You will need to install a physical machine that will act as the "container" for your VMs. This method is very similar to the method for installing compute nodes.

On the frontend, execute:

```
# insert-ethers
```

You will see a screen that looks like:



```
Insert Ethernet Addresses -- version 5.0
Opened kickstart access to 10.0.0.0/255.0.0.0 network
```

Choose Appliance Type

Select An Appliance Type:

Select the ’VM Container’ appliance, then hit ’OK’.

Now PXE boot the physical machine that will be your VM container. Just like a compute node, the VM container will be recognized by insert-ethers and installed. The default name of the node will be `vm-container-X-Y`.

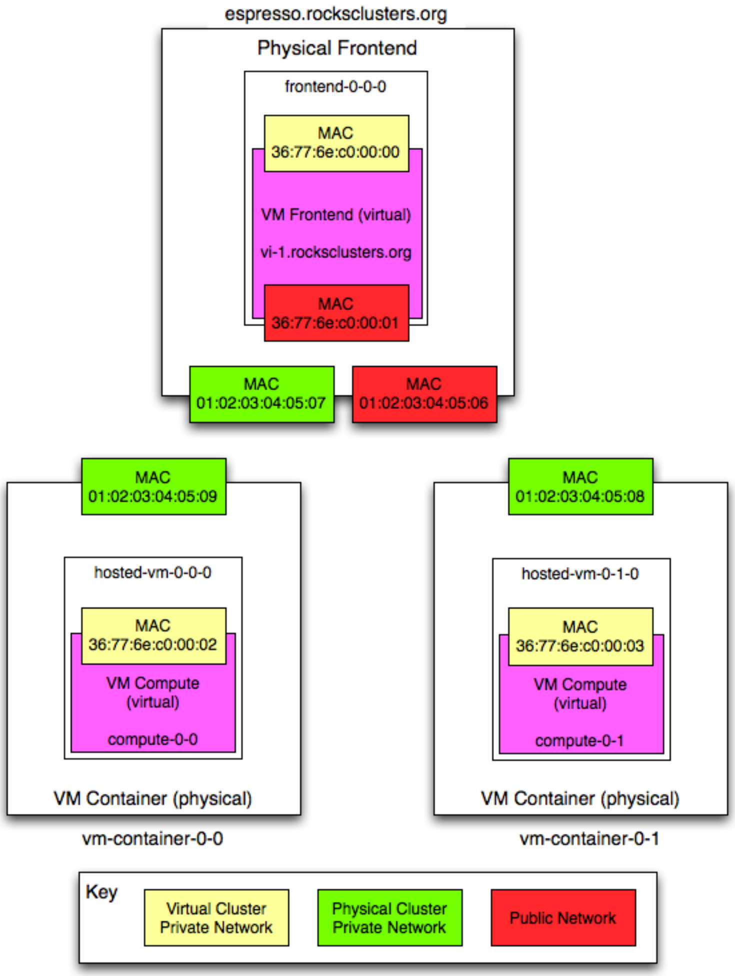
You can install as many VM containers as you like.

3.3. Installing Virtual Clusters

3.3.1. Provisioning a Virtual Cluster

After you install a VM Server and at least one VM Container, you are ready to provision a virtual cluster.

We’ll use the following illustration as a guide to help keep track of the names of the physical machines and the virtual machines.



In the above picture, "espresso.rocksclusters.org" is a physical machine. Also, "vm-container-0-0" and "vm-container-0-1" are physical machines that were kickstarted by "espresso". The machine "frontend-0-0-0" is a virtual machine that is hosted by "espresso". The machines "hosted-vm-0-0-0" and "hosted-vm-0-1-0" are VMs that are associated with "frontend-0-0-0" (they are all in the same VLAN).

Depending on your perspective, the virtual machines have different names. Dom0 is a physical machine that hosts (multiple) virtual systems. DomU are guests and generally refer to names by usual convention. The equivalence is:

Table 3-1. Virtual Machine Names

Host	Dom0 Name (physical)	DomU Name (virtual)
37:77:6e:c0:00:00	frontend-0-0-0	vi-1.rocksclusters.org
37:77:6e:c0:00:01	hosted-vm-0-0-0	compute-0-0
37:77:6e:c0:00:02	hosted-vm-0-1-0	compute-0-1

An important point is that the only common thing between the physical side and the virtual side is the MAC address (in yellow). We will use the MAC address of a virtual machine to control it (e.g., to initially power it on).

The names in the virtual cluster look like the names in a traditional cluster -- the frontend is named "vi-1.rocksclusters.org" and its compute nodes are named "compute-0-0" and "compute-0-1". If you login to "vi-1.rocksclusters.org", you would be hard pressed to tell the difference between this virtual cluster and a traditional physical cluster.

You must select your own IP address for your virtual frontend. The IP address "137.110.119.118" is managed by UCSD and should not be used by you.

They are only used here to show you a concrete example.

First, we'll add a virtual cluster to the VM Server's database. In this example, we'll add a frontend with the IP of "137.110.119.118" and we'll associate 2 compute nodes with it:

```
# rocks add cluster ip="137.110.119.118" num-computes=2
```

The above command will take some time and then output something similar to:

```
created frontend VM named: frontend-0-0-0
 created compute VM named: hosted-vm-0-0-0
 created compute VM named: hosted-vm-0-1-0
```

The command adds entries to the database for the above nodes and establishes a VLAN that will be used for the private network (eth0 inside the VM).

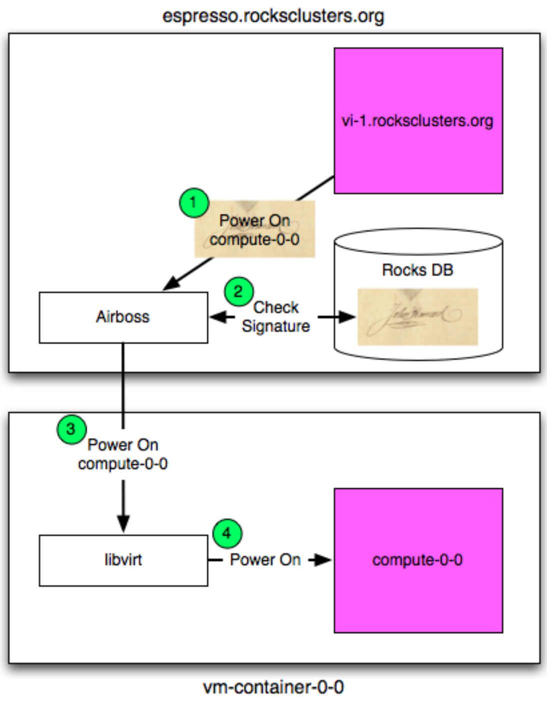
Info about all the defined clusters on the VM Server (including the physical cluster) can be obtained with the command: `rocks list cluster`:

```
# rocks list cluster
FRONTEND                    CLIENT NODES     TYPE
espresso.rocksclusters.org: ---------------- physical
:                           vm-container-0-0 physical
:                           vm-container-0-1 physical
frontend-0-0-0-public:      ---------------- VM
:                           hosted-vm-0-0-0  VM
:                           hosted-vm-0-1-0  VM
```

3.3.2. The Airboss

In Rocks, we've developed a service known as the "Airboss" that resides on the physical frontend (in Dom0) and it allows non-root users to control their VMs. The motivation for this service is that libvirt (a virtualization API written by RedHat that can control several different virtualization implementations) assumes "root" access to control and monitor VMs.

The Airboss in Rocks is a small service that uses digitally signed messages to give non-root users access to their virtual cluster (and only their virtual cluster). The Airboss relies upon public/private key pairs to validate messages. The administrator of the physical hosting cluster must issue a single command to associate a public key with a particular virtual cluster. At that point, the full process of booting and installing a virtual cluster can be controlled by the (authorized) non-root user.



In the above picture, a user that is logged in to vi-1.rocksclusters.org wants to power on compute-0-0 (one of the VMs associated with the virtual cluster). The user executes the "power on" command. The command creates a "power on" message, signs it with a private key, then sends it to the Airboss that is running on espresso.rocksclusters.org. The Airboss verifies the message signature. If the signature is valid, then the Airboss instructs libvirt on vm-container-0-0 to start ("power on") compute-0-0.

3.3.3. Creating an RSA Key Pair

Before we can install a VM, we must create an RSA key pair. These keys will be used to authenticate Airboss commands. To create a key pair, execute:

```
# rocks create keys key=private.key
```

The above command will ask for a pass phrase for the private key. If you would like a "passphraseless" private key, execute:

```
# rocks create keys key=private.key passphrase=no
```

The above command will place your private key into the file private.key and it will output the public key for your private key:

```
# rocks create keys key=private.key
Generating RSA private key, 1024 bit long modulus
............++++++
.......++++++
e is 65537 (0x10001)
Enter pass phrase for private.key:
Verifying - Enter pass phrase for private.key:
Enter pass phrase for private.key:
writing RSA key
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDMoCPmR/Kev64znRBxvtsniXIF
dyQMxR/bBFKNDmvmzPuPUim5jmD3TLilnH75/KidtJCwlb+Lhr5Cs6/9sRzX6rX2
ExVUZsgo4A+O+XMk8KeowO/c2rPc+YdXaBir3Aesm/MCfCZaidZae8QLmVKW7Va5
qErl9gyhhR7uDX+hgwIDAQAB
-----END PUBLIC KEY-----
```

Now save the public key to file, that is, copy the above public key:

```
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDMoCPmR/Kev64znRBxvtsniXIF
dyQMxR/bBFKNDmvmzPuPUim5jmD3TLilnH75/KidtJCwlb+Lhr5Cs6/9sRzX6rX2
ExVUZsgo4A+O+XMk8KeowO/c2rPc+YdXaBir3Aesm/MCfCZaidZae8QLmVKW7Va5
qErl9gyhhR7uDX+hgwIDAQAB
-----END PUBLIC KEY-----
```

And save your public key into a file (e.g., $HOME/public.key).

We now want to associate your public key with the virtual cluster you provisioned. This will allow you to use your private key to send authenticated commands to control your cluster. To associate your public key with your virtual cluster, execute:

```
# rocks add host key frontend-0-0-0 key=public.key
```

We can see the relationship by executing:

```
# rocks list host key
HOST            ID PUBLIC KEY
frontend-0-0-0: 7  -----BEGIN PUBLIC KEY-----
:                  MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDMoCPmR/Kev64znRBxvtsniXIF
:                  dyQMxR/bBFKNDmvmzPuPUim5jmD3TLilnH75/KidtJCwlb+Lhr5Cs6/9sRzX6rX2
:                  ExVUZsgo4A+O+XMk8KeowO/c2rPc+YdXaBir3Aesm/MCfCZaidZae8QLmVKW7Va5
:                  qErl9gyhhR7uDX+hgwIDAQAB
:                  -----END PUBLIC KEY-----
:                  ----------------------------------------------------------------
```

We see that the public key is associated with "frontend-0-0-0" (the name of the VM in Dom0).

3.3.4. Installing a VM Frontend

Now, we'll want to install the virtual frontend. First, login to the physical frontend (e.g., espresso). To start the VM frontend install, we'll need to power on and install the VM frontend:

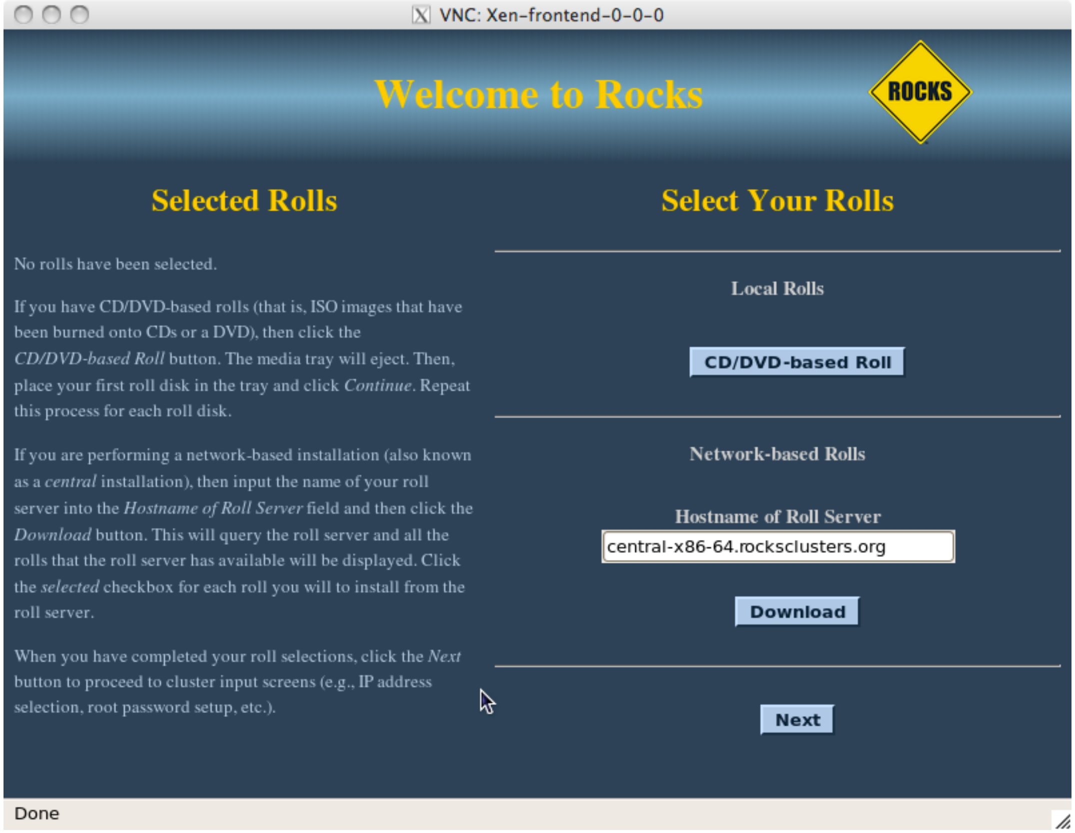
```
# rocks set host power frontend-0-0-0 action=install key=private.key
```

The action of "install" ensures that the VM will be put into install mode, then it will be powered on.

Then, to connect to the VM's console, execute:

```
# rocks open host console frontend-0-0-0 key=private.key
```

Soon you will see the familiar frontend installation screen:



No rolls have been selected.

If you have CD/DVD-based rolls (that is, ISO images that have been burned onto CDs or a DVD), then click the *CD/DVD-based Roll* button. The media tray will eject. Then, place your first roll disk in the tray and click *Continue*. Repeat this process for each roll disk.

If you are performing a network-based installation (also known as a *central* installation), then input the name of your roll server into the *Hostname of Roll Server* field and then click the *Download* button. This will query the roll server and all the rolls that the roll server has available will be displayed. Click the *selected* checkbox for each roll you will to install from the roll server.

When you have completed your roll selections, click the *Next* button to proceed to cluster input screens (e.g., IP address selection, root password setup, etc.).

Hostname of Roll Server: central-x86-64.rocksclusters.org

In the "Hostname of Roll Server" field, insert the FQDN of your VM Server (the name of the physical machine that is hosting the VM frontend). Then click "Download".

From here, you want to follow the standard procedure for bringing up a frontend[2] starting at Step 8.

After the VM frontend installs, it will reboot. After it reboots, login and then we'll begin installing VM compute nodes.

3.3.5. Installing VM Compute Nodes

Login to the VM frontend (the virtual machine named "vi-1.rocksclusters.org" in the example picture at the top of this page), and execute:

```
# insert-ethers
```

Select "Compute" as the appliance type.

In another terminal session on vi-1.rocksclusters.org, we'll need to set up the environment to send commands to the Airboss on the physical frontend. We'll do this by putting the RSA private key that we created in section Creating an RSA Key Pair (e.g., private.key) on vi-1.rocksclusters.org.

Prior to sending commands to the Airboss, we need to establish a ssh tunnel between the virtual frontend (e.g., vi-1) and the physical frontend (e.g., espresso, where the Airboss runs). This tunnel is used to securely pass Airboss messages. On the virtual frontend (e.g., vi-1), execute:

```
# ssh -f -N -L 8677:localhost:8677 espresso.rocksclusters.org
```

Now we can securely send messages to the Airboss.

Now, we're ready to install compute nodes. But, there's a problem - when we first login to vi-1.rocksclusters.org, the only machine we know about is ourself (i.e., vi-1.rocksclusters.org). There are no other nodes in the virtual frontend's database. But the physical machine knows about the MAC addresses of the virtual compute nodes (e.g., hosted-vm-0-0-0 and hosted-vm-0-1-0) that are associated with this virtual cluster. The good news is, we can ask the Airboss on the physical frontend for a list of MAC addresses that are assigned to our virtual cluster:

```
# rocks list host macs vi-1.rocksclusters.org key=private.key
```

Which outputs:

```
MACS IN CLUSTER
36:77:6e:c0:00:02
36:77:6e:c0:00:00
36:77:6e:c0:00:03
```

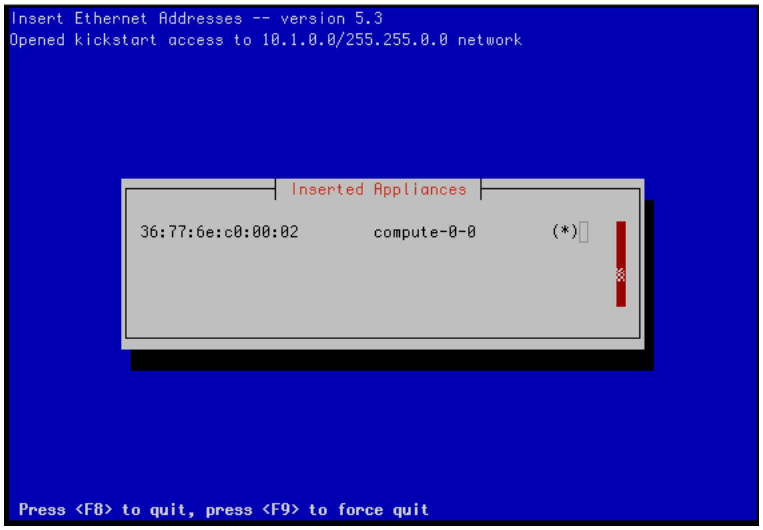
The MAC address 36:77:6e:c0:00:00 is ourself (the VM frontend) and the other two MACs (36:77:6e:c0:00:02 and 36:77:6e:c0:00:03) are the VM compute nodes that are associated with our VM frontend.

We can use the MAC address of the VM compute nodes to power up and install our compute nodes:

```
# rocks set host power 36:77:6e:c0:00:02 key=private.key action=install
```

The action of "install" ensures that the VM will be put into install mode, then it will be powered on.

Soon, you should see insert-ethers discover the VM compute node:



```
Insert Ethernet Addresses -- version 5.3
Opened kickstart access to 10.1.0.0/255.255.0.0 network

                 Inserted Appliances

  36:77:6e:c0:00:02        compute-0-0        (*)

Press <F8> to quit, press <F9> to force quit
```

After the virtual compute node is discovered by insert-ethers, we can open a console to the node by executing:

```
# rocks open host console compute-0-0 key=private.key
```

Lastly, to power off a virtual compute node (e.g., compute-0-0), execute:

```
# rocks set host power compute-0-0 key=private.key action=off
```

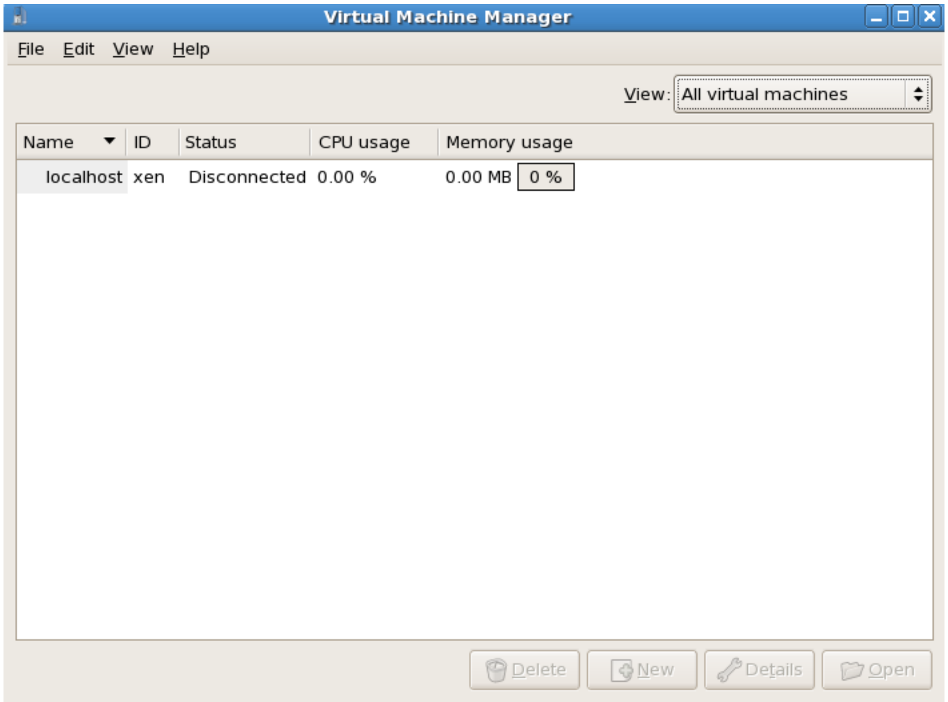
3.3.6. Using RedHat’s Virt-Manager (Root Users Only)

Virt-manager is a program produced by RedHat that is a desktop user interface for managing virtual machines. This section describes how to use some of virt-manager’s features to control and monitor VMs on a Rocks cluster.

To interact with the VM frontend’s console, on the physical frontend, you need to start "virt-manager":

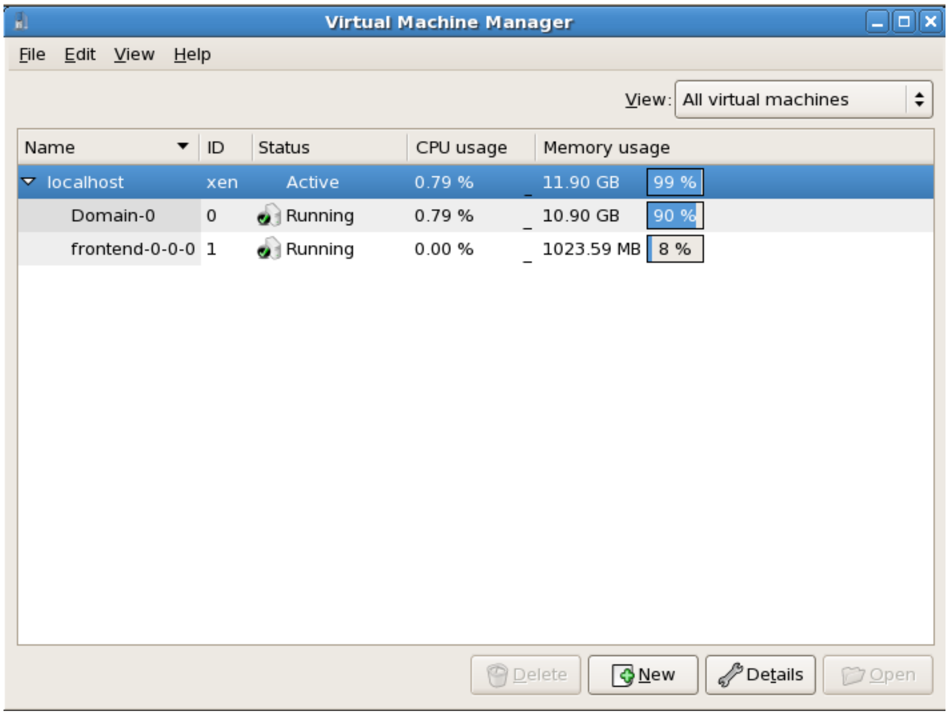
```
# virt-manager
```

This will display a screen similar to:



Name	ID	Status	CPU usage	Memory usage
localhost	xen	Disconnected	0.00 %	0.00 MB 0 %

Double click on the "localhost" entry and then you'll see:

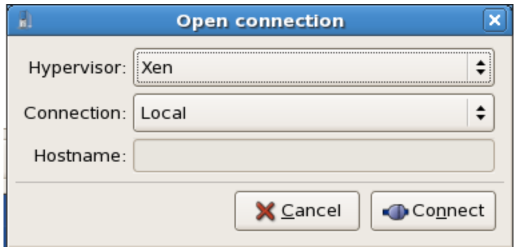


Name	ID	Status	CPU usage	Memory usage
localhost	xen	Active	0.79 %	11.90 GB 99 %
Domain-0	0	Running	0.79 %	10.90 GB 90 %
frontend-0-0-0	1	Running	0.00 %	1023.59 MB 8 %

To bring the up the console for the VM frontend, double click on "frontend-0-0-0".

Now we'll describe how to connect to the console for the virtual compute node "compute-0-0". In the example configuration described at the top of this page, the VM "compute-0-0" is hosted on the physical machine named "vm-container-0-0" so we'll need to tell "virt-manager" to open a connection to "vm-container-0-0".

Inside "virt-manager", click on "File" then "Open connection...". This brings up a window that looks like:

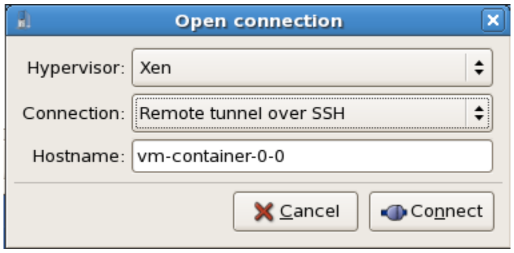


Hypervisor: Xen

Connection: Local

Hostname:

Now change the "Connection:" field to "Remote tunnel over SSH" and enter "vm-container-0-0" for the "Hostname:" field:



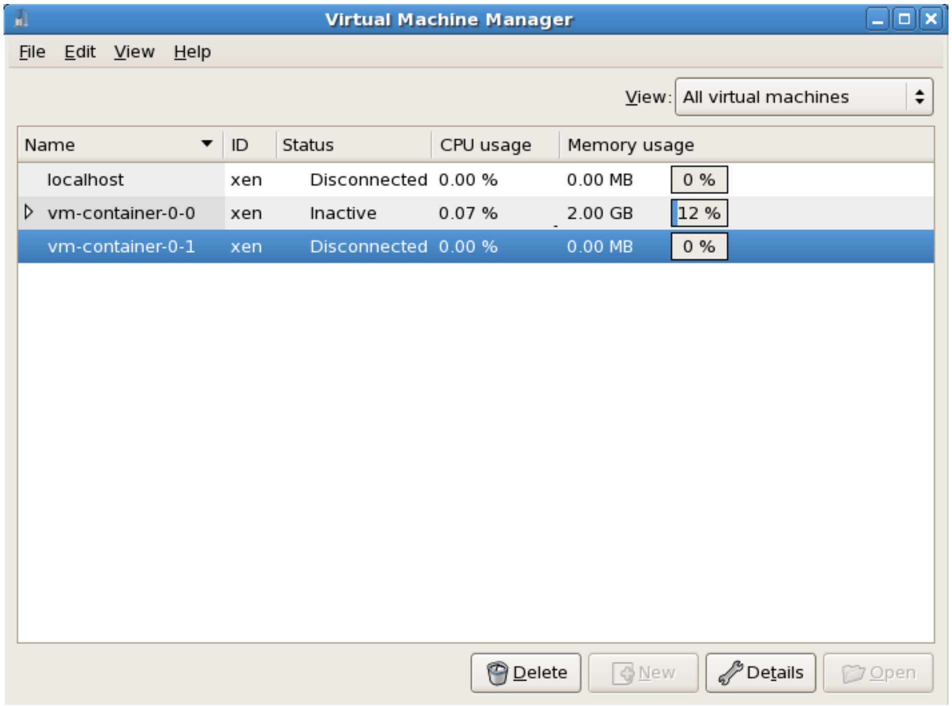
Hypervisor: Xen

Connection: Remote tunnel over SSH

Hostname: vm-container-0-0

Then click "Connect".

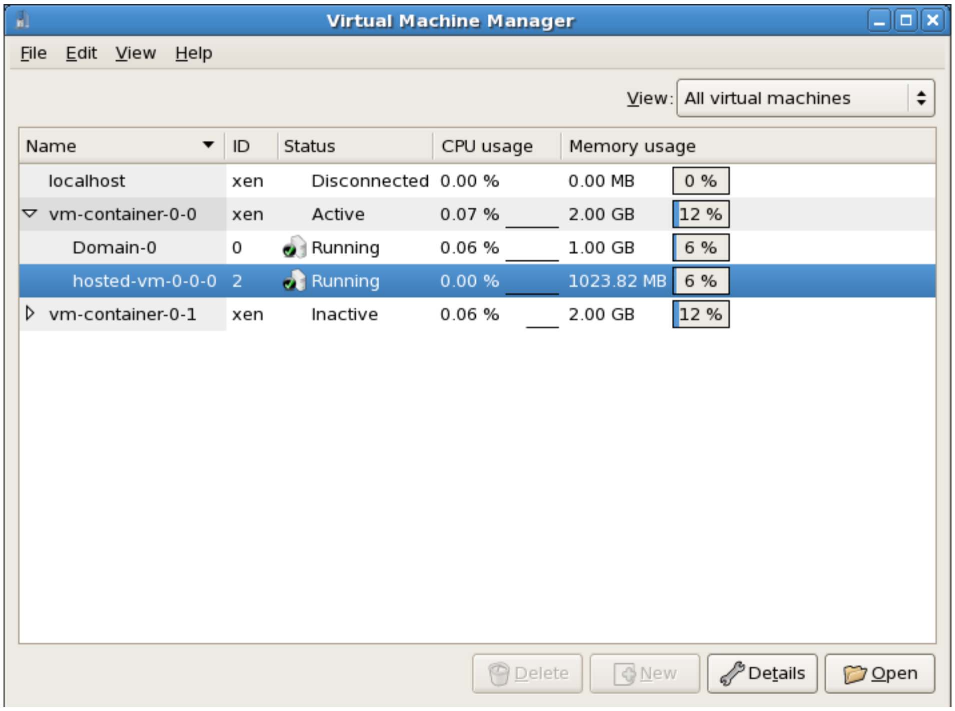
In the "virt-manager" window, you should see something similar to:



View: All virtual machines

Name	ID	Status	CPU usage	Memory usage	
localhost	xen	Disconnected	0.00 %	0.00 MB	0 %
vm-container-0-0	xen	Inactive	0.07 %	2.00 GB	12 %
vm-container-0-1	xen	Disconnected	0.00 %	0.00 MB	0 %

Double click on "vm-container-0-0" and then you'll see:

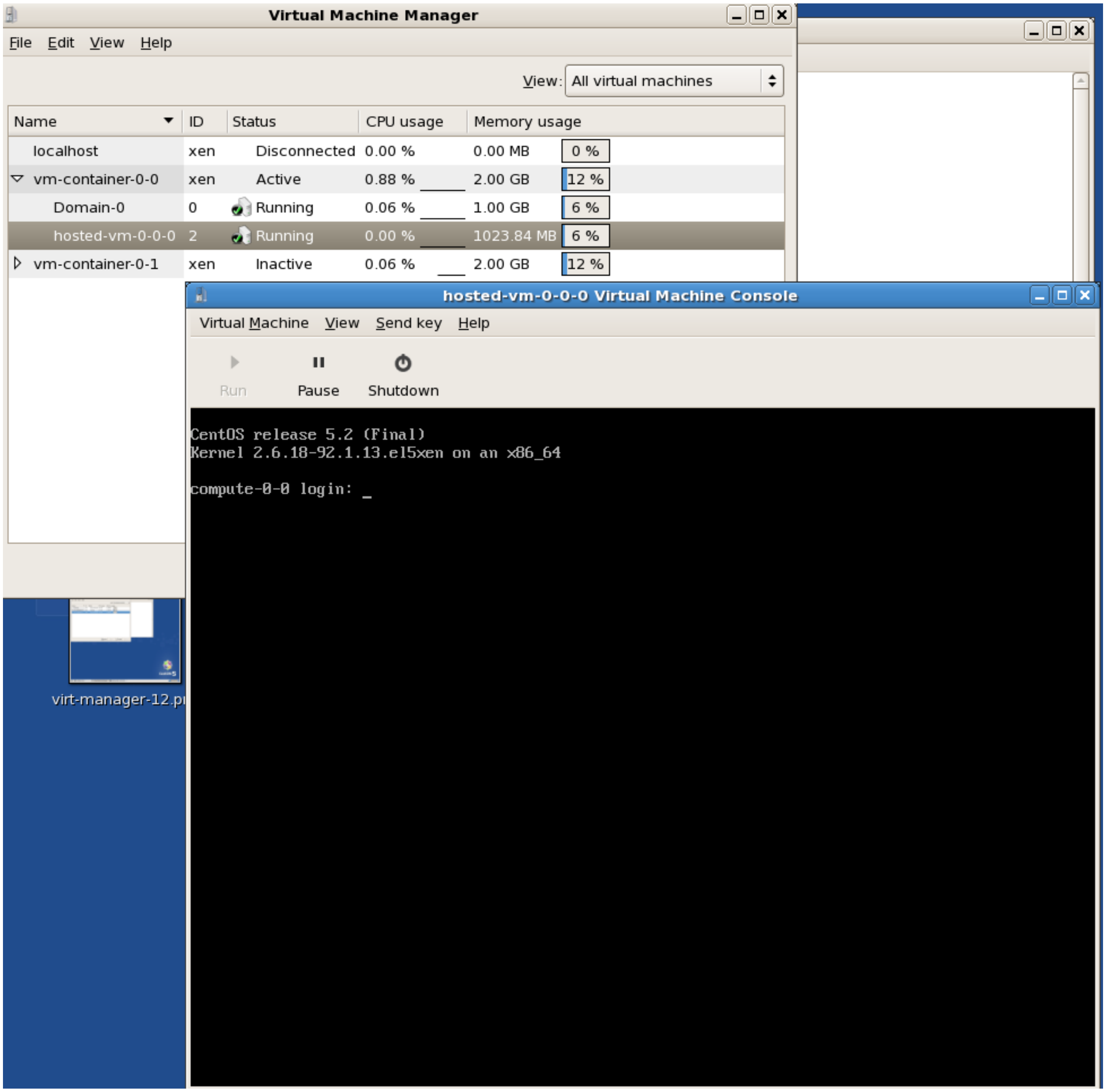


View: All virtual machines

Name	ID	Status	CPU usage	Memory usage
localhost	xen	Disconnected	0.00 %	0.00 MB 0 %
vm-container-0-0	xen	Active	0.07 %	2.00 GB 12 %
Domain-0	0	Running	0.06 %	1.00 GB 6 %
hosted-vm-0-0-0	2	Running	0.00 %	1023.82 MB 6 %
vm-container-0-1	xen	Inactive	0.06 %	2.00 GB 12 %

Now to connect to the compute node's console, double click on "hosted-vm-0-0-0". Recall that from the perspective of the physical frontend (the VM Server), "hosted-vm-0-0-0" is the name for the VM "compute-0-0" (again, see the figure at the top of this page).

You should now see the console for compute-0-0:


Virtual Machine Manager
File Edit View Help
View: All virtual machines
hosted-vm-0-0-0 Virtual Machine Console
Virtual Machine View Send key Help
Run
Pause
Shutdown
virt-manager-12.p

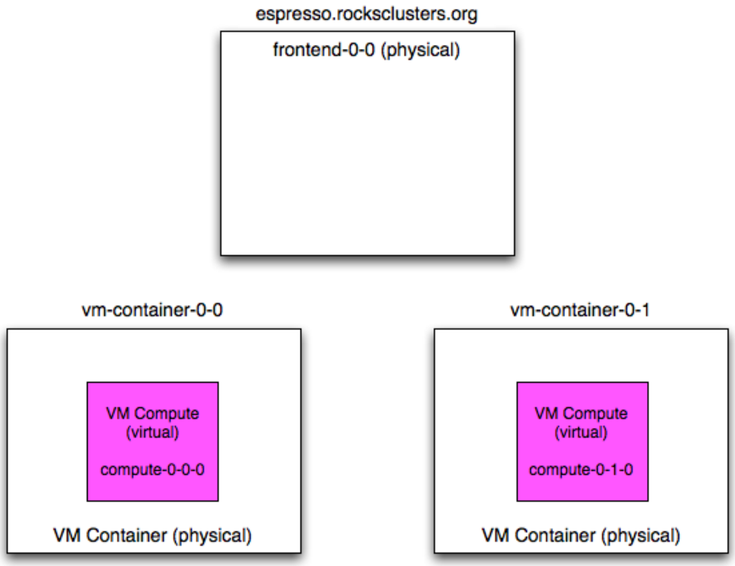
Name	ID	Status	CPU usage	Memory usage	
localhost	xen	Disconnected	0.00 %	0.00 MB	0 %
vm-container-0-0	xen	Active	0.88 %	2.00 GB	12 %
Domain-0	0	Running	0.06 %	1.00 GB	6 %
hosted-vm-0-0-0	2	Running	0.00 %	1023.84 MB	6 %
vm-container-0-1	xen	Inactive	0.06 %	2.00 GB	12 %

```
CentOS release 5.2 (Final)
Kernel 2.6.18-92.1.13.el5xen on an x86_64

compute-0-0 login: _
```

3.4. Physical Frontend with Virtual Compute Nodes

In this scenario, the frontend is a physical machine (not a VM) and the compute nodes are virtual machines.



In the above picture, "frontend-0-0" is a physical machine (with the public name of "espresso.rocksclusters.org". The physical machine "frontend-0-0" controls two VM compute nodes named "compute-0-0-0" and "compute-0-1-0". This means that "compute-0-0-0" and "compute-0-1-0" are configured by "frontend-0-0". This is opposed to the "virtual cluster scenario" (Installing Virtual Clusters), where the virtual frontend ("frontend-0-0-0") configured the VM compute nodes, and the physical machine that housed "frontend-0-0-0" only started and stopped the virtual compute nodes.

3.4.1. Adding, Installing and Booting VMs with a Physical Frontend

In the common case, you will execute three Rocks commands over the lifetime of your VMs: add (to add VM info to the database), start (to boot or install a VM) and stop (to shutdown an installed VM).

To add a VM to the system, you need to associate a VM with a physical machine (i.e., a VM container) and you need to assign an appliance type to the VM. Here's an example:

```
# rocks add host vm vm-container-0-0 membership="Compute"
```

The above command will output a message similar to:

```
added VM compute-0-0-0 on physical node vm-container-0-0
```

This tells us that, in the database, the compute VM named "compute-0-0-0" has been assigned to the physical machine "vm-container-0-0".

The next step is to install the VM.

VMs are installed with the `start`. Here's how to install the VM that was added above:

```
# rocks start host vm compute-0-0-0
```

After running the command above, you may see the following message:

```
libvir: Xen Daemon error : POST operation failed: (xend.err "Error creating domain: Disk isn't accessible")
```

This is not a problem. The above means the file that will be used for the VMs disk space was not present when the VM was started. The "rocks start host vm" command eventually creates it and starts the VM. That is, while you may see the error message above, the VM is actually running and installing.

The above command will start the standard Rocks installation process for the VM named "compute-0-0-0". After the installation process initializes the network inside the VM, you can monitor the installation just like a physical machine installation by executing:

```
# rocks-console compute-0-0-0
```

After the installation completes, the VM will reboot. After the VM boots, you can interact with the VM just like any other physical machine.

3.5. Remotely Controlling VMs with Pilot

3.5.1. Pilot Overview

"Pilot" is a program that can be used on Windows, Mac OS X and Linux systems to remotely and securely control VMs that are housed on Rocks Clusters. Pilot communicates to the "Airboss" (see section The Airboss for more info) on a Rocks frontend. Pilot can power off, power on, install and connect to the console of VMs.

3.5.2. Installing Pilot

3.5.2.1. Installing Pilot on Mac OS X

Open a "terminal" session and execute:

```
$ curl -o pilot.py http://www.rocksclusters.org/ftp-site/pub/rocks/extra/pilot/pilot.py
$ chmod a+x pilot.py
$ curl -o TightVncViewer.jar http://www.rocksclusters.org/ftp-site/pub/rocks/extra/pilot/osx/TightVn
$ curl -o foundation-python-extras.dmg http://www.rocksclusters.org/ftp-site/pub/rocks/extra/pilot/o
```

Install the "foundation-python-extras.dmg" package:

```
$ open foundation-python-extras.dmg
```

This will open a window with a picture of an open box icon with the label "foundation-python-extras.pkg" under it. Double click on the open box icon.

This will open an installation window. Click on the default settings to properly install this package.

3.5.2.2. Installing Pilot on Windows

The following procedure has been tested on Windows 7, 32-bit.

First, you'll need to install "curl" on your machine. Download and unzip the following file:

```
http://am.net/lib/TOOLS/curl/curl-7.21.2-ssl-sspi-zlib-static-bin-w32.zip
```

Then, from the Rocks web site, download:

```
http://www.rocksclusters.org/ftp-site/pub/rocks/extra/pilot/windows/DLSupport.bat
```

Open a Windows Command Prompt and execute:

```
PATH=%PATH%;<directory for unzipped curl>
DLsupport
```

The above will download: Pilot, VCredist, Python v2.7, OpenSSL and M2Crypto.

Install VCredist, Python v2.7, OpenSSL and M2Crypto by executing:

```
vcredist_x86.exe
python-2.7.msi
Win32OpenSSL_Light-1_0_0a.exe
M2Crytpo-0.20.2.win32-py2.7.exe
```

If you don't already have an SSH client, you'll need to download one. We suggest Putty:

```
http://the.earth.li/~sgtatham/putty/latest/x86/putty-0.60-installer.exe
```

Most Windows PCs already have Java installed. Pilot has been tested with versions available from:

```
http://www.java.com/en/download/
```

3.5.3. Create a Public/Private Key Pair

Pilot uses a private/public key pair to authenticate its messages with the Airboss. If you haven't already created a key pair and associated the public key with the virtual frontend of the cluster you wish to remotely control, then login to the physical frontend that houses your virtual cluster and execute the procedure found here: Creating an RSA Key Pair.

Then copy the private key to the same directory on your machine where you have installed "pilot.py".

3.5.4. Open an SSH Tunnel from Your Machine to the Physical Frontend

Pilot sends messages to the Airboss on port 8677. We'll need to open a secure tunnel and forward port 8677 from your local machine to the Rocks frontend where the Airboss is running.

3.5.4.1. Mac OS X SSH Tunneling

Open a new terminal session and execute:

```
$ ssh -L 8677:localhost:8677 root@beopen.rocksclusters.org
```

Be sure to replace "beopen.rocksclusters.org" with the FQDN of your frontend.

3.5.4.2. Windows SSH Tunneling

If using PUTTY, a guide to setting up tunnels can be found here:

```
http://docs.cs.byu.edu/general/ssh_tunnels.html#use-putty-to-set-up-a-tunnel
```

3.5.5. Using Pilot for Remote VM Power Control

Before you get started, open a new terminal session on your machine. Pilot is a command-line utility.

To power on a VM, you must know the MAC address of the VM you wish to power on. Assuming you want to power on the VM named "frontend-0-0-0" on your physical frontend, execute:

```
# rocks list host interface frontend-0-0-0
SUBNET  IFACE MAC               IP              NETMASK       MODULE NAME                  VLAN OPTI
private eth0  76:77:6e:40:00:00 10.1.255.251    255.255.0.0   xennet frontend-0-0-0        2    ----
public  eth1  76:77:6e:40:00:01 137.110.119.118 255.255.255.0 xennet frontend-0-0-0-public 0    ----
```

The VM frontend-0-0-0 has the MAC addresses of 76:77:6e:40:00:00 and 76:77:6e:40:00:01. You can use either with pilot.

To power on frontend-0-0-0, execute:

```
$ ./pilot.py set host power host=76:77:6e:40:00:00 key=private.key action=on
```

To power off frontend-0-0-0, execute:

```
$ ./pilot.py set host power host=76:77:6e:40:00:00 key=private.key action=off
```

To power on and to force frontend-0-0-0 to install, execute:

```
$ ./pilot.py set host power host=76:77:6e:40:00:00 key=private.key action=install
```

3.5.6. Using Pilot to Connect to a VM's Console

You can also connect to a VMs console with pilot by executing:

```
$ ./pilot.py open host console host=76:77:6e:40:00:00 key=private.key
```

When you connect to a VM's console, you will see two mouse pointers, that is, the mouse pointers are not in "sync". You will have to experiment to find which pointer is the true pointer for the console.

3.6. Networking with XEN Roll

In order to support XEN Virtual machines, it is necessary to have a particular network configuration on the hosting servers. In particular bridges must be set up to provide Virtual Machine with network connectivity. In this section, we describe the various kinds of bridging scenarios for virtual machines and how to set them up. For these examples, the physical machine will be called vm-container-0-0.

3.6.1. VM Network Bridging to Physical Devices

When a VM is bridged to the physical device, it must be assigned in the same subnet as the physical device with a compatible IP address (this case is exactly what is described in Physical Frontend with Virtual Compute Nodes).

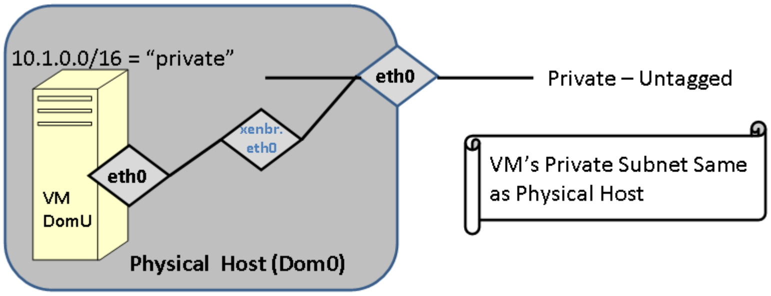


FIGURE: The Virtual machine is bridged to eth0. In this case eth0 of the VM is in the same subnet (with a compatible IP) address. The VM and the container will be able to ping each other. This was the only configuration supported in Rocks 5.0

The following example shows this most basic of bridging scenarios. The guest (compute-0-0-1) and the container (vm-container-0-0) are in the same IP subnet and will be able to ping each other.

```
[root@tranquil images]# rocks list host interface vm-container-0-0 compute-0-0-1
HOST              SUBNET    IFACE    MAC               IP             NETMASK       MODULE NAME
compute-0-0-1:    private   eth0     00:16:3e:00:00:11 172.16.254.192 255.255.255.0 xennet compute-0
vm-container-0-0: private   eth0     00:09:6b:89:39:68 172.16.254.238 255.255.255.0 e1000  vm-contai
```

3.6.2. Logical VLAN Devices

In this scenario, The guest (hosted-vm-0-0-0) and the host (vm-container-0-0) are not in the same logical network (this is the scenario described in Provisioning a Virtual Cluster).

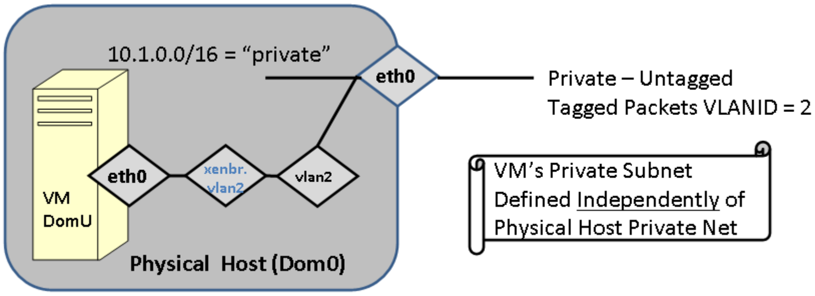

10.1.0.0/16 = “private”
eth0
Private – Untagged
Tagged Packets VLANID = 2
eth0
xenbr.
vlan2
vlan2
VM’s Private Subnet Defined Independently of Physical Host Private Net
VM
DomU
Physical Host (Dom0)

FIGURE: Guest VM is bridged through a logical VLAN device.

```
[root@rocks ~]# rocks list host interface vm-container-0-0 hosted-vm-0-0-0
HOST              SUBNET  IFACE MAC               IP             NETMASK   MODULE NAME             V
hosted-vm-0-0-0:  ------- eth0  00:16:3e:00:00:05 -------------- --------- ------ hosted-vm-0-0-0  2
vm-container-0-0: private eth0  00:0e:0c:5d:7e:5e 10.255.255.254 255.0.0.0 e1000  vm-container-0-0 -
vm-container-0-0: private vlan2 ----------------- -------------- --------- ------ ---------------- 2
```

In the above configuration, Logical VLAN device vlan2 (with tag=2) will be on the physical network eth0 on vm-container-0-0. The hosted-vm-0-0-0 (a Rocks "appliance" that simply holds a generic VM guest) will have have its interface on VLAN=2. The physical machine must have a Logical vlan device with the same tag.

Below we give a more complicated configuration and walk through exactly what is bridged where.

```
[root@rocks ~]# rocks list host interface vm-container-0-0
SUBNET  IFACE   MAC               IP             NETMASK       MODULE NAME             VLAN
private eth0    00:0e:0c:5d:7e:5e 10.255.255.254 255.0.0.0     e1000  vm-container-0-0 ------
net10   eth1    00:10:18:31:74:84 192.168.1.10   255.255.255.0 tg3    vm-net10-0-0     ------
net10   vlan100 ----------------- -------------- ------------- ------ ---------------- 100
private vlan2   ----------------- -------------- ------------- ------ ---------------- 2

[root@rocks ~]# rocks list host interface hosted-vm-0-0-0
SUBNET IFACE MAC               IP NETMASK MODULE NAME            VLAN
------ eth0  00:16:3e:00:00:05 -- ------- ------ hosted-vm-0-0-0 2
------ eth1  00:16:3e:00:00:80 -- ------- ------ --------------- 100
```

In the above scenario, if hosted-vm-0-0-0 (Xen guest, DomU) were to be booted on physical host vm-container-0-0 (Dom0), the packets from the guest on eth0 will be tagged with VLAN=2, and eth1 with VLAN=100. The host machine must have Logical VLAN interfaces called "vlan*.". To make the proper bridge configuration, Rocks will match the VLANs of the guest interfaces to the VLANs on the host. On the host, logical interface vlan2 is labeled as being on the private network (eth0) and logical vlan100 is labeled as being on the net10 network (eth1).

3.6.3. Networking for Virtual Clusters

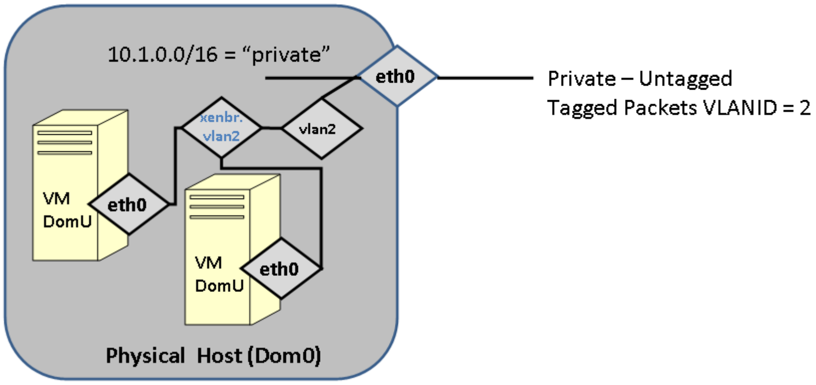


FIGURE: Multiple VMs communicating on a Logical VLAN interface.

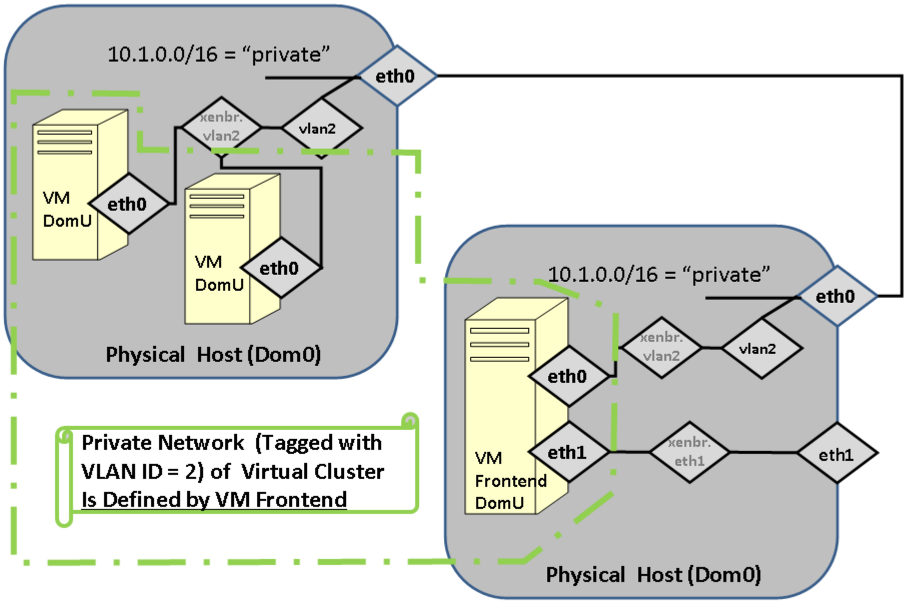


FIGURE: Fully Virtualized cluster, including a virtual frontend.

Notes

1. /roll-documentation/base/5.5/install-frontend.html
2. /roll-documentation/base/5.5/install-frontend.html

Chapter 4. Command Reference

4.1. add

4.1.1. add cluster

rocks add cluster {ip} {num-computes} [container-hosts=*string*] [cpus-per-compute=*string*] [disk-per-compute=*string*] [disk-per-frontend=*string*] [fe-container=*string*] [fe-name=*string*] [ip=*string*] [mem-per-compute=*string*] [num-computes=*string*] [virt-type=*string*] [vlan=*string*]

Add a VM-based cluster to an existing physical cluster.

arguments

ip

The IP address for the virtual frontend.

num-computes

The number of compute nodes VMs to associate with the frontend.

parameters

[container-hosts=*string*]

A list of VM container hosts that will be used to hold the VM compute nodes. This must be a space-separated list (e.g., container-hosts="vm-container-0-0 vm-container-0-1"). The default is to allocate the compute nodes in a round robin fashion across all the VM containers.

[cpus-per-compute=*string*]

The number of CPUs to allocate to each VM compute node. The default is 1.

[disk-per-compute=*string*]

The size of the disk (in gigabytes) to allocate to each VM compute node. The default is 36.

[disk-per-frontend=*string*]

The size of the disk (in gigabytes) to allocate to the VM frontend node. The default is 36.

[fe-container=*string*]

Hosting machine for virtual frontend. Defaults to the physical frontend

[fe-name=*string*]

name to for labeling the frontend. defaults to frontend-0-0-n, where n is assigned

[ip=*string*]

Can be used in place of the ip argument.

[mem-per-compute=*string*]

The amount of memory (in megabytes) to allocate to each VM compute node. The default is 1024.

[num-computes=*string*]

Can be used in place of the num-computes argument.

[virt-type=*string*]

Defines the virtualization type as either paravirtualized (para) or Hardware Virtualized (hvm). Default is para.

[vlan=*string*]

The VLAN ID to assign to this cluster. All network communication between the nodes of the virtual cluster will be encapsulated within this VLAN. The default is the next free VLAN ID.

examples

# rocks add cluster 1.2.3.4 2

Create one frontend VM, assign it the IP address '1.2.3.4', and create 2 compute node VMs.

4.1.2. add host vm

rocks add host vm {host...} {membership} [cpus=*string*] [disk=*string*] [disk-size=*string*] [ip=*string*] [mac=*string*] [mem=*string*] [membership=*string*] [name=*string*] [num-macs=*string*] [slice=*string*] [subnet=*string*] [sync-config=*bool*] [virt-type=*string*] [vlan=*string*]

Add a VM specification to the database.

arguments

host

One or more physical host names.

membership

The membership to assign to the VM.

parameters

[cpus=*string*]

The number of CPUs to assign to this VM. The default is: 1.

[disk=*string*]

A disk specification for this VM. The default is:
file:/<largest-partition-on-physical-node>/xen/disks/<vm-name>.hda,hda,w

[disksize=*string*]

The amount of disk space in gigabytes to assign to the disk specification. The default is: 36.

[ip=*string*]

The IP address to assign to the VM. If no IP address is provided, then one will be automatically assigned.

[mac=*string*]

A MAC address to assign to this VM. If no MAC address is specified, the next free MAC address will be selected.

[mem=*string*]

The amount of memory in megabytes to assign to this VM. The default is: 1024.

[membership=*string*]

Can be used in place of the membership argument.

[name=*string*]

The name to assign to the VM (e.g., ’compute-0-0-0’).

[num-macs=*string*]

The number of MAC addresses to automatically assign to this VM. The default is 1.

[slice=*string*]

The ’slice’ id on the physical node. Each VM on a physical node has a unique slice number The default is the next available free slice number.

[subnet=*string*]

The subnet to associate to this VM. The default is: private.

[sync-config=*bool*]

Decides if ’rocks sync config’ should be run after the VM is added. The default is: yes.

[virt-type=*string*]

Virtualization Type. Valid Values are [para,hvm]. Defaults to para for paravirtualized.

[vlan=*string*]

The vlan ID to set for each interface. If you supply multiple MACs (e.g., ’num-macs’ > 1), you can specify multiple vlan IDs by a comma separated list (e.g., vlan="3,4,5"). To not specify a vlan for a MAC, use the keyword ’none’. For example, if you want to specify a vlan ID for interface 1 and 3, but not interface 2, type: vlan="3,none,5". The default is to not assign a vlan ID.

examples

# rocks add host vm

Create a default VM.

# rocks add host vm mem=4096

Create a VM and allocate 4 GB of memory to it.

4.2. create

4.2.1. create host vm

rocks create host vm {host...}

Create a VM slice on a physical node. This command will configure a VM and install it. This can be used for the initial setup of a VM or to reconfigure an existing VM.

arguments

host

A list of one or more VM host names.

examples

# rocks create host vm compute-0-0-0

Create VM host compute-0-0-0.

4.3. dump

4.3.1. dump host vm

rocks dump host vm [host...]

Dump host VM information as Rocks commands.

arguments

[host]

Zero, one or more host names. If no host names are supplied, information for all hosts will be listed.

examples

$ rocks dump host vm compute-0-0-0

Dump VM info for compute-0-0-0.

$ rocks dump host vm

Dump VM info for all configured virtual machines.

related commands

add host vm

4.4. list

4.4.1. list cluster

rocks list cluster [cluster...] [status=*bool*]

Lists a cluster, that is, for each frontend, all nodes that are associated with that frontend are listed.

arguments

[cluster]

Zero, one or more frontend names. If no frontend names are supplied, information for all clusters will be listed.

parameters

[status=*bool*]

If true, then for each VM-based cluster node, output the VM’s status (e.g., ’active’, ’paused’, etc.).

examples

$ rocks list cluster frontend-0-0

List the cluster associated with the frontend named ’frontend-0-0’.

$ rocks list cluster

List all clusters.

4.4.2. list host vm

rocks list host vm [host...] [showdisks=*bool*] [status=*bool*]

Lists the VM configuration for hosts.

arguments

[host]

Zero, one or more host names. If no host names are supplied, information for all hosts will be listed.

parameters

[showdisks=*bool*]

If true, then output VM disk configuration. The default is 'false'.

[status=*bool*]

If true, then output each VM's status (e.g., 'active', 'paused', etc.).

examples

$ rocks list host vm compute-0-0

List the VM configuration for compute-0-0.

$ rocks list host vm compute-0-0 compute-0-1

List the VM configuration for compute-0-0 and compute-0-1.

4.5. move

4.5.1. move host vm

rocks move host vm {host} {physhost} {file}

Move a VM from its current physical node to another.

arguments

host

The name of the VM host to move.

physhost

The name of the physical host in which to move the VM.

file

The name of the file that stores the running VM’s state.

examples

# rocks move host vm compute-0-0-0 vm-container-1-0

Move VM host compute-0-0-0 to physical host vm-container-1-0.

4.6. pause

4.6.1. pause host vm

rocks pause host vm {host...}

Pauses a VM slice on a physical node.

arguments

host

A list of one or more VM host names.

examples

# rocks pause host vm compute-0-0-0

Pause VM host compute-0-0-0.

4.7. remove

4.7.1. remove cluster

rocks remove cluster [cluster...]

Remove a virtual cluster.

arguments

[cluster]

One or more virtual frontend names.

examples

# rocks rmeove cluster frontend-0-0-0

Remove the cluster associated with the frontend named 'frontend-0-0'.

4.7.2. remove host vm

rocks remove host vm {host...}

Remove the configuration info in the database for the supplied hosts.

arguments

host

A list of one or more VM host names.

examples

# rocks remove host vm compute-0-0-0

Remove the configuration info in the database for compute-0-0-0.

4.8. report

4.8.1. report host vm

rocks report host vm {host}

Outputs a configuration file used by rocks-pygrub in order to boot a VM.

arguments

host

One VM host name (e.g., compute-0-0-0).

examples

$ rocks report host vm compute-0-0-0

Create the VM configuration file for host compute-0-0-0

$ rocks report host vm compute-0-0-0

Create the VM configuration file for host compute-0-0-0.

4.8.2. report host vm config

rocks report host vm config {host...}

Reports the XML Configuration for VM that will be handed to libvirt for startup.

arguments

host

One or more VM host names.

examples

$ rocks report host vm config compute-0-0-0

list the XML configuration of Report XML Config of VM compute-0-0-0.

4.8.3. report host vm virt_type

rocks report host vm virt_type {host}

Output the type of virtualization used for a VM.

arguments

host

One VM host name (e.g., compute-0-0-0).

examples

$ rocks report host vm virt-type compute-0-0-0

Report the vitualization type used.

4.8.4. report host xen bridge

rocks report host xen bridge

Generates the Xen networking bridge configuration script for a host.

examples

$ rocks report host xen bridge

4.8.5. report host xen xendomains

rocks report host xen xendomains

Generates the /etc/sysconfig/xendomains configuration file for a host.

examples

$ rocks report host xen xendomains vm-container-0-0

4.8.6. report vm nextmac

rocks report vm nextmac

Outputs the next free MAC address that can be used for a VM.

examples

$ rocks report vm nextmac

4.9. restore

4.9.1. restore host vm

rocks restore host vm {host...} {file}

Restore a VM on a physical node. This command restores a previously saved VM.

arguments

host

A list of one or more VM host names.

file

The file name the saved VM state is stored in. If you don't supply this parameter, then the default file name is: /<largest-partition-on-physical-host>/xen/disks/<vm-name%gt;.saved. For example, on a physical node with the default partitioning, the file that contains the state for VM compute-0-0-0 is: /state/partition1/xen/disks/compute-0-0-0.saved

examples

# rocks restore host vm compute-0-0-0

Restore VM host compute-0-0-0.

4.10. resume

4.10.1. resume host vm

rocks resume host vm {host...}

Resume a paused VM slice on a physical node.

arguments

host

A list of one or more VM host names.

examples

# rocks resume host vm compute-0-0-0

Resume paused VM host compute-0-0-0.

4.11. save

4.11.1. save host vm

rocks save host vm {host...} {file}

Save a VM on a physical node. This command saves a currently running VM, then halts the VM. This saved state can be used to restart the VM with the command 'rocks restore host vm'.

arguments

host

A list of one or more VM host names.

file

The file name the saved VM state will be stored in. If you don't supply this parameter, then the default file name will be: /<largest-partition-on-physical-host>/xen/disks/<vm-name>.saved. For example, on a physical node with the default partitioning, the saved file for VM compute-0-0-0 will be named: /state/partition1/xen/disks/compute-0-0-0.saved

examples

# rocks save host vm compute-0-0-0

Save VM host compute-0-0-0.

4.12. set

4.12.1. set cluster power

rocks set cluster power {host} [action=*string*] [delay=*string*] {key=*string*}

Turn the power on or off for each client host in a virtual cluster. This command will *not* affect a virtual frontend.

arguments

host

The host name of a virtual frontend.

parameters

[action=*string*]

The power setting. This must be one of 'on', 'off' or 'install'. The 'install' action will turn the power on and force the host to install.

[delay=*string*]

Sets the time (in seconds) to delay between each power command. Default is '1'.

key=*string*

A private key that will be used to authenticate the request. This should be a file name that contains the private key.

examples

$ rocks set cluster power frontend-0-0-0 action=on

Turn on the power for each client node that is associated with frontend-0-0-0.

4.12.2. set host vm

rocks set host vm
{host} [disk=*string*] [disksize=*string*] [mem=*string*] [physnode=*string*] [slice=*string*] [virt-type=*string*]

Change the VM configuration for a specific VM.

arguments

host

One or more VM host names.

parameters

[disk=*string*]

A VM disk specification. More than one disk can be supplied. Each disk specification must separated by a space.

[disksize=*string*]

The size of the VM disk in gigabytes.

[mem=*string*]

The amount of memory in megabytes to assign to this VM.

[physnode=*string*]

The physical machine this VM should run on.

[slice=*string*]

The slice ID for this VM.

[virt-type=*string*]

Set the virtualization type for this VM. This can be ’para’ or ’hvm’.

examples

# rocks set host vm compute-0-0-0 mem=4096

Change the memory allocation for VM compute-0-0-0 to 4 GB.

4.13. start

4.13.1. start host vm

rocks start host vm {host...}

Boots a VM slice on a physical node.

arguments

host

A list of one or more VM host names.

examples

# rocks start host vm compute-0-0-0

Start VM host compute-0-0-0.

# rocks start host vm compute-0-0-0

Start VM host compute-0-0-0.

4.13.2. start service airboss

rocks start service airboss [foreground=`boolean`]

Starts the VM Control service. This service validates commands from remote hosts and, if the command is accepted, the command is parsed and applied to VMs that are managed by this host.

parameters

[foreground=`boolean`]

If set to to 'yes', this service will stay in the foreground. Default is 'no'.

4.14. stop

4.14.1. stop host vm

rocks stop host vm {host...}

Destroy a VM slice on a physical node.

arguments

host

A list of one or more VM host names.

examples

# rocks stop host vm compute-0-0-0

Stop VM host compute-0-0-0. This is equivalent to a ’hard power off’, (i.e., pulling the power cord from a node).

4.14.2. stop service airboss

rocks stop service airboss

4.15. sync

4.15.1. sync host network xen

rocks sync host network xen

Reconfigure and restart the network for the named hosts.

examples

# rocks sync host network compute-0-0

Reconfigure and restart the network on compute-0-0.

Appendix A. Rocks® Copyright

Rocks(r)
www.rocksclusters.org
version 5.5 (Mamba)
version 6.0 (Mamba)

Copyright (c) 2000 - 2012 The Regents of the University of California.
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice unmodified and in its entirety, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. All advertising and press materials, printed or electronic, mentioning features or use of this software must display the following acknowledgement:

> "This product includes software developed by the Rocks(r) Cluster Group at the San Diego Supercomputer Center at the University of California, San Diego and its contributors."

4. Except as permitted for the purposes of acknowledgment in paragraph 3, neither the name or logo of this software nor the names of its authors may be used to endorse or promote products derived from this software without specific prior written permission. The name of the software includes the following terms, and any derivatives thereof: "Rocks", "Rocks Clusters", and "Avalanche Installer". For licensing of the associated name, interested parties should contact Technology Transfer & Intellectual Property Services, University of California, San Diego, 9500 Gilman Drive, Mail Code 0910, La Jolla, CA 92093-0910, Ph: (858) 534-5815, FAX: (858) 534-7345, E-MAIL:invent@ucsd.edu

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS “AS IS AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Appendix B. Third Party Copyrights and Licenses

This section enumerates the licenses from all the third party software components of this Roll. A "best effort" attempt has been made to insure the complete and current licenses are listed. In the case of errors or ommisions please contact the maintainer of this Roll. For more information on the licenses of any components please consult with the original author(s) or see the Rocks® CVS repository[1].

B.1. xen

```
GNU General Public License
--------------------------

Most files in this repository are licensed under the terms of the GNU
General Public License (GPL), a copy of which is attached at the end
of this notice. Note that the only valid version of the GPL as far as
the files in this repository are concerned is _this_ particular
version of the license (i.e., *only* v2, not v2.2 or v3.x or
whatever), unless explicitly otherwise stated.

Licensing Exceptions (the relaxed BSD-style license)
----------------------------------------------------

For the convenience of users and those who are porting OSes to run as
Xen guests, certain files in this repository are not subject to the
GPL when distributed separately or included in software packages
outside this repository. Instead we specify a much more relaxed
BSD-style license. Affected files include the Xen interface headers
(xen/include/public/COPYING), and various drivers, support functions
and header files within the Linux sparse source trees. In all such
cases, license terms are stated at the top of the file or in a COPYING
file in the same directory. Note that _any_ file that is modified and
then distributed within a Linux kernel is still subject to the GNU GPL.

 -- Keir Fraser (on behalf of the Xen team)

=====================================================================

                    GNU GENERAL PUBLIC LICENSE
                       Version 2, June 1991

 Copyright (C) 1989, 1991 Free Software Foundation, Inc.
                       59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                            Preamble
```

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation’s software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author’s protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors’ reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone’s free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains

a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program’s source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients’ exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR

REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
    <one line to give the program’s name and a brief idea of what it does.>
    Copyright (C) <year>  <name of author>

    This program is free software; you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation; either version 2 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program; if not, write to the Free Software
    Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
    Gnomovision version 69, Copyright (C) year name of author
    Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type ‘show c’ for details.
```

The hypothetical commands ‘show w’ and ‘show c’ should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than ‘show w’ and ‘show c’; they could even be

```
mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the program, if
necessary.  Here is a sample; alter the names:

  Yoyodyne, Inc., hereby disclaims all copyright interest in the program
  ‘Gnomovision’ (which makes passes at compilers) written by James Hacker.

  <signature of Ty Coon>, 1 April 1989
  Ty Coon, President of Vice

This General Public License does not permit incorporating your program into
proprietary programs.  If your program is a subroutine library, you may
consider it more useful to permit linking proprietary applications with the
library.  If this is what you want to do, use the GNU Library General
Public License instead of this License.
```

Notes

1. http://cvs.rocksclusters.org