Murano Administrator's Guide

Murano Administrator's Guide

v0.2

Publication date 2013-09-09

Abstract

This document is intended for individuals who wish to install and use our product or intend to contribute.

Table of Contents

List of Tables

Chapter 1. General Deployment Steps

Prepare A Lab For Murano

This section provides basic information about lab's system requirements. It also contains a description of a test which you may use to check if your hardware fits the requirements. To do this, run the test and compare the results with baseline data provided.

Lab Requirements

Table 1.1. Hardware requirements

Criteria	Minimal	Recommended
CPU	4 core @ 2.4 GHz	24 core @ 2.67 GHz
RAM	8 GB	24 GB or more
HDD	2 x 500 GB (7200 rpm)	4 x 500 GB (7200 rpm)
RAID	Software RAID-1 (use mdadm as it will improve read performance almost two times)	Hardware RAID-10

There are a few possible storage configurations except the shown above. All of them were tested and were working well.

- 1x SSD 500+ GB
- 1x HDD (7200 rpm) 500+ GB and 1x SSD 250+ GB (install the system onto the HDD and mount the SSD drive to folder where VM images are)
- 1x HDD (15000 rpm) 500+ GB

The list of OSes which we used in our lab is shown below.

Table 1.2. OS Requirements

List
Ubuntu Server 12.04 LTS

Test Your Lab Host Performance

We have measured time required to boot 1 to 5 instances of Windows system simultaneously. You can use this data as the baseline to check if your system is fast enough.

You should use sysprepped images for this test, to simulate VM first boot.

Steps to reproduce test:

1. Prepare Windows 2012 Standard (with GUI) image in QCOW2 format. Let's assume that its name is ws-2012-std.qcow2
2. Ensure that there is NO KVM PROCESSES on the host. To do this, run command:

```
># ps aux | grep kvm
```

3. Make 5 copies of Windows image file:

```
># for i in $(seq 5); do \
   cp ws-2012-std.qcow2 ws-2012-std-$i.qcow2; done
```

4. Create script start-vm.sh in the folder with .qcow2 files:

```
#!/bin/bash
[ -z $1 ] || echo "VM count not provided!"; exit 1
for i in $(seq $1); do
    echo "Starting VM $i ..."
    kvm \
      -m 1024 \
      -drive file=ws-2012-std-$i.qcow2,if=virtio \
      -net user -net nic,model=virtio \
      -nographic \
      -usbdevice tablet \
      -vnc :$i &
done
```

5. Start ONE instance with command below (as root) and measure time between VM's launch and the moment when Server Manager window appears. To view VM's desktop, connect with VNC viewer to your host to VNC screen :1 (port 5901):

```
># ./start-vm.sh 1
```

6. Turn VM off. You may simply kill all KVM processes by

```
># killall kvm
```

7. Start FIVE instances with command below (as root) and measure time interval between ALL VM's launch and the moment when LAST Server Manager window appears. To view VM's desktops, connect with VNC viewer to your host to VNC screens :1 thru :5 (ports 5901-5905):

```
># ./start-vm.sh 5
```

8. Turn VMs off. You may simply kill all KVM processes by

```
># killall kvm
```

Baseline Bata

The table below provides baseline data which we've got in our environment.

Avg. Time refers to the lab with recommended hardware configuration, while **Max. Time** refers to minimal hardware configuration.

	Boot ONE instance	**Boot FIVE instances**
Avg. Time	3m:40s	8m
Max. Time	5m	20m

Host Optimizations

Default KVM installation could be improved to provide better performance.

The following optimizations may improve host performance up to 30%:

- change default scheduler from **CFQ** to **Deadline**
- use **ksm**
- use **vhost-net**

Install OpenStack

We use Devstack (http://devstack.org/) to build our lab environment.

Devstack is a set of scripts to automate OpenStack installation using sources from Git repositories. No packages are built and installed during this process, all OpenStack components are executed in separate consoles (via screen utility).

It is a great tool to quickly install and start to develop OpenStack on you server. However, it has a few caveats that should be kept in mind:

- Devstack scripts must be executed each time the host reboots. This also reset your OpenStack environment to it's 'just installed' state.
- Multi-node configurations are not as easy in installation as single-node.

Below are the general steps that required to install OpenStack using Devstack in a Single-node mode.

Use Devstack's guide to install single VM OpenStack (http://devstack.org/guides/single-vm.html [http://devstack.org/guides/single-vm.html])

localrc example.

```
HOST_IP=
FLAT_INTERFACE=
FLOATING_RANGE=

ADMIN_PASSWORD=swordfish
MYSQL_PASSWORD=swordfish
RABBIT_PASSWORD=swordfish
SERVICE_PASSWORD=swordfish
SERVICE_TOKEN=tokentoken

ENABLED_SERVICES+=,heat,h-api,h-api-cfn,h-api-cw,h-eng

# Image's cache is in $TOP_DIR/files
IMAGE_URLS+=",http://fedorapeople.org/groups/heat/prebuilt-jeos-images/"
```

```
IMAGE_URLS+="F17-x86_64-cfntools.qcow2"

# /etc/nova/nova.conf
EXTRA_OPTS=(force_config_drive=true libvirt_images_type=qcow2 force_raw_images=false)

# Logging
SCREEN_LOGDIR=/opt/stack/log/
LOGFILE=$SCREEN_LOGDIR/stack.sh.log
```

If you need to image builder only, then install only packages required to run **KVM** (see below).

Configure OpenStack

Note

Additional OpenStack configuration usually doesn't required in case you've installed OpenStack with Devstack scripts.

To configure OpenStack is not a simple task and depends on various options (hardware configuration, network layout, and so on) and in general is out of scope of this document.

Chapter 2. Install Murano Components

This chapter describes how to install Murano components on a separate devbox. We strongly recommend to use a separate host (virtual machine or real host) for Murano devbox as it prevents you from various dependency conflicts.

Automatic Installation

There is a script to automate Murano installation onto devbox.

- Create a folder to hold cloned repositories

```
>#  mkdir -p /opt/git
```

- Clone murano-deployment repository

```
>#  cd /opt/git
>#  git clone git://github.com/stackforge/murano-deployment.git
```

- Change directory to **murano-deployment** and switch to required branch (e.g. **master**)

```
>#  cd /opt/git/murano-deployment
>#  git checkout -b master origin/master
```

- Install prerequisites

```
>#  cd /opt/git/murano-deployment/devbox-scripts
>#  ./murano-git-install.sh prerequisites
```

- Configure the following parameters in lab binding configuration file **/etc/murano-deployment/lab-binding.rc**
  - **LAB_HOST** - IP or nostname of the lab. Actually, this address/name should point to the host where Keystone is installed.
  - **ADMIN_USER** - OpenStack **admin** user
  - **ADMIN_PASSWORD** - A password for OpenStack **admin** user
  - **RABBITMQ_USER** - User to connect to RabbitMQ host
  - **RABBITMQ_PASSWORD** - Password for that user
  - **RABBITMQ_VHOST** - vHost which will be used by Murano components. Provides additional layer of isolation from other devboxes.
  - **RABBITMQ_HOST** - (optional) IP address or hostname of the host where RabbitMQ is installed IF it is not the same host as LAB_HOST points to
  - **RABBITMQ_HOST_ALT** - (optional) IP address or hostname of the RabbitMQ host to connect from inside the Windows instance. In some cases the addresses like LAB_HOST or RABBITMQ_HOST are inaccessible from instances, and they must use different address.
  - **FILE_SHARE_HOST** - (optional) IP address or hostname of the host where file share with prerequisites is located IF it is not the same host as LAB_HOST points to.

- **BRANCH_NAME** - branch name from which all Murano components will be fetched for installation
- **SSL_ENABLED** - Set '**true**' if OpenStack is configured with SSL support and '**false**' otherwise.
- **SSL_CA_FILE** - Path to CA certificate for certificate validation on client side.

- Install Murano components

```
>#  ./murano-git-install.sh install
```

- Login to the Dashboard using URL **http://<your VM IP>/dashboard** or **http://<your VM IP>/horizon**

Manual Installation

Each Murano component has it's own installation scripts called **setup.sh** (for Ubuntu) and **setup-centos.sh** (for CentOS). They are easy to use and will do all the work for you.

For more information please consult the Murano Developer's guide.

Chapter 3. Image Builder

Murano requires a Windows Image in QCOW2 format to be builded and uploaded into Glance.

The easiest way to build Windows image for Murano is to build it on the host where your OpenStack is installed.

Install Required Packages

Note

Please check that hardware virtualization supported and enabled in BIOS.

The following packages should be installed on any host which will be used to build Windows Image:

- ipxe-qemu
- kvm-ipxe
- qemu-kvm
- munin-libvirt-plugins
- python-libvirt
- libvirt-bin
- libvirt0
- munin-libvirt-plugins
- python-libvirt
- virt-goodies
- virt-manager
- virt-top
- virt-what
- virtinst
- python

On Ubuntu you could install them using the command below:

```
>apt-get install ipxe-qemu kvm-ipxe qemu-kvm virt-goodies \
          virtinst virt-manager libvirt0 libvirt-bin \
          munin-libvirt-plugins python python-libvirt \
          python-libxml2 python-minimal python-pycurl \
          python-pyorbit python-requests python-six \
```

```
samba samba-common openssh-server virt-top virt-what
```

Configure Shared Resource

Configure samba based share.

```
># mkdir -p /opt/samba/share
># chown -R nobody:nogroup /opt/samba/share
```

Configure samba server (/etc/samba/smb.conf).

```
...
[global]
   ...
   security = user
...
[share]
   comment = Deployment Share
   path = /opt/samba/share
   browsable = yes
   read only = no
   create mask = 0755
   guest ok = yes
   guest account = nobody
...
```

Restart services.

```
># service smbd restart
># service nmbd restart
```

Prerequisites

Download the files below and copy them into their places in your **${SHARE_PATH}** folder (we usually use **/opt/samba/share** as **${SHARE_PATH}**):

- Windows 2012 Server ISO evaluation version
  - ${SHARE_PATH}/libvirt/images/ws-2012-eval.iso
  - http://technet.microsoft.com/en-us/evalcenter/hh670538.aspx [http://technet.microsoft.com/en-us/evalcenter/hh670538.aspx]
- VirtIO drivers for Windows
  - ${SHARE_PATH}/libvirt/images/virtio-win-0.1-52.iso
  - http://alt.fedoraproject.org/pub/alt/virtio-win/stable/virtio-win-0.1-52.iso [http://alt.fedoraproject.org/pub/alt/virtio-win/stable/virtio-win-0.1-52.iso]
- CloudBase-Init for Windows

    - ${SHARE_PATH}/share/files/CloudbaseInitSetup_Beta.msi
    - http://www.cloudbase.it/downloads/CloudbaseInitSetup_Beta.msi [http://www.cloudbase.it/downloads/CloudbaseInitSetup_Beta.msi]
- Far Manager
    - ${SHARE_PATH}/share/files/Far30b3367.x64.20130426.msi
    - http://www.farmanager.com/files/Far30b3525.x64.20130717.msi [http://www.farmanager.com/files/Far30b3525.x64.20130717.msi]
- Git client
    - ${SHARE_PATH}/share/files/Git-1.8.1.2-preview20130201.exe
    - https://msysgit.googlecode.com/files/Git-1.8.3-preview20130601.exe [https://msysgit.googlecode.com/files/Git-1.8.3-preview20130601.exe]
- Sysinternals Suite
    - ${SHARE_PATH}/share/files/SysinternalsSuite.zip
    - http://download.sysinternals.com/files/SysinternalsSuite.zip [http://download.sysinternals.com/files/SysinternalsSuite.zip]
- unzip.exe tool
    - ${SHARE_PATH}/share/files/unzip.exe
    - https://www.dropbox.com/sh/zthldcxnp6r4flm/-k1Om_V6XR [https://www.dropbox.com/sh/zthldcxnp6r4flm/-k1Om_V6XR]
- PowerShell v3
    - ${SHARE_PATH}/share/files/Windows6.1-KB2506143-x64.msu
    - http://www.microsoft.com/en-us/download/details.aspx?id=34595 [http://www.microsoft.com/en-us/download/details.aspx?id=34595]
- .NET 4.0
    - ${SHARE_PATH}/share/files/dotNetFx40_Full_x86_x64.exe
    - http://www.microsoft.com/en-us/download/details.aspx?id=17718 [http://www.microsoft.com/en-us/download/details.aspx?id=17718]
- .NET 4.5
    - ${SHARE_PATH}/share/files/dotNetFx45_Full_setup.exe
    - http://www.microsoft.com/en-us/download/details.aspx?id=30653 [http://www.microsoft.com/en-us/download/details.aspx?id=30653]
- Murano Agent
    - ${SHARE_PATH}/share/files/MuranoAgent.zip

- https://www.dropbox.com/sh/zthldcxnp6r4flm/-k1Om_V6XR [https://www.dropbox.com/sh/zthldcxnp6r4flm/-k1Om_V6XR]

Additional Software

This section describes additional software which is required to build an Windows Image.

Windows ADK

Windows Assessment and Deployment Kit (ADK) for Windows® 8 is required to build your own answer files for auto unattended Windows installation.

You can dowload it from http://www.microsoft.com/en-us/download/details.aspx?id=30652 [http://www.microsoft.com/en-us/download/details.aspx?id=30652].

PuTTY

PuTTY is a useful tool to manage your Linux boxes via SSH.

You can download it from http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html [http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html].

Windows Server 2012 ISO image

We use the following Windows installation images:

- Windows Server 2008 R2
  - Image Name: 7601.17514.101119-1850_x64fre_server_eval_en-us-GRMSXEVAL_EN_DVD.iso
  - URL: http://www.microsoft.com/en-us/download/details.aspx?id=11093 [http://www.microsoft.com/en-us/download/details.aspx?id=11093]
- Windows Server 2012
  - Image Name: 9200.16384.WIN8_RTM.120725-1247_X64FRE_SERVER_EVAL_EN-US-HRM_SSS_X64FREE_EN-US_DV5.iso
  - URL: http://technet.microsoft.com/en-US/evalcenter/hh670538.aspx?ocid=&wt.mc_id=TEC_108_1_33 [http://technet.microsoft.com/en-US/evalcenter/hh670538.aspx?ocid=&wt.mc_id=TEC_108_1_33]

VirtIO Red Hat drivers ISO image

Warning

Please, choose stable version instead of latest, We've got errors with unstable drivers during guest unattended install.

Download drivers from http://alt.fedoraproject.org/pub/alt/virtio-win/stable/ [http://alt.fedoraproject.org/pub/alt/virtio-win/stable/]

Floppy Image With Unattended File

Run following commands as root:

1. Create emtpy floppy image in your home folder

```
>#  dd bs=512 count=2880 \
   if=/dev/zero of=~/floppy.img \
   mkfs.msdos ~/floppy.img
```

2. Mount the image to **/media/floppy**

```
>#  mkdir /media/floppy mount -o loop \
   ~/floppy.img /media/floppy
```

3. Download **autounattend.xml** file from https://raw.github.com/stackforge/murano-deployment/master/image-builder/share/files/ws-2012-std/autounattend.xml [https://raw.github.com/stackforge/murano-deployment/master/image-builder/share/files/ws-2012-std/autounattend.xml]

```
>#  cd ~
>#  wget https://raw.github.com/stackforge/murano-deployment\
/master/image-builder/share/files/ws-2012-std/autounattend.xml
```

4. Copy our **autounattend.xml** to **/media/floppy**

```
>#  cp ~/autounattend.xml /media/floppy
```

5. Unmount the image

```
>#  umount /media/floppy
```

Build Windows Image (Automatic Way)

1. Clone **murano-deployment** repository

```
>#  git clone git://github.com/stackforge/murano-deployment.git
```

2. Change directory to **murano-deployment/image-builder** folder.

3. Create folder structure for image builder

```
>#  make build-root
```

4. Create shared resource

   Add to /etc/samba/smb.conf.

```
[image-builder-share]
   comment = Image Builder Share
   browsable = yes
   path = /opt/image-builder/share
   guest ok = yes
```

```
guest user = nobody
read only = no
create mask = 0755
```

Restart samba services.

```
>#  restart smbd && restart nmbd
```

5. Test that all required files are in place

```
>#  make test-build-files
```

6. Get list of available images

```
>#  make
```

7. Run image build process

```
>#  make ws-2012-std
```

8. Wait until process finishes
9. The image file **ws-2012-std.qcow2** should be stored under **/opt/image-builder/share/images** folder.

Build Windows Image (Manual Way)

Warning

Please note that the preferred way to build images is to use **Automated Build** described in the previous chapter.

Get Post-Install Scripts

There are a few scripts which perform all the required post-installation tasks.

Package installation tasks are performed by script named **wpi.ps1**.

Download it from https://raw.github.com/stackforge/murano-deployment/master/image-builder/share/scripts/ws-2012-std/wpi.ps1 [https://raw.github.com/stackforge/murano-deployment/master/image-builder/share/scripts/ws-2012-std/wpi.ps1]

Note

There are a few scripts named **wpi.ps1**, each supports only one version of Windows image. The script above is intended to be used to create Windows Server 2012 Standard. To build other version of Windows please use appropriate script from **scripts** folder.

Clean-up actions to finish image preparation are performed by **Start-Sysprep.ps1** script.

Download it from https://raw.github.com/stackforge/murano-deployment/master/image-builder/share/scripts/ws-2012-std/Start-Sysprep.ps1 [https://raw.github.com/stackforge/murano-deployment/master/image-builder/share/scripts/ws-2012-std/Start-Sysprep.ps1]

These scripts should be copied to the shared resource folder, subfolder **Scripts**.

Create a VM

This section describes steps required to build an image of Windows Virtual Machine which could be used with Murano. There are two possible ways to create it - from CLI or using GUI tools. We describe both in this section.

Note

Run all commands as root.

Way 1: Using CLI Tools

This section describes the required step to launch a VM using CLI tools only.

1. Preallocate disk image

```
>qemu-img create -f raw /var/lib/libvirt/images/ws-2012.img 40G
```

2. Start the VM

```
>virt-install --connect qemu:///system --hvm --name WinServ \
   --ram 2048 --vcpus 2 --cdrom /opt/samba/share/9200.16384.WIN8_RTM\
.120725-1247_X64FRE_SERVER_EVAL_EN-US-HRM_SSS_X64FREE_EN-US_DV5.ISO \
  --disk path=/opt/samba/share/virtio-win-0.1-52.iso,device=cdrom \
  --disk path=/opt/samba/share/floppy.img,device=floppy \
  --disk path=/var/lib/libvirt/images/ws-2012.qcow2\
,format=qcow2,bus=virtio,cache=none \
  --network network=default,model=virtio \
  --memballoon model=virtio --vnc --os-type=windows \
  --os-variant=win2k8 --noautoconsole \
  --accelerate --noapic --keymap=en-us --video=cirrus --force
```

Way 2: Using virt-manager UI

A VM also could be lauched via GUI tools like virt-manager.

1. Launch *virt-manager* from shell as root
2. Set a name for VM and select Local install media
3. Add one cdrom and attach Windows Server ISO image to it
4. Select OS type **Windows** and it's version **Windows Server 2008**
5. Set CPU and RAM amount
6. Deselect option **Enable storage for this virtual machine**
7. Select option **Customize configuration before install**
8. Add second cdrom for ISO image with virtio drivers
9. Add a floppy drive and attach our floppy image to it

10.Add (or create new) HDD image with Disk bus **VirtIO** and storage format **RAW**

11.Set network device model **VirtIO**

12.Start installation process and open guest vm screen through **Console** button

Convert the image from RAW to QCOW2 format. The image must be converted from RAW format to QCOW2 before being imorted into Glance.

```
>#  qemu-img convert -O qcow2 /var/lib/libvirt/images/ws-2012.raw \
   /var/lib/libvirt/images/ws-2012-ref.qcow2
```

Upload Image Into Glance

Services deployed by Murano require specially prepared images, that can be created manually or via automated scripts. Please refer to corresponding chapters of this book to create image. After images are created they should be registered in Openstack Glance - image operation service.

1. Use the glance image-create command to import your disk image to Glance:

   ```
   >$ glance image-create --name <NAME> \
        --is-public true --disk-format qcow2 \
        --container-format bare \
        --file <IMAGE_FILE> \
        --property <IMAGE_METADATA>
   ```

   Replace the command line arguments to glance image-create with the appropriate values for your environment and disk image:

   - Replace <**NAME**> with the name that users will refer to the disk image by. E.g. '**ws-2012-std**'
   - Replace <**IMAGE_FILE**> with the local path to the image file to upload. E.g. '**ws-2012-std.qcow2**'.
   - Replace <**IMAGE_METADATA**> with the following property string

     ```
     murano_image_info='{"title": "Windows 2012 Core Edition", "type": "ws-2012-core
     ```

     where

     - title - user-friendly description of the image
     - type - one of possible image types
       - ws-2012-std - Windows Server 2012 Standart Edition
       - ws-2012-core - Windows 2012 Core Edition
       - ws-2008r2-std - Windows Server 2008R2 Standart Edition
       - ws-2008r2 - Windows Server 2012R2

2. To update metadata of the existing image run the command:

```
>$ glance image-update <IMAGE-ID> --property <IMAGE_MATADATA>
```

- Replace **<IMAGE-ID>** with image id from the previous command output.
- Replace **<IMAGE_METADATA>** with murano_image_info property, e.g.

```
murano_image_info='{"title": "Windows 2012 Core Edition", "type": "ws-2012-core
```

Warning

The value of the **--property** argument named **murano_image_info** is a JSON string. Only double quotes are valid in JSON, so please type the string exactly as in the example above.

After these steps desired image can be chosen in Murano dashboard and used for services platform.

Chapter 4. Troubleshooting

General Notes. The following debug sequence should be used when you have no idea about why the system isn't working. If you have one, you may skip unnecessary sections.

Set debug options to "True" in the following Murano configuration files:

- /etc/murano-api/murano-api.conf
- /etc/murano-conductor/conductor.conf

Stop both **murano-api** and **murano-conductor** services. We will start them one by one from the console.

murano-api. First, the murano-api must be started.

- Open new console
- Start **murano-api** service manually

```
>#  murano-api --config-dir /etc/murano-api 2>&1 \
   > /var/log/murano-api-live.log &
>#  tailf /var/log/murano-api-live.log
```

- Open dashboard, create and send to deploy some simple environment.
- Open RabbitMQ web console, open your vhost and ensure that queues were created and there is at least one message.
- Check log for errors - there shouldn't be any
- Keep **murano-api** service running

murano-conductor. Next to the **murano-api** the **murano-conductor** should be started

- Open new console
- Start conductor from console

```
>#  muranoconductor --config-dir /etc/murano-conductor \
   > /var/log/murano-conductor-live.log &
>#  tailf /var/log/murano-conductor-live.log
```

- Check that there is no python exceptions in the log. Some errors like 404 are ok, as conductor tries to delete environment that doesn't exist
- Check heat stack status. It should not be in FAILED state. If it is - check heat and nova error log to find the cause.
- Keep murano-conductor service running.

Log Files. There are various log files which will help you to debug the system.

Murano Log Files

- /var/log/murano-api.log
- /var/log/murano-conductor.log

- /var/log/apache2/errors.log
- /var/log/httpd/errors.log

Windows Log Files

- C:\Program Files (x86)\CloudBase Solutions\logs\log.txt
- C:\Murano\Agent\log.txt
- C:\Murano\PowerShell.log