Murano Getting Started Guide

Murano Getting Started Guide

v0.3

Table of Contents

Chapter 1. About Murano

Murano Project introduces an application catalog, which allows application developers and cloud administrators to publish various cloud-ready applications in a browsable categorised catalog, which may be used by the cloud users (including the unexperienced ones) to pick-up the needed applications and services and composes the reliable environments out of them in a “push-the-button” manner. Key goal is to provide UI and API which allows to compose and deploy composite environments on the Application abstraction level and then manage their lifecycle. The Service should be able to orchestrate complex circular dependent cases in order to setup complete environments with many dependant applications and services. However, the actual deployment itself will be done by the existing software orchestration tools (such as Heat), while the Murano project will become an integration point for various applications and services.

The service addresses following use cases:

- Self-deploying of predefined Services with their dependencies
- Automation of administrative tasks during data center roll-out
- Custom windows application as a Windows Service
- Custom linux application as a Linux Service

The solution provides higher level of abstraction for manipulation applications. Key concepts are:

- Windows Service - a service such as Active Directory, MsSQL, or IIS, which usually consists of multiple virtual machines (VM) and has multiple dependencies.
- Linux Service - a service such as Apache Server can be easily preconfigured and installed in 1-Click
- Environment - a logical unit for one or several Services.
- VM instance - a VM which hosts a Service. A Service might be deployed over several VM instances.
- Metadata Service - Service that provides opportunity to manage custom services definitions.

The Key Features of the Service are the following:

1. Native to OpenStack
2. Introduces abstraction level for Windows Environments
3. Supports Availability Zones and Disaster Recovery scenarios
4. Uses native Windows features for High Availability (HA) solutions

Document change history

The following table describes the most recent changes:

Revision Date	Version of Murano	Summary of Changes
September. 4, 2013	Release-0.2	• Initial document creation.

Revision Date	Version of Murano	Summary of Changes
December. 9, 2013	Release-0.4	• Update with Release-0.4 information

Chapter 2. Before you begin

Naming Conventions

To clearly separate commands and parts of configuration files we use boxes, like shown below:

```
[Some config]
...
some_key1 = some_value1
some_key2 = some_value2
...
```

```
>$ echo ‘Execute this command as regular user’
># echo ‘Execute this command as root’
```

All commands start either with ‘>$’ mark or with ‘>#’ mark. The difference is that first should be executed as regular user, while second - as superuser (‘root’).

Use Appropriate Branch

Master is a brunch for a development purposes. For stable releases appropriate branch names are using: release-0.1, release-0.2 and so on.

Use Separate vHost in RabbitMQ

In general it is OK to configure Murano services to use the root (‘/’) vHost in RabbitMQ and use the same user credentials as OpenStack services use. However, we recommend to create a separate vHost with separate user for each Murano devbox. There are a few reasons for that:

- this prevents queue name collisions
- this prevents message stealing from queues
- this simplify debugging

If you are planning to use only one devbox then you may stay vHost with ‘/’. Required steps to configure your own vHost are described in RabbitMQ Configuration Notes.

System requirements

- **OpenStack Lab** - a single node Openstack environment. See setup lab environment

  Supported OS:

  - Ubuntu Server 12.04 x64

  Hardware requirements:

  - CPU: 8+ cores
  - RAM: 12+ GB

- HDD:
  - 1x SSD 500+ GB
  - 1x HDD (7200 rpm) 500+ GB and 1x SSD 250+ GB (system onto the HDD and SSD drive for VM images)
  - 1x HDD (15000 rpm) 500+ GB
- NIC: 1x Ethernet 100#bps (recommended 1Gbps)

Software requirements:

- Heat
- RabbitMQ
- Windows Server 2012 Standard image imported into Glance. See Build Windows Image chapter.
- Samba share with prerequisites. See Install Samba
- OpenStack metadata service.

- **Devbox** - a machine where all murano services will be running. This machine should have access to the Openstack Lab. See Setup Devbox chapter.

  We suggest to use virtual machine for Murano, as it allows you to backup your VM easily. Any type of hypervisor software which supports linux as guest OS could be used. KVM, VMWare and VirtualBox were tested with success.

  Supported OS:

  - Ubuntu Server 12.04 LTS x86_64
  - CentOS 6.4 x86_64

  "Hardware" requirements:

  - RAM: 512 MB
  - CPU: 1 core
  - HDD: 20 GB
  - NIC: 1x Ethernet 100#bps

  Software Requirements:

  - Packages:

    Ubuntu:

    - gcc
    - python-pip
    - python-dev

- libxml2-dev
- libxslt-dev
- libffi-dev

CentOS

- gcc
- python-pip
- python-devel
- libxml2-devel
- libxslt-devel
- libffi-devel

- X Server is NOT required and system runlevel 3 is preferred.

Chapter 3. Setup Lab Environment

Install OpenStack

To install Murano you need a working copy of OpenStack. If you already have a Openstack installation make sure it meets the system requirements. To install Openstack that will be ready to use with Murano follow the instructions below.

Using Devstack

Currently the most simple way to build a lab is to use devstack. The steps are quite simple:

- Install and configure OS on your hardware. The supported OS is Ubuntu Server 12.04 x64.
- Install all the latest updates:

```
>apt-get update
>apt-get -y upgrade
```

- Create a user **stack**:

```
>adduser stack
```

- Add user **stack** to sudoers:

```
>echo ‘stack ALL=(ALL) NOPASSWD:ALL' > /etc/sudoers.d/stack
>chmod 0440 /etc/sudoers.d/stack
```

- Create a folder for OpenStack installation files:

```
>mkdir /opt/stack
>chown stack:stack /opt/stack
```

- Clone the **devstack** repo

```
>su stack
>cd
>git clone https://github.com/openstack-dev/devstack.git
```

- Checkout the **stable/havana** branch

```
>cd devstack
>git checkout stable/havana
```

- Get `localrc` and `local.sh` files

```
>$ wget https://raw.github.com/stackforge/murano-deployment/release-0.4/
                getting-started/localrc
>$ wget https://raw.github.com/stackforge/murano-deployment/release-0.4/
                getting-started/local.sh
>$ chmod +x local.sh
```

- Start *devstack*

```
>$ ./stack.sh
```

When stack.sh finishes execution your OpenStack installation is ready.

Prepare Shared Prerequisites

Configure Samba Share

- Install SAMBA:

```
>$ apt-get update
>$ apt-get install samba
```

- Create shared folder:

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

- Edit /etc/samba/smb.conf:

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

[image-builder-share]
    comment = Image Builder Share
```

```
    path = /opt/image-builder/share
    browsable = yes
    guest ok = yes
    guest account = nobody
    read only = no
    create mask = 0755
```

- Restart services:

```
># restart smbd
># restart nmdb
```

Copy Prerequisites Into The Share

- Create folder structure:

```
># mkdir -p “/opt/samba/share/Prerequisites/IIS”
># mkdir -p “/opt/samba/share/Prerequisites/SQL Server/2012”
># mkdir -p “/opt/samba/share/Prerequisites/SQL Server/Tools”
```

- Download and put the following files into the specified paths under `/opt/samba/share` folder:

  1. ASP .NET MVC 4 setup package [http://www.asp.net/mvc/mvc4]: `Prerequisites/AspNetMVC4Setup.exe`
  2. Additional files to build ASP application [https://www.dropbox.com/sh/zthldcxnp6r4flm/Y8gxbIZGF_/WebApplications.exe]: should be placed to `Prerequisites/WebApplications.exe`
  3. MS SQL Server 2012 [http://www.microsoft.com/en-us/download/details.aspx?id=29066]: Extract the content of the ISO/EXE file to the `Prerequisites/SQL Server/2012/`. After that `setup.exe` should be located inside this directory.
  4. Microsoft® Windows PowerShell Extensions for Microsoft® SQL Server®2012 [http://go.microsoft.com/fwlink/?LinkID=239656&clcid=0x409]: `Prerequisites/SQL Server/Tools/PowerShellTools.msi`
  5. Microsoft® SQL Server® 2012 Shared Management Objects [http://go.microsoft.com/fwlink/?LinkID=239659&clcid=0x409]: `Prerequisites/SQL Server/Tools/SharedManagementObjects.msi`
  6. Microsoft® System CLR Types for Microsoft® SQL Server® 2012 [http://go.microsoft.com/fwlink/?LinkID=239644&clcid=0x409]: `Prerequisites/SQL Server/Tools/SQLSysClrTypes.msi`

Build Windows Image

A pre-built Windows Image is required to create environments in Murano. Because of its size it’s better to build the image on the same host where OpenStack is installed. This section describe steps required to build such an image.

Prepare Build Environment

- Samba should be already installed (See Configure samba share)
- Clone **murano-deployment** repository:

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

- Change directory to `murano-deployment/image-builder` folder:

```
>#  cd /opt/git/murano-deployment/image-builder
```

- Create folder structure for image builder:

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

- Download and put the following files into the specified paths under `/opt/image-builder` folder:

  1. Windows 2012 Server ISO evaluation version [http://technet.microsoft.com/en-us/evalcenter/hh670538.aspx]: libvirt/images/ws2012eval.iso
  2. VirtIO drivers [http://alt.fedoraproject.org/pub/alt/virtio-win/stable/virtio-win-0.1-52.iso] should be placed to libvirt/images/virtio-win-0.1-52.iso
  3. CloudBase-Init for Windows: [http://www.cloudbase.it/downloads/CloudbaseInitSetup_Beta.msi] share/files/CloudbaseInitSetup_Beta.msi
  4. Far Manager: [http://www.farmanager.com/files/Far30b3525.x64.20130717.msi] share/files/Far30b3367.x64.20130426.msi
  5. Git client [https://msysgit.googlecode.com/files/Git-1.8.3-preview20130601.exe] share/files/Git-1.8.1.2-preview20130201.exe
  6. Sysinternals [http://download.sysinternals.com/files/SysinternalsSuite.zip] Suiteshare/files/SysinternalsSuite.zip
  7. unzip.exe tool [https://www.dropbox.com/sh/zthldcxnp6r4flm/-k1Om_V6XR] share/files/unzip.exe
  8. PowerShell v3 [http://www.microsoft.com/en-us/download/details.aspx?id=34595] share/files/Windows6.1-KB2506143-x64.msu
  9. .NET 4.0: [http://www.microsoft.com/en-us/download/details.aspx?id=17718] share/files/dotNetFx40_Full_x86_x64.exe
  10. .NET 4.5: [http://www.microsoft.com/en-us/download/details.aspx?id=30653] share/files/dotNetFx45_Full_setup.exe
  11. Murano Agent [https://www.dropbox.com/sh/zthldcxnp6r4flm/-k1Om_V6XR] share/files/MuranoAgent.zip

- Test that all required files are in place:

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

Build The Image

- Get list of supported images:

```
>#  make
```

- Run image build process:

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

- Wait until process finishes
- The image file `ws-2012-std.qcow2` will be stored in `/opt/image-builder/share/images` folder.

Import Windows Image Into Glance

Now when you've built a Windows Image it must be imported into Glance.

- Import `openrc` file which contains environment variables definitions required by OpenStack components:

```
>$  source openrc
```

- Import the Windows Server 2012 image into Glance:

```
>$ cd /opt/image-builder/share/images
>$ glance image-create --name ws-2012-std --disk-format qcow2           \
    --container-format bare --file ws-2012-std.qcow2 --is-public true     \
    --property murano_image_info='{"type":"windows.2012", "title":"Windows Server 2012 Sta
```

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.

RabbitMQ Configuration Notes

OpenStack rabbitMQ default credentials could be used for murano services, but prefered way is to make additional changes into rabbitMQ configuration, like own vhost, login and password. These steps require superuser rights and should be run only on the OpenStack controller node where rabbitMQ service resides.

How to do this:

```
>#  rabbitmqctl add_user muranouser muranopassword
>#  rabbitmqctl set_user_tags muranouser administrator
>#  rabbitmqctl add_vhost muranovhost
>#  rabbitmqctl set_permissions -p muranovhost muranouser ".*" ".*" ".*"
```

Keystone Configuration Notes

Service entry and endpoint associated with it for murano-api and murano-repository could also be configured in the keystone. If there is no record about murano-api in the keystone murano-dashboard would try to reach murano-api service at localhost on the default murano-api port 8082.(8084 in case of murano-repository)

How to add service record into the keystone using python keystone client:

```
>#  keystone service-create --name muranoapi --type murano \
                --description "Murano-Api Service"
```

And for murano-repository:

```
>#  keystone service-create --name murano-repository --type murano-repository \
                --description "Murano-Repository Service"
```

This command returns UUID of the created service record, which should be used below.

```
>#  keystone endpoint-create \
              --service-id UUID_from_above \
              --publicurl http://murano_vm_address:8082 \
              --internalurl http://murano_vm_address:8082 \
              --adminurl http://murano_vm_address:8082 \
```

Chapter 4. Setup Devbox

There are a few ways to install Murano devbox:

- using Vagrant automation tool

  or

- using a script to install all components automatically

> **Note**
> The preferred way is to use script for automated installation. It is described in Automated Way section below.

Vagrant Way

The simplest way to get working Murano devbox is Vagrant tool. On this page describes how to create the Murano DevBox on Ubuntu Desktop.

Prepare Environment

- Install VirtualBox:

```
>#  apt-get install virtualbox
```

- Install VirtualBox Extension Pack.

  To install Extension Pack please go to the Dowload section on virtualbox.org (https://www.virtualbox.org/wiki/Downloads), then select and download appropriate Extension Pack package. When downloaded, open it with VirtualBox (you could simply double-click the file) and install the package.

- Install Vagrant:

```
>#  apt-get install vagrant --no-install-recommends
```

- Upgrade the Vagrant:

```
>#  wget http://files.vagrantup.com/
            packages/7ec0ee1d00a916f80b109a298bab08e391945243/
            vagrant_1.2.7_x86_64.deb
>#  dpkg --install vagrant_1.2.7_x86_64.deb
```

Launch The Box

- Clone murano-deployment repository:

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

- Change directory to cloned repository folder:

```
>#  cd murano-deployment/getting-started
>#  git checkout -b release-0.4 origin/release-0.4
```

- **IMPORTANT STEP**: RabbitMQ credentials are specified to default in `murano-deployment/getting-started/localrc` so don't forget to change them by replacing all the markers '***' to your settings:

```
# Lab Settings
#-------------
# Address of the host which provides Keystone service.
#
# LAB_HOST='192.168.1.2'
LAB_HOST='***.***.***.***'
...
```

- Launch the box:

```
>$  ./launch-the-box.sh
```

- The script will do the following:

  - Download the predefined vagrnat box with Ubuntu Server 12.04;

  - Register it in Vagrant;

  - Start the box using vagrant.

- - Vagrant will do the rest:

  - Launch and initialize the box;

  - Execute the provision.sh script which will install Murano onto the box.

- When everything is done open the link http://127.0.0.1:8080/horizon

Automated Way

This easy way can be used on Ubuntu Server 12.04 x86-64 and CentOS 6.4 x86-64. Also, you can use virtual machines on VMWare, VirtualBox, KVM with these operating systems.

- Create a folder to hold git repositories:

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

- Clone murano-deployment repository from branch release-0.4:

```
>#  git clone git://github.com/stackforge/murano-deployment.git -b release-0.4
```

- Install prerequisites and configure devbox.

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

- Press Enter to configure lab binding configuration file `/etc/murano-deployment/lab-binding.rc`

```
LAB_HOST=''

ADMIN_USER=''
ADMIN_PASSWORD=''

RABBITMQ_LOGIN=''
RABBITMQ_PASSWORD=''
RABBITMQ_VHOST=''
RABBITMQ_PORT=''

# Equals to LAB_HOST by default
#RABBITMQ_HOST=''
#RABBITMQ_HOST_ALT=''

#FILE_SHARE_HOST=''

BRANCH_NAME='release-0.4'

# Only 'true' or 'false' values are allowed!
SSL_ENABLED='false'
SSL_CA_FILE=''
SSL_CERT_FILE=''
SSL_KEY_FILE=''

#BRANCH_MURANO_API=''
#BRANCH_MURANO_DASHBOARD=''
#BRANCH_MURANO_CLIENT=''
#BRANCH_MURANO_CONDUCTOR=''
#BRANCH_MURANO_REPOSITORY=''
```

It's recommended to use separate vHost in RabbitMQ.

Then some additional system packages will be installed.

- Install Murano components

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

- When everything is done login to the Dashboard using URL http://your_IP/horizon for Ubuntu and http://your_IP/dashboard for CentOS

Chapter 5. Troubleshooting

Set debug options to "true" in all config files - dashboard, api, conductor.

Note

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.

First, stop both murano-api and murano-conductor services. We will start them one by one from the console.

Murano-Api

- Open new console
- Start api service manually

```
>#  murano-api --config-dir /etc/murano-api \
        > /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

- 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.

Now, the environment should be created, and instance(s) launched.

Next, check if instances were configured correctly by the cloudbase init tool.

Log in to any instance and open powershell log file at `C:\Murano\PowerShell.log`. There shouldn't be any exceptions logged. Other symptoms of successful configuration is that the instance was renamed and you have to press <Ctrl>+<Alt>+<Del> to log into. Unconfigured instance has autologon enabled for the first logon, so if console is open, the instance is not configured (yet).

Check that Murano Agent has correct config file. If there is a .bak file, then it was changed by the init script. Check the file, ensure that it has correct values.

Check Murano Agent log file. There should be logged all tasks received by the agent from the conductor.

Check PowerShell log. There should be messages about all functions, executed on the instance.

Log files

Murano Log Files

- /var/log/murano/murano-api.log
- /var/log/murano/murano-conductor.log
- /var/log/murano/murano-dashboard.log
- /var/log/murano/murano-repository.log
- /var/log/apache2/errors.log
- /var/log/httpd/errors.log

Windows Log Files (on VM's)

- C:\Murano\PowerShell.log
- C:\Murano\Agent\log.txt