Murano

Murano is an online application catalog for OpenStack. It provides a user friendly interface to interact with cloud-ready applications.

Cloud users -- including inexperienced ones -- can then use the catalog to compose reliable application environments with the push of a button.

The key goal is to provide UI and API to compose and deploy composite environments on the Application abstraction level and then manage their lifecycle. The Service can orchestrate complete environments with many dependent applications and services by the existing software orchestration tools (such as Heat) and Murano will act as an integration point for various applications and services.

Murano provides tools for defining and packaging applications on preconfigured VMs (either Linux or Windows). Applications can be arbitrarily complex: from individual KVM-based VMs equipped for software development, testing or small-scale website deployments, to multi-tier Linux and vCenter applications that auto-scale onto many VMs, self-heal and provide robust support for Big Data analysis and other critical work.

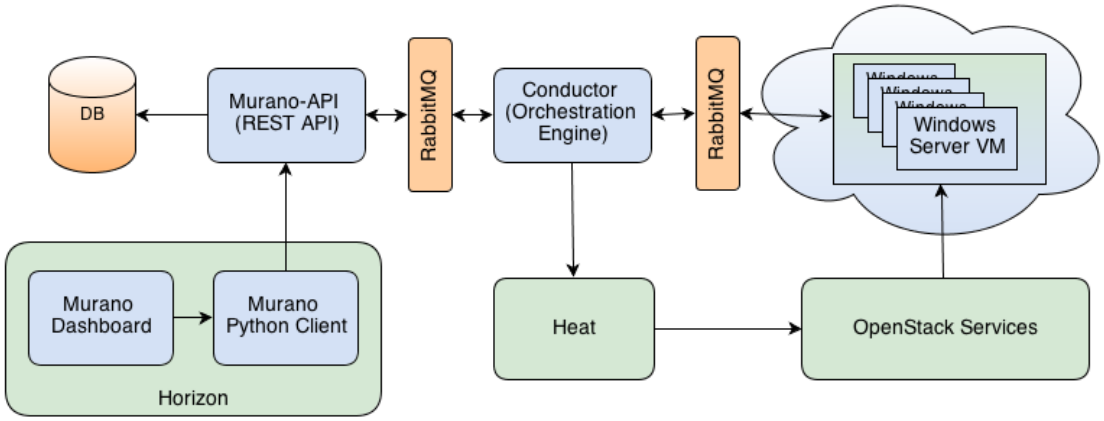
Project Murano is tightly integrated with core OpenStack services, including Keystone, Glance, Heat, and Horizon. Murano is useful to third-party application developers as it offers to:

- Publish applications.
- Include deployment rules and requirements.
- Perform Application configurations.
- Output parameters and billing rules.
- Track billing and usage information.
- Handle modifications on run-time.

1. Murano Integration with Heat

Murano deals with applications while Heat orchestrates infrastructure. In fact Murano utilizes an infrastructure orchestration layer such as Heat to provision the underlying VMs, networking and storage. With Murano it is possible to upload applications written in the HOT (Heat Orchestration Template) format. This format is supported by the Murano CLI tool, which automatically generates a proper Murano application package from the HOT template supplied by the user.

2. Murano Architecture



3. Murano High Level Work Flow

1. User send request via Murano dashboard to Murano python client
2. Murano python client send request to Murano API server
3. Murano API server send the request to rabbitmq
4. Murano conductor picks the request message from queue
5. Murano conductor parses the message and sends the heat template to heat engine
6. Murano conductor sends deployment execution plan to the rabbitmq as Murano agent task.
7. Heat deploy the whole IaaS level infrastructure via various OpenStack services
8. OpenStack services provision the VM instances with Murano-agent enabled.
9. On VMs Murano-agent pick up the execution plan assigned by the Murano conductor and execute them via various scripts on certain OS

4. Deploying Cloud Application with Heat

4.1. Composing a package

Murano support applications defined in different formats. As a first step to universality, support of a heat orchestration template was added. It means that any heat template could be added as a separate application into the Application Catalog.

Following steps are used for package composing:

Step 1. Choosing the desired heat orchestration template

Step 2. Rename it to template.yaml

Step 3. Prepare an application logo (optional step)

It could be any picture associated with the application.

Step 4. Create manifest.yaml file

All service information about the application is contained here. Specify the following parameters:

Format: defines an application definition format; should be set to Heat.HOT/1.0

Type: defines a manifest type, should be set to Application

FullName: a unique name which will be used to identify the application in Murano Catalog

Description: text information about an application

Author: a name of an application author or a company

Tags: keywords associated with the application

Logo: a name of a logo file for an application

Following is an example:

```
Format: Heat.HOT/1.0
Type: Application
FullName: io.murano.apps.Chef-Server
Name: Chef Server
Description: "Heat template to deploy Open Source CHEF server on a VM"
Author: John Doe
Tags:
  - hot-based
Logo: logo.png
```

Step 5: Create a zip archive, containing the specified files: template.yaml, manifest.yaml, logo.png

Step6: Upload the zip file to Murano Package.

Using the horizon dashboard it is possible to upload the zip archive to the murano package.

5. vIMS Orchestration with Murano using Heat

Following steps will be followed for composing vIMS package:

Step 1: Download vIMS Heat Template package from GitHub (https://github.com/Metaswitch/clearwater-heat/). These templates don't provide monitoring and scaling of vIMS cluster. For full orchestration we will need to modify these yaml templates to add ceilometer monitoring and heat scaling policy implementation.

Step 2: OpenStack Liberty should be deployed as nested files support was added in Liberty release.

Step 3: clearwater.yaml is the top-level template, and depends on the other templates to create a network and Clearwater servers.

Step 4: All files need to be placed in the "Resources/HotFiles" and HOT template will use get_file to reference them.

Step 5: Import the Ubuntu 14.04 cloud image into your OpenStack deployment. This image should have Murano client and other configurational optimizations installed, to interoperate with Murano for automated application deployment and management.

Step 6: Prepare an application logo (optional step).

Step 7: Create manifest.yaml file

Step 8: Create a zip archive, containing the specified files: All Clearwater yaml files, manifest file and logo picture. For Murano package composition the following command has to be run:

```
murano package-create  --template wordpress/clearwater.yaml
```

Step 9: Upload the zip file to Murano Package.

Step 10: Use Murano GUI to deploy vIMS cluster.

6. Cloudify Plugin for Murano

There exist a cloudify plugin for Murano. The cloudify plugin for Murano enables you to deploy Cloudify TOSCA blue prints from Murano. This plugin will help us to use Cloudify underneath Murano to Orchestrate Clearwater vIMS. Murano is an Application Catalog and Cloudify is a Cloud Orchestrator which is designed to manage the lifecycle of the TOSCA applications. The process is simple to implement:

1. Install Murano in an Openstack environment.
2. Install Cloudify Manager in a separate environment.
3. Edit the Murano Config file to add the IP of Cloudify Manager.
4. Zip Cloudify application package. This package will be imported to Murano later.
5. Install Cloudify Murano Plugin into the Murano environment.
6. Restart Murano services.
7. Import Cloudify Application package from the Murano Dashboard inside Horizon.
8. Restart Murano services again.

6.1. Deploying vIMS

1. Once the Cloudify Plugin has been imported, import vIMS files to Murano.
2. Press "Quick Deploy" in the Applications tab underneath Application Catalog.
3. This uploads the blueprint to the Cloudify Manager and installation of vIMS is started.