networking-powervm Documentation

Release 4.0.2.dev2

IBM

July 13, 2018

Contents

This project will provide an ML2 compatible agent for PowerVM Shared Ethernet Adapter integration with OpenStack Neutron.

Documentation on Neutron can be found at the Neutron Devref.

CHAPTER 1

Networking-PowerVM Overview

Contents:

PowerVM Neutron ML2 Agent

The IBM PowerVM hypervisor provides virtualization on POWER hardware. PowerVM operators can see benefits in their environments by making use of OpenStack. This project implements a ML2 compatible agent that provides capability for PowerVM admins to natively use OpenStack Neutron. This agent is tied to the Shared Ethernet Adapter technology which is currently the typical scenario for PowerVM network virtualization.

Problem description

This project provides a ML2 compatible agent for the PowerVM hypervisor. It is paired to the nova-powervm driver.

This PowerVM agent provides support for VLAN networks across Shared Ethernet Adapters. It provisions the VLANs on the Virtual I/O Servers (VIOS) to support the client workload, via the PowerVM REST API. The Nova component will set up the peer adapter as part of VIF plugging.

Only networks of physical type VLAN are supported.

Use Cases

- Deploy a VLAN to the specified Virtual I/O Server (or pair of servers) as deploys occur.
- Periodic heal of the systems (similar to Open vSwitch agent design).
- Periodic optimization (removal of unused VLANs from the Shared Ethernet Adapters) of the system.
- Heartbeat of the agent.

Project Priority

None

Data model impact

None

REST API impact

None

Security impact

None

Notifications impact

None

Other end user impact

None to end user.

Performance Impact

No performance impact. Deploy operations should not be impacted by using this agent.

Other deployer impact

The operator needs to obtain the agent from the code repository. The cloud administrator needs to install the agent on both the Neutron controller as well as on the compute node.

The operator will then need to configure the bridge_mappings, to define in the CONF file how to map the physical networks to the adapters. No further configuration is required for the operator. If only one physical network exists (the default), and a single Shared Ethernet Adapter, no bridge_mapping configuration is required. The agent will assume the default network maps to that single Shared Ethernet Adapter (or single pair SEAs set up for redundancy).

Redundant Shared Ethernet Adapters (as defined by the PowerVM Redbook) are fully supported by this agent.

Developer impact

None

Implementation

Assignee(s)

Primary assignee: thorst

Other contributors: wpward svenkat efried

Dependencies

- The Neutron ML2 Plugin.
- Utilizes the PowerVM REST API specification for management. Will utilize future versions of this specification as it becomes available: http://ibm.co/1lThV9R
- Builds on top of the pypowervm library. An open-source, python based library that interacts with the PowerVM REST API.

Testing

Tempest Tests

Since the tempest tests should be implementation agnostic, the existing tempest tests should be able to run against the PowerVM agent without issue.

Thorough unit tests exist within the agent that validate specific functions for this implementation.

Functional Tests

A third party functional test environment has been created. It monitors incoming Neutron change sets. Once it detects a new change set, it should execute the existing lifecycle API tests. A non-gating vote (+1 or -1) will be provided with information provided (logs) based on the result.

Work continues in this area.

API Tests

No changes (no new APIs)

References

- Neutron ML2 Plugin: https://wiki.openstack.org/wiki/Neutron/ML2
- PowerVM REST API Initial Specification (may require newer versions as they become available): http://ibm.co/1lThV9R
- PowerVM Virtualization Introduction and Configuration: http://www.redbooks.ibm.com/abstracts/sg247940.html
- PowerVM Best Practices: http://www.redbooks.ibm.com/abstracts/sg248062.html

CHAPTER 2

Networking-PowerVM Policies

Contents:

Networking-PowerVM Policies

In the Policies Guide, you will find documented policies for developing with Networking-PowerVM. This includes the processes we use for blueprints and specs, bugs, contributor onboarding, and other procedural items.

Policies

Networking-PowerVM Bugs

Networking-PowerVM maintains all of its bugs in Launchpad. All of the current open Networking-PowerVM bugs can be found in that link.

Bug Triage Process

The process of bug triaging consists of the following steps:

1. Check if a bug was filed for a correct component (project). If not, either change the project or mark it as "Invalid".
2. Add appropriate tags. Even if the bug is not valid or is a duplicate of another one, it still may help bug submitters and corresponding sub-teams.
3. Check if a similar bug was filed before. If so, mark it as a duplicate of the previous bug.
4. Check if the bug description is consistent, e.g. it has enough information for developers to reproduce it. If it's not consistent, ask submitter to provide more info and mark a bug as "Incomplete".
5. Depending on ease of reproduction (or if the issue can be spotted in the code), mark it as "Confirmed".
6. Assign the importance. Bugs that obviously break core and widely used functionality should get assigned as "High" or "Critical" importance. The same applies to bugs that were filed for gate failures.
7. (Optional). Add comments explaining the issue and possible strategy of fixing/working around the bug.

Contributing to Networking-PowerVM

If you would like to contribute to the development of OpenStack, you must follow the steps in the "If you're a developer, start here" section of this page:

http://wiki.openstack.org/HowToContribute

Once those steps have been completed, changes to OpenStack should be submitted for review via the Gerrit tool, following the workflow documented at:

http://wiki.openstack.org/GerritWorkflow

Pull requests submitted through GitHub will be ignored.

Bugs should be filed on Launchpad, not GitHub:

https://bugs.launchpad.net/neutron-powervm

Code Reviews

Code reviews are a critical component of all OpenStack projects. Code reviews provide a way to enforce a level of consistency across the project, and also allow for the careful onboarding of contributions from new contributors.

Code Review Practices

Networking-PowerVM follows the code review guidelines as set forth for all OpenStack projects. It is expected that all reviewers are following the guidelines set forth on that page.

Indices and tables

- genindex
- modindex
- search

CHAPTER 3

Networking-PowerVM Devref

Contents:

Developer Guide

In the Developer Guide, you will find information on how to develop for Networking-PowerVM and how it interacts with Neutron. You will also find information on setup and usage of Networking-PowerVM.

Internals and Programming

Setting Up a Development Environment

This page describes how to setup a working Python development environment that can be used in developing networking-powervm.

These instructions assume you're already familiar with Git and Gerrit, which is a code repository mirror and code review toolset, however if you aren't please see this Git tutorial for an introduction to using Git and this guide for a tutorial on using Gerrit and Git for code contribution to Openstack projects.

Getting the code

Grab the code:

```
git clone git://git.openstack.org/openstack/networking-powervm
cd networking-powervm
```

Setting up your environment

The purpose of this project is to provide the 'glue' between OpenStack Networking (Neutron) and PowerVM. The pypowervm project is used to control PowerVM systems.

It is recommended that you clone down the OpenStack Neutron project along with pypowervm into your respective development environment.

Running the tox python targets for tests will automatically clone these down via the requirements. When run with tox, it pulls the necessary requirements into a virtualenv.

Additional project requirements may be found in the requirements.txt file.

Usage

To make use of the PowerVM drivers, a PowerVM system set up with NovaLink is required. The networking-powervm agent should be installed on the management VM. That agent code also is required to be installed on the Neutron controller as well.

The NovaLink architecture is such that the network agent runs directly on the PowerVM system. No external management element (e.g. Hardware Management Console or PowerVC) is needed. Management of the virtualization is driven through a thin virtual machine running on the PowerVM system.

Configuration of the PowerVM system and NovaLink is required ahead of time. The Shared Ethernet Adapters should be set up and configured beforehand. SR-IOV physical port labels must be set to the name of the neutron physical network to which they are cabled. For example, to associate SR-IOV physical port with location code U78C9.001.WZS094N-P1-C7-T2 with the neutron network named 'prod_net':

```
pvmctl sriov update --loc U78C9.001.WZS094N-P1-C7-T2 -s label=prod_net
```

Any un-labeled SR-IOV physical ports will be assumed to belong to the 'default' neutron physical network.

The operator does not need to add VLANs; those will be managed by the networking-powervm agent directly.

Configuration File Options

You must identify which mechanism driver(s) neutron should use. In the [ml2] section of the ML2 configuration file (e.g. `/etc/neutron/plugins/ml2/ml2_conf.ini`), the value of `mechanism_drivers` should be set to a comma-separated list of the desired drivers. The drivers provided by networking-powervm are:

- `pvm_sea`: Shared Ethernet Adapter (SEA) mechanism driver.
- `pvm_sriov`: SR-IOV mechanism driver for virtual NIC.

If using only `pvm_sea` and there is only a single Shared Ethernet Adapter (or adapter pair) using the default physical network, no further configuration is required (but see Optional Configuration below).

If using `pvm_sriov`, you must inform the compute driver which physical networks are allowed to be used by VMs. Each SR-IOV physical port must be labeled with its corresponding neutron network name as described in Usage above; and each authorized network must be listed in the `passthrough_whitelist` in the `[pci]` section of the nova configuration file (e.g. `/etc/nova/nova.conf`). For example, to authorize networks named `default` and `prod_net`, include the following in the nova configuration file:

```
[pci]
passthrough_whitelist = [{"physical_network": "default"}, {"physical_network": "prod_net"}]
```

Optional Configuration The following options go in the `[AGENT]` section of the ML2 configuration file.

Configuration option = Default Value	Agent	Description
bridge_mappings = ''	SEA	(StrOpt) The Network Bridge mappings (defined by the SEA) that describe how the neutron physical networks map to the Shared Ethernet Adapters. This is required if using a network other than the default; or if using more than one SEA (or redundant SEA pair). Format: <phnet1>:<sea1>:<vio1>,<phnet2>:<sea2>:<vio2> Example: default:ent5:vios_1,speedy:ent6:vios_1
automated_powervm_vlan_cleanup = True	SEA	Determines whether or not the VLANs will be removed from the Network Bridge if a VM is removed and it is the last VM on the system to use that VLAN. By default, the agent will clean up VLANs to improve the overall system performance (by reducing broadcast domain). Will only apply to VLANs not on the primary PowerVM virtual Ethernet adapter of the SEA.
vnic_required_vfs = 2	SRIOV	(Integer) Redundancy level for the vNIC created to back an SR-IOV port. The value represents the number of SR-IOV logical ports to create (one per physical port). The binding will fail if the agent cannot find enough physical ports with sufficient free capacity to satisfy this setting.
vnic_vf_capacity = None	SRIOV	(Float) Value between 0.0000 and 1.0000 indicating the minimum guaranteed capacity of the VFs backing the SR-IOV vNIC. Must be a multiple of each physical port's minimum capacity granularity, or the binding will fail. If unspecified, the platform defaults the capacity for each VF to its backing physical port's minimum capacity granularity. [1]

SR-IOV-Backed Neutron Port Creation

To create a neutron port on network with ID `$netid` that will be serviced by an SR-IOV vNIC:

```
neutron port-create --vnic-type direct $netid
```

[1] For more details on SR-IOV logical port capacity, see section 1.3.3 of the IBM Power Systems SR-IOV Technical Overview and Introduction.

Testing

Running Networking-PowerVM Tests

This page describes how to run the Networking-PowerVM tests. This page assumes you have already set up an working Python environment for Networking-PowerVM development.

With *tox*

Networking-PowerVM, like other OpenStack projects, uses tox for managing the virtual environments for running test cases. It uses Testr for managing the running of the test cases.

Tox handles the creation of a series of virtualenvs that target specific versions of Python.

Testr handles the parallel execution of series of test cases as well as the tracking of long-running tests and other things.

For more information on the standard tox-based test infrastructure used by OpenStack and how to do some common test/debugging procedures with Testr, see this wiki page:

https://wiki.openstack.org/wiki/Testr

PEP8 and Unit Tests Running pep8 and unit tests is as easy as executing this in the root directory of the Networking-PowerVM source code:

```
tox
```

To run only pep8:

```
tox -e pep8
```

To restrict the pylint check to only the files altered by the latest patch changes:

```
tox -e pep8 HEAD~1
```

To run only the unit tests:

```
tox -e py27
```

Indices and tables

- genindex
- modindex
- search