Mellanox ConnectX-4/ ConnectX-4 LX Plugin for RedHat OpenStack Platform 9 – version 1-1.1.0

User Manual

Rev 1.0

NOTE:
THIS HARDWARE, SOFTWARE OR TEST SUITE PRODUCT (“PRODUCT(S)”) AND ITS RELATED DOCUMENTATION ARE PROVIDED BY MELLANOX TECHNOLOGIES “AS-IS” WITH ALL FAULTS OF ANY KIND AND SOLELY FOR THE PURPOSE OF AIDING THE CUSTOMER IN TESTING APPLICATIONS THAT USE THE PRODUCTS IN DESIGNATED SOLUTIONS. THE CUSTOMER'S MANUFACTURING TEST ENVIRONMENT HAS NOT MET THE STANDARDS SET BY MELLANOX TECHNOLOGIES TO FULLY QUALIFY THE PRODUCT(S) AND/OR THE SYSTEM USING IT. THEREFORE, MELLANOX TECHNOLOGIES CANNOT AND DOES NOT GUARANTEE OR WARRANT THAT THE PRODUCTS WILL OPERATE WITH THE HIGHEST QUALITY. ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT ARE DISCLAIMED. IN NO EVENT SHALL MELLANOX BE LIABLE TO CUSTOMER OR ANY THIRD PARTIES FOR ANY DIRECT, INDIRECT, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES OF ANY KIND (INCLUDING, BUT NOT LIMITED TO, PAYMENT FOR 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 FROM THE USE OF THE PRODUCT(S) AND RELATED DOCUMENTATION EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Mellanox Technologies
350 Oakmead Parkway Suite 100
Sunnyvale, CA 94085
U.S.A.
www.mellanox.com
Tel: (408) 970-3400
Fax: (408) 970-3403

Table of Contents

List of Tables

Document Revision History

Table 1: Document Revision History

Revision	Date	Description
1.0	February 1st, 2017	Updated description and parameter requirements.

Definitions, Acronyms and Abbreviations

Term	Description
SR-IOV	Single Root I/O Virtualization (SR-IOV), is a specification that allows a PCI device to appear virtually on multiple Virtual Machines (VMs), each of which has its own virtual function. This specification defines virtual functions (VFs) for the VMs and a physical function for the hypervisor. Using SR-IOV in a cloud infrastructure helps to achieve higher performance since traffic bypasses the TCP/IP stack in the kernel.
iSER	iSCSI Extensions for RDMA (iSER), is an extension of the data transfer model of iSCSI, a storage networking standard for TCP/IP. iSER enables the iSCSI protocol to take advantage of the RDMA protocol suite to supply higher bandwidth for block storage transfers (zero time copy behavior). iSER eliminates the TCP/IP processing overhead while preserving the compatibility with iSCSI protocol.
RDMA	Remote Direct Memory Access (RDMA) is a technology that enables data's read and write from a remote server without involving the CPU and allowing it to to perform other tasks. It reduces latency and increases throughput.
ConnectX-4	ConnectX-4 adapter cards with Virtual Protocol Interconnect (VPI), supporting EDR 100Gb/s InfiniBand and 100Gb/s Ethernet connectivity, provide the highest performance and most flexible solution for high-performance, Web 2.0, Cloud, data analytics, database, and storage platforms.
ConenctX-4 LX	ConnectX-4 Lx EN Network Controller with 1/10/25/40/50Gb/s Ethernet connectivity addresses virtualized infrastructure challenges, delivering best-in-class and highest performance to various demanding markets and applications. Providing true hardware-based I/O isolation with unmatched scalability and efficiency, achieving the most cost-effective and flexible solution for Web 2.0, Cloud, data analytics, database, and storage platforms.
VXLAN Offload	Virtual Extensible LAN (VXLAN) is a network virtualization technology that attempts to improve the scalability problems associated with large cloud computing deployments.
VF	VF is virtual NIC that will be available for VMs on Compute nodes.
NEO	Mellanox NEO™ is a powerful platform for managing scale-out computing networks. Mellanox NEO™ enables data center operators to efficiently provision, monitor and operate the modern data center fabric.

1 Mellanox REHL-OSP Plugin

The Mellanox Redhat Openstack plugin is a bundle of scripts, packages and metadata that extends Red Hat Openstack and adds Mellanox features such as SR-IOV for networking and iSER protocol for storage.

Red Hat Openstack allows the configuration of Mellanox ConnectX-4 and ConnectX-4 LX network adapters to accelerate the performance of compute and storage traffic.

This implements the following performance enhancements over Ethernet:

- Compute nodes network enhancements:
  - SR-IOV based networking
  - VXLAN traffic offload
- Cinder nodes use iSER block storage as the iSCSI transport rather than the default iSCSI over TCP.

These features reduce CPU overhead, boost throughput, reduce latency, and enable network traffic to bypass the software switch layer (e.g. Open vSwitch).

Mellanox Plugin integration with Mellanox NEO SDN Controller enables switch VLAN auto provisioning and port configuration for Ethernet networks over private VLAN networks.

2 Package Directory Tree

mellanx-rhel-osp	deployment_sources/	environment_conf/ [Folder holding the scripts that generate the env yaml file used in the deployment command] • config.yaml [a file holding the basic parameters used to create the conf template] • env-template.yaml [a template file used to generate another env yaml file based on user inputs and selected flavors] • multiple_backends.yaml [yaml template that enables iSER configurations]
		network/ [Folder holding example yaml files needed to support multiple nics deployment] • cinder-storage.yaml [network based configuration file] • compute.yaml [network based configuration file] • controller.yaml [network based configuration file] • network_env.yaml [network template file holding network details that users manipulate based on given topology properties]

		overcloud_scripts/
		[Folder holding scripts and resources used in heat templates while deploying the env]
		• scripts/ [Folder holding scripts that are called by the heat yaml files]
		  • common , reboot_servers_conditionally.sh , install_ofed.sh , mellanox_settings.py , sriov.sh
		• puppet/ [Folder holding puppet scripts used in post configurations for the openstack services]
		  • manifests/
		  • compute_sriov.pp , controller_sriov.pp , configure_mlnx_neo.pp ,
		• mellanox_first_boot_conf.yaml [Heat template that runs tasks on first boot of the overcloud nodes]
		• mellanox_post_deployment.yaml [Heat template that runs post deployment tasks]
		• mellanox_pre_deployment.yaml [Heat template that runs tasks prior to the post deployment phase]

		deploy.sh [An Example shell script that deploys the openstack with a generated env.yaml file and the number of overcloud nodes]
		prepare_overcloud_image.py [Script that updates the downloaded overcloud image with mlnx prerequisites]
		create_conf_template.py [a script that loads the env-template.yaml and manipulates it based on flavors passed as arguments to it]

mellanx-rhel-osp	deployment_sources/	summarize_interfaces.py [a script that generates a yaml file holding current nodes interfaces details and profiles]
		params.py [a script that loads all values of needed parameters and defines the package files topology in absolute paths]
		utils.py
		iface_rename.py [a script that generates a 70 persistent udev rules file based on new names in iterfaces_drivers.yaml]
		create_network_conf.py [a script used to generate role.yaml files under network directory to be used in network_env.yaml based on neutron network type in config.yaml vlan/vxlan]
		rm_ovc_image_subscriptions.py [a script that unregisters and removes all subscriptions from the overcloud images]
	README.md [MD format documentation]	
	metadata.yaml [File holding package release details]	
	specs/ [Folder holding spec file that builds rpm out of the package]	
	build.sh [for building an RPM out of the package]	

3 Mellanox RHEL-OSP Plugin Configuration

3.1 Prerequisites

1. Download the Mellanox rhel-osp package from http://bgate.mellanox.com/openstack/openstack_plugins/red_hat_director_plugins/9.0/ and extract the package:

```
tar -zxvf mellanox-rhel-osp-1-1.1.0.tar.gz  -C <stack homedir>
```

2. Install undercloud, register and configure it.
   For help, please refer to RedHat Director 9 Installation section. Appendix A:

3. Register ironic nodes, run introspection, tag their profiles, their root disks and install their swift data to a swift data directory. You can follow the steps detailed in Preparing and Configuring the Overcloud Installation.

3.2 Configuring Mellanox RHEL-OSP Plugin

1. Check the available interfaces and their drivers for all registered ironic nodes from the stack user home directory <stack home directory>/mellanox-rhel-osp/deployment_sources/. Run:

```
$ python summarize_interfaces.py --path_to_swift_data PATH_TO_SWIFT_DATA
```

   Note: This step is crucial for summarizing the interfaces' names and their drivers.

2. Edit the config.yaml file with the parameters and flavors required to create the env template file. This is the main file to configure before deploying TripleO over Mellanox NICs. After configuring this file, TripleO basic configurations file will be generated in this directory. You can use this file or further customize it before starting the deployment.

```
Sriov:
sriov_enable: true
number_of_vfs: 16
sriov_interface: 'enp3s0f0'

Neutron:
physnet_vlan: 'default_physnet'
physnet_vlan_range: '21:36'
network_type: 'vlan'

Iser:
iser_enable: true
storage_network_port: 'enp3s0f1'

Neo:
neo_enable: false
neo_ip:
neo_username: 'admin'
neo_password: 123456
InterfacesValidation: true
```

To configure the flavor of the package:

	Parameter	Required	Description
SR-IOV	sriov_enable	Always	(BoolOpt) If true, burn SR-IOV enabled firmware and enable creating virtual functions over physical interfaces.
	number_of_vfs	Only if sriov_enable is True.	(IntOpt) If > 0, the number of vfs to be burned in bootstrap. Note that if SR-IOV is enabled, number_of_vfs must be an even number between > 0 and <= 32. Otherwise please set an empty value or 0.
	sriov_interface	Always	(StrOpt) The sriov interface, choose from interfaces_drivers.yaml.
Neutron	physnet_vlan	Always	(StrOpt) The physnet name to use for VLAN based deployments.
	physnet_vlan_range		(StrOpt) The vlans range to use for the vlan network.
	network_type		(StrOpt) Can be either VLAN or VXLAN.
iSER	iser_enable	Always	(BoolOpt) If true, enable iSCSI Extensions for RDMA.
	storage_network_port		(StrOpt) The storage network port. Choose from the interfaces list generated from the Summarize interfaces script (from step 1) found in interfaces_drivers.yaml
NEO	neo_enable	Always	(BoolOpt) If true, enable Mellanox NEO to efficiently provision, monitor and operate the modern data center fabric.
	neo_ip	Only when neo_enable is True	(StrOpt) The IP address of the NEO machine.
	neo_username		(StrOpt) The username of the NEO machine.
	neo_password		(StrOpt) The password of the NEO machine.
	InterfacesValidation	Always	(BoolOpt) If true, enable checking the following rules: • Chosen ports availability in the interfaces_drivers.yaml. • Storage port RDMA capability. Using the same port for both SR-IOV and iSER.

An example of the config.yaml file is available in <stack homedir>/mellanox-rhel-osp/deployment_sources/environment_conf/config.yaml.

3. Generate the overcloud environment template configuration files:

```
python create_conf_template.py
```

This script uses the interfaces_driver.yaml (generated from step 1) and config.yaml files to generate the following file:

<stack homedir>/mellanox-rhel-osp/deployment_sources/environment_conf/env.yaml that contains the flavor of the Overcloud deployment.
The network configuration file is located in <stack homedir>/mellanox-rhel-osp/deployment_sources/network/network_env.yaml, which must be modified according to the environment setup.
Example:

```
Parameter_defaults:

  # Required parameters
  ProvisioningInterface: eno1
  ExternalInterface: eno2
  StorageInterface: enp3s0f0
  TenantInterface: enp3s0f1
  NeutronExternalNetworkBridge: "br-ex"
  # Just provide the cider for the required networks (Here are
TenantNetCidr, StorageNetCidr, ExternalNetCidr)
  TenantNetCidr: 172.17.0.0/24
  StorageNetCidr: 172.18.0.0/24
  ExternalNetCidr:  <external_net_cidr>
  #InternalApiNetCidr: 172.16.0.0/24
  #StorageMgmtNetCidr: 172.19.0.0/24
  #ManagementNetCidr: 172.20.0.0/24


  # Just provide the Allocation pools for required networks
  TenantAllocationPools: [{'start': '172.17.0.10', 'end':
'172.17.0.200'}]
  StorageAllocationPools: [{'start': '172.18.0.10', 'end':
'172.18.0.200'}]
  ExternalAllocationPools: [{'start': '<external_ip_range_start>', 'end':
'<external_ip_range_end>'}]
  #StorageMgmtAllocationPools: [{'start': '172.19.0.10', 'end':
'172.19.0.200'}]
  #ManagementAllocationPools: [{'start': '172.20.0.10', 'end':
'172.20.0.200'}]
  #InternalApiAllocationPools: [{'start': '172.16.0.10', 'end':
'172.16.0.200'}]

  ExternalInterfaceDefaultRoute: <external_gateway> # The gateway of the
external network
  ControlPlaneDefaultRoute: <undercloud_internal_ip># The internal IP of
the undercloud
  EC2MetadataIp: <undercloud_internal_ip># The internal IP of the
undercloud
  DnsServers: ["8.8.8.8","8.8.4.4"]
  NtpServer: <ntp_server>
```

The network heat templates are generated using the following script:

```
python create_network_conf.py
```

This script is used to generate <role>.yaml files under network directory.
It is used in network_env.yaml based on neutron network type in config.yaml vlan/vxlan in the folder: <stack homedir>/mellanox-rhel-osp /deployment_sources/network.

4. Prepare the overcloud image from the folder: `<stack homedir>/mellanox-rhel-osp/deployment_sources`

```
[stack@vm deployment_sources]$ python prepare_overcloud_image.py
--iso_ofed_path ISO_OFED_PATH
--overcloud_images_path OVERCLOUD_IMAGES_PATH
--root_password ROOT_PASSWORD
--redhat_account_username REDHAT_ACCOUNT_USERNAME
--redhat_account_password REDHAT_ACCOUNT_PASSWORD
```

```
[--lldp_package_path LLDP_PACKAGE_PATH]
[--python_mlnx_package_path PYTHON_MLNX_PACKAGE_PATH]
Log file: <stack home directory>/mellanox-rhel-
osp/deployment_sources/mellanox-rhel-osp.log

optional arguments:
  -h, --help            show this help message and exit
  --iso_ofed_path ISO_OFED_PATH
                        path to the OFED iso file
  --overcloud_images_path OVERCLOUD_IMAGES_PATH
                        path to the overcloud images
  --root_password ROOT_PASSWORD
                        set root password for the overcloud nodes
  --redhat_account_username REDHAT_ACCOUNT_USERNAME
                        username of the Redhat account for registering
  --redhat_account_password REDHAT_ACCOUNT_PASSWORD
                        password of the Redhat account for registering
   [--lldp_package_path LLDP ]
                        Lldp package path (optional)
   [--python_mlnx_package_path PYTHON_MLNX_PACKAGE_PATH]
                        Python Mellanox package path.
```

4 Deploying the Overcloud

Validate details of the configuration files and run the Deployment command based on your setup. You can also modify and run the example deploy.sh script in the package using the relevant files.

```
openstack \
overcloud deploy \
--stack ovcloud \
--control-scale         1 \
--compute-scale         1 \
--block-storage-scale   1 \
--control-flavor control \
--compute-flavor compute \
--block-storage-flavor block-storage \
--templates    \
--ntp-server time.nist.gov
-e ./environment_conf/env.yaml \
-e ./network/network_env.yaml \
-e ./environment_conf/multiple_backends.yaml \
#--validation-errors-fatal \
#--timeout 180 \
$@
```

- `-stack` specifies the name of the overcloud stack.
- `--control-scale`, `--compute-scale` and `--block-storage-scale` are the number of nodes for each role
- `--control-flavor`, `--compute-flavor` and `--block-storage-flavor` are the nova flavors.
- `-Templates` uses the templates provided in ~/templates or the additional templates provided using -e <path of the template file>.

Note: In the HA deployment, it is required to provide an ntp server to deploy using –-ntp-server. Also, you need to change the envirenoment configuration yaml file to the generated env.yaml from step 3.

Table 2: Tested guest Images

Supported OS	Tested Kernel
CenOS 7	3.10.0-327.36.1.el7.x86_64
Ubuntu 14.04	4.4.0-22-generic

This RedHat Openstack Mellanox plugin v1-1.1.0 uses MLNX_OFED_LINUX version 4.

5 Troubleshooting

- To debug the undercloud installation you can use the following command:

```
openstack stack resource list ovcloud -n $depth
```

- You can also review the log files for issues:

```
<stack homedir>/mellanox-rhel-osp/deployment_sources/mellanox-rhel-
osp.log
/var/log/heat/heat-engine.log
```

- Before deleting the overcloud images, make sure to unregister the overcloud images and remove all subscriptions using the following script

```
[stack@vm deployment_sources]$ python rm_ovc_image_subscriptions.py
usage: rm_ovc_image_subscriptions.py [-h] --overcloud_images_path
OVERCLOUD_IMAGES_PATH
Log file: /<stack home directory>/mellanox-rhel-osp
/deployment_sources/mellanox-rhel-osp.log
optional arguments:
-h, --help show this help message and exit
--overcloud_images_path OVERCLOUD_IMAGES_PATH
path to the overcloud images
```

- To debug the overcloud node installation in case of network configurations issues, use the following command:

```
os-net-config -c /etc/os-net-config/config.json -d
or view the following log files for the deployment steps output:
/var/log/mellanox-rhel-osp.log
/var/log/messages
```

6 Contact Support

To suggest new features and enhancements or to report a problem, please contact Mellanox Support (support@mellanox.com).

Appendix A: RedHat Director 9 Installation

A.1 Undercloud Installation

The steps below are based on the official RedHat documentaion.

1. Install RHEL 7.2 VM.
   Download the ISO file from RedHat official website downloads site.

   Note: The RHEL version will later be automatically updated to v7.3.

2. Configure the OS for the undercloud installation.

   a. Create a director installation user:

```
# useradd stack
# passwd stack
# echo "stack ALL=(root) NOPASSWD:ALL" | tee -a /etc/sudoers.d/stack
# chmod 0440 /etc/sudoers.d/stack
# su - stack
```

3. Prepare and register the system:

   a. Add a workaround for director node boot failure after undercloud installation https://access.redhat.com/solutions/2327921:

   b. Create new home directory.

```
$ sudo mkdir /deploy_directory
$ sudo chown stack:stack /deploy_directory
```

   c. Move the home line form the /etc/fstab.

```
$ sudo sed -i 's/\(.*home.*\)/#\1/' /etc/fstab
```

   d. Change the home user directory from /home/stack to new path.

```
$ sudo sed -i 's/\/home\/stack/\/deploy_directory/' /etc/passwd
```

   e. Copy bash scripts from home.

```
$ cp /home/stack/.bash* /deploy_directory/
```

   f. Set hostname for the system and editing the /etc/hosts file.

```
$ sudo hostnamectl set-hostname  <hostname.hostdomain>
$ sudo hostnamectl set-hostname --transient <hostname.hostdomain >
```

   g. Edit the "/etc/hosts" file to be something like that.

```
127.0.0.1   hostname.hostdomain hostname localhost
localhost.localdomain localhost4 localhost4.localdomain4
```

   h. Register the system.

```
subscription-manager register --username RHEL_USER --password
RHEL_PASS --auto-attach
```

   i. Enable the required packages.

```
$ sudo subscription-manager repos --disable=*
$ sudo subscription-manager repos --enable=rhel-7-server-rpms --
enable=rhel-7-server-extras-rpms --enable=rhel-7-server-openstack-9-
rpms --enable=rhel-7-server-openstack-9-director-rpms --enable=rhel-7-
server-rh-common-rpms
$ sudo yum update -y
```

   j. Reboot the driver.

k. Log in as a stack user as all next steps should be performed as stack user.

```
# su - stack
```

l. Create directories for the images and templates.

```
$ mkdir ~/images
$ mkdir ~/templates
```

4. Install the Undercloud.

a. Install the Director packages.

```
$ sudo yum install -y python-tripleoclient
```

b. Configure the Director.

i. Copy the configuration file to `~/undercloud.conf`

```
$ cp /usr/share/instack-undercloud/undercloud.conf.sample
~/undercloud.conf
```

For the configuration of undercloud we can leave all the commented lines as it's and just change the "local_interface" parameter, Or

ii. Edit the **undercloud.conf** file to be like that

```
[DEFAULT]
local_ip                    = 172.21.1.254/24
undercloud_public_vip       = 172.21.1.253
undercloud_admin_vip        = 172.21.1.252
local_interface             = <local_interface>
masquerade_network          = 172.21.1.0/24
dhcp_start                  = 172.21.1.150
dhcp_end                    = 172.21.1.199
network_cidr                = 172.21.1.0/24
network_gateway             = 172.21.1.254
discovery_interface         = br-ctlplane
discovery_iprange           = 172.21.1.100,172.21.1.149
undercloud_debug            = true
[auth]
```

`<local_interface>`: the interface that is connected to the local network.
Run "ip addr" to get interface name.

c. Install the undercloud

```
$ openstack undercloud install
```

To use the undercloud commands line tools

```
$ source ~/stackrc
```

A.2 Preparing and Configuring the Overcloud Installation

These steps are based on the official link from RedHat:

https://access.redhat.com/documentation/en/red-hat-openstack-platform/9/paged/director-installation-and-usage/chapter-5-configuring-basic-overcloud-requirements

1. Obtain the overcloud images and upload them:

```
$ sudo yum install rhosp-director-images rhosp-director-images-ipa
$ cd ~/images
$ for i in /usr/share/rhosp-director-images/overcloud-full-latest-9.0.tar
/usr/share/rhosp-director-images/ironic-python-agent-latest-9.0.tar; do
tar -xvf $i; done
$ openstack overcloud image upload --image-path ~/images/
```

2. Set a nameserver on the undercloud neutron subnet:

```
$ neutron subnet-update `neutron subnet-list | grep '|'| tail -1 | cut -
d'|' -f2` --dns-nameserver 8.8.8.8
```

3. Obtain the basic TripleO heat template:

```
$ cp -r /usr/share/openstack-tripleo-heat-templates ~/templates
```

4. Register ironic nodes and assign profiles to them
5. Prepare the json file **`~/instackenv.json`** according to the following:

```
{
    "nodes":[
        {
            "mac":[
                "bb:bb:bb:bb:bb:bb"
            ],
            "cpu":"4",
            "memory":"6144",
            "disk":"40",
            "arch":"x86_64",
            "pm_type":"pxe_ipmitool",
            "pm_user":"ADMIN",
            "pm_password":"ADMIN",
            "pm_addr":"192.0.2.205"
        },
        {
            "mac":[
                "cc:cc:cc:cc:cc:cc"
            ],
            "cpu":"4",
            "memory":"6144",
            "disk":"40",
            "arch":"x86_64",
            "pm_type":"pxe_ipmitool",
            "pm_user":"ADMIN",
            "pm_password":"ADMIN",
            "pm_addr":"192.0.2.206"
        }
    ]
}
```

6. Inspect the hardware of the nodes:

Import the json file to the director and start the introspection process

```
$ openstack baremetal import --json ~/instackenv.json
$ openstack baremetal configure boot
$ openstack baremetal introspection bulk start
```

To see the nodes use:

```
$ ironic node-list
```

- Inspect the hardware of the nodes (download the file data that contains the description of the hardware detected per each node)
- The ironic python agent does not have an OS, so we can't merge codes or package with it during the introspection phase
- The introspection is a pre step for the deployment and we may encounter some problems while detecting and updating cards (e.g. if it's a VPI type cards, it will be configured to IB not to eth)

7. Tag the nodes and define the root disk for nodes.

   a. Tag nodes into profiles.

```
$ ironic node-update <NodeID> add
properties/capabilities='profile:<PROFILE>,boot_option:local'
Profiles list [compute, control, swift-storage, ceph-storage, block-
storage]
```

   b. Define the root disk for nodes.

```
$ mkdir swift-data
$ cd swift-data
$ export IRONIC_DISCOVERD_PASSWORD=`sudo grep admin_password
/etc/ironic-inspector/inspector.conf | egrep -v '^#'  | awk '{print
$NF}' | cut -d'=' -f2 | cut -d'=' -f2`
$ for node in $(ironic node-list | grep -v UUID| awk '{print $2}'); do
swift -U service:ironic -K $IRONIC_DISCOVERD_PASSWORD download ironic-
inspector inspector_data-$node; done
$ for node in $(ironic node-list | awk '!/UUID/ {print $2}'); do echo
$node; serial=`cat inspector_data-$node | jq '.inventory.disks' | awk
'/"serial": /{print $2}' | head -1` ;ironic node-update $node add
properties/root_device='{"serial": '$serial'}'; done
```