

Mellanox ConnectX-4/ ConnectX-4 Lx Plugin for RedHat OpenStack Platform 10

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

© Copyright 2017. Mellanox Technologies Ltd. All Rights Reserved.

Mellanox®, Mellanox logo, Accelio®, BridgeX®, CloudX logo, CompustorX®, Connect-IB®, ConnectX®, CoolBox®, CORE-Direct®, EZchip®, EZchip logo, EZappliance®, EZdesign®, EZdriver®, EZsystem®, GPUDirect®, InfiniHost®, InfiniBridge®, InfiniScale®, Kotura®, Kotura logo, Mellanox CloudRack®, Mellanox CloudXMellanox®, Mellanox Federal Systems®, Mellanox HostDirect®, Mellanox Multi-Host®, Mellanox Open Ethernet®, Mellanox OpenCloud®, Mellanox OpenCloud Logo®, Mellanox PeerDirect®, Mellanox ScalableHPC®, Mellanox StorageX®, Mellanox TuneX®, Mellanox Connect Accelerate Outperform logo, Mellanox Virtual Modular Switch®, MetroDX®, MetroX®, MLNX-OS®, NP-1c®, NP-2®, NP-3®, Open Ethernet logo, PhyX®, PlatformX®, PSIPHY®, SiPhy®, StoreX®, SwitchX®, Tiler®, Tiler logo, TestX®, TuneX®, The Generation of Open Ethernet logo, UFM®, Unbreakable Link®, Virtual Protocol Interconnect®, Voltaire® and Voltaire logo are registered trademarks of Mellanox Technologies, Ltd.

All other trademarks are property of their respective owners.

For the most updated list of Mellanox trademarks, visit <http://www.mellanox.com/page/trademarks>

Table of Contents

| | |
|---|-----------|
| Document Revision History..... | 5 |
| Definitions, Acronyms and Abbreviations..... | 6 |
| 1 Mellanox REHL-OSP Plugin..... | 7 |
| 2 Package Directory Tree..... | 8 |
| 3 Plugin Configuration..... | 10 |
| 3.1 Prerequisites..... | 10 |
| 3.2 Configuring RedHat OSP 10 Plugin..... | 11 |
| 4 Deploying the Overcloud..... | 14 |
| 5 Troubleshooting..... | 15 |
| 6 Contact Support..... | 16 |
| Appendix A: RedHat Director 10 Installation..... | 17 |
| A.1 Undercloud Installation..... | 17 |
| A.2 Preparing and Configuring the Overcloud Installation..... | 19 |

List of Tables

| | |
|--|----|
| Table 1: Document Revision History | 5 |
| Table 2: Tested guest Images | 14 |

Document Revision History

Table 1: Document Revision History

| Revision | Date | Description |
|----------|---------------|---|
| 1.0 | May 7th, 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. |

1 Mellanox REHL-OSP Plugin

The Mellanox Red Hat 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:

- 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.
- Active-Backup Bonding over Mellanox interface.

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

2 Package Directory Tree

| | | |
|------------------------------|---------------------------------|---|
| <pre>mellanx- rhel-osp</pre> | <pre>deployment_sourc es/</pre> | <pre>environment_conf/</pre> <p>[Folder holding the scripts that generate the env yaml file used in the deployment command]</p> <ul style="list-style-type: none"> • config.yaml [a file holding the basic parameters used to create the conf template] • environment-rhel-registration.yaml [a file holding the environment rhel registration heat 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] • neutron-sriov-template.yaml [yaml template that enables neutron SR-IOV configuration.] <hr/> <pre>network/</pre> <p>[Folder holding example yaml files needed to support multiple nics deployment]</p> <ul style="list-style-type: none"> • 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] <hr/> <pre>overcloud_scripts/</pre> <p>[Folder holding scripts and resources used in heat templates while deploying the env]</p> <ul style="list-style-type: none"> • scripts/ [Folder holding scripts that are called by the heat yaml files] <ul style="list-style-type: none"> • 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] <ul style="list-style-type: none"> • manifests/ <ul style="list-style-type: none"> • compute_sriov.pp , controller_sriov.pp and configure_mlnx_neo.pp [Not supported in this version.] • 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] • rhel-unregistration.yaml [Heat template that runs the un-registration of the deployment.] <hr/> <pre>deploy.sh</pre> <p>[An Example shell script that deploys the openstack with a generated env.yaml file and the number of overcloud nodes]</p> |
|------------------------------|---------------------------------|---|

| | |
|--|---|
| | <pre>prepare_overcloud_image.py</pre> <p>[Script that updates the downloaded overcloud image with mlnx prerequisites]</p> |
| | <pre>create_conf_template.py</pre> <p>[a script that loads the env-template.yaml and manipulates it based on flavors passed as arguments to it]</p> |
| | <pre>summarize_interfaces.py</pre> <p>[a script that generates a yaml file holding current nodes interfaces details and profiles]</p> |
| | <pre>params.py</pre> <p>[a script that loads all values of needed parameters and defines the package files topology in absolute paths]</p> |
| | <pre>utils.py</pre> |
| | <pre>iface_rename.py</pre> <p>[a script that generates a 70 persistent udev rules file based on new names in interfaces_drivers.yaml]</p> |
| | <pre>create_network_conf.py</pre> <p>[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]</p> |
| | <pre>rm_ovc_image_subscriptions.py</pre> <p>[a script that unregisters and removes all subscriptions from the overcloud images]</p> |
| | <pre>README.md</pre> <p>[MD format documentation]</p> |
| | <pre>metadata.yaml</pre> <p>[File holding package release details]</p> |

3 Plugin Configuration

3.1 Prerequisites

1. Download the Mellanox rhel-osp package from http://www.mellanox.com/page/products_dyn?product_family=273&mtag=cloudx_software and extract the package:

```
tar -zxvf mellanox-rhel-osp-2.0.0.tar.gz -C <stack homedir>
```

2. Install undercloud, register and configure it.
For help, please refer to [RedHat Director 10 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 _____

4. [Preparing and Configuring the Overcloud](#) Installation.

3.2 Configuring RedHat OSP 10 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 RHEL-OSP 10 over Mellanox NICs. After configuring this file, RHEL-OSP 10 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 is set to false by default as it is not supported in this version.

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

Bond:
    bond_enable: true

InterfacesValidation: false
```

To configure the flavor of the package:

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 cidr 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>
```

- If bonding is enabled:
 - i. Assign required network vlan ID in network_env.yaml:

```
# Use these parameters when using VLANS or Bonding
#InternalApiNetworkVlanID: 20
#StorageNetworkVlanID: 30
#StorageMgmtNetworkVlanID: 40
#TenantNetworkVlanID: 50
#ExternalNetworkVlanID: 10
```

- ii. Customize bonding option in network_env.yaml:

```
# Customize bonding options, e.g. "mode=4 lacp_rate=1 updelay=1000
miimon=100"
#BondInterfaceOvsOptions: "bond_mode=active-backup"
```

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
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
  --lldp_package_path LLDAP
                        Lldp package path (optional)
  --python_mlnx_package_path PYTHON_MLNX_PACKAGE_PATH
                        Python Mellanox package path. (optional)
```

5. Update the environment rhel registration file environment-rhel-registration.yaml from the folder: <stack homedir>/mellanox-rhel-osp/deployment_sources/environment_conf with the registration activation key, the auto attach option and the other options as in the following example:

```
# Note this can be specified either in the call
# to heat stack-create via an additional -e option
# or via the global environment on the seed in
# /etc/heat/environment.d/default.yaml

parameter_defaults:
  rhel_reg_activation_key: "<reg_activation_key>"
  rhel_reg_auto_attach: "False"
  rhel_reg_base_url: ""
  rhel_reg_environment: ""
  rhel_reg_force: ""
  rhel_reg_machine_name: ""
  rhel_reg_org: "<your_organization_name>"
  rhel_reg_password: ""
  rhel_reg_pool_id: ""
  rhel_reg_release: ""
  rhel_reg_repos: ""
  rhel_reg_sat_url: ""
  rhel_reg_server_url: ""
  rhel_reg_service_level: ""
  rhel_reg_user: ""
  rhel_reg_type: ""
  rhel_reg_method: "portal"
  rhel_reg_sat_repo: "rhel-7-server-rpms, rhel-7-server-extras-pms"
```

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.



NOTE: `deploy.sh` is an example script/template. Please pay attention when using this file to make sure your requirements are met and the structure is valid.

```
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              \
  -e environment_conf/environment-rhel-registration.yaml \
  -e network/network_env.yaml \
  -e environment_conf/env.yaml \
  -e environment_conf/neutron-sriov.yaml \
  -e environment_conf/multiple_backends.yaml > overcloud_install.log \
  #-e ./environment_conf/multiple_backends.yaml \
  #--validation-errors-fatal \
  #--timeout 180 \
  #--ntp-server <add_your_ntp> \
  $@
```

- `-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>`.
- If SR-IOV is enabled: add the `neutron-sriov.yaml` file in `deploy.sh`
- If iSER is enabled: add the `multiple_backends.yaml` file in `deploy.sh`

Note: In the HA deployment, it is required to provide an ntp server to deploy using `--ntp-server`. Also, you need to change the environment 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 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
```

- 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 10 Installation

A.1 Undercloud Installation

The steps below are based on the official RedHat documentation.

https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/10/html/director_installation_and_usage/chap-installing_the_undercloud

1. On the KVM host can install RHEL 7.2 VM using the following commands:

- Installs required packages:

```
# yum install libvirt-client libvirt-daemon qemu-kvm libvirt-
daemondriver-qemu libvirt-daemon-kvm virt-install bridge-utils rsync
```

- Create the undercloud virtual machine on the KVM host and create two virtual NICs that connect the respective KVM bridges:

```
# virt-install --name undercloud --memory=16384 --vcpus=4 -location
/var/lib/libvirt/images/rhel-server-7.3-x86_64-dvd.iso --disk size=100 -
-network bridge=<kvm_external_bridge> --network
bridge=<kvm_local_bridge> --graphics=vnc --hvm --os-variant=rhel7
```

This starts a **libvirt** domain. Connect to it with **virt-manager** and walk through the install process. Set the VM public IP via the **virt-manager** viewer.

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:

b. Add a workaround for director node boot failure after undercloud installation

<https://access.redhat.com/solutions/2327921>:

c. Create new home directory.

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

d. Move the home line from the /etc/fstab.

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

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

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

f. Copy bash scripts from home.

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

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

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

h. Edit the “/etc/hosts” file to be something like that.

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

- i. Register the system.

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

- j. 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-rh-common-rpms
--enable=rhel-ha-for-rhel-7-server-rpms --enable=rhel-7-server-
openstack-10-rpms
$ sudo yum update -y
```

- k. Reboot the system.

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

```
# su - stack
```

- m. 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 package.

- 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. Run "ip addr" to get interface name and the undercloud IP.

- iii. Edit the **undercloud.conf** file to be like that

```
[DEFAULT]
local_interface           = <local_interface>
generate_service_certificate = true
enable_ui                 = true
undercloud_public_vip    = <undercloud_public_ip>
[auth]
```

where:

<local_interface>: the interface that is connected to the local network.

<undercloud_public_ip>: the public IP of the undercloud VM.

- c. Run the "install" command to 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-us/red_hat_openshift_platform/10/html/director_installation_and_usage/

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-10.0.tar /usr/share/rhosp-director-images/ironic-python-agent-latest-10.0.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
$ for node in $(openstack baremetal node list -c UUID -f value) ; do
openstack baremetal node manage $node ; done
$ openstack overcloud node introspect --all-manageable --provide
```

To see the nodes use:

```
$ openstack baremetal 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
$ sudo yum install crudini
$ export SWIFT_PASSWORD=`sudo crudini --get /etc/ironic-
inspector/inspector.conf swift password`
$ for node in $(ironic node-list | grep -v UUID| awk '{print $2}'); do
swift -U service:ironic -K $SWIFT_PASSWORD download ironic-inspector
inspector_data-$node; done
$ for node in $(ironic node-list | awk '!/UUID/ {print $2}'); do echo
"NODE: $node" ; cat inspector_data-$node | jq '.inventory.disks' ;
echo "-----" ; done
```