Multitenancy in Bare Metal service

Overview

It is possible to use dedicated tenant networks for provisioned nodes, which extends the current Bare Metal service capabilities of providing flat networks. This works in conjunction with the Networking service to allow provisioning of nodes in a separate provisioning network. The result of this is that multiple tenants can use nodes in an isolated fashion. However, this configuration does not support trunk ports belonging to multiple networks.

Network interface is one of the driver interfaces, that manages network switching for nodes. Currently there are 3 network interfaces available in the Bare Metal service:

- `noop` interface is used for standalone deployments, and does not perform any network switching;
- `flat` interface places all provisioned nodes and nodes being deployed into a single layer 2 network, separated from the cleaning network;
- `neutron` interface provides tenant-defined networking by integrating with neutron, while also separating tenant networks from the provisioning and cleaning provider networks.

Configuring Ironic

Below is an example flow of how to setup ironic so that node provisioning will happen in a multitenant environment (which means using `neutron` network interface as stated above):

1. Network interfaces can be enabled on ironic-conductor by adding them to `enabled_network_interfaces` configuration option under the default section:

   ```
   [DEFAULT]
   ...
   enabled_network_interfaces=noop,flat,neutron
   ```

   Please note that `enabled_network_interfaces` has to be set in the ironic-api configuration file too, as its value is used on the API side to check if the requested network interface is available.

   Keep in mind that, ideally, all conductors should have the same list of enabled network interfaces, but it may not be the case during conductor upgrades. This may cause problems if one of the conductors dies and some node that is taken over is mapped to a conductor that does not support the node's network interface. Any actions that involve calling the node's driver will fail until that network interface is installed and enabled on that conductor.

2. It is recommended to set the default network interface via `default_network_interface` configuration option under the default section:

   ```
   [DEFAULT]
   ...
   default_network_interface=neutron
   ```

   It will be used for all nodes that will be created without explicitly specifying the network interface.

   If it is not set, the default network interface is determined by looking at the `[dhcp]dhcp_provider` configuration option value. If it is `neutron` - `flat` network interface becomes the default, otherwise `noop` is the default.

3. Define a provider network in neutron, which we shall refer to as the "provisioning" network, and add it in under the neutron section in ironic-conductor configuration file. Using `neutron` network interface requires that `provisioning_network_uuid` and `cleaning_network_uuid` configuration options are set to a valid neutron network UUIDs, otherwise ironic-conductor will fail to start:

```
[neutron]
...
cleaning_network_uuid=$CLEAN_UUID
provisioning_network_uuid=$PROVISION_UUID
```

Please refer to Configure the Bare Metal service for cleaning for more information about cleaning.

Note: The "provisioning" and "cleaning" networks may be the same neutron provider network, or may be distinct networks. To ensure communication between ironic and the deploy ramdisk works, it's important to ensure that security groups are disabled for these networks, *or* the default security groups allow:

- DHCP
- TFTP
- egress port used for ironic (6385 by default)
- ingress port used for ironic-python-agent (9999 by default)
- if using the iSCSI deploy method (`pxe_*` and `iscsi_*` drivers), the ingress port used for iSCSI (3260 by default)
- if using the direct deploy method (`agent_*` drivers), the egress port used for swift (typically 80 or 443)
- if using iPXE, the egress port used for the HTTP server running on the ironic conductor nodes (typically 80).

4. Install and configure a compatible ML2 mechanism driver which supports bare metal provisioning for your switch. See ML2 plugin configuration manual for details.

5. Restart the ironic conductor and API services after the modifications:

   - Fedora/RHEL7/CentOS7:

     ```
     sudo systemctl restart openstack-ironic-api
     sudo systemctl restart openstack-ironic-conductor
     ```

   - Ubuntu:

     ```
     sudo service ironic-api restart
     sudo service ironic-conductor restart
     ```

6. Make sure that the conductor is reachable over the provisioning network by trying to download a file from a TFTP server on it, from some non-control-plane server in that network:

   ```
   tftp $TFTP_IP -c get $FILENAME
   ```

   where FILENAME is the file located at the TFTP server.

Configuring nodes

1. Multitenancy support was added in the 1.20 API version. The following examples assume you are using python-ironicclient version 1.5.0 or higher. They show the usage of both `ironic` and `openstack baremetal` commands.

   If you're going to use `ironic` command, set the following variable in your shell environment:

   ```
   export IRONIC_API_VERSION=1.20
   ```

   If you're using ironic client plugin for openstack client via `openstack baremetal` commands, export the following variable:

   ```
   export OS_BAREMETAL_API_VERSION=1.20
   ```

2. Node's `network_interface` field should be set to valid network interface that is listed in the `[DEFAULT]/enabled_network_interfaces` configuration option in the ironic-api config. Set it to `neutron` to use neutron ML2 driver:

- `ironic` command:

```
ironic node-create --network-interface neutron \
--driver agent-ipmitool
```

- `openstack` command:

```
openstack baremetal node create --network-interface neutron \
--driver agent-ipmitool
```

Note: If the `[DEFAULT]/default_network_interface` configuration option was set, the `--network-interface` option does not need to be specified when defining the node.

3. To update existing node's network interface, use the following commands:

   - `ironic` command:

```
ironic node-update $NODE_UUID_OR_NAME add network_interface=neutron
```

   - `openstack` command:

```
openstack baremetal node set $NODE_UUID_OR_NAME \
--network-interface neutron
```

4. The Bare Metal service provides the `local_link_connection` information to the Networking service ML2 driver. The ML2 driver uses that information to plug the specified port to the tenant network.

`local_link_connection` fields

Field	Description
`switch_id`	Required. Identifies a switch and can be a MAC address or an OpenFlow-based `datapath_id`.
`port_id`	Required. Port ID on the switch, for example, Gig0/1.
`switch_info`	Optional. Used to distinguish different switch models or other vendor specific-identifier. Some ML2 plugins may require this field.

   Create a port as follows:

   - `ironic` command:

```
ironic port-create -a $HW_MAC_ADDRESS -n $NODE_UUID \
-l switch_id=$SWITCH_MAC_ADDRESS -l switch_info=$SWITCH_HOSTNAME \
-l port_id=$SWITCH_PORT --pxe-enabled true
```

   - `openstack` command:

```
openstack baremetal port create $HW_MAC_ADDRESS --node $NODE_UUID \
--local-link-connection switch_id=$SWITCH_MAC_ADDRESS \
--local-link-connection switch_info=$SWITCH_HOSTNAME \
--local-link-connection port_id=$SWITCH_PORT --pxe-enabled true
```

5. Check the port configuration:

   - `ironic` command:

```
ironic port-show $PORT_UUID
```

   - `openstack` command:

```
openstack baremetal port show $PORT_UUID
```

After these steps, the provisioning of the created node will happen in the provisioning network, and then the node will be moved to the tenant network that was requested.