Quantum Support for Virtual Machine’s Network Interface Configuration

This document describes a proposal for OpenStack Quantum Service to be able to support multiple types of attachment configurations for the network interface on the virtual machine to connect to the underlying network. The need for this proposal arises from the observation that for nova-compute to craft the correct network interface configuration for a VM (prior to spawning the VM), in certain cases, some added information might be needed from Quantum (apart from what nova-compute already posses on the compute side). The proposal touches on what support should be provided by Quantum, and how it can be consumed by nova-compute.

The discussion in this proposal is driven from a libvirt configuration perspective. However, the need to get network information (as summarized above) would be generic, especially on account of the checks performed by the entities analogous to libvirt, say, during a virtual machine migration.

Issue

Let's discuss this with respect to a specific example. As of today, nova-compute generates the domain configuration (libvirt.xml), including the network interface configuration as below:

```
        <interface type='bridge'>
            <source bridge='br100'/>
            <mac address='02:16:3e:78:4d:84'/>
            <filterref filter="nova-instance-instance-00000001">
                <parameter name="IP" value="10.1.0.2" />
                <parameter name="DHCPSERVER" value="10.1.0.1" />
                <parameter name="PROJNET" value="10.1.0.0" />
                <parameter name="PROJMASK" value="255.255.255.128" />
            </filterref>
        </interface>
```

Going forward, if nova-compute will still generate the network interface configuration (as above), before spawning the VM, nova-compute would need to know the value “br100”. If the network boundary is drawn between the VM’s network interface and the bridge, the knowledge of “br100” should ideally come from Quantum.

It should be noted that the need to get information from Quantum is not just limited to the type “bridge”. Even in the case of type “direct” (e.g. 802.1Qbh), information would be required from Quantum as shown below:

```
    <interface type='direct'>
      <mac address='02:16:3e:65:a9:e7'/>
      <source dev='eth4' mode='private'/>
      <virtualport type='802.1Qbh'>
        <parameters profileid='enterprise-xyz'/>
      </virtualport>
      <model type='virtio'/>
    </interface>
```

To construct the above configuration, the source dev, i.e. “eth4” and the profileid, i.e. “enterprise-xyz” should ideally be acquired from Quantum.

Proposal

One option could be to add a core API to achieve this. However, this can probably be done in a cleaner way by adding an attribute to the "Port" resource as shown below. It's called "configuration" here (but could be any other name) and could have zero or more key-value pairs.

```
<Port>
        <port-id> Universally Unique ID </id>
        <VIF-id> Universily Unique ID or None </id>
        <state>Attached/Unattached</state>
        <configuration>
                <conf key=”key-1”>value-1</conf>
                <conf key=”key-2”>value-2</conf>
                ...
        </configuration>
</Port>
```

Again, as discussed, the “configuration” information will be required prior to the interface cofiguration for the VM being created (at least in some cases). As such, for nova-compute to be able to generically process all interface types, we are proposing that this "configuration" attribute be present as a core attribute (and not as an extension attribute). If present, nova-compute will always make the core API call to "list port details", after making a call to "create port", and before creating the configuration file for the VM.

Also note that nova-compute uses a template file to construct the VM configuration and it is possible to change this template by way of configuring a flag in nova.conf. Using that option, one could craft different template files, depending on which network interface configuration needs to be used for a VM on that host. The attribute values which go into that interface configuration would either be already available to nova-compute, or could come by way of the "configuration" key-value pairs as described above.

Let us take the example of an 802.1Qbh interface configuration to make this clearer. Let's assume that "create network" and "create port" calls have already been made. Now, nova-compute makes a call to "list port details" and gets back the following response:

```
<Port>
        <port-id>xyz</id>
        <VIF-id>abc</id>
        <state>Unattached</state>
        <configuration>
             <conf key=”source_dev”>eth4</conf>
              <conf  key=”profile_id”>enterprise-profile</conf>
        </configuration>
</Port>
```

nova-compute can iterate through the key-value pairs of “configuration” and store them in a list (without having to understand them). If the keys used here correspond to the placeholders used in the template file, the corresponding values for those keys can be easily substituted in the template file to construct the VM configuration (just as it is done today with all other place holders in this template file, and again, without nova-compute understanding any of it). The example template file shown below, maps the keys "source_dev" and "profile_id" to placeholders, such that the values "eth4" and "enterprise-profile" will be substitued when nova-compute generates the VM configuration file.

```
<domain type='${type}'>
    <name>${name}</name>
    <memory>${memory_kb}</memory>
   ...
   ...
   <devices>
   ...
   ...
    <interface type='direct'>
      <mac address='${mac_address}'/>
      <source dev='${source_dev}' mode='private'/>
      <virtualport type='802.1Qbh'>
        <parameters profileid='${profile_id}'/>
      </virtualport>
      <model type='virtio'/>
            <filterref filter="nova-instance-${name}">
                <parameter name="IP" value="${ip_address}" />
                <parameter name="DHCPSERVER" value="${dhcp_server}" />
#if $getVar('extra_params', False)
                ${extra_params}
#end if
#if $getVar('ra_server', False)
                <parameter name="RASERVER" value="${ra_server}" />
#end if
            </filterref>
    </interface>

    ...
    ...
    </devices>
</domain>
```

Summary

The proposal outlined in this document is to:

a) Add the "configuration" element to the Port resource, and,

b) Establish the convention that nova-compute makes a call to obtain the port details after it creates the Port, and before it creates the VM configuration.