..

 This work is licensed under a Creative Commons Attribution 3.0 Unported

 License.

 http://creativecommons.org/licenses/by/3.0/legalcode

================================

Network Segment Range Management

================================

Launchpad blueprint:

https://blueprints.launchpad.net/neutron/+spec/network-segment-range-management

Currently, network segment ranges are configured as an entry in ML2 config

file [1]\_ that is statically defined for tenant network allocation and

therefore must be managed as part of the host deployment and management. When a

normal tenant user creates a network, Neutron assigns the next free

segmentation ID (VLAN ID, VNI etc.) from the configured segment ranges; only an

administrator can assign a specific segment ID via the provider extension.

This spec introduces an extension which exposes the segment range management to

be administered via the neutron API. In addition, it introduces the ability for

the administrator to control the segment ranges globally or on a per-tenant

basis.

Problem Description

===================

Self-service networks, aka tenant networks primarily enable general

(non-privileged) projects to manage networks without involving administrators.

These networks are entirely virtual and require virtual routers to interact

with provider and external networks such as the Internet. In most cases,

self-service networks use overlay protocols such as VXLAN or GRE because they

can support many more networks than layer-2 segmentation using VLAN tagging

(802.1q) which typically requires additional configuration of physical network

infrastructure. Nevertheless, in some other cases, OpenStack Networking also

supports the self-service network isolation using Flat and VLAN etc. [2]\_.

The current neutron implementation sets entries in a config file to statically

define network segment ranges for tenant network allocation. It does not

support another dynamic way (e.g. at the API level) after initialization.

Neutron services have to be restarted in order to make any change to the

network segment ranges take effect. However, cloud network infrastructure

deployments can be complex and non-homogeneous. This rigidity brings about

complexity and difficulty to cloud administrators when dealing with varied

requirements.

Network segment range management provides an added level of flexibility at the

API level which enables full network orchestration of all the types of tenant

networks using standard REST API rather than needing to interact with the host

configuration directly. Apart from this, for VLAN tenant networks especially,

network segment range management enables a control over the management of the

cloud/network infrastructure for tenant networks. It also facilitates

per-tenant assignments (including shared). This enables a cloud administrator

to directly control the VLAN tenant segment mapping, and ultimately the

underlying layer2/layer3 path of the tenant traffic without exposing specific

network segment range information to the tenant user. The allocations for

tenant networks will be allocated from the assigned projects of the network

segment ranges.

Several example use cases are demonstrated below.

UseCase I

---------

Enable cloud administrators to create and assign per-tenant network segment

ranges. This gives an administrator the privilege to manage the underlying

network segment ranges. When a tenant creates a network, it will allocate a

segment ID from the segment ranges assigned to the tenant or shared if no

tenant specific range available. It helps map the VMs created by that tenant to

map with an explicit set of existing networks, for privacy or dedicated

business connection needs. One example illustration is shown below:

::

 +------------------+

 | Physical Network |

 +------------+---------+ Infrastructures +-------------------+

 | | +---------+--------+ |

 | | | |

 | | | |

 +----+----+ +----+----+ +----+----+ +----+----+

 | Tenant 0| | Tenant 1|.........| Tenant k|.................| Tenant n|

 +----+----+ +----+----+ +----+----+ +----+----+

 | | | |

 | | | |

 + range-0 + range-1 + range-k1 + range-n

 range-k2

\* One cloud is connecting and mapping with a large number of existing l2

 physical network infrastructures.

\* n+1 tenants: Tenant 0, ...Tenant k..., Tenant n are available in this cloud.

\* Each tenant needs to be assigned with one (or more) network segment range(s)

 respectively (range-0, ...range-k1..., ...range-k2..., range-n) by the cloud

 administrator, so that the networks it creates can reach the corresponding

 dedicated physical network infrastructure it would like to connect to

 according to different business requirements.

A possible workflow is presented as follows:

1. Cloud administrator lists tenants available in the cloud:

 \* openstack project list

2. Cloud administrator lists all network segment ranges existing:

 \* openstack network segment range list

3. If no network segment range created for one target tenant, then cloud

 administrator could create one for it:

 \* openstack network segment range create

 --name <network\_segment\_range\_name>

 --shared <shared>

 --project <project\_id>

 --network\_type <network\_type>

 --physical\_network <physical\_network\_name>

 --range <network\_segment\_range>

4. Login as normal tenant user, create tenant networks by following the usual

 steps applicable [3]\_:

 \* openstack create network net

 \* The network created will automatically allocate a segment ID from the

 segment ranges assigned to the tenant (per step 3) or shared if no tenant

 specific range available.

This helps decide on choosing the correct VLAN mapping when multiple tenants

exist. It also provides the possibility to dynamically create a network segment

range for each tenant, even if it's not pre-deployed in the host configuration.

UseCase II

----------

In some VLAN tenant network scenarios, it is not uncommon that the existing

physical network infrastructures’ segment configurations have been changed.

Thus, it is a must for cloud administrators to update the VLAN allocation range

for tenant networks at the same time in order to keep aligned with the

connection or mapping. Hence offering the ability to dynamically manage segment

ranges of self-service networks, as this spec proposes, is imperative for VLAN

tenant networks and is definitely a plus for other overlay tenant networks.

A possible workflow is presented as follows:

1. Cloud administrator lists all network segment ranges existing and

 identifies the one needs to update:

 \* openstack network segment range list

2. Cloud administrator updates a network segment range based on the actual

 requirement changing:

 \* openstack network segment range update

 --name <network\_segment\_range\_name>

 --range <network\_segment\_range>

3. Login as normal tenant user, create tenant networks by following the usual

 steps applicable [3]\_:

 \* openstack create network net

 \* The network created will automatically allocate free segment ID from the

 updated network segment ranges available.

Proposed Change

===============

To address the above Use-cases, this spec introduces a new resource called

network\_segment\_ranges together with its implementation.

Currently by default, all pre-configured segment information (e.g.

network\_vlan\_ranges, vni\_ranges etc. defined in [1]\_) is loaded into

"ml2\_xxx\_allocations" DBs by ML2 type drivers once the Neutron services are up.

For all self-service networks’ segmentation ID sync, allocations and

releases, they will be based on this information.

All this information will be maintained. The network segment ranges introduced

in this spec will augment this initial allocation that is loaded from the

configuration. It proposes an API way that user with administrative privilege

can create and manage various network segment ranges for all network types

supported by ML2.

Data Model Impact

-----------------

The following new table is added as part of the network network management

feature::

 CREATE TABLE network\_segment\_ranges (

 id CHAR(36) NOT NULL PRI KEY,

 name VARCHAR(255),

 shared BOOL NOT NULL,

 project\_id VARCHAR(255) NOT NULL,

 network\_type ENUM('flat', 'vlan', 'vxlan', 'gre', 'geneve') NOT NULL,

 physical\_network VARCHAR(64),

 minimum INT,

 maximum INT

 );

For different network types, the validation strategies and responses should

have the following variants:

\* FLAT: minimum and maximum integers input are not allowed. A forbidden or

 failed response should return if one of which is given.

\* VLAN: minimum = 1, maximum = 4094.

\* VXLAN: minimum = 1, maximum = 2 \*\* 24 - 1.

\* GRE: minimum = 1, maximum = 2 \*\* 32 - 1.

\* Geneve: minimum = 1, maximum = 2 \*\* 24 - 1.

Notes: Other validation rules like minimum <= maximum are always applicable and

should be paid attention to. Most of the cited above have been supported in

neutron\_lib.plugins.utils.

Mixin classes to add the network segment range management extension should be

provided. The DB operation logic should be handled by the ML2 type manager and

the type drivers. For the values present in the existing ML2 configuration

options [1]\_ (e.g. ml2\_type\_vlan, ml2\_type\_vxlan etc.), they will be loaded as

`shared` segment ranges into segment\_range DB in order to provide backward

compatibility for initial deployment when this extension is present.

Resource Extension

------------------

The following new resource is being introduced and its attributes maps would be

like:

.. code-block:: python

 NETWORK\_TYPE\_LIST = [TYPE\_FLAT, TYPE\_VLAN, TYPE\_VXLAN, TYPE\_GRE, TYPE\_GENEVE]

 RESOURCE\_ATTRIBUTE\_MAPS = {

 'network\_segment\_ranges': {

 'id': {'allow\_post': False, 'allow\_put': False, 'is\_visible': True},

 'name': {'allow\_post': True, 'allow\_put': True, 'is\_visible': True,

 'default': None,

 'validate': {

 'type:not\_empty\_string': db\_const.NAME\_FIELD\_SIZE}},

 'shared': {'allow\_post': True, 'allow\_put': False,

 'convert\_to': converters.convert\_to\_boolean,

 'is\_visible': True, 'default': True},

 'project\_id': {'allow\_post': True, 'allow\_put': False,

 'validate': {

 'type:string': db\_const.PROJECT\_ID\_FIELD\_SIZE},

 'required\_by\_policy': True,

 'is\_filter': True,

 'is\_sort\_key': True,

 'is\_visible': True},

 'network\_type': {'allow\_post': True, 'allow\_put': False,

 'validate': {'type:values': NETWORK\_TYPE\_LIST},

 'default': constants.ATTR\_NOT\_SPECIFIED,

 'enforce\_policy': True,

 'is\_filter': True,

 'is\_visible': True},

 'physical\_network': {'allow\_post': True, 'allow\_put': False,

 'validate': {

 'type:string': PHYSICAL\_NETWORK\_MAX\_LEN},

 'default': constants.ATTR\_NOT\_SPECIFIED,

 'enforce\_policy': True,

 'is\_filter': True,

 'is\_visible': True},

 'minimum': {'allow\_post': True, 'allow\_put': True,

 'convert\_to': converters.convert\_to\_int, 'is\_visible': True},

 'maximum': {'allow\_post': True, 'allow\_put': True,

 'convert\_to': converters.convert\_to\_int, 'is\_visible': True},

 },

 }

REST API Impact

---------------

The idea is to add a new resource extension with the below defined attributes.

Resource extension network\_segment\_ranges:

+-----------------+--------+-----+--------+-----------+-----------------------+

| Attribute Name | Type | Req | CRUD | Default | Description |

| | | | | Value | |

+=================+========+=====+========+===========+=======================+

| id | String | N/A | R | | Identifier of network |

| | | | | | segment range |

+-----------------+--------+-----+--------+-----------+-----------------------+

| name | String | No | CRU | None | Name of network |

| | | | | | segment range |

+-----------------+--------+-----+--------+-----------+-----------------------+

| shared | Bool | Yes | CR | False | Shared with other |

| | | | | | projects |

+-----------------+--------+-----+--------+-----------+-----------------------+

| project\_id | String | No | CR | Current | Owner of network |

| | | | | project\_id| range. Optional when |

| | | | | | `shared` is True. |

+-----------------+--------+-----+--------+-----------+-----------------------+

| network\_type | Enum | Yes | CR | 1st tenant| Flat, VLAN, VxLAN, |

| | | | | network | GRE, Geneve |

| | | | | type [1]\_ | |

+-----------------+--------+-----+--------+-----------+-----------------------+

| physical\_network| String | No | CR | None | Optional. Only |

| | | | | | applicable for Flat, |

| | | | | | VLAN. |

+-----------------+--------+-----+--------+-----------+-----------------------+

| minimum | INT | Yes | CRU | None | Floor integer of the |

| | | | | | segment range |

+-----------------+--------+-----+--------+-----------+-----------------------+

| maximum | INT | Yes | CRU | None | Ceiling integer of the|

| | | | | | segment range |

+-----------------+--------+-----+--------+-----------+-----------------------+

To specify a range with single item, min equals to max can do the trick. For

discrete segment ranges of one given network type, they are represented as

several ones, each with a min and a max.

The following network segment range management Rest APIs will be provided in

line with the new resources previously introduced:

\* List all network segment ranges.

 GET /v2.0/network\_segment\_ranges

::

 GET /v2.0/network\_segment\_ranges

 Accept: application/json

 {

 "network\_segment\_ranges": [

 {

 "id": "d23abc8d-2991-4a55-ba98-2aaea84cc72f",

 "name": "network\_segment\_range\_physnet1",

 "shared": False,

 "project\_id": "45977fa2dbd7482098dd68d0d8970117",

 "network\_type": "vlan",

 "physical\_network": "physnet1",

 "minimum": 100,

 "maximum": 200,

 }

 ]

 }

\* List a network segment range information.

 GET /v2.0/network\_segment\_ranges/<network\_segment\_range-id>

::

 GET /v2.0/network\_segment\_ranges/d23abc8d-2991-4a55-ba98-2aaea84cc72f

 Accept: application/json

 {

 "network\_segment\_range": {

 "id": "d23abc8d-2991-4a55-ba98-2aaea84cc72f",

 "name": "network\_segment\_range\_physnet1",

 "shared": False,

 "project\_id": "45977fa2dbd7482098dd68d0d8970117",

 "network\_type": "vlan",

 "physical\_network": "physnet1",

 "minimum": 100,

 "maximum": 200,

 }

 }

\* Create a network segment range for a given tenant.

 POST /v2.0/network\_segment\_ranges/<network\_segment\_range-id>

::

 POST /v2.0/network\_segment\_ranges/d23abc8d-2991-4a55-ba98-2aaea84cc72f

 Accept: application/json

 {

 "network\_segment\_range": {

 "name": "network\_segment\_range\_physnet1",

 "shared": False,

 "project\_id": "45977fa2dbd7482098dd68d0d8970117",

 "network\_type": "vlan",

 "physical\_network": "physnet1",

 "minimum": 100,

 "maximum": 200,

 }

 }

\* Delete a network segment range by id.

 DELETE /v2.0/network\_segment\_ranges/<network\_segment\_range-id>

::

 DELETE /v2.0/network\_segment\_ranges/d23abc8d-2991-4a55-ba98-2aaea84cc72f

 Accept: application/json

\* Update a network segment range with given data.

 PUT /v2.0/network\_segment\_ranges/<network\_segment\_range-id>

::

 PUT /v2.0/network\_segment\_ranges/d23abc8d-2991-4a55-ba98-2aaea84cc72f

 Accept: application/json

 {

 "network\_segment\_range": {

 "minimum": 200,

 "maximum": 300,

 }

 }

Command Line Client Impact

--------------------------

Openstack Client would add network segment range management related CLIs. They

should be admin only CLI commands. For example:

\* openstack network segment range list

\* openstack network segment range show <network\_segment\_range-id>

\* openstack network segment range create

 [--name <range\_name>]

 [--shared <shared>]

 [--project\_id <project\_id>]

 [--network\_type <network\_type>]

 [--physical\_network <physical\_network>]

 [--minimum <range\_minimum>]

 [--maximum <range\_maximum>]

 <network\_segment\_range>

 \* Argument --name is optional; --project\_id is optional when `shared` is

 True; --physical\_network is optional and only applicable for FLAT and VLAN.

 All the other parameters should be required.

\* openstack network segment range update [--name <range\_name>]

 [--minimum <range\_minimum>] [--maximum <range\_maximum>] <project\_id>

\* openstack network segment range delete <network\_segment\_range-id>

Other Impact

------------

\* ML2 plugin, plugin manager and type drivers will need to be refined and added

 with several new methods correspondingly in order to support this feature.

\* The existing ML2 configuration will populate the proposed

 network\_segment\_ranges as a shared range. When this extension is loaded, a

 change is expected in the processing of the ML2 configuration.

\* Validation work is needed for quite a few cases, including but not limited

 to:

 \* Admin privilege should always be checked before performing any network

 segment range operation cited previously.

 \* When updating or deleting one network segment range, the operation should

 be rejected if one of the network segments is in use.

 \* We're maintaining the consistency by "ml2\_xxx\_allocations" DBs and relying

 on them to do the eventual validation and the underlying segment

 allocation. This means network segment ranges can be configured before

 agents are actually mapped to specific physical network mappings.

\* Theoretically, multiple network segment ranges can be created for one

 tenant (while one network segment range cannot be owned by several tenants).

 If a tenant has more than one segment range configured, it would pick up the

 next free segmentation ID (VLAN ID, VNI etc.) from all its owned network

 segment ranges.

 Backwards compatibility comes from having the default behavior of segment

 ranges being assigned as a `shared` resource to tenants. If both of `shared`

 and `specified` segment range resources are exposed to a tenant, the

 `specified` should override the `shared`.

\* The extension will be optional (an option in [4]\_ with functionality disabled

 by default should be added), in which case the existing ML2 configuration

 options will be applicable.

Other End User Impact

---------------------

Users with admin privilege will be able to dynamically manage network segment

ranges for all ML2 supported network types, all tenants and tenant networks.

If no dynamic network segment range is created for a given tenant, or the

feature is disabled due to the backwards compatibility consideration, there

will be no impact to end users.

Other Deployer Impact

---------------------

Deployers should have an option available to enable or disable this

functionality so that they can continue to use the configuration file as

before. They also need to be strongly warned to update their operational

documentation to ensure that the new network segment information is managed

using the correct facility. If the feature is disabled, nothing at the

deployment level would be impacted.

Performance Impact

------------------

Performance testing must be conducted to see what is the overhead of enabling

this feature, of course that if the feature is disabled no performance impact

should be noticed.

Implementation

==============

Assignee(s)

-----------

\* Kailun Qin <kailun.qin@intel.com>

Work Items

----------

\* Adjust the DB model and add the new table.

\* Extend current API.

\* Modify type drivers as well as all related references.

\* Add related tests.

\* Add CLI openstackclient.

\* Documentation.

Dependencies

============

None

Testing

=======

Unit tests, functional tests, API tests and scenario tests are necessary.

Documentation Impact

====================

The Neutron API reference will need to be updated.

References

==========

.. [1] /etc/neutron/plugins/ml2/ml2\_conf.ini

.. [2] `OpenStack Networking`

 https://docs.openstack.org/neutron/latest/admin/intro-os-networking.html

.. [3] `Self-service network`

 https://docs.openstack.org/newton/install-guide-rdo/launch-instance-networks-selfservice.html

.. [4] /etc/neutron/neutron.conf

Related Information

-------------------

Neutron v2 API: https://developer.openstack.org/api-ref/network/v2/