We have based this off the Atlas API but made changes to it to reflect the use cases provided by users and operators of Neutron LBaaS and also off of things we have learned. SSL Termination, SSL Re-encryption, and Content Switching (L7 Switching) are not being shown. There is a content switching section with an idea we have but it is not perfect. We also have ideas for SSL Termination and SSL Re-encryption through this API but we can visit that later if this does go forward. Also, if there are attributes missing or features missing it is probably due to us wanting to show the most common use cases. We feel all of those will fall into place once a draft is agreed upon. First thing is first though, reasons we made some of the decisions we made:

Q: Why only one port and one protocol per load balancer?
A: One port and one protocol simplifies everything. Sharing VIPs across load balancers will allow for multiple ports through the same VIP. The only use cases we could come up with for even having multiple ports on a load balancer is SSL Termination. However, this can still be accomplished with sharing VIPs. If there are any use cases we are not thinking about please let us know.

Q: Why only one pool per load balancer?
A: One pool per load balancer also simplifies everything. The ony use case we can come up with is content switching (L7 switching) and active/passive pools. Active/passive pools can probably be accomplished by using PRIMARY and SECONDARY member types. Content switching is definitely a good reason to have multiple pools but we wanted the load balancer to always be consistent and it sometimes having a "pool" and other times having "pools" is not consistent. If load balancers took multiple pools just for the sake of being consistent because of content switching then that can be confusing to a user that does not use content switching. We have thought of an option for content switching that is outlined below but it is definitely not perfect. We would prefer a perfect and elegant solution so this needs more thought and time.

Q: Why not share pool members?
A: Pool member status is in relation to a pool
Ex: Draining state for a node may not need drained in the other pool reference

Q: Why not share health monitors?
A: Sharing a health monitor could remove a healthy node from a pool
Ex: Actual server running multiple services where one fails would remove the node accross all pools being monitored where should only be from the one pool

Q: Should the pool take a network id instead of a subnet id?
A:

==========LOAD BALANCERS==========

POST /loadbalancers

Description: Creates a single load balancer in one call. VIPs and Pools can be created on the fly by providing their appropriate attributes to create a new one. If an existing VIP or Pool is wanted to be used then providing just the id attribute in that entity's body is required. If using an existing VIP, the load balancer protocol must not already be in use by another load balancer using that same VIP.

Request Body:

```
{
   "loadbalancer":{
      "name": "Single Call LB",
      "port":80,
      "protocol": "HTTP",
      "vips":[
         {
            "type":"IPv4",
            "subnet_id":"SUBNET_UUID"
         }
      ],
      "pool":{
         "name": "pool1”,
         “subnet_id”: “SUBNET_UUID”,
         "algorithm": "ROUNDROBIN",
         "session_persistence": "HTTP_COOKIE",
         "members": [{
            "ip": "10.1.1.1",
            "port": 80,
            "enabled": true
         }],
         "health_monitor": {
            "type": "HTTP",
            "delay": 1,
            "timeout": 10,
            "interval": 1
         }
      }
}
```

Response Body:

```
{"loadbalancer": {
      "id":"LB_UUID",
      ...
      "vips":[{
         "id":"VIP_UUID",
         ...
```

```
      }],
      "pool": {
         "id": "POOL_UUID",
         ...
         "members":[{
            "id": "MEMBER_UUID",
            ...
         }]
      }
   }
}
```

GET /loadbalancers
Description: Lists all load balancers.
Request Body:
None
Response Body:

```
{"loadbalancers": [{
      "id":"LB_UUID",
      ...
      "vips":[{
         "id":"VIP_UUID",
         ...
      }],
      "pool": {
         "id": "POOL_UUID",
         ...
         "members":[{
            "id": "MEMBER_UUID",
            ...
         }]
      }
      "content_switching": {
         "enabled":true,
         "pools":[{
            "id": "POOL_UUID",
            ...
            "members": [{
               "id": "MEMBER_UUID",
               ...
            }],
            "rules": [{
               "id": "RULE_UUID",
```

```
                ...
            }]
         }]
      }
   }]
}
```

PUT /loadbalancers
Disallowed

DELETE /loadbalancers
Disallowed

POST /loadbalancers/{lb_id}
Disallowed

GET /loadbalancers/{lb_id}
Description: Shows the details of the configuration of a specific load balancer.
Request Body:
None
Response Body:

```
{"loadbalancer": {
      "id":"LB_UUID",
      ...
      "vips":[{
         "id":"VIP_UUID",
         ...
      }],
      "pool": {
         "id": "POOL_UUID",
         ...
         "members":[{
            "id": "MEMBER_UUID",
            ...
         }]
      }
      "content_switching": {
         "enabled":true,
         "pools":[{
            "id": "POOL_UUID",
            ...
            "members": [{
               "id": "MEMBER_UUID",
```

```
            ...
         }],
         "rules": [{
            "id": "RULE_UUID",
            ...
         }]
      }]
   }
}
```

PUT /loadbalancers/{lb_id}
Description: Updates the attributes of an existing load balancer.
Request Body

```
{"loadbalancer": {
   "port": 81,
   "protocol": "HTTP"
   }
}
```

DELETE /loadbalancers/{lb_id}
Description: Completely deletes a load balancer. Any pools or VIPs that this load balancer is using will be detached but not deleted. These must be deleted using the /vips and the /pools resources.
Request Body
None
Respone Body
None

==========LOAD BALANCER VIPS==========

POST /loadbalancers/{lb_id}/vips
Option #1: Creates and attaches a virtual IP to an existing load balancer.
Request Body:

```
{"vip":{
   "type": "IPv4",
   "subnet_id": "SUBNET_UUID"
   }
}
```

Option #2: Attaches an existing virtual IP to an existing load balancer.
Request Body:

```
{"vip":{
```

```
   "id": "VIP_UUID"
   }
}
```
Response Body:
```
{"vip":{
   "id": "VIP_UUID",
   ...
   }
}
```

GET /loadbalancers/{lb_id}/vips
Description: Lists all virtual IPs attached to an existing load balancer.
Request Body:
None
Response Body:
```
{"vips": [{
   "id": "VIP_UUID",
   ...
   },{
   "id": "VIP_UUID",
   ...
   }]
}
```

PUT /loadbalancers/{lb_id}/vips
Disallowed

DELETE /loadbalancers/{lb_id}/vips
Disallowed
POST /loadbalancers/{lb_id}/vips/{vip_id}
Disallowed

GET /loadbalancers/{lb_id}/vips/{vip_id}
Description: Shows details about an existing virtual IP attached to an existing load balancer.
Request Body:
None
Response Body
```
{"vip":{
   "id": "VIP_UUID",
   ...
   }
}
```

PUT /loadbalancers/{lb_id}/vips/{vip_id}
Disallowed. Update a virtual IP by using /vips/{vip_id}

DELETE /loadbalancer/{lb_id}/vips/{vip_id}
Description: Detaches an existing virtual IP from an existing load balancer. Does not actually delete the virtual IP.
Request Body:
None
Response Body:
None

==========LOAD BALANCER POOLS==========

POST /loadbalancers/{lb_id}/pools
Option #1: Creates and attaches a new pool to an existing load balancer.
Request Body:

```
{"pool":{
   "name":"pool1",
   "algorithm":"ROUNDROBIN",
   "session_persistence":"SOURCE_IP",
   "subnet_id":"SUBNET_UUID",
   "members": [{
      "ip":"10.1.1.1",
      "port": 80
      }]
   }
}
```

Option #2: Attaches an existing pool to an existing load balancer.
Request Body:

```
{"pool": {
   "id":"POOL_UUID"
   }
}
```

Response Body:

```
{"pool": {
   "id": "POOL_UUID",
   ...
   }
}
```

GET /loadbalancer/{lb_id}/pools
Description: Lists all pools attached to an existing load balancer.

Request Body:
None
Response Body:

```
{"pools": [{
   "id": "POOL_UUID",
   ...
   }]
}
```

PUT /loadbalancers/{lb_id}/pools
Disallowed

DELETE /loadbalancers/{lb_id}/pools
Disallowed

POST /loadbalancers/{lb_id}/pools/{pool_id}
Disallowed

GET /loadbalancers/{lb_id}/pools/{pool_id}
Description: Shows details of an existing pool attached to an existing load balancer.
Request Body:
None
Response Body:

```
{"pool": {
   "id": "POOL_UUID",
   ...
   }
}
```

PUT /loadbalancers/{lb_id}/pools/{pool_id}
Disallowed. Updates to a pool should be done through /pools/{pool_id}.

DELETE /loadbalancers/{lb_id}/pools/{pool_id}
Description: Detaches an existing pool from an existing load balancer. Does not delete the pool. That is done through the /pools/{pool_id} resource.
Request Body:
None
Response Body:
None

==========VIPS==========

POST /vips

Description: Creates a new virtual IP.
Request Body:

```
{"vip":{
   "type": "IPv4",
   "subnet_id": "SUBNET_UUID"
   }
}
Response Body:
{"vip":{
   "id": "VIP_UUID",
   ...
   }
}
```

GET /vips
Description: List all virtual IPs with back references to the load balancers in which they are attached.
Request Body:
None
Response Body:

```
{"vips":[{
   "id": "VIP_UUID",
   "loadbalancers":[LB_UUID, LB_UUID],
   ...
   },{
   "id": "VIP_UUID",
   "loadbalancers":[LB_UUID, LB_UUID],
   ..
   }]
}
```

PUT /vips
Disallowed

DELETE /vips
Disallowed

POST /vips/{vip_id}
Disallowed

GET /vips/{vip_id}
Description: Shows details of a virtual IP with back references to the load balancers in which it is attached..

Request Body:
None
Response Body:

```
{"vip": {
   "id": "VIP_UUID",
   "loadbalancers":[LB_UUID, LB_UUID],
   ...
   }
}
```

PUT /vips/{vip_id}
Disallowed for now pending further discussion.

DELETE /vips/{vip_id}
Description: Completely deletes a virtual IP only if it is not attached to any load balancers.
Request Body:
None
Response Body:
None

==========POOLS==========

POST /pools
Description: Creates a new pool.
Request Body:

```
{"pool":{
   "name":"pool1",
   "algorithm":"ROUNDROBIN",
   "session_persistence": "SOURCE_IP",
   "subnet_id":"SUBNET_UUID",
   "members":[{
      "ip":"10.1.1.1",
      "port": 80,
      "enabled": true"
    }]
   }
}
```

Response Body:

```
{"pool":{
   "id": "POOL_UUID",
   "..."
```

```
   "members":[{
      "id": "MEMBER_UUID",
      ...
   }]
   }
}
```

GET /pools
Description: Lists all pools with back references to the load balancers in which they are attached.
Request Body:
None
Response Body:

```
{"pools":[
   {
      "id": "POOL_UUID",
      "loadbalancers":[LB_UUID, LB_UUID],
      ...
   },
   {
      "id": "POOL_UUID",
      "loadbalancers":[LB_UUID, LB_UUID],
      ...
   }
  ]
}
```

PUT /pools
Disallowed

DELETE /pools
Disallowed

POST /pools/{pool_id}
Disallowed

GET /pools/{pool_id}
Description: Shows the details of an existing pool with back references to the load balancers in which it is attached.
Request Body:
None
Response Body:

```
{"pool":{
```

```
   "id": "POOL_UUID",
   "loadbalancers":[LB_UUID, LB_UUID],
   ...
   }
}
```

PUT /pools/{pool_id}
Description: Updates the attributes of an existing pool.
Request Body:

```
{"pool":{
   "algorithm": "LEAST_CONNECTIONS",
   "session_persistence":"HTTP_COOKIE"
   }
}
```

Response Body:

```
{"pool":{
   "id": "POOL_UUID",
   ...
   }
}
```

DELETE /pools/{pool_id}
Description: Completely deletes a pool only if that pool is not attached to any load balancers. Must detach the pool from all load balancers before it can be deleted.
Request Body:
None
Response Body:
None

==========POOL MEMBERS==========

POST /pools/{pool_id}/members
Description: Adds a new member to an existing pool.
Request Body:

```
{"member":{
   "ip":"10.1.1.1",
   "port": 80,
   "enabled": true
   }
}
```

Response Body:

```
{"member":{
```

```
   "id":"MEMBER_UUID",
   ...
   }
}
```

GET /pools/{pool_id}/members
Description: Lists all the members in an existing pool.
Request Body:
None
Response Body:

```
{"members": [{
   "id": "MEMBER_UUID",
   ...
   }]
}
```

PUT /pools/{pool_id}/members
Disallowed

DELETE /pools/{pool_id}/members
Disallowed

POST /pools/{pool_id}/members/{member_id}
Disallowed

GET /pools/{pool_id}/members/{member_id}
Description: Shows details of an existing pool member.
Request Body:
None
Response Body:

```
{"member": {
   "id": "MEMBER_UUID"
   ...
   }
}
```

PUT /pools/{pool_id}/members/{member_id}
Description: Updates the attributes of an existing pool member.
Request Body:

```
{"member": {
   "ip": "10.1.1.1",
   "port": 80,
   "enabled": true
```

```
    }
}
```
Response Body:
```
{"member": {
    "id": "MEMBER_UUID"
    ...
    }
}
```

DELETE /pools/{pool_id}/members/{member_id}
Description: Completely deletes a pool member.
Request Body:
None
Response Body:
None

==========HEALTH MONITORS==========

POST /pools/{pool_id}/health_monitor
Description: Creates a health monitor on an existing pool.
Request Body:
```
{"health_monitor":{
    "type": "PING",
    "delay": 1,
    "interval": 1,
    "timeout": 10
    }
}
```
Response Body:
```
{"health_monitor":{
    "type": "PING",
    "delay": 1,
    "interval": 1,
    "timeout": 10
    }
}
```

GET /pools/{pool_id}/health_monitor
Description: Shows the details of an existing pool's health monitor.
Request Body:
None
Response Body:

```
{"health_monitor":{
   "type": "PING",
   "delay": 1,
   "interval": 1,
   "timeout": 10
   }
}
```

PUT /pools/{pool_id}/health_monitor
Description: Updates attributes of an existing pool's health monitor.
Request Body:

```
{"health_monitor":{
   "delay": 2
   }
}
```

Response Body:

```
{"health_monitor":{
   "type": "PING",
   "delay": 2,
   "interval": 1,
   "timeout": 10
   }
}
```

DELETE /pools/{pool_id}/health_monitor
Description: Removes a health monitor from a load balancer.
Request Body:
None
Response Body:
None

==========CONTENT SWITCHING==========

Content switching is a work in progress. The option below is something we came up and has a 1:1 relationship to a load balancer.

Q: Rules can't be directly attached to a pool due to pool sharing. If one load balancer wants a pool to have different rules than another load balancer then that creates a conflict. How to handle this?

Q: How do we handle multiple rules on a pool? Are they all AND or are they all OR or should we having another way to define ANDs and ORs? DSL?

POST /loadbalancers/{lb_id}/content_switching
Option #1: Adds content switching to an existing load balancer using pools created in this call.
Request Body:

```
{"content_switching": {
      "enabled": true,
      "pools": [{
         "name": "pool2",
         "subnet_id": "SUBNET_UUID",
         "members": [{
            "ip":"10.1.1.2",
            "port": 80,
            "enabled": true
         },
         "rules":[{
            "type": "path",
            "match": "/index.html"
         }]
      }]]
   }
}
```

Option #2: Adds content switching to an existing load balancer using existing pools.
Request Body:

```
{"content_switching": {
      "enabled": true,
      "pools": [{
         "id": "POOL_UUID,
         "rules":[{
            "type": "path",
            "match": "/index.html"
         }]
      }]]
   }
}
```

Response Body:

```
{"content_switching": {
      "enabled": true,
      "pools": [{
         "id": "POOL_UUID",
         "name": "pool2",
         "subnet_id": "SUBNET_UUID",
         "members": [{
            "ip":"10.1.1.2",
            "port": 80,
```

```
            "enabled": true
         },
         "rules":[{
            "id": "RULE_UUID",
            "type": "path",
            "match": "/index.html"
         }]
      }]]
   }
}
```

GET /loadbalancers/{lb_id}/content_switching
Description: Shows details about content switching on an existing load balancer.
Request Body:
None
Response Body:

```
{{"content_switching": {
      "enabled": true,
      "pools": [{
         "id": "POOL_UUID",
         "name": "pool2",
         "subnet_id": "SUBNET_UUID",
         "members": [{
            "ip":"10.1.1.2",
            "port": 80,
            "enabled": true
         },
         "rules":[{
            "id": "RULE_UUID",
            "type": "path",
            "match": "/index.html"
         }]
      }]]
   }
}
```

PUT /loadbalancers/{lb_id}/content_switching
Description: Updates attributes of content switching on an existing load balancer.
Request Body:

```
{"content_switching":{
   "enabled": false
   }
}
```

Response Body:

```
{"content_switching": {
      "enabled": false,
      "pools": [{
         "id": "POOL_UUID",
         "name": "pool2",
         "subnet_id": "SUBNET_UUID",
         "members": [{
            "ip":"10.1.1.2",
            "port": 80,
            "enabled": true
         },
         "rules":[{
            "id": "RULE_UUID",
            "type": "path",
            "match": "/index.html"
         }]
      }]]
   }
}
```

DELETE /loadbalancers/{lb_id}/content_switching
Description: Completely removes content switching and its configuration from an existing load balancer.
Request Body:
None
Response Body:
None

POST /loadbalancers/{lb_id}/content_switching/pools
Option #1: Creates and attaches a new pool to the content switching on an existing load balancer.
Request Body:

```
{"pool": {
      "name": "pool2",
      "subnet_id": "SUBNET_UUID",
      "members": [{
         "ip":"10.1.1.2",
         "port": 80,
         "enabled": true
      },
      "rules":[{
         "type": "path",
```

```
         "match": "/index.html"
      }]
   }
}
```

Option #2: Attaches an existing pool to the content switching on an existing load balancer.

```
{"pool": {
      "id":"POOL_UUID",
      "rules":[{
         "type": "path",
         "match": "/index.html"
      }]
   }
}
Response Body:
{"pool": {
      "id": "POOL_UUID",
      "name": "pool2",
      "subnet_id": "SUBNET_UUID",
      "members": [{
         "ip":"10.1.1.2",
         "port": 80,
         "enabled": true
      },
      "rules":[{
         "id": "RULE_UUID",
         "type": "path",
         "match": "/index.html"
      }]
   }
}
```

==========CLI Examples==========

To fully create a load balancer through CLI:

Create Vip:
neutron lb-vip-create --subnet_id “SUBNET_UUID” --type “IPv4”

Create Pool:
neutron lb-pool-create --name “pool1” --subnet_id “SUBNET_UUID” --algorithm “ROUNDROBIN” --session_persistence “SOURCE_IP”

Add Member To Pool

neutron lb-pool-member-add --ip “10.1.1.1” --port “80” --pool_id “POOL_UUID”

Create LB:
neutron lb-create --name “lb1” --vip_id “VIP_UUID” --pool_id “POOL_UUID” --protocol “HTTP” port “80”