Climate REST API

Lease Service API

- Lease
  Lease is the basic object in our component. Using this type of object you can create single reservation of different virtual, or physical resources.

  Lease REST API ops

Veb	Link	Description
GET	/v1/leases	List all leases
GET	/v1/leases/{lease_id}	Show specified lease info
POST	/v1/leases	Add new lease
PUT	/v1/leases/{lease_id}	Update (prolong) lease
DEL	/v1/leases/{lease_id}	Destroy lease

  Create action
  When we want create lease of virtual resource we need to know 4 things:
  - Type of resource
  - Resource spec (I mean parameters)
  - Time to start lease
  - Time to end lease/period
  - Default actions when the lease will end (notifications, etc.)

Request body (virtual:instance)

```
{
    "name": "lease_foo",
    "start_date": "1234",
    "end_date": "2345",
    "reservations": [
        {
            "id": "fake_resource_id",
            "resource_id": "1234-1234-1234",
            "resource_type": "virtual:instance",
        }
```

```
        ],
                "events":[
                        {
                                "event_type": "notification",
                                "event_date":  "3456"
                        },
                        {
                                "event_type": "notification",
                                "event_date":  "4567"
                        }
                ]
                }
```

Request body (physical:host)

```
    {
                "name": "lease_foo",
                "start_date": "1234",
                "end_date": "2345",
                "reservations": [
                        {
                            "id": "fake_resource_id",
                            "resource_id": "1234-1234-1234",
                            "resource_type": "physical:host",
                            "best-effort":"true|false",
                            "min":"10",
                            "max":"15",

                            "resource-properties": ["and", [">=","$ram_gb","64"],
                        [">=","$usable-cores","16"],["=","cpu-model","SandyBridge"]
                        ,[">=","usable-gpus","4"],["=","gpu-model","k20"]]
                            "hypervisor-properties": ["and",
                        ["=","hypervisor-type","xen"], ["=",
                        ["cpu-overcommitment-ratio", "1:1"],
                        [ram-overcommitment-ratio", "1:1"]]
                        }
        ],
                "events":[
                        {
                                "event_type": "notification",
                                "event_date":  "3456"
                        },
                        {
```

“event_type”: “notification”,
“event_date”: “4567”
}
]
}

Request:
POST /v1/leases

Normal Response Code: 202 (ACCEPTED)

Destroy

When we want to end lease manually, or the time has come we need one parameter lease ID that will prove that we have rights to end the lease.

Request body:
{lease_id: xxxx-xxxx-xxxx}

Request:
DEL /v1/leases/{lease_id}

Normal Response Code: 204 (NO CONTENT)

Update (prolong)

We want to prolong our lease, so we want to take 3 parameters lease ID, token and new time of end/period.

Request body:
{lease_id: xxxx-xxxx-xxxx,
end_date: timestamp}

Request:
PUT /v1/leases/{lease_id}

Normal Response Code: 200 (OK)

List

For some undiscovered reasons you may want to list your lease, and we will give you such ability.

Request:
GET /v1/leases

Normal Response Code: 200 (OK)


```
{
  "leases": [
    {
      "id": "aaaa-bbbb-cccc-dddd",
      "name": "lease_foo",
      "start_date": "1234",
      "end_date": "2345",
      "reservations": [
        {
          "id": "qrdjndlfb",
          "lease_id": "aaaa-bbbb-cccc-dddd",
          "resource_id": "12346789876543",
          "resource_type": "virtual:instance",
          "status": "Reserved"
        }
      ]
    },
    {
      "id": "asjdaldb",
      "name": "lease_foo_2",
      "start_date": "1234",
      "end_date": "2345",
      "reservations": [
        {
          "id": "aslflf",
          "lease_id": "aaaa-bbbb-cccc-dddd",
          "resource_id": "12346789876543",
          "resource_type": "physical:host",
          "status": "Reserved"
        }
      ]
    }
  ]
}
```


Get info

May be one day you want to remember of your lease and that’s what you need.

Request:
GET /v1/leases/aaaa-bbbb-cccc-dddd

Normal Response Code: 200 (OK)


```
{
  "id": "aaaa-bbbb-cccc-dddd",
  "name": "lease_foo",
  "start_date": "1234",
  "end_date": "2345",
  "reservations": [
    {
      "id": "qrdjndlfb",
      "lease_id": "aaaa-bbbb-cccc-dddd",
      "resource_id": "12346789876543",
      "resource_type": "virtual:instance",
      "status": "Reserved"
    }
  ]
}
```


Request:
GET /v1/leases/{lease_id}

List plugins for the lease
(now we'll have one plugin for one lease, because in the most simple case one resource will be reserved in one lease)
GET /v1/leases/{lease_id}/plugins
200 OK


```
{
        plugins: [
                {
                        "id": "aaaa-bbbb-cccc-dddd",
                        "name": "plugin_name_1",
                        "resource_type": "virtual:instance",
                        “description": "Starts VM when lease begins and deletes it when
                lease ends."
                }
        ]
}
```


- Plugins & events

  List plugins

_____It's needed for UI side.

List events (admin-only)
_____It's needed for admins/devops.

TBD

- add list of possible statuses for both physical and virtual resources

Hosts Reservation Admin API

Description

Provides an ability to mark Nova compute hosts reservable and to prevent Nova boot vm on them (using aggregates or smth else). I think that it should look like hosts selector. In fact it's an API to define pool of compute hosts that could be reserved by end users.

API

(Reservable hosts are inside a Pcloud, so the APIs used to create it are either the Nova API or the Pcloud API)

Hosts Reservation User API

Description

Provides an ability to reserve Nova compute hosts from the pool defined using "Hosts Reservation Admin API". These reserved hosts could be used for instances provisioning in future. This API will use "Lease Service API" to register start/end date and actions that should be occured on the specified host.

API

Arguments :

- Number of hosts
- Capabilities (Json format ?). They should match the extra-specs of the eligible aggregates.

Veb	Link	Description
POST	/v1/leases	Add new lease (host properties as param)
GET	/v1/flavors	List the available host “flavors”

POST /v1/leases:

```
{
    "name": "lease_foo",
    "start_date": "1234",
    "end_date": "2345",
    "reservations":
        {
            "id": "fake_resource_id",
            "resource_id": "1234-1234-1234",
```

```
                    "resource_type": "physical:host",
                    “best-effort”:”true|false”,
                    “min”:”10”,
                    “max”:”15”,

                    “resource-properties”: [“and”, [">=","$ram_gb",”64”],
                [“>=”,”$usable-cores”,”16”],[“=”,”cpu-model”,”SandyBridge”]
                ,[“>=”,“usable-gpus”,”4”],[“=”,”gpu-model”,”k20”]]
                    “hypervisor-properties”: [“and”,
                [“=”,”hypervisor-type”,”xen”], [“=”,
                [“cpu-overcommitment-ratio”, “1:1”],
                [ram-overcommitment-ratio”, “1:1”]]
                }
        ],
            “events”:[
                {
                    “event_type”: “notification”,
                    “event_date”:  “3456”
                },
                {
                    “event_type”: “notification”,
                    “event_date”:  “4567”
                }
            ]
            }
```

Request:
POST /v1/leases

Normal Response Code: 202 (ACCEPTED)

GET /v1/flavors:

```
{
    “capabilities”: [“cpu.model”, “cpu.cache.size”, “memory.size”, “storage.size”,
“network.rate”]
}
```

Host Reservation Workflow

1.1/ Admin provisions hosts to one host aggregate (calling it freepool) using either the "Host Reservation Admin API" (yet to be defined) or directly thru nova CLI. As a Climate v1, we could assess to leave on provisioning by hand.

1.2/ Admin enrolls hosts to Climate DB by defining thru Admin API (or by using a script in Climate v1) their capabilities (GPU, nGPU, CPU, nCPU, etc.)

2/ User wants to reserve 1:N "hosts with extra specs and overcommitment ratios" (ie. defines for a lease 1:N reservations), he calls the "Host Reservation User API", which creates either 1:N pcloud(s) or an host aggregate(s) (pcloud/hostaggregate <=> reservation), stores the details of the lease->reservations(s) (#of hosts, extra_specs) in the Climate DB and passes the 1:N pcloud_uuid or the hostaggregate_id to the "Lease API" *(prodiving Lease API allows to provide 1:N resources as parameters, could you Dina confirm this ?)*
The Lease API returns a lease_uuid to the Host Reservation User API which itself returns it to the end-user.

3/ When a lease has to start, the Lease Scheduler calls the "wake_up" Host Reservation plugin interface by providing resource_ids of all the empty pcloud(s)/aggregate(s), from which the plugin selects from the freepool the number of hosts with right capabilities (elected by looking at Climate DB hosts capabilities) and moves them in each of the pclouds. It updates the status field corresponding to the host in the Climate DB (free => allocated).

4/ When a lease ends, the Lease Scheduler calls the Host Manager "delete" (or whatever) method (mapped by the conf file to the on_end event). This method moves each of the hosts from each of the plcoud(s)/aggregate(s) ending to a “PostReserved” host aggregate where Host(s) VMs still live. Its'up in a Climate v1 to the operator responsibility to periodically verify the status of each of the hosts to make sure there are no longer any running VMs and if so, move back the host to the corresponding freepool.