1 BlockDeviceMapping

https://docs.openstack.org/developer/nova/block_device_mapping.html

```python
class BlockDeviceMapping(base.NovaPersistentObject, base.NovaObject):
    fields = {
        'id': fields.IntegerField(),
        'instance_uuid': fields.UUIDField(),
        'instance': fields.ObjectField('Instance', nullable=True),
        'source_type': fields.StringField(nullable=True),
        'destination_type': fields.StringField(nullable=True),
        'guest_format': fields.StringField(nullable=True),
        'device_type': fields.StringField(nullable=True),
        'disk_bus': fields.StringField(nullable=True),
        'boot_index': fields.IntegerField(nullable=True),
        'device_name': fields.StringField(nullable=True),
        'delete_on_termination': fields.BooleanField(default=False),
        'snapshot_id': fields.StringField(nullable=True),
        'volume_id': fields.StringField(nullable=True),
        'volume_size': fields.IntegerField(nullable=True),
        'image_id': fields.StringField(nullable=True),
        'no_device': fields.BooleanField(default=False),
        'connection_info': fields.StringField(nullable=True),
    }
```

2 flavor

https://docs.openstack.org/admin-guide/compute-flavors.html

https://docs.openstack.org/admin-guide/cli-manage-flavors.html

在 OpenStack 中，flavors 定义了 nova 实例的 CPU、内存、存储容量等数值。简单地说，一个 flavor 相当于一个实例的硬件配置。

一个 flavor 包含了如下参数：

1. Flavor ID，这是一个 flavor 的 uuid。uuid 一般是自动产生的。
2. Name，一个 flavor 的名称。
3. VCPUS，虚拟 CPU 的数量。
4. Memory MB，RAM 的大小，单位为 MB。
5. Root Disk GB，root 分区的磁盘空间的大小，单位为 GB。
6. Ephemeral Disk GB，临时分区的磁盘空间的大小，单位为 GB，默认值为 0。需要知道的是，当虚拟机关闭时，临时分区所有数据将丢失。而且制作快照时，不会考虑临时分区中的数据。

7. Swap，交换空间的大小，单位为 MB，默认值为 0。
8. RXTX Factor，这是一个可选属性，用于创建不同带宽的 server，默认值为 1.0。RXTX Factor 仅适用于基于 Xen 或 NSX 的系统。
9. Is Public，用于决定是否任何用户都可以使用这个 flavor，默认值为 True。
10. Extra Specs，键和值的 pair，用于定义 flavor 可以在哪些 compute node 上运行。

对于 Newton 而言，openstack 没有默认的 flavor，而 Mitaka 和更早的版本有如下的默认 flavor：

Flavor	VCPUs	Disk (in GB)	RAM (in MB)
m1.tiny	1	1	512
m1.small	1	20	2048
m1.medium	2	40	4096
m1.large	4	80	8192
m1.xlarge	8	160	16384

2.1 管理 flavor

在 openstack 中，可以使用 openstack flavor 命令行工具来管理 flavor。
常用的 openstack flavor 命令如下所示：

1. 列出 flavors，并显示出 flavor 的属性，命令如下：

```
openstack flavor list
```

2. 创建 flavor，命令如下：

```
openstack flavor create FLAVOR_NAME --id FLAVOR_ID --ram RAM_IN_MB --disk
    ROOT_DISK_IN_GB --vcpus NUMBER_OF_VCPUS
```

可以通过如下命令查看 create 更多的选项：

```
openstack help flavor create
```

3. 将 flavor 分配给一个 project，命令如下：

```
# FLAVOR是flavor的名称或ID
# TENANT_ID是project的ID
nova flavor-access-add FLAVOR TENANT_ID
```

4. 删除 flavor，命令如下：

```
openstack flavor delete FLAVOR_ID
```

5. 查看 flavor 命令的帮助手册，命令如下：

```
openstack flavor --help
```

2.2 Extra Specs

这个网页可以查看 Extra Specs 上的值：

```
https://docs.openstack.org/admin-guide/compute-flavors.html#extra-specs
```

3 Instance 类

https://developer.openstack.org/api-guide/compute/server_concepts.html

```
# nova/objects/instance.py Instance
Instance.save()  # 用于修改数据库
```

```
class Instance(BASE, NovaBase):
    """Represents a guest VM."""
    __tablename__ = 'instances'
    __table_args__ = (
        Index('uuid', 'uuid', unique=True),
        Index('project_id', 'project_id'),
        Index('instances_host_deleted_idx',
              'host', 'deleted'),
        Index('instances_reservation_id_idx',
              'reservation_id'),
        Index('instances_terminated_at_launched_at_idx',
              'terminated_at', 'launched_at'),
        Index('instances_uuid_deleted_idx',
              'uuid', 'deleted'),
        Index('instances_task_state_updated_at_idx',
              'task_state', 'updated_at'),
        Index('instances_host_node_deleted_idx',
              'host', 'node', 'deleted'),
        Index('instances_host_deleted_cleaned_idx',
              'host', 'deleted', 'cleaned'),
```

```
21      )
22      injected_files = []
23
24      id = Column(Integer, primary_key=True, autoincrement=True)
25
26      @property
27      def name(self):
28          try:
29              base_name = CONF.instance_name_template % self.id
30          except TypeError:
31              # Support templates like "uuid-%(uuid)s", etc.
32              info = {}
33              # NOTE(russellb): Don't use self.iteritems() here, as it will
34              # result in infinite recursion on the name property.
35              for column in iter(orm.object_mapper(self).columns):
36                  key = column.name
37                  # prevent recursion if someone specifies %(name)s
38                  # %(name)s will not be valid.
39                  if key == 'name':
40                      continue
41                  info[key] = self[key]
42              try:
43                  base_name = CONF.instance_name_template % info
44              except KeyError:
45                  base_name = self.uuid
46          return base_name
47
48      @property
49      def _extra_keys(self):
50          return ['name']
51
52      user_id = Column(String(255))
53      project_id = Column(String(255))
54
55      image_ref = Column(String(255)) # instance的后端镜像id
56      kernel_id = Column(String(255))
57      ramdisk_id = Column(String(255))
58      hostname = Column(String(255))
59
60      launch_index = Column(Integer)
61      key_name = Column(String(255))
62      key_data = Column(MediumText())
63
64      power_state = Column(Integer) # instance的power_state
65      vm_state = Column(String(255))
66      task_state = Column(String(255)) # instance的task_state
67
68      memory_mb = Column(Integer)
69      vcpus = Column(Integer)
70      root_gb = Column(Integer)
71      ephemeral_gb = Column(Integer)
72      ephemeral_key_uuid = Column(String(36))
73
74      # This is not related to hostname, above.  It refers
75      #  to the nova node.
76      # instance所在的宿主机
```

```
77      host = Column(String(255))  # , ForeignKey('hosts.id'))
78      # To identify the "ComputeNode" which the instance resides in.
79      # This equals to ComputeNode.hypervisor_hostname.
80      node = Column(String(255))
81
82      # *not* flavorid, this is the internal primary_key
83      instance_type_id = Column(Integer)
84
85      user_data = Column(MediumText())
86
87      reservation_id = Column(String(255))
88
89      scheduled_at = Column(DateTime)
90      launched_at = Column(DateTime)
91      terminated_at = Column(DateTime)
92
93      availability_zone = Column(String(255))
94
95      # User editable field for display in user-facing UIs
96      display_name = Column(String(255))
97      display_description = Column(String(255))
98
99      # To remember on which host an instance booted.
100     # An instance may have moved to another host by live migration.
101     launched_on = Column(MediumText())
102
103     # NOTE(jdillaman): locked deprecated in favor of locked_by,
104     # to be removed in Icehouse
105     locked = Column(Boolean)
106     locked_by = Column(Enum('owner', 'admin'))
107
108     os_type = Column(String(255))
109     architecture = Column(String(255))
110     vm_mode = Column(String(255))
111     uuid = Column(String(36))
112
113     root_device_name = Column(String(255))
114     default_ephemeral_device = Column(String(255))
115     default_swap_device = Column(String(255))
116     config_drive = Column(String(255))
117
118     # User editable field meant to represent what ip should be used
119     # to connect to the instance
120     access_ip_v4 = Column(types.IPAddress())
121     access_ip_v6 = Column(types.IPAddress())
122
123     auto_disk_config = Column(Boolean())
124     progress = Column(Integer)
125
126     # EC2 instance_initiated_shutdown_terminate
127     # True: -> 'terminate'
128     # False: -> 'stop'
129     # Note(maoy): currently Nova will always stop instead of terminate
130     # no matter what the flag says. So we set the default to False.
131     shutdown_terminate = Column(Boolean(), default=False)
132
```

```
133     # EC2 disable_api_termination
134     disable_terminate = Column(Boolean(), default=False)
135
136     # OpenStack compute cell name.  This will only be set at the top of
137     # the cells tree and it'll be a full cell name such as 'api!hop1!hop2'
138     cell_name = Column(String(255))
139     internal_id = Column(Integer)
140
141     # Records whether an instance has been deleted from disk
142     cleaned = Column(Integer, default=0)
```

4 image metadata

image metadata 的概念:

```
1     Another common term for "image properties" is "image metadata" because what
          we're talking about here are properties that describe the image data that
          can be consumed by various OpenStack services
```

image metadata 的示例:

```
1     {'status': u'queued',
2      'name': u'snap1',
3      'deleted': False,
4      'container_format': u'bare',
5      'created_at': datetime.datetime(2017, 2, 28, 5, 38, 57, tzinfo=<iso8601.
           iso8601.Utc object at 0x7fc8fc5a3d10>),
6      'disk_format': u'qcow2',
7      'updated_at': datetime.datetime(2017, 2, 28, 5, 38, 57, tzinfo=<iso8601.
           iso8601.Utc object at 0x7fc8fc5a3d10>),
8      'id': u'9ce400ab-7785-445c-9e89-9ea35d6de063',
9      'owner': u'd6fda80e5d464008825d806edf4ecc20',
10     'min_ram': 0,
11     'checksum': None,
12     'min_disk': 40,
13     'is_public': False,
14     'deleted_at': None,
15     'properties': {u'instance_uuid': u'ed38cd61-3b9c-47b8-b3e0-b9c511053b23',
16                    u'instance_type_memory_mb': u'4096',
17                    u'user_id': u'ea66cd61e4564f88b2a877868fe1b8a4',
18                    u'image_type': u'snapshot',
19                    u'instance_type_id': u'1',
20                    u'instance_type_name': u'm1.medium',
21                    u'instance_type_ephemeral_gb': u'0',
22                    u'instance_type_rxtx_factor': u'1.0',
23                    u'instance_type_root_gb': u'40',
24                    u'network_allocated': u'True',
25                    u'instance_type_flavorid': u'3',
26                    u'instance_type_vcpus': u'2',
27                    u'instance_type_swap': u'0',
28                    u'base_image_ref': u'1ed1b0e2-f9ae-4a9a-a0aa-166f6a75d5f2'},
29     'size': 0}
```

5 Image

```
class Image(BASE, GlanceBase):
    """Represents an image in the datastore."""
    __tablename__ = 'images'
    __table_args__ = (Index('checksum_image_idx', 'checksum'),
                      Index('visibility_image_idx', 'visibility'),
                      Index('ix_images_deleted', 'deleted'),
                      Index('owner_image_idx', 'owner'),
                      Index('created_at_image_idx', 'created_at'),
                      Index('updated_at_image_idx', 'updated_at'))

    id = Column(String(36), primary_key=True,
                default=lambda: str(uuid.uuid4()))
    name = Column(String(255))
    disk_format = Column(String(20))
    container_format = Column(String(20))
    size = Column(BigInteger().with_variant(Integer, "sqlite"))
    virtual_size = Column(BigInteger().with_variant(Integer, "sqlite"))
    status = Column(String(30), nullable=False)
    visibility = Column(Enum('private', 'public', 'shared', 'community',
                        name='image_visibility'), nullable=False,
                        server_default='shared')
    checksum = Column(String(32))
    min_disk = Column(Integer, nullable=False, default=0)
    min_ram = Column(Integer, nullable=False, default=0)
    owner = Column(String(255))
    protected = Column(Boolean, nullable=False, default=False,
                       server_default=sql.expression.false())
```

6 用户请求 Request

```
class Request(webob.Request):
    """Add some OpenStack API-specific logic to the base webob.Request."""
```