Nova Grouping API

Senhua Huang (Cisco), Alex Glikson (IBM) and Gary Kotton (Red Hat)

The document will propose an API for VM grouping. A group will maintain a collection of VMs and the relationship between them. For example, in order to provide high availability of an application, VM's of the same group should be deployed on different hosts (anti-affinity). The API will provide building blocks for advanced features:

- VM Ensembles[1]
- Virtual clusters[2]

The aforementioned will allow a tenant to deploy a multi-VM application that is designed for VM fault tolerance in a way that application availability is actually resilient to physical host failure. It will also allow a tenant to request that VMs will be placed in close proximity to each other to optimize performance.

This can later be extended to have the following support:

- Groups of groups
- Cross project groups
- Fine-grained topology-aware placement

API

A group will be a dictionary that contains the following:

- id - a unique UUID
- name - human readable name (this does not need to be unique)
- tenant_id - the ID of the tenant that owns the group
- policies - a list of policies for the group. The following policies can be supported: anti affinity, network proximity and host capabilities. In future release, quantitative policies can be added (e.g., QoS).
- metadata - a way to store arbitrary key value pairs on a group
- members - UUIDs of all of the instances that are members of the group.

The groups will be saved in the database.

Groups can be created and modified by a user as follows:

Verb	**URI**	**Description**
GET	/v1.0/{tenant_id}/groups[3]	Lists all configured groups
POST	/v1.0/{tenant_id}/groups	Register a group

[1] https://blueprints.launchpad.net/nova/+spec/vm-ensembles. One of the ideas discussed at summit was to have VM allocations and then have the option of the group deployment.

[2] https://blueprints.launchpad.net/nova/+spec/vcluster-api-extension. This will allow users/tenants to create virtual clusters with bundled storage, compute, and network requests.

[3] The versioning string v1.0 is not final.

GET	/v1.0/{tenant_id}/groups/{id}	Get a specific group
UPDATE	/v1.0/{tenant_id}/groups/{id}	Update a specific group. Membership can not be changed via this API. Policy can be changed only for empty groups.
DELETE	/v1.0/tenant_id/groups/{id}	Delete a specific group

A group entry will have the following attributes:

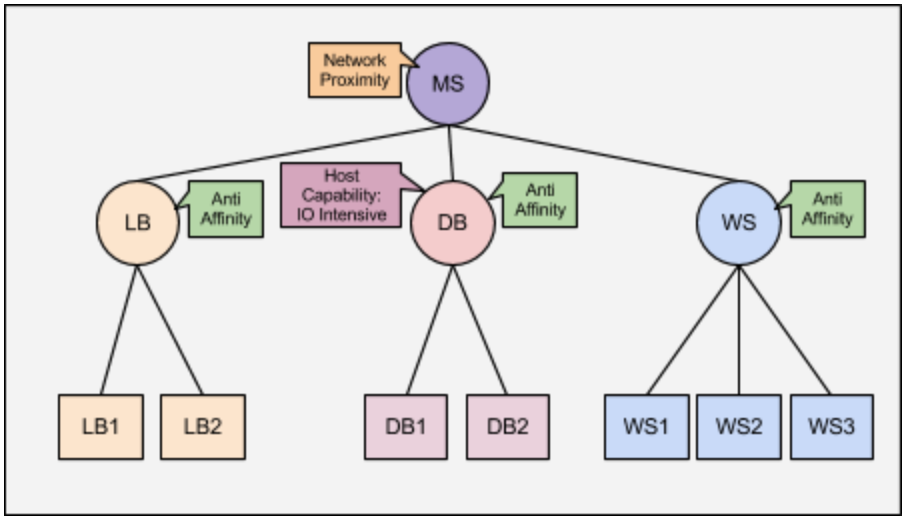
Attribute	**Type**	**CRUD**	**Required**	**Default Value**	**Description**
id	uuid	CR	y	generated	UUID of the policy
name	str	CRU	n	None	A user friendly name for the group.
tenant_id	uuid	CRU	n		Keystone tenant
policies	list()	CRU	n	[anti-affinity]	The group policy
metadata	dict{}	CRU	n	None	Arbitrary key value pairs
members	list(uuid)	R	n	[]	The UUID's of the members in the group.

The API will support JSON and XML formats.

Flow

There are a number of ways in which the groups can be used. The following example will

make use of the application below:



Groups will be created for the following:

- LB - load balancers. The group attribute will be anti-affinity
- DB - database servers. The group attribute will be anti-affinity and host capabilities.
- WS - web servers. The group attribute will be anti-affinity

Each of the above mentioned groups will be created with no members. When an instance is deployed and scheduled a hint will be passed to the scheduler indicating the group id (for example --hint group=UUID). The scheduler in turn will be able to schedule the instance according to the group and its properties. Group membership will be removed once VM instance is removed.