cloudify-openstack-plugin Documentation

Release 2.0

GigaSpaces Technologies Ltd.

Sep 28, 2017

Contents

The OpenStack plugin allows users to use an OpenStack based cloud infrastructure for deploying services and applications. For more information about OpenStack, please refer to: https://www.openstack.org/.

Contents:

CHAPTER 1

Openstack Configuration

The Openstack plugin requires credentials and endpoint setup information in order to authenticate and interact with Openstack.

This information will be gathered by the plugin from the following sources, each source possibly partially or completely overriding values gathered from previous ones:

1. environment variables for each of the configuration parameters.
2. JSON file at `~/openstack_config.json` or at a path specified by the value of an environment variable named `OPENSTACK_CONFIG_PATH`
3. values specified in the `openstack_config` property for the node whose operation is currently getting executed (in the case of relationship operations, the `openstack_config` property of either the **source** or **target** nodes will be used if available, with the **source**'s one taking precedence).

The structure of the JSON file in section (2), as well as of the `openstack_config` property in section (3), is as follows:

```
{
    "username": "",
    "password": "",
    "tenant_name": "",
    "auth_url": "",
    "region": "",
    "nova_url": "",
    "neutron_url": "",
    "custom_configuration": ""
}
```

- `username` username for authentication with Openstack Keystone service.
- `password` password for authentication with Openstack Keystone service.
- `tenant_name` name of the tenant to be used.
- `auth_url` URL of the Openstack Keystone service.

> **Attention:** New in 2.0
>
> `auth_url` must include the full keystone auth URL, including the version number.

- `region` Openstack region to be used. This may be optional when there's but a single region.
- `nova_url` (**DEPRECATED** - instead, use `custom_configuration` to pass `endpoint_override` directly to the Nova client) explicit URL for the Openstack Nova service. This may be used to override the URL for the Nova service that is listed in the Keystone service.
- `neutron_url` (**DEPRECATED** - instead, use `custom_configuration` to pass `endpoint_url` directly to the Neutron client) explicit URL for the Openstack Neutron service. This may be used to override the URL for the Neutron service that is listed in the Keystone service.
- `custom_configuration` a dictionary which allows overriding or directly passing custom configuration parameter to each of the Openstack clients, by using any of the relevant keys: `keystone_client`, `nova_client`, `neutron_client` or `cinder_client`. * Parameters passed directly to Openstack clients using the `custom_configuration` mechanism will override other definitions (e.g. any of the common Openstack configuration parameters listed above, such as `username` and `tenant_name`) * The following is an example for the usage of the `custom_configuration` section in a blueprint:

```
custom_configuration:
  nova_client:
    endpoint_override: nova-endpoint-url
    nova_specific_key_1: value_1
    nova_specific_key_2: value_2
  neutron_client:
    endpoint_url: neutron-endpoint-url
  keystone_client:
    ..
  cinder_client:
    ..
```

The environment variables mentioned in (1) are the standard Openstack environment variables equivalent to the ones in the JSON file or `openstack_config` property. In their respective order, they are:

- `OS_USERNAME`
- `OS_PASSWORD`
- `OS_TENANT_NAME`
- `OS_AUTH_URL`
- `OS_REGION_NAME`
- `NOVACLIENT_BYPASS_URL`
- `OS_URL`

Note: `custom_configuration` doesn't have an equivalent standard Openstack environment variable.

> The Openstack manager blueprint stores the Openstack configuration used for the bootstrap process in a JSON file as described in (2) at `~/openstack-config.json`. Therefore, if they've been used for bootstrap, the Openstack configuration for applications isn't required as the plugin will default to these same settings.

CHAPTER 2

Types

Node Types

`cloudify.openstack.nodes.Server`

An OpenStack server.

Derived from: cloudify.nodes.Compute

Properties:

openstack_config **default:** `{}`

see Openstack Configuraion

resource_id name to give to the new resource or the name or ID of an existing resource when the `use_external_resource` property is set to `true` (see the using existing resources section). Defaults to ‘’ (empty string).

image The image for the server. May receive either the ID or the name of the image. note: This property is currently optional for backwards compatibility, but will be modified to become a required property in future versions (Default: ‘’).

management_network_name Cloudify’s management network name. Every server should be connected to the management network. If the management network’s name information is available in the Provider Context, this connection is made automatically and there’s no need to override this property (See the Misc section for more information on the Openstack Provider Context). Otherwise, it is required to set the value of this property to the management network name as it was set in the bootstrap process. Note: When using Nova-net Openstack (see the Nova-net Support section), don’t set this property. Defaults to ‘’ (empty string).

create_if_missing **default:** `False`

If use_external_resource is `true` and the resource is missing, create it instead of failing.

server default: `{}`

key-value server configuration as described in OpenStack compute create server API. (DEPRECATED - Use the args input in create operation instead)

flavor The flavor for the server. May receive either the ID or the name of the flavor. note: This property is currently optional for backwards compatibility, but will be modified to become a required property in future versions (Default: '').

use_password default: `False`

A boolean describing whether this server image supports user-password authentication. Images that do should post the administrator user's password to the Openstack metadata service (e.g. via cloudbase); The password would then be retrieved by the plugin, decrypted using the server's keypair and then saved in the server's runtime properties.

use_external_resource type: `boolean` **default:** `False`

a boolean for setting whether to create the resource or use an existing one. See the using existing resources section.

`cloudify.openstack.nodes.WindowsServer`

This type has the same properties and operations-mapping as the type above (as it derives from it), yet it overrides some of the agent and plugin installations operations-mapping derived from the built-in cloudify.nodes.Compute type. Use this type when working with a Windows server.

Additionally, the default value for the use_password property is overridden for this type, and is set to true. When using an image with a preset password, it should be modified to false.

Derived from: *cloudify.openstack.nodes.Server*

Properties:

agent_config type: `cloudify.datatypes.AgentConfig` **default:** `{'port':  5985}`

(updates the defaults for the agent_config for Windows)

use_password default: `True`

Default changed for derived type because Windows instances need a password for agent installation

os_family default: `windows`

(updates the os_family default as a convenience)

`cloudify.openstack.nodes.KeyPair`

Derived from: cloudify.nodes.Root

Properties:

openstack_config default: `{}`

endpoints and authentication configuration for Openstack. Expected to contain the following nested fields: username, password, tenant_name, auth_url, region.

resource_id the name that will be given to the resource on Openstack (excluding optional prefix). If not provided, a default name will be given instead. If use_external_resource is set to "true", this exact value (without any prefixes applied) will be looked for as either the name or id of an existing keypair to be used.

create_if_missing default: `False`

If use_external_resource is `true` and the resource is missing, create it instead of failing.

keypair default: `{}`

the keypair object as described by Openstack. This parameter can be used to override and pass parameters directly to Nova client. Note that in the case of keypair, the only nested parameter that can be used is "name".

private_key_path required

the path (on the machine the plugin is running on) to where the private key should be stored. If use_external_resource is set to "true", the existing private key is expected to be at this path.

use_external_resource type: `boolean` **default:** `False`

a boolean describing whether this resource should be created or rather that it already exists on Openstack and should be used as-is.

`cloudify.openstack.nodes.Image`
Derived from: cloudify.nodes.Root

Properties:

openstack_config default: `{}`

endpoints and authentication configuration for Openstack. Expected to contain the following nested fields: username, password, tenant_name, auth_url, region.

resource_id name to give to the new resource or the name or ID of an existing resource when the `use_external_resource` property is set to `true` (see the using existing resources section). Defaults to '' (empty string).

image default: `{}`

Required parameters are (container_format, disk_format). Accepted types are available on http://docs.openstack.org/developer/glance/formats.html To create an image from the local file its path should be added in data parameter.

create_if_missing default: `False`

If use_external_resource is `true` and the resource is missing, create it instead of failing.

image_url The openstack resource URL for the image.

use_external_resource default: `False`

a boolean for setting whether to create the resource or use an existing one. See the using existing resources section.

`cloudify.openstack.nodes.SecurityGroup`
Derived from: cloudify.nodes.SecurityGroup

Properties:

openstack_config default: `{}`

endpoints and authentication configuration for Openstack. Expected to contain the following nested fields: username, password, tenant_name, auth_url, region.

resource_id name to give to the new resource or the name or ID of an existing resource when the `use_external_resource` property is set to `true` (see the using existing resources section). Defaults to '' (empty string).

rules default: `[]`

key-value security_group_rule configuration as described in http://developer.openstack.org/api-ref-networking-v2.html#security_groups

create_if_missing default: `False`

If use_external_resource is `true` and the resource is missing, create it instead of failing.

security_group default: `{}`

key-value security_group configuration as described in http://developer.openstack.org/api-ref-networking-v2-ext.html#createSecGroup. (**DEPRECATED - Use the 'args' input in create operation instead**)

disable_default_egress_rules default: `False`

a flag for removing the default rules which https://wiki.openstack.org/wiki/Neutron/SecurityGroups#Behavior. If not set to *true*, these rules will remain, and exist alongside any additional rules passed using the *rules* property.

use_external_resource type: `boolean` **default:** `False`

a boolean for setting whether to create the resource or use an existing one. See the using existing resources section.

description type: `string`

SecurityGroup description.

`cloudify.openstack.nodes.Router`
Derived from: cloudify.nodes.Router

Properties:

openstack_config default: `{}`

endpoints and authentication configuration for Openstack. Expected to contain the following nested fields: username, password, tenant_name, auth_url, region.

resource_id name to give to the new resource or the name or ID of an existing resource when the `use_external_resource` property is set to `true` (see the using existing resources section). Defaults to '' (empty string).

create_if_missing default: `False`

If use_external_resource is `true` and the resource is missing, create it instead of failing.

default_to_managers_external_network default: `True`

A boolean which determines whether to use the Cloudify Manager's external network if no other external network was given (whether by a relationship, by the *external_network* property or by the nested *external_gateway_info* key in the *router* property). This is only relevant if the manager's external network appears in the Provider-context. Defaults to *true*.

router default: `{}`

key-value router configuration as described in http://developer.openstack.org/api-ref-networking-v2.html#layer3. (**DEPRECATED - Use the 'args' input in create operation instead**)

external_network An external network name or ID. If given, the router will use this external network as a gateway.

use_external_resource **type:** `boolean` **default:** `False`

a boolean for setting whether to create the resource or use an existing one. See the using existing resources section.

`cloudify.openstack.nodes.Port`
Derived from: cloudify.nodes.Port

Properties:

openstack_config **default:** `{}`

endpoints and authentication configuration for Openstack. Expected to contain the following nested fields: username, password, tenant_name, auth_url, region.

resource_id name to give to the new resource or the name or ID of an existing resource when the `use_external_resource` property is set to `true` (see the using existing resources section). Defaults to '' (empty string).

fixed_ip may be used to request a specific fixed IP for the port. If the IP is unavailable (either already taken or does not belong to a subnet the port is on) an error will be raised.

create_if_missing **default:** `False`

If use_external_resource is `true` and the resource is missing, create it instead of failing.

use_external_resource **type:** `boolean` **default:** `False`

a boolean for setting whether to create the resource or use an existing one. See the using existing resources section.

port **default:** `{}`

key-value port configuration as described in http://developer.openstack.org/api-ref-networking-v2.html#ports. (**DEPRECATED - Use the 'args' input in create operation instead**)

`cloudify.openstack.nodes.Network`
Derived from: cloudify.nodes.Network

Properties:

resource_id name to give to the new resource or the name or ID of an existing resource when the `use_external_resource` property is set to `true` (see the using existing resources section). Defaults to '' (empty string).

openstack_config **default:** `{}`

endpoints and authentication configuration for Openstack. Expected to contain the following nested fields: username, password, tenant_name, auth_url, region.

network **default:** `{}`

key-value network configuration as described in http://developer.openstack.org/api-ref-networking-v2.html#networks. (**DEPRECATED - Use the 'args' input in create operation instead**)

create_if_missing default: `False`

If use_external_resource is `true` and the resource is missing, create it instead of failing.

use_external_resource type: `boolean` **default:** `False`

a boolean for setting whether to create the resource or use an existing one. See the using existing resources section.

`cloudify.openstack.nodes.Subnet`
Derived from: cloudify.nodes.Subnet

Properties:

subnet default: `{}`

key-value subnet configuration as described at http://developer.openstack.org/api-ref-networking-v2.html#subnets. (**DEPRECATED - Use the 'args' input in create operation instead**)

use_external_resource type: `boolean` **default:** `False`

a boolean for setting whether to create the resource or use an existing one. See the using existing resources section.

openstack_config default: `{}`

endpoints and authentication configuration for Openstack. Expected to contain the following nested fields: username, password, tenant_name, auth_url, region.

create_if_missing default: `False`

If use_external_resource is `true` and the resource is missing, create it instead of failing.

resource_id name to give to the new resource or the name or ID of an existing resource when the `use_external_resource` property is set to `true` (see the using existing resources section). Defaults to '' (empty string).

`cloudify.openstack.nodes.FloatingIP`
Derived from: cloudify.nodes.VirtualIP

Properties:

resource_id name to give to the new resource or the name or ID of an existing resource when the `use_external_resource` property is set to `true` (see the using existing resources section). Defaults to '' (empty string).

openstack_config default: `{}`

endpoints and authentication configuration for Openstack. Expected to contain the following nested fields: username, password, tenant_name, auth_url, region.

use_external_resource type: `boolean` **default:** `False`

a boolean for setting whether to create the resource or use an existing one. See the using existing resources section.

create_if_missing default: `False`

If use_external_resource is `true` and the resource is missing, create it instead of failing.

floatingip default: `{}`

key-value floatingip configuration as described in http://developer.openstack.org/api-ref-networking-v2.html#layer3. (**DEPRECATED - Use the 'args' input in create operation instead**)

`cloudify.openstack.nodes.Volume`
Derived from: cloudify.nodes.Volume

Properties:

openstack_config default: `{}`

endpoints and authentication configuration for Openstack. Expected to contain the following nested fields: username, password, tenant_name, auth_url, region.

resource_id name to give to the new resource or the name or ID of an existing resource when the `use_external_resource` property is set to `true` (see the using existing resources section). Defaults to '' (empty string).

boot type: `boolean` **default:** `False`

If a Server instance is connected to this Volume by a relationship, this volume will be used as the boot volume for that Server.

create_if_missing default: `False`

If use_external_resource is `true` and the resource is missing, create it instead of failing.

device_name default: `auto`

The device name this volume will be attached to. Default value is *auto*, which means openstack will auto-assign a device. Note that if you do explicitly set a value, this value may not be the actual device name assigned. Sometimes the device requested will not be available and openstack will assign it to a different device, this is why we recommend using *auto*.

volume default: `{}`

key-value volume configuration as described in http://developer.openstack.org/api-ref-blockstorage-v1.html#volumes-v1. (**DEPRECATED - Use the 'args' input in create operation instead**)

use_external_resource type: `boolean` **default:** `False`

a boolean for setting whether to create the resource or use an existing one. See the using existing resources section.

`cloudify.openstack.nodes.Project`
Derived from: cloudify.nodes.Root

Properties:

openstack_config default: `{}`

endpoints and authentication configuration for Openstack. Expected to contain the following nested fields: username, password, tenant_name, auth_url, region.

resource_id name to give to the new resource or the name or ID of an existing resource when the `use_external_resource` property is set to `true` (see the using existing resources section). Defaults to '' (empty string).

quota default: `{}`

A dictionary mapping service names to quota definitions for a proejct

e.g:

```
quota:
  neutron: <quota>
  nova: <quota>
```

create_if_missing default: `False`

If use_external_resource is `true` and the resource is missing, create it instead of failing.

project default: `{}`

key-value project configuration.

use_external_resource default: `False`

a boolean for setting whether to create the resource or use an existing one. See the using existing resources section.

users default: `[]`

List of users assigned to this project in the following format: { name: string, roles: [string] }

Types' Common Behaviors

Validations

All types offer the same base functionality for the `cloudify.interfaces.validation.creation` interface operation:

- If it's a new resource (`use_external_resource` is set to `false`), the basic validation is to verify there's enough quota to allocate a new resource of the given type.
- When [using an existing resource](#using-existing-resources), the validation ensures the resource indeed exists.

Runtime Properties

Node instances of any of the types defined in this plugin get set with the following runtime properties during the `cloudify.interfaces.lifecycle.create` operation:

- `external_id` the Openstack ID of the resource
- `external_type` the Openstack type of the resource
- `external_name` the Openstack name of the resource

The only exceptions are the two *floating-ip* types - Since floating-ip objects on Openstack don't have a name, the `external_name` runtime property is replaced with the `floating_ip_address` one, which holds the object's actual IP address.

Default Resource Naming Convention

When creating a new resource (i.e. `use_external_resource` is set to `false`), its name on Openstack will be the value of its `resource_id` property. However, if this value is not provided, the name will default to the following schema:

`<openstack-resource-type>_<deployment-id>_<node-instance-id>`

For example, if a server node is defined as so:

```
node_templates:
  myserver:
    type: cloudify.openstack.nodes.Server
    ...
```

Yet without setting the `resource_id` property, then the server's name on Openstack will be `server_my-deployment_myserver_XXXXX` (where the XXXXX is the autogenerated part of the node instance's ID).

Using Existing Resources

It is possible to use existing resources on Openstack - whether these have been created by a different Cloudify deployment or not via Cloudify at all.

All Cloudify Openstack types have a property named `use_external_resource`, whose default value is `false`. When set to `true`, the plugin will apply different semantics for each of the operations executed on the relevant node's instances. Specifically, in the case of the `cloudify.interfaces.lifecycle.create` operation, rather than creating a new resource on Openstack of the given type, the plugin will behave as follows:

1. Try to find an existing resource on Openstack whose name (or IP, in the case of one of the **floating-ip** types) is the value specified for the `resource_id` property. If more than one is found, an error is raised.
2. If no resource was found, the plugin will use the value of the `resource_id` property to look for the resource by ID instead. If a resource still isn't found, an error is raised.
3. If a single resource was found, the plugin will use that resource, and set the node instance with the appropriate runtime properties according to the resource's data.

The semantics of other operations are affected as well:

- The `cloudify.interfaces.lifecycle.start` operation, where applicable, will only validate that the resource is indeed started, raising an error if it isn't.
- The `cloudify.interfaces.lifecycle.stop` operation, where applicable, won't have any effect.
- The `cloudify.interfaces.lifecycle.delete` operation will not actually delete the resource from Openstack (but will clear the runtime properties from the node instance).
- The `cloudify.interfaces.validation.creation` operation will verify that a resource with the given name or ID indeed exists, or otherwise print a list of all available resources of the given type.
- The `cloudify.interfaces.relationship_lifecycle.establish` operation will behave as normal if the related node is not set with `use_external_resource` as `true`; However if both nodes have this property set to `true`, the operation will only attempt to verify that they're indeed "connected" on Openstack as well ("connected" in this case also refers to a security-group imposed on a server, floating-ip associated with a server, etc.).

Notes

- As mentioned in the [Relationships section](#relationships), some relationships take effect in non-relationship operations. When `use_external_resource` is set to `true`, the existence of such connections is validated as well.
- Using an existing resource only makes sense for single-instance nodes.

Relationships

Not all relationships have built-in types (i.e., some types may simply get connected using standard Cloudify relationships such as `cloudify.relationships.connected_to`).

Some relationships take effect in non-relationship operations, e.g. a subnet which is connected to a network actually gets connected on subnet's creation (in the `cloudify.interfaces.lifecycle.create` operation) and not in a `cloudify.interfaces.relationship_lifecycle.establish` operation - this occurs whenever the connection information is required on resource creation.

`cloudify.openstack.server_connected_to_port`
A relationship for connecting a server to a port. The server will use this relationship to automatically connect to the port upon server creation.

Derived from: cloudify.relationships.connected_to

`cloudify.openstack.port_connected_to_security_group`
A relationship for a port to a security group.

Derived from: cloudify.relationships.connected_to

`cloudify.openstack.server_connected_to_keypair`
Derived from: cloudify.relationships.connected_to

`cloudify.openstack.port_connected_to_subnet`
A relationship for connecting a port to a subnet. This is useful when a network has multiple subnets, and a port should belong to a specific subnet on that network. The port will then receive some IP from that given subnet.

Note that when using this relationship in combination with the port type's property *fixed_ip*, the IP given should be on the CIDR of the subnet connected to the port.

Note: This relationship has no operations associated with it; The port will use this relationship to automatically connect to the subnet upon port creation.

Derived from: cloudify.relationships.connected_to

`cloudify.openstack.server_connected_to_security_group`
A relationship for setting a security group on a server.

Derived from: cloudify.relationships.connected_to

`cloudify.openstack.subnet_connected_to_router`
A relationship for connecting a subnet to a router.

Derived from: cloudify.relationships.connected_to

`cloudify.openstack.port_connected_to_floating_ip`
A relationship for associating a floating ip with a port. If that port is later connected to a server, the server will be accessible via the floating IP.

Derived from: cloudify.relationships.connected_to

`cloudify.openstack.server_connected_to_floating_ip`

A relationship for associating a floating ip with a server.

Derived from: cloudify.relationships.connected_to

`cloudify.openstack.volume_attached_to_server`

A relationship for attaching a volume to a server.

Derived from: cloudify.relationships.connected_to

CHAPTER 3

Nova-net Support

The Openstack plugin includes support for Nova-net mode - i.e. an Openstack installation which does not have the Networking API (Neutron service).

In such an environment, there is but a single preconfigured private network, which all servers make use of automatically. There are no subnets, networks, routers or ports. Since these resource types don't exist, the plugin's equivalent types aren't valid to use in such an environment.

There are, however, some resource types whose API is available via both the Nova and Neutron services - These had originally been on the Nova service, and later were moved and got extended implementation in the Neutron one, but were also kept in the Nova service for backward compatibility.

For these resource types, the Openstack plugin defines two separate types - one in the plugin's standard types namespace (`cloudify.openstack.nodes.XXX`), which uses the newer and extended API via the Neutron service; and Another in a special namespace (`cloudify.openstack.nova_net.nodes.XXX`), which uses the older API via the Nova service. This is why you may notice two separate types defined for [Floating](#cloudifyopenstacknodesfloatingip) [IP](#cloudifyopenstacknovanetnodesfloatingip), as well as for [Security](#cloudifyopenstacknodessecuritygroup) [Group](#cloudifyopenstacknovanetnodessecuritygroup).

To summarize, ensure that when working in a Nova-net Openstack environment, Neutron types aren't used - these include all types whose resources' APIs are natively available only via the Network API, as well as the types which are in the `cloudify.openstack.nova_net.Nodes` namespace.

On the opposite side, when using an Openstack environment which supports Neutron, it's recommended to use the Neutron-versions of the relevant types (i.e. avoid any types defined under the `cloudify.openstack.nova_net.Nodes` namespace), as they offer more advanced capabilities. However, it's important to mention that this is not required, and using the Nova-versions of some types in a Neutron-enabled environment is possible and will work as well.

Nova-net Node Types

`cloudify.openstack.nova_net.nodes.FloatingIP`

Derived from: cloudify.nodes.VirtualIP

Properties:

resource_id name to give to the new resource or the name or ID of an existing resource when the `use_external_resource` property is set to `true` (see the using existing resources section). Defaults to ‘’ (empty string).

openstack_config default: `{}`

endpoints and authentication configuration for Openstack. Expected to contain the following nested fields: username, password, tenant_name, auth_url, region.

use_external_resource type: `boolean` **default:** `False`

a boolean for setting whether to create the resource or use an existing one. See the using existing resources section.

create_if_missing default: `False`

TODO: CREATE. THIS IS MISSING

floatingip default: `{}`

key-value floatingip configuration as described in http://developer.openstack.org/api-ref-compute-v2-ext.html#ext-os-floating-ips. (**DEPRECATED - Use the ‘args‘ input in create operation instead**)

`cloudify.openstack.nova_net.nodes.SecurityGroup`
Derived from: cloudify.nodes.SecurityGroup

Properties:

openstack_config default: `{}`

endpoints and authentication configuration for Openstack. Expected to contain the following nested fields: username, password, tenant_name, auth_url, region.

resource_id name to give to the new resource or the name or ID of an existing resource when the `use_external_resource` property is set to `true` (see the using existing resources section). Defaults to ‘’ (empty string).

rules default: `[]`

key-value security group rule as described in http://developer.openstack.org/api-ref-compute-v2-ext.html#ext-os-security-group-default-rules.

create_if_missing default: `False`

TODO: CREATE. THIS IS MISSING

security_group default: `{}`

key-value security_group configuration as described in http://developer.openstack.org/api-ref-compute-v2-ext.html#ext-os-security-groups. (**DEPRECATED - Use the ‘args‘ input in create operation instead**)

use_external_resource type: `boolean` **default:** `False`

a boolean for setting whether to create the resource or use an existing one. See the using existing resources section.

description required

security group description

CHAPTER 4

Examples

Example I

This example will show how to use most of the types in this plugin, as well as how to make the relationships between them.

We'll see how to create a server with a security group set on it and a floating_ip associated to it, on a subnet in a network.

The following is an excerpt from the blueprint's *blueprint*.'nodes' section:

```
my_floating_ip:
  type: cloudify.openstack.nodes.FloatingIP
  interfaces:
    cloudify.interfaces.lifecycle:
      create:
        inputs:
          args:
            floating_network_name: Ext-Net


my_network:
  type: cloudify.openstack.nodes.Network
  properties:
    resource_id: my_network_openstack_name


my_subnet:
  type: cloudify.openstack.nodes.Subnet
  properties:
    resource_id: my_subnet_openstack_name
  interfaces:
    cloudify.interfaces.lifecycle:
      create:
        inputs:
          args:
            cidr: 1.2.3.0/24
            ip_version: 4
    cloudify.interfaces.validation:
      creation:
        inputs:
          args:
            cidr: 1.2.3.0/24
            ip_version: 4
  relationships:
    - target: my_network
      type: cloudify.relationships.contained_in


my_security_group:
  type: cloudify.openstack.nodes.SecurityGroup
  properties:
    resource_id: my_security_group_openstack_name
    rules:
      - remote_ip_prefix: 0.0.0.0/0
        port: 8080


my_server:
  type: cloudify.openstack.nodes.Server
  properties:
    resource_id: my_server_openstack_name
  interfaces:
    cloudify.interfaces.lifecycle:
      create:
        inputs:
          args:
            image: 8672f4c6-e33d-46f5-b6d8-ebbeba12fa02
            flavor: 101
    cloudify.interfaces.validation:
      creation:
        inputs:
          args:
            image: 8672f4c6-e33d-46f5-b6d8-ebbeba12fa02
            flavor: 101
  relationships:
    - target: my_network
      type: cloudify.relationships.connected_to
    - target: my_subnet
      type: cloudify.relationships.depends_on
    - target: my_floating_ip
      type: cloudify.openstack.server_connected_to_floating_ip
    - target: my_security_group
      type: cloudify.openstack.server_connected_to_security_group
```

1. Creates a floating IP, whose node name is `my_floating_ip`, and whose floating_network_name is `Ext-Net` (This value represents the name of the external network).
2. Creates a network, whose node name is `my_network`, and whose name on Openstack is `my_network_openstack_name`.
3. Creates a subnet, whose node name is `my_subnet`, and whose name on Openstack is `my_subnet_openstack_name`. The subnet's address range is defined to be 1.2.3.0 - 1.2.3.255 us-

ing the cidr parameter, and the subnet's IP version is set to version 4. The subnet will be set on the my_network_openstack_name network because of the relationship to the my_network node.

4. Creates a security_group, whose node name is my_security_group, and whose name on Openstack is my_security_group_openstack_Name. The security group is set with a single rule, which allows all traffic (since we use the address range 0.0.0.0/0) to port 8080 (default direction is *ingress*).
5. Creates a server, whose node name is my_server, and whose name on openstack is my_server_openstack_name. The server is set with an image and flavor IDs. The server is set with multiple relationships:

- A relationship to the my_network node: Through this relationship, the server will be automatically placed on the my_network_openstack_name network.
- A relationship to the my_subnet node: This relationship is strictly for ensuring the order of creation is correct, as the server requires the my_subnet_openstack_name subnet to exist before it can be created on it.
- A relationship to the my_floating_ip node: This designated relationship type will take care of associating the server with the floating IP represented by the my_floating_ip node.
- A relationship with the my_security_group node: This relationship will take care of setting the server up with the security group represented by the my_security_group node.

Example II

This example will show how to use the router and port types, as well as some of the relationships that were missing from Example I.

We'll see how to create a server connected to a port, where the port is set on a subnet in a network, and has a security group set on it. Finally, we'll see how this subnet connects to a router and from there to the external network.

The following is an excerpt from the blueprint's blueprint.``node_templates`` section:

```
my_network:
  type: cloudify.openstack.nodes.Network
  properties:
    resource_id: my_network_openstack_name


my_security_group:
  type: cloudify.openstack.nodes.SecurityGroup
  properties:
    resource_id: my_security_group_openstack_name
    rules:
      - remote_ip_prefix: 0.0.0.0/0
        port: 8080


my_subnet:
  type: cloudify.openstack.nodes.Subnet
  properties:
    resource_id: my_subnet_openstack_name
  interfaces:
    cloudify.interfaces.lifecycle:
      create:
        inputs:
          args:
            cidr: 1.2.3.0/24
            ip_version: 4
    cloudify.interfaces.validation:
      creation:
        inputs:
          args:
            cidr: 1.2.3.0/24
            ip_version: 4
  relationships:
    - target: my_network
      type: cloudify.relationships.contained_in
    - target: my_router
      type: cloudify.openstack.subnet_connected_to_router


my_port:
  type: cloudify.openstack.nodes.Port
  properties:
    resource_id: my_port_openstack_name
  relationships:
    - target: my_network
      type: cloudify.relationships.contained_in
    - target: my_subnet
      type: cloudify.relationships.depends_on
    - target: my_security_group
      type: cloudify.openstack.port_connected_to_security_group


my_router:
  type: cloudify.openstack.nodes.Router
  properties:
    resource_id: my_router_openstack_Name


my_server:
  type: cloudify.openstack.nodes.Server
  properties:
    cloudify_agent:
      user: ubuntu
  interfaces:
    cloudify.interfaces.lifecycle:
      create:
        inputs:
          args:
            image: 8672f4c6-e33d-46f5-b6d8-ebbeba12fa02
            flavor: 101
    cloudify.interfaces.validation:
      creation:
        inputs:
          args:
            image: 8672f4c6-e33d-46f5-b6d8-ebbeba12fa02
            flavor: 101
  relationships:
    - target: my_port
      type: cloudify.openstack.server_connected_to_port
```

1. Creates a network. See Example I for more information.
2. Creates a security group. See Example I for more information.

3. Creates a subnet. This is again similar to what we've done in Example I. The difference here is that the subnet has an extra relationship set towards a router.
4. Creates a port, whose node name is `my_port`, and whose name on Openstack is `my_port_openstack_name`. The port is set with multiple relationships:

- A relationship to the `my_network` node: Through this relationship, the port will be automatically placed on the `my_network_openstack_name` network.
- A relationship to the `my_subnet` node: This relationship is strictly for ensuring the order of creation is correct, as the port requires the `my_subnet_openstack_name` subnet to exist before it can be created on it.
- A relationship to the `my_security_group` node: This designated relationship type will take care of setting the `my_security_group_openstack_name` security group on the port.

5. Creates a router, whose node name is `my_router`, and whose name on Openstack is `my_router_openstack_name`. The router will automatically have an interface in the external network.
6. Creates a server, whose node name is `my_server`, and whose name on Openstack is **the node's ID** (since no `name` parameter was supplied under the `server` property). The server is set with an image and flavor IDs. It also overrides the `cloudify_agent` property of its parent type to set the username that will be used to connect to the server for installing the Cloudify agent on it. Finally, it is set with a relationship to the `my_port` node: This designated relationship type will take care of connecting the server to `my_port_openstack_name`.

Example III

This example will show how to use the `volume` type, as well as `volume_attached_to_server` relationship.

The following is an excerpt from the blueprint's `blueprint`.``node_templates`` section:

```
my_server:
  type: cloudify.openstack.nodes.Server
  properties:
    cloudify_agent:
      user: ubuntu
  interfaces:
    cloudify.interfaces.lifecycle:
      create:
        inputs:
          args:
            image: 8672f4c6-e33d-46f5-b6d8-ebbeba12fa02
            flavor: 101
    cloudify.interfaces.validation:
      creation:
        inputs:
          args:
            image: 8672f4c6-e33d-46f5-b6d8-ebbeba12fa02
            flavor: 101

my_volume:
  type: cloudify.openstack.nodes.Volume
  properties:
    resource_id: my_openstack_volume_name
    device_name: /dev/vdb
  interfaces:
    cloudify.interfaces.lifecycle:
      create:
        inputs:
          args:
            size: 1
  relationships:
    - target: my_server
      type: cloudify.openstack.volume_attached_to_server
```

1. Creates a server, with name `my_server`, and with name on Openstack **the node's ID** (since no `name` parameter was supplied under the `server` property). The server is set with an image and flavor IDs.
2. Creates a volume. It is set with a relationship to the `my_server` node: This designated relationship type will take care of attaching the volume to Openstack server node.

Example IV

This example will show how to use a Windows server with a Cloudify agent on it.

The following is an excerpt from the blueprint's `blueprint`.''node_templates'' section:

```
my_keypair:
  type: cloudify.openstack.nodes.KeyPair
  properties:
    private_key_path: /tmp/windows-test.pem

my_server:
  type: cloudify.openstack.nodes.WindowsServer
  relationships:
    - type: cloudify.openstack.server_connected_to_keypair
      target: keypair
  interfaces:
    cloudify.interfaces.lifecycle:
      create:
        inputs:
          args:
            server:
              image: 8672f4c6-e33d-46f5-b6d8-ebbeba12fa02
              flavor: 101
              name: my-server
              userdata: |
                #ps1_sysnative
                winrm quickconfig -q
                winrm set winrm/config/winrs '@{MaxMemoryPerShellMB="300"}'
                winrm set winrm/config '@{MaxTimeoutms="1800000"}'
                winrm set winrm/config/service '@{AllowUnencrypted="true"}'
                winrm set winrm/config/service/auth '@{Basic="true"}'
                &netsh advfirewall firewall add rule name="WinRM 5985" protocol=TCP␣
↪dir=in localport=5985 action=allow
                &netsh advfirewall firewall add rule name="WinRM 5986" protocol=TCP␣
↪dir=in localport=5986 action=allow

                msiexec /i https://www.python.org/ftp/python/2.7.6/python-2.7.6.msi␣
↪TARGETDIR=C:\Python27 ALLUSERS=1 /qn
    cloudify.interfaces.validation:
      creation:
        inputs:
          args:
            server:
              image: 8672f4c6-e33d-46f5-b6d8-ebbeba12fa02
              flavor: 101
              name: my-server
              userdata: |
                #ps1_sysnative
                winrm quickconfig -q
                winrm set winrm/config/winrs '@{MaxMemoryPerShellMB="300"}'
                winrm set winrm/config '@{MaxTimeoutms="1800000"}'
                winrm set winrm/config/service '@{AllowUnencrypted="true"}'
                winrm set winrm/config/service/auth '@{Basic="true"}'
                &netsh advfirewall firewall add rule name="WinRM 5985" protocol=TCP␣
↪dir=in localport=5985 action=allow
                &netsh advfirewall firewall add rule name="WinRM 5986" protocol=TCP␣
↪dir=in localport=5986 action=allow

                msiexec /i https://www.python.org/ftp/python/2.7.6/python-2.7.6.msi␣
↪TARGETDIR=C:\Python27 ALLUSERS=1 /qn
        cloudify.interfaces.worker_installer:
          install:
            inputs:
              cloudify_agent:
                user: Admin
                password: { get_attribute: [SELF, password] }
```

1. Creates a keypair. the private key will be saved under `/tmp/windows-test.pem`.
2. Creates a Windows server:

- It is set with a relationship to the `my_keypair` node, which will make the server use the it as a public key for authentication, and also use this public key to encrypt its password before posting it to the Openstack metadata service.
- The worker-installer interface operations are given values for the user and password for the `cloudify_agent` input - the password uses the [get_attribute]({{< relref "blueprints/spec-intrinsic-functions.md#get-attribute" >}}) feature to retrieve the decrypted password from the Server's runtime properties (Note that in this example, only the `install` operation was given with this input, but all of the worker installer operations as well as the plugin installer operations should be given with it).
- We define custom userdata which configures WinRM and installs Python on the machine (Windows Server 2012 in this example) once it's up. This is required for the Cloudify agent to be installed on the machine.

CHAPTER 5

Tips

- It is highly recommended to **ensure that Openstack names are unique** (for a given type): While Openstack allows for same name objects, having identical names for objects of the same type might lead to ambiguities and errors.
- To set up DNS servers for Openstack servers (whether it's the Cloudify Manager or application VMs), one may use the Openstack `dns_nameservers` parameter for the [Subnet type](#cloudifyopenstacknodessubnet) - that is, pass the parameter directly to Neutron by using the `args` input of the operations in Subnet node, e.g.:

```
my_subnet_node:
  interfaces:
    cloudify.interfaces.lifecycle:
      create:
        inputs:
          args:
            dns_nameservers: [1.2.3.4]
    cloudify.interfaces.validation:
      creation:
        inputs:
          args:
            dns_nameservers: [1.2.3.4]
```

  This will set up `1.2.3.4` as the DNS server for all servers on this subnet.
- Public keys, unlike the rest of the Openstack resources, are user-based rather than tenant-based. When errors indicate a missing keypair, make sure you're using the correct user rather than tenant.
- ICMP rules show up on Horizon (Openstack GUI) as ones defined using `type` and `code` fields, rather than a port range. However, in the actual Neutron (and Nova, in case of Nova-net security groups) service, these fields are represented using the standard port range fields (i.e., `type` and `code` correspond to `port_range_min` and `port_range_max` (respectively) on Neutron security groups, and to `from_port` and `to_port` (respectively) on Nova-net security groups).

  **** For example, to set a security group rule which allows ping from anywhere, the following setting may be declared in th**

    – `protocol: icmp`

  - port_range_min: 0 (type)
  - port_range_max: 0 (code)
  - remote_ip_prefix: 0.0.0.0/0

- To use Openstack Neutron's ML2 extensions, use the args input for the Network's create operation. For example, the provider network may be set in the following way:

```
my_network:
  type: cloudify.openstack.nodes.Network
  ...
  interfaces:
    cloudify.interfaces.lifecycle:
      create:
        inputs:
          args:
            # Note that for this parameter to work, OpenStack must be configured␣
↪to use Neutron's ML2 extensions
            provider:network_type: vxlan
```

- Ordering NICs in the Openstack plugin can be done in the 1.4 version of the Openstack plugin by simply stating the relationships to the various networks (or ports) in the desired order, e.g.:

```
node_templates:
  server:
    type: cloudify.openstack.nodes.Server
    relationships:
      - target: network1
        type: cloudify.relationships.connected_to
      - target: network2
        type: cloudify.relationships.connected_to

  network1:
    type: cloudify.openstack.nodes.Network
    properties:
      resource_id: network1

  network2:
    type: cloudify.openstack.nodes.Network
    properties:
      resource_id: network2
```

  In the example above, network1 will be connected to a NIC preceding the one network2 will - however these wont be eth0/eth1, but rather eth1/eth2 - because by default, the management network will be prepended to the networks list (i.e. it'll be assigned to eth0). To avoid this prepending, one should explicitly declare a relationship to the management network, where the network's represented in the blueprint by an existing resource (using the "use_external_resource" property). This will cause the management network adhere the NICs ordering as the rest of them. Example:

```
node_templates:
  server:
    type: cloudify.openstack.nodes.Server
    properties:
      management_network_name: network2
    relationships:
      - target: network1
        type: cloudify.relationships.connected_to
      - target: network2
```

```
        type: cloudify.relationships.connected_to
      - target: network3
        type: cloudify.relationships.connected_to

  network1:
    type: cloudify.openstack.nodes.Network
    properties:
      resource_id: network1

  network2:
    type: cloudify.openstack.nodes.Network
    properties:
      use_external_resource: true
      resource_id: network2

  network3:
    type: cloudify.openstack.nodes.Network
    properties:
      use_external_resource: true
      resource_id: network3
```

In this example, "network2" represents the management network, yet it'll be connected to eth1, while "network1" will take eth0, and "network3" (which also happened to already exist) will get connected to eth2.

> The server's property "management_network_name: network2" is not mandatory for this to work - this was just to make the example clear - yet the management network can also be inferred from the provider context (which is what happens when this property isn't explicitly set). Were the provider context to have "network2" set as the management network, this example would've worked just the same with this property omitted.

CHAPTER 6

Misc

- The plugin's operations are each **transactional** (and therefore also retryable on failures), yet not **idempotent**. Attempting to execute the same operation twice is likely to fail.
- Over this documentation, it's been mentioned multiple times that some configuration-saving information may be available in the Provider Context. The Openstack manager blueprint and Openstack provider both create this relevant information, and therefore if either was used for bootstrapping, the Provider Context will be available for the Openstack plugin to use.

The exact details of the structure of the Openstack Provider Context are not documented since this feature is going through deprecation and will be replaced with a more advanced one.

CHAPTER 7

Changelog

2.0.1:

- Don't overwrite server['image'] when server is booted from volume
- Fix loading auth_url from environment (OPENSTACK-101)
- Raise an error if server is not attached to a network. Previously an IndexError would be raised.
- Make sure security_group is removed if a later step (rule creation) fails (OPENSTACK-106)
- Fix attempt to access *volume.display_name* (is now .name) (OPENSTACK-108)
- Correctly handle nova_url and neutron_url in openstack_configuration (these are deprecated) (OPENSTACK-109)

2.0:

- Don't require a Server image to be specified if a boot_volume is attached
- Add support for keystone auth v3. auth_url setting must now include version
- Upgraded openstack library dependencies
- Use availability_zone from connected boot_volume if Server doesn't specify
- Embed full docs in plugin repo. Now using sphinxify sphinx extension

1.5:

- Create project, assign existing users with roles and customize quotas.
- Create image from file (local workflow only) or url.
- Add conditional creation to all resources. Create a resource only if it doesn't already exist. Previously, could either use an existing resource, or create it.
- Boot server from volume. Support boot from block storage and not only from image like in previous versions.
- Fix connect port to security group race-condition.
- Get mac address from port after creation.

- Raise error also when external network is missing in floating ip creation. Previously, an error was raised only when floating network id or name was missing.

CHAPTER 8

Plugin Requirements

- Python versions:
  - 2.7.x
- If the plugin is installed from source, then the following system dependencies are required:
  - `gcc`
  - `gcc-c++`
  - `python-devel`

CHAPTER 9

Compatibility

- *Mitaka* official support
- *Liberty* official support
- *Kilo* official support
- *Juno*, *Icehouse* previously supported, not currently tested.

Attention: New in 2.0

The full Keystone URL in *Openstack Configuration* is now required in the `openstack_config` `auth_url` property: eg `http://192.0.2.200:5000/v2.0` or `http://192.0.2.200:5000/v3`.

The Openstack plugin uses various Openstack clients packages. The versions used in Openstack Plugin are as follows:

- keystoneauth1 - 2.12.1
- Keystone client - 3.5.0
- Nova client - 7.0.0
- Neutron client - 6.0.0
- Cinder client - 1.9.0
- Glance client - 2.5.0

CHAPTER 10

Indices and tables

- genindex
- modindex
- search

Cloudify Types Index