OpenStack Instantiation, Provisioning, and Testing

Exercising OpenStack

As most NFV implementors either rely or at least support OpenStack, we have created a repository containing API tests to validate your OpenStack environment as well as a highly configurable script that is responsible for setting up your VMs/VNFs which can be located at https://nougat.cablelabs.com/SNAPS/provisioning.

Tests

While creating the main script responsible for setting up and provisioning the OpenStack networking and VM instances, several dozen tests had been written that not only helps determine issues with the environment deployment script, but also can help quickly determine if your newly deployed OpenStack cluster is running properly.

Environment Setup

Required Python Packages

An attempt has been made to identify all Python packages generally not included with the Python SDK. The set below may not be complete but will give a sense on how to setup your Python 2.7 runtime environment:

- shutil
- time
- ansible
- Crypto
- neutronclient
- glanceclient
- keystoneclient
- novaclient

SSH

Host Keys

As Ansible is being used to not only configure hosts with multiple NICs, it also can be used to provision the newly minted VMs. As Ansible uses the SSH protocol exclusively to perform these tasks, one may encounter issues the first time Ansible attempts to connect with the machine due to the fact SSH clients generally prompt a user to store the generated host key upon first contact. The following stub when entered in your ~/.ssh/config file, when attempting to provision an environment in Lab 2.

Lab 2 settings

```
Host 10.197.123.*
    StrictHostKeyChecking no
    UserKnownHostsFile=/dev/null
```

Proxy

As all communications to Lab 1 must be routed through a proxy, there are several means to configure your SSH client to send its communication through a proxy server. The method shown below uses an executable named "corkscrew" to help facilitate. Attached is an executable that has been tested on OSx and must be placed into your ~/.ssh directory: corkscrew. The following stub should be added to your ~/.ssh/config file and demonstrates connecting to the Lab 1 environment.

Lab 1 SSH Proxy Settings

```
Host 10.197.103.*
    ProxyCommand ~/.ssh/corkscrew localhost 3128 %h %p
```

Configuration

To run the tests, the {{ repo_dir }}/openstack/tests/conf/os_env.yaml must be edited with your environmental settings:

- username - the tenant's user
- password - the tenant's user password
- os_auth_url - the URL for OpenStack API access
- tenant_name - the name of the tenant on which the tests should run
- ext_net - the name of the external network that will be routed to the test network
- http_proxy - the {{ host }}:{{ port }} through which the web service API traffic must be sent

Test Descriptions

- ./openstack/tests/create_image_tests.py - Exercises basic Glance capabilities
- ./openstack/tests/neutron_utils_tests.py - Exercises basic Neutron capabilities
- ./openstack/tests/nova_utils_tests.py - Exercises basic Nova capabilities
- ./openstack/tests/create_keypairs_tests.py - Tests the creation of key/pairs which are required for SSH access to deployed Nova instances
- ./openstack/tests/create_network_tests.py - Test the creation of a tenant network
- ./openstack/tests/create_instance_tests.py - Tests the proper creation of instances including the configuration of multiple NICs (only the first one "eth0" is able to obtain an IP address)
- ./provisioning/ansible/tests/ansible_utils_tests.py - Tests the provisioning of a VM via Ansible

Deployment

The central Python script in question here is ./VNF/deploy_vnfs.py where it takes one required argument which is the path to the configuration YAML file.

Configuration

The configuration file used to deploy and provision a virtual environment has been designed to describe the required images, networks, SSH public and private keys, associated VMs, as well as any required post deployment provisioning tasks. A fully formed sample can be found in the ./provisioning/ansible/unimgr/deploy-unimgr.yaml that can be dowloaded from here. Please note that many of the more esoteric optional supported attributes still have not been fully tested.

- openstack: - the top level tag that denotes configuration for the OpenStack components
  - connection: - contains the credentials and endpoints required to connect with OpenStack
    - username: - the tenant's user (required)
    - password: - the tentant's user password (required)
    - auth_url: - the URL to the OpenStack APIs (required)
    - tenant_name: - the name of the OpenStack tenant for the user (required)
    - http_proxy: - the {{ host }}:{{ port }} of the proxy server the HTTPPhotoman01(optional)
  - images: - describes each image
    - name: The unique image name. If the name already exists for your tenant, a new one will not be created (required)
    - format: The format type of the image i.e. qcow2 (required)
    - download_url: The HTTP download location of the image file (required)
    - local_download_path: The local directory used to stage the image prior to sending it to OpenStack
  - networks:
    - network:
      - name: The name of the network to be created. If one already exists, a new one will not be created (required)
      - subnet:
        - name: The name of the network to be created. If one already exists, a new one will not be created. Note: although OpenStack allows for multiple subnets to be applied to any given network, we have not included support as our current use cases does not utilize this functionality (required)
        - cidr: The subnet mask value (required)
        - dns_nameservers: A list of IP values used for DNS resolution (default: 8.8.8.8)
        - ip_version: 4|6 (default: 4)
        - tenant_id: ID of the tenant who owns the network. Note: only administrative users can specify tenants other than their own (optional)
        - allocation_pools: A dictionary containing the valid start and end addresses to be assigned (optional)
        - start: The start address for allocation_pools (optional)
        - end: The ending address for allocation_pools (optional)
        - gateway_ip: The IP address to the gateway (optional)
        - enable_dhcp: T|F (optional)
        - host_routes: A list of host route dictionaries (optional)

- For example:

```
"host_routes":[
{
"destination":"0.0.0.0/0",
"nexthop":"123.456.78.9"
},
{
"destination":"192.168.0.0/24",
"nexthop":"192.168.0.1"
}
]
```

        - destination: The destination for a static route (optional)
        - nexthop: The next hop for the destination (optional)
        - ipv6_ra_mode: Valid values: "dhcpv6-stateful", "dhcpv6-stateless", and "slaac" (optional)
        - ipv6_address_mode: Valid values: "dhcpv6-stateful", "dhcpv6-stateless", and "slaac" (optional)
    - router:
        - name: The name of the router to be created. If one already exists, a new one will not be created (required)
        - external_gateway: A dictionary containing the external gateway parameters: "network_id", "enable_snat", "external_fixed_ips" (optional)
        - enable_snat: T|F (default True)
        - external_fixed_ips: Dictionary containing the IP address parameters (optional)
- keypairs:
    - keypair:
        - name: The name of the keypair to be created. If one already exists, a new one will not be created but simply loaded from its configured file location (required)
        - public_filepath: The path to where the generated public key will be stored if it does not exist (optional but really required for provisioning purposes)
        - private_filepath: The path to where the generated private key will be stored if it does not exist (optional but really required for provisioning purposes)
- instances:
    - instance:
        - name: The unique instance name for tenant. (required)
        - flavor: Must be one of the preconfigured flavors (required)
        - imageName: The name of the image to be used for deployment (required)
        - keypair_name: The name of the keypair to attach to instance (optional but required for NIC configuration and Ansible provisioning)
        - sudo_user: The name of a sudo_user that is attached to the keypair (optional but required for NIC configuration and Ansible provisioning)
        - ports: A list of port configurations (should contain at least one)
            - port: Denotes the configuration of a NIC
                - name: The unique port name for tenant (required)
                - network_name: The name of the network to which the port is attached (required)
                - ip: The assigned IP address (when null, OpenStack will assign an IP to the port)
                - admin_state_up: T|F (default True)
                - tenant_id: The ID of the tenant who owns the network. Only administrative users can specify a the tenant ID other than their own (optional)
                - mac_address: The desired MAC for the port (optional)
                - fixed_ips: A dictionary that allows one to specify only a subnet ID, OpenStack Networking allocates an available IP from that subnet to the port. If you specify both a subnet ID and an IP address, OpenStack Networking tries to allocate the specified address to the port. (optional)
                - seurity_groups: A list of security group IDs (optional)
                - allowed_address_pairs: A dictionary containing a set of zero or more allowed address pairs. An address pair contains an IP address and MAC address. (optional)
                - opt_value: The extra DHCP option value (optional)
                - opt_name: The extra DHCP option name (optional)
                - device_owner: The ID of the entity that uses this port. For example, a DHCP agent (optional)
                - device_id: The ID of the device that uses this port. For example, a virtual server (optional)
        - floating_ip: Configure when instance requires external access (optional)
            - ext_net: The name of the external network on which to attach the floating IP (required)
            - port_name: The name of the port on which to bind the port (required)
- ansible:
    - playbook_location: The absolute or relative path to the playbook to execute (required)
    - hosts: A list of hosts to which the playbook will be executed (required)
    - variables: Should your Ansible scripts require any substitution values to be applied with Jinga2 templates, the values defined here will be used to for substitution
        - tag name = substitution variable names. For instance, for any file being pushed to the host being provisioned containing a value such as {{ foo }}, you must specify a tag name of "foo"
            - vm_name:
            - type: string|port

- when type == string, an tag name "value" must exist and its value will be used for template substituion
- when type == port, custom code has been written to extract certain assigned values to the port:
    - port_name: The name of the port from which to extract the substitution values (required)
    - port_value: The port value. Currently only supporting "mac_address" and "ip_address" (only the first)

Execution

Once your Python environment has the necessary libraries installed you will also have to set your PYTHONPATH environment variable to execute deploy_vnfs.py from the command line. Prior to executing the main script from the repository home directory, export the PYTHONPATH value within the same shell as follows:

Setting up shell session

```
export PYTHONPATH=`pwd`:$PYTHONPATH
```

Once set, your shell session is all set to execute any of the Python scripts within the repository. To deploy and provision the UNIManager also from the repository home directory:

Deployment/Provisioning Invokation

```
./VNF/deploy_vnfs.py provisioning/ansible/unimgr/deploy-unimgr.yaml
```

Planned Future Enhancements

- Add the ability to undeploy environment based on configuration
- Add testing to more esoteric settings
- Add support for multiple subnets on a network
- Container support
- ???