Remote and centralized OpenStack installation and management

Experience at PG
Mirko Mariotti
mirko.mariotti@unipg.it
acknow. AMS / ASI and PG colleagues

OpenStack at PG

- Small OpenStack installation (~1000 cores)
- Computational resources for local researcher, students, labs, events.
- Not only services, base for our R&D on cloud technologies

OpenStack at PG (some detail)

- AA federated with INFN-AAI and Unipg IDM
- Network virtualization via neutron and VLAN backend
- Storage: cinder, ceph
- Two installation, one production (Mitaka), one development (Rocky)
- OpenStack core machine also virtualized (outside OS)
- Resources also in ASI (Rome) and Chieti

We will talk about our small experience about:

- Automatic OpenStack installation
- Integration of remote resources

Automatic OpenStack installation via Ansible

- We chose Ansible as automatic installation system
- The approach is inventory-centric (keep as much as possible within nodes variables and build configurations out of it)
- Use ansible roles to map different OpenStack/Ceph elements
- Idempotency of playbooks

OpenStack inventory

```
farm_os_rocky:
  hosts:
    rockydb1:
    rockydb2:
    rockydb3:
    rockycc:
      os_services:
        - glance-api
        - glance-registry
        - cinder-scheduler
        - nova-api
        - nova-consoleauth
        - nova-scheduler
        - nova-conductor
        - nova-novncproxy
        - neutron-server
    rockyns:
      provider_interface_name: ens6
      overlay_interface_ip_address: 10.13.100.4
      os_services:
        - neutron-linuxbridge-agent
        - neutron-dhcp-agent
        - neutron-metadata-agent
        - neutron-l3-agent
  vars:
    redmon_style: "rclocal"
    authorized_keys_of_the_group: []
    authorized_key_list:
      - name: root
        state: present
        authorized_keys: "{{ mainkey }} + {{ authorized_keys_of_the_group }}"
    redmon_enabled: "yes"
    redmon_views:
      - rocky os load
```

```
farmexecution_rocky:
  hosts:
    farm-comp-13-01:
      clustername: clusterdev
      checked_roles: cephconsumer
      vm_backend: ceph
      provider_interface_name: enp5s0f1np1
      overlay_interface_ip_address: 10.13.113.1
    farm-comp-13-02:
      vm_backend: local
      provider_interface_name: eno2
      overlay_interface_ip_address: 10.13.113.2
    farm-comp-13-03:
      vm_backend: ceph
      provider_interface_name: enp65s0f1np1
      overlay_interface_ip_address: 10.13.113.3
  vars:
    checked_roles: rockynovaconfig
    os_services:
    #  - neutron-dhcp-agent
      - neutron-linuxbridge-agent
    #  - neutron-metadata-agent
      - nova-compute
    redmon_style: "rclocal"
    authorized_keys_of_the_group:
      - key: "{{ lookup('file', './keys/managers') }}"
    authorized_key_list:
      - name: root
        state: present
        authorized_keys: "{{ mainkey }} + {{ authorized_keys_of_the_group }}"
    redmon_enabled: "yes"
    redmon_views:
      - rocky os load
```

OpenStack Installation:

OpenStack components mapped to ansible roles:

- Common configuration present in every node (rockycommon)
- Compute installation (rockycompute)
- Compute configuration (rockynovaconfig)
- Network configuration (rockyneutronconfig)

OpenStack playbooks

```yaml
- hosts: '{{ host }}'
  roles:
    - pgdefault
    - rockycommon
    - rockycompute
    - cephclient
    - rockynovaconfig
    - rockyneutronconfig

  tasks:

    - name: Restart Openstack Services
      systemd:
        state: restarted
        daemon_reload: no
        name: "{{ item }}"
      loop: "{{ os_services }}"

    - name: Load the Hypervisors
      command: su -s /bin/sh -c "nova-manage cell_v2 discover_hosts --verbose" nova
      delegate_to: rockycc
```

```
ansible-playbook -i inventory.yaml pb_create_rocky_compute.yaml
--extra-vars='{"host" : "farm-comp-13-01" }'
```

OpenStack roles example

Rockyneutronconfig role

```
- name: Copy neutron.conf
  template:
    src: neutron.conf.j2
    dest: /etc/neutron/neutron.conf
    owner: root
    group: neutron
    mode: '0640'


- name: Copy ml2_conf.ini
  template:
    src: ml2_conf.ini.j2
    dest: /etc/neutron/plugins/ml2/ml2_conf.ini
    owner: root
    group: neutron
    mode: '0644'


- name: Copy linuxbridge_agent.ini
  template:
    src: linuxbridge_agent.ini.j2
    dest: /etc/neutron/plugins/ml2/linuxbridge_agent.ini
    owner: root
    group: neutron
    mode: '0644'
  when: provider_interface_name is defined
```

- The playbook apply the roles
- roles contain the rules to specialize the configurations using variables from the inventory
- The resulting filled in configuration file is pushed on the node.

linuxbridge_agent.ini.j2

```
# Comma-separated list of <physical_network>:<physical_interface> tuples
# mapping physical network names to the agent's node-specific physical network
# interfaces to be used for flat and VLAN networks. All physical networks
# listed in network_vlan_ranges on the server should have mappings to
# appropriate interfaces on each agent. (list value)
physical_interface_mappings = provider:{{ provider_interface_name }}

# List of <physical_network>:<physical_bridge> (list value)
#bridge_mappings =
```

Ceph Installation

- Ceph components mapped to ansible roles:
  - Cephmaster (ceph-deploy)
  - Cephnode (storage nodes and admin nodes)
  - Cephconsumer (nodes that use the storage)

Ceph Installation - The inventory

```
farm_ceph_master_01:
  hosts:
    ceph-master-01:
      checked_roles:
        - pgdefault
        - cephmaster
      farmmgm_net: 10.13.0.0/16
      ceph_net: 10.23.0.0/16
      clustername: clusterceph01
      groupname: farm_ceph_cluster_01
      ceph_master_key: storagemasterprod
      info_enabled: "yes"
      redmon_update: "yes"
      redmon_enabled: "yes"
      redmon_style: "systemd"
      redmon_metrics:
        - { name: "load1", config_template: "redmonload1.j2", script_template: "rmscript_load1.j2", script_target: "rmscript_load1" , grafana_dashboards: [] , redmon_views: ["ceph load"]}
        - { name: "netinvlan402", config_template: "redmonnetin.j2", script_template: "rmscript_netin.j2", script_target: "rmscript_netinvlan402" , nic: "vlan402", grafana_dashboards: [] , redmon_views: ["ceph net"]}
        - { name: "netoutvlan402", config_template: "redmonnetout.j2", script_template: "rmscript_netout.j2", script_target: "rmscript_netoutvlan402" , nic: "vlan402", grafana_dashboards: [] , redmon_views: ["ceph net"]}
      authorized_keys_of_the_group:
        - key: "{{ lookup('file', './keys/dipstors') }}"
      authorized_key_list:
        - name: root
          state: present
          authorized_keys: "{{ mainkey }} + {{ authorized_keys_of_the_group }}"
        - name: storagemaster
          state: present
          authorized_keys: "{{ mainkey }} + {{ authorized_keys_of_the_group }}"


farm_ceph_cluster_01:
  hosts:
    farm-ceph-03:
      is_mon: true
      is_initial_mon: true
      is_mgr: true
      is_initial_mgr: true
      is_admin: true
      checked_roles:
        - pgdefault
        - cephnode
      main_if: eno1
      ceph_if: eno2
      ceph_ip: 10.23.237.3
      farmmgm_if: vlan402
      farmmgm_ip: 10.13.237.3
      is_netboot: "yes"
      osds:
        - { device: "/dev/disk/by-id/ata-ST1000NM0018-2F2130_ZFA0D6BV" }
        - { device: "/dev/disk/by-id/ata-TOSHIBA_MG04ACA400NY_X85TK3QBF7DE" }
```

```
pools:
  glance-images:
    pg_num: "128"
    size: "3"
  volumes:
    pg_num: "128"
    size: "3"
  vms:
    pg_num: "128"
    size: "2"
cephx:
  cinder:
    key: "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
    caps mon: "\"profile rbd\""
    caps osd: "\"profile rbd pool=volumes, profile rbd pool=vms, profile rbd-read-only pool=glance-images\""
  glance:
    key: "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
    caps mon: "\"profile rbd\""
    caps osd: "\"profile rbd pool=glance-images\""
```

Ceph Roles examples

Node

```
- name: Configure osds
  when: item[0].device == item[1].item.device and item[1].stdout == 'false'
  args:
    chdir: /home/storagemaster/{{ clustername }}
  shell: |
    /usr/bin/ceph-deploy osd create --data {{ item[0].device }} {{ inventory_hostname }}
  loop: "{{ osds | product(configurated_osds.results) | list  }}"
  delegate_to: "{{ cephmaster }}"
  become: yes
  become_user: storagemaster


- name: Check if monitors are running
  shell: |
    ( /usr/bin/pgrep ceph-mon > /dev/null  && echo "true" ) ||  echo "false"
  register: monrunning

#- debug: var=monrunning


- name: Create admin nodes
  shell: |
    /usr/bin/ceph-deploy admin {{ inventory_hostname }}
  args:
    chdir: /home/storagemaster/{{ clustername }}
  delegate_to: "{{ cephmaster }}"
  become: yes
  become_user: storagemaster
  when:
    - is_admin is defined
    - is_admin == true


- name: Create new monitors
  shell: |
    /usr/bin/ceph-deploy mon create {{ inventory_hostname }}
  args:
    chdir: /home/storagemaster/{{ clustername }}
  delegate_to: "{{ cephmaster }}"
  become: yes
  become_user: storagemaster
  when:
    - is_mon is defined
    - is_mon == true
    - monrunning.stdout == "false"
```

Master

```
- name: "Update ssh client config on {{ cephmaster }}"
  shell: |
    echo "Host {{ item }}" >> /home/storagemaster/.ssh/config
    echo "\tHostname {{ item }}" >> /home/storagemaster/.ssh/config
    echo "\tUser storagemaster" >> /home/storagemaster/.ssh/config
    echo >> /home/storagemaster/.ssh/config
  loop: "{{ groups[groupname] }}"
  become: yes
  become_user: storagemaster
```

Consumer

```
- name: Build hosts file
  lineinfile: dest=/etc/hosts regexp='.*{{ item }}$' line="{{ hostvars[item].farmmgm_ip }} {{item}}" state=present
  loop: "{{ groups[groupname] }}"

- name: Keyring creation
  template:
    src: keyring.j2
    dest: /etc/ceph/ceph.client.{{ item.key }}.keyring
    owner: root
    group: root
    mode: '0644'
  loop: "{{ lookup('dict',cephx, wantlist=True) }}"

~
~
```

Ceph Cluster creation and Management examples

```
- hosts: '{{ host }}'
  tasks:
  - include_role:
      name: '{{ role }}'
    when:
      - role is defined
      - checked_roles is defined
      - role in checked_roles
```

ansible-playbook -i inventory.yaml pb_apply_role.yaml --extra-vars='{"host" : "farmmgmtest" , "role": "cephconsumer" }'

ansible-playbook -i inventory.yaml pb_create_ceph_cluster.yaml --extra-vars='{"ceph_master" : "ceph-master-01" , "ceph_nodes": "farm_ceph_cluster_01" }'

```
- hosts: '{{ ceph_master }}'
  tasks:
  - include_role:
      name: pgdefault
  - include_role:
      name: cephmaster
    when:
      - checked_roles is defined
      - '"cephmaster" in checked_roles'

- hosts: '{{ ceph_nodes }}'
  tasks:
  - include_role:
      name: pgdefault
  - include_role:
      name: cephnode
    when:
      - checked_roles is defined
      - '"cephnode" in checked_roles'
```

Integration of remote resources

A single OpenStack installation … but the sites have to be as much autonomous as possible especially regarding:

- Storage
- Outbound connectivity

Cross-site operations have to be possible (knowing the risks).

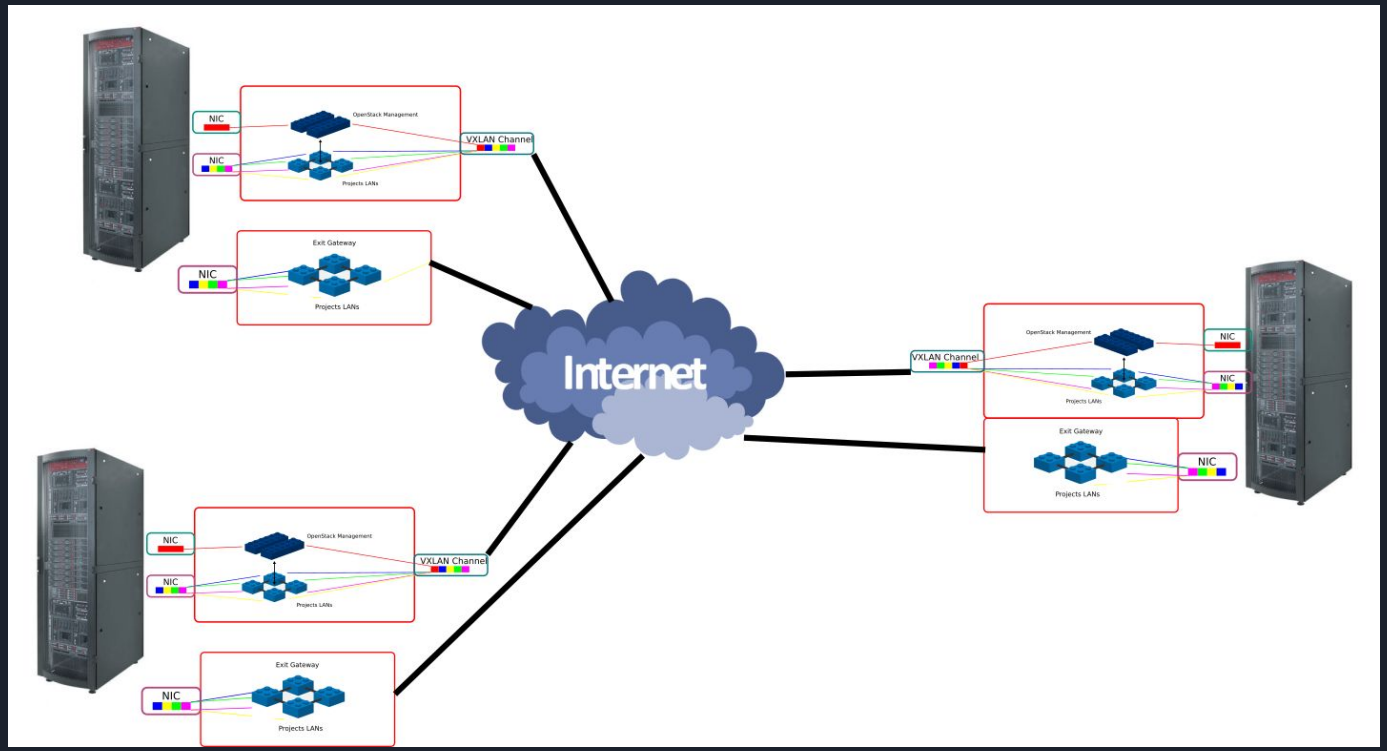
Ideally the traffic among sites would be only the OpenStack management one.

Resource organized in different zones logically correspondent to different geographical locations.

SDN (software-defined networking) solution to connect the different zones.

All build with standard servers and Linux systems.

Overall


NIC
NIC
OpenStack Management
Projects LANs
VXLAN Channel
Exit Gateway
NIC
Projects LANs
Internet
VXLAN Channel
OpenStack Management
NIC
NIC
Projects LANs
Exit Gateway
NIC
Projects LANs
NIC
NIC
OpenStack Management
Projects LANs
VXLAN Channe
Exit Gateway
NIC
Projects LANs

Network Requirements for a remote site

- 2 Public IPs (one for the link with the central site, one for the outbound connectivity)
- VLAN capable switches
- Hypervisors with 2 NICs (Optionally)

SDN

Software-Defined Networking is a way to overlay multiple networks to a single physical fabric and to control them via software.

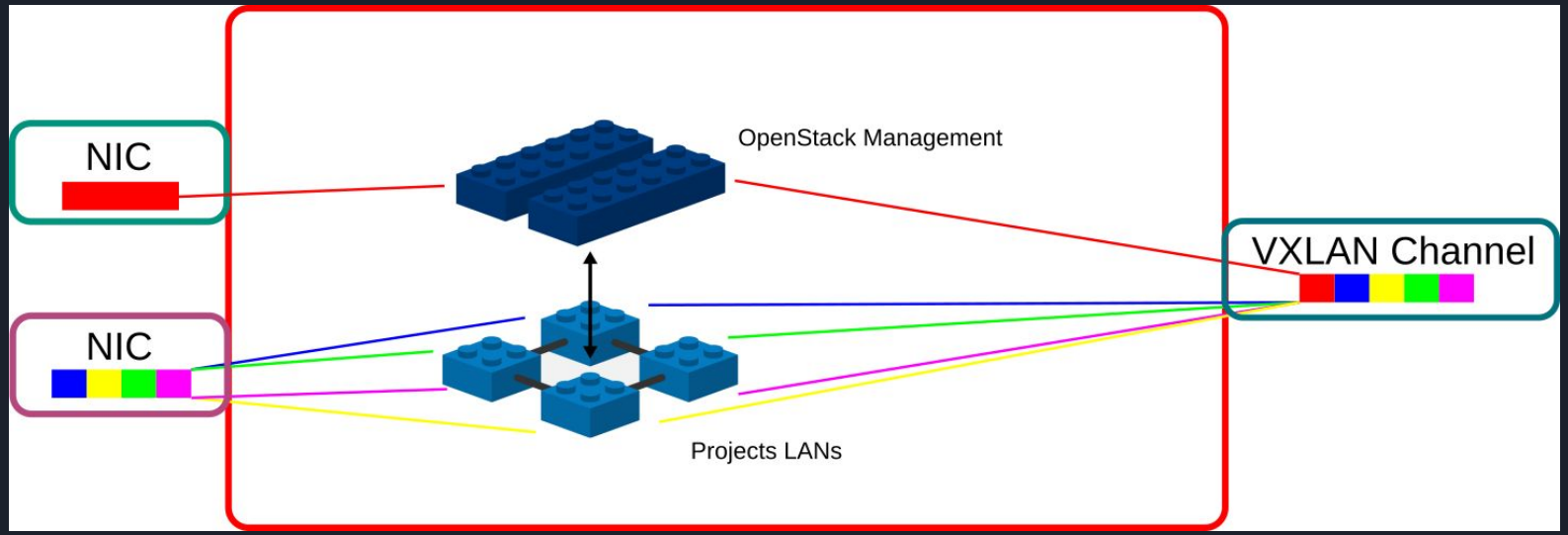
Openvswitch is an open source project for SDN

Used for network virtualization in many cloud framework

We are using this approach and Openvswitch also to the physical infrastructure.

How we use SDN (from OpenStack perspective)

We use a Linux box for each site to “virtualize” the openstack LAN (Both management and projects) and transport it to other sites.



SDN configuration

```
[asigw]> /root #  ovs-vsctl show
14f2df09-7723-41df-beb4-67d1ad48d064
    Bridge asitunnel
        Port asitunnel
            Interface asitunnel
                type: internal
        Port "enp4s3"
            trunks: [402, 1016, 1065]
            Interface "enp4s3"
        Port "vxlan0"
            tag: 0
            trunks: [402, 1016, 1065]
            Interface "vxlan0"
                type: vxlan
                options: {df_default="false", key=flow, remote_ip="10.199.190.4"}
        Port ams
            tag: 0
            Interface ams
                type: internal
    ovs_version: "2.5.2"
[asigw]> /root #
```

```
auto asitunnel
allow-ovs asitunnel
iface asitunnel inet manual
        ovs_type OVSBridge
        ovs_ports ams enp4s3 vxlan0

allow-asitunnel enp4s3
iface enp4s3 inet manual
        ovs_bridge asitunnel
        ovs_type OVSPort
        ovs_options vlan_mode=trunk trunks=402,1016,1065

allow-asitunnel vxlan0
iface vxlan0 inet manual
        ovs_bridge asitunnel
        ovs_type OVSTunnel
        ovs_tunnel_type vxlan
        ovs_options tag=0 vlan_mode=native-untagged
trunks=402,1016,1065
        ovs_tunnel_options options:remote_ip=10.199.190.4
options:key=flow options:df_default=false
```

Remote Automation

The sites are L2 connected, every automatic installation/configuration mechanism available on the master site works out of the box on the remote sites.

The ansible playbook can easily be used on remote sites.

Start-up and Management requirements (for the site Admin)

Start-up:

- Operating system installation on nodes with the public IP
- Nodes cabling
- Switch configuration (also from remote)
- Operating system installation on nodes
- Nodes network configuration on management VLAN
- Enable access to the Ansible server to the nodes

Management:

- Check on hardware failures

Current WorkFlows

Currently on this infrastructure we run different workflows.

For AMS:

- DODAS batch system
- HVO, a website on ASI exposed at PG

For FERMI:

- DODAS batch system

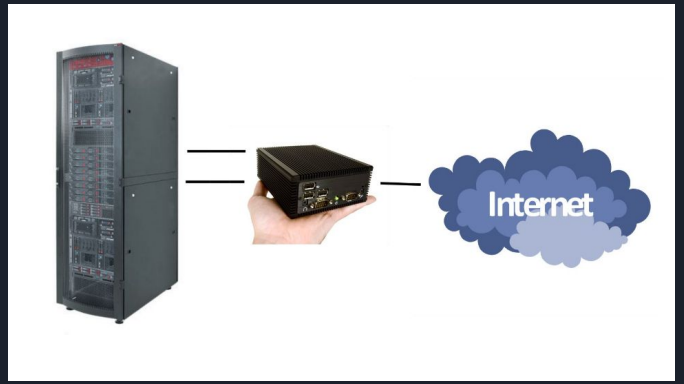
None of this workflows require effort from remote sites admin

Further possible development

More elements can be automated:

- Switches configuration (openflow or ansible)
- Openvswitch (or linuxbridge) via Ansible
- The Operating System installation on inner nodes.

It is possible to pack everything in an appliance to extend an OpenStack installation



Summary

We found an easy, net-centric, way to use remote resources within our OpenStack installation.

The solution has low impact on the remote site admins because:

- It is a single OS installation
- It is build in a automated way

Backup slides

Bandwidth requirements

Traffic on the OS controller node:

AVG in: ~ 50 kBit/s/hwnode

AVG out: ~ 25 kBit/s/hwnode

Traffic on a DB/rabbitmq node:

AVG in: ~ 17 kBit/s/hwnode

AVG out: ~35 kBit/s/hwnode

100 Nodes ~ 7MBit/s

Some additional overhead from VM is also expected.