

PA200 - Cloud Computing

Lecture 8: OpenStack cloud administration and cloud software development

by Ilya Etingof, Red Hat

Warm-up

Let's rehearse on the previous lectures...

Q: OpenStack instance information

- Machine flavor
- Boot image name
- Network name
- Storage options
- SSH key name
- Heat template

Q: Heat requires

- Heat template (HOT)
- Environment file
- User configuration file
- Stacks

Q: Why Heat?

- Hyperconverged Environment and Automation Tracker (HEAT)
- Heat evaporates a liquid and raises clouds
- To honor H.E.A.T. rock band

Q: OpenStack design pillars

- Based on open protocols
- Loosely coupled services
- Interacting over REST APIs
- Each service is a project backed by a team
- Coded in Python programming language

Q: Typical OpenStack service structure

- REST API
- Database
- Business logic engine

- Message bus
- Remote agent
- UI

Q: OpenStack key services

- Nova (compute service)
- Neutron (network service)
- Glance (image service)
- Swift (object storage)
- Cinder (block storage)
- Heat (orchestration)
- Ironic (bare metal)
- Keystone (authentication service)

Q: OpenStack development pillars

- Open source
- Open community
- Open design
- Open development

Q: OpenStack future

- Integration with PaaS (containers)
- Software Defined Networking
- Network Function Virtualization
- Integration with hardware
- Stretching the cloud (edge)

In this lecture...

- HOT resources
- Working with Heat
- Install and manage OpenStack cloud

Install OpenStack client

```
$ python3 -m venv osc
$ source osc/bin/activate
$ pip install openstackclient
```

Configure OpenStack for MUNI

```
$ cat ~/.config/openstack/clouds.yaml
clouds:
  muni-cloud:
    auth:
      auth_url: https://ostack.ics.muni.cz/...
      project_name: ...
      username: <U#O>@ucn.muni.cz
      password: <secondary password>
      region_name: ...
```

See also:

- <https://wiki.ics.muni.cz/openstack>

Create SSH keypair

```
$ openstack --os-cloud muni-cloud keypair create muni-key > ~/.ssh/muni
$ cat ~/.ssh/muni
-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEArGE3HRUB0QvkBmLdoIULYbLE9gEWgRuYBpcs7MQHMSq4ocZ7
...
1tMgsAX56h9P6DcDoZymcs4HNAtprmtXMkdTI9EeG0c6gzYsRnY=
-----END RSA PRIVATE KEY-----
```

Create HOT template and environment

```
$ vi pa200-project-2.yaml
```

See also:

<https://docs.openstack.org/heat/latest/>

HOT parameters

Declaring HOT parameters:

```
parameters:
  key_name:
    type: string
    default: my_key
  instance_type:
    type: string
    default: m1.small
    constraints:
      - allowed_values: [m1.tiny, m1.small, m1.medium, m1.large, m1.xlarge]
```

HOT resource: network (1/2)

Allocating virtual network:

```

my_net:
  type: OS::Neutron::Net

my_subnet:
  type: OS::Neutron::Subnet
  depends_on: [ my_net ]
  properties:
    dns_nameservers:
      - 8.8.8.8
    network: { get_resource: my_net }
    ip_version: 4
    cidr: 10.0.0.0/24
    allocation_pools:
      - { start: 10.0.0.10, end: 10.0.0.250 }

```

HOT resource: network (2/2)

Allocating virtual router:

```

my_router:
  type: OS::Neutron::Router
  properties:
    external_gateway_info: {"network": { get_param: public_network }}

my_router_subnet_connect:
  type: OS::Neutron::RouterInterface
  depends_on: [ my_subnet, my_router ]
  properties:
    router: { get_resource: my_router }
    subnet: { get_resource: my_subnet }

```

HOT resource: security groups

Set up firewall:

```

my_sec_group:
  type: OS::Neutron::SecurityGroup
  properties:
    rules:
      - { direction: ingress,
          remote_ip_prefix: 0.0.0.0/0,
          port_range_min: 80,
          port_range_max: 80,
          protocol: tcp }

```

HOT resource: network port (1/2)

Allocate network port (NIC):

```

my_server_port:
  type: OS::Neutron::Port
  properties:
    network_id: { get_resource: my_net }

```

```
fixed_ips:
  - subnet_id: { get_resource: my_subnet }
security_groups: [{ get_resource: my_sec_group }]
```

HOT resource: network port (2/2)

Allocate floating (public) IP:

```
my_server_floating_ip:
  type: OS::Neutron::FloatingIP
  properties:
    floating_network: { get_param: public_network }
    port_id: { get_resource: my_server_port }
```

HOT resource: instance

Allocating a compute instance:

```
my_server:
  type: OS::Nova::Server
  properties:
    image: { get_param: image_id }
    flavor: { get_param: instance_type }
    key_name: { get_param: key_name }
    networks:
      - port: { get_resource: my_server_port }
```

HOT resource: block storage

Allocating block storage device:

```
my_volume:
  type: OS::Cinder::Volume
  properties:
    size: 1
my_volume_attachment:
  type: OS::Cinder::VolumeAttachment
  properties:
    instance_uuid: { get_resource: my_server }
    volume_id: { get_resource: my_volume }
```

HOT resource: instance configuration

Running cloud-init bootstrap script:

```
my_server:
  type: OS::Nova::Server
  properties:
    user_data:
      str_replace:
        template: |
          #!/bin/bash -v
```

```

voldata_id="%voldata_id%"
voldata_dev="/dev/disk/by-id/virtio-$(echo ${voldata_id} | cut -c -20)"
mkfs.ext4 ${voldata_dev}
mkdir -pv /var/lib/dbfiles
echo "${voldata_dev} /var/lib/dbfiles ext4 defaults 1 2" >> /etc/fstab
mount /var/lib/dbfiles

params:
  "%voldata_id%": { get_resource: DatabaseVolume }

```

Create the infrastructure

```

$ openstack --os-cloud muni-cloud stack create --wait --template pa200.yaml pa200
$ openstack --os-cloud muni-cloud stack list
$ openstack --os-cloud muni-cloud stack show pa200

```

Inspect created stack

```

$ openstack --os-cloud rdo-cloud stack resource list pa200
...
$ openstack --os-cloud rdo-cloud stack server list
...
| fcs...588 | pa200-web | ACTIVE | pa200-net-10.0.0.22, 38.145.35.152 | CentOS-7-x86_64-GenericCloud-1804_02 | ml.small |
$ ssh centos@38.145.35.152
centos@pa200-web $

```

Debugging OpenStack project (1/4)

```

$ openstack --os-cloud muni-cloud stack create --wait --template pa200.yaml pa200
2019-04-15 06:58:29Z [pa200]: CREATE_IN_PROGRESS Stack CREATE started
2019-04-15 06:58:29Z [pa200.DatabaseVolume]: CREATE_IN_PROGRESS state changed
2019-04-15 06:58:30Z [pa200.net]: CREATE_IN_PROGRESS state changed
2019-04-15 06:58:31Z [pa200.net]: CREATE_COMPLETE state changed
2019-04-15 06:58:32Z [pa200.router]: CREATE_IN_PROGRESS state changed
2019-04-15 06:58:32Z [pa200.DatabaseVolume]: CREATE_COMPLETE state changed
2019-04-15 06:58:33Z [pa200.subnet]: CREATE_IN_PROGRESS state changed
...
2019-04-15 06:58:59Z [pa200]: CREATE_COMPLETE Stack CREATE completed successfully

```

Debugging OpenStack project (2/4)

```

$ openstack --os-cloud muni-cloud stack show pa200
...
stack_status      | CREATE_FAILED
stack_status_reason | Resource CREATE failed: Conflict: resources.DatabaseServer:
                    | Multiple possible networks found, use a Network ID to be more
                    | specific.
...
$ openstack --os-cloud muni-cloud stack delete -y pa200

```

Debugging OpenStack project (3/4)

```

$ openstack --os-cloud muni-cloud console log show pa200-web
CentOS Linux (3.10.0-862.2.3.el7.x86_64) 7 (Core)
[ 0.000000] Initializing cgroup subsys cpu
[ 0.000000] Initializing cgroup subsys cpuset
[ 0.000000] Linux version 3.10.0-862.2.3.el7.x86_64 (builder@kbuilder.dev.centos.org) (gcc version 4.8.5 20150623 (Red Hat 4.8.5-28) (GCC) ) #1 SMP Wed May 9 18:05:47 UTC 2018
[ 0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-3.10.0-862.2.3.el7.x86_64 root=UUID=5f1871e2-c19c-4e86-8d6c-04d5fd71a0a ro console=tty0 console=ttyS0,115200n8 crashkernel=auto console=ttyS0,115200 LANG=en_US.UTF-8
[ 0.000000] e820: BIOS-provided physical RAM map:
[ 0.000000] BIOS-e820: [mem 0x00000000-0x00000000] usable
...
[ 43.720025] cloud-init[1155]: Cloud-init v. 0.7.9 finished at Mon, 15 Apr 2019 07:07:19 +0000.

```

Debugging OpenStack project (4/4)

```

$ ssh centos@38.145.35.152
centos@pa200-web $ less /var/log/cloud-init-output.log

```

OpenStack cloud installation

Many approaches:

- Manual installation
- PackStack
- DevStack
- Infrared
- Fuel
- TripleO i.e. OpenStack-on-OpenStack
- ... and others

Manual installation (1/2)

- Assign roles to the baremetal machines
- Install and configure OS
- Install OpenStack packages and dependencies
- Configure OpenStack services

Manual installation (2/2)

- Time consuming
- Error prone
- Not easily repeatable
- Manual maintenance

Automated installation (1/2)

- PackStack: via Puppet modules
- DevStack: via bash scripts
- Infrared: via Ansible playbooks

Automated installation (2/2)

- Relatively lightweight and quick
- Tend to focus specific scenarios
- Cloud maintenance might be lacking
- Troubleshooting can be hard

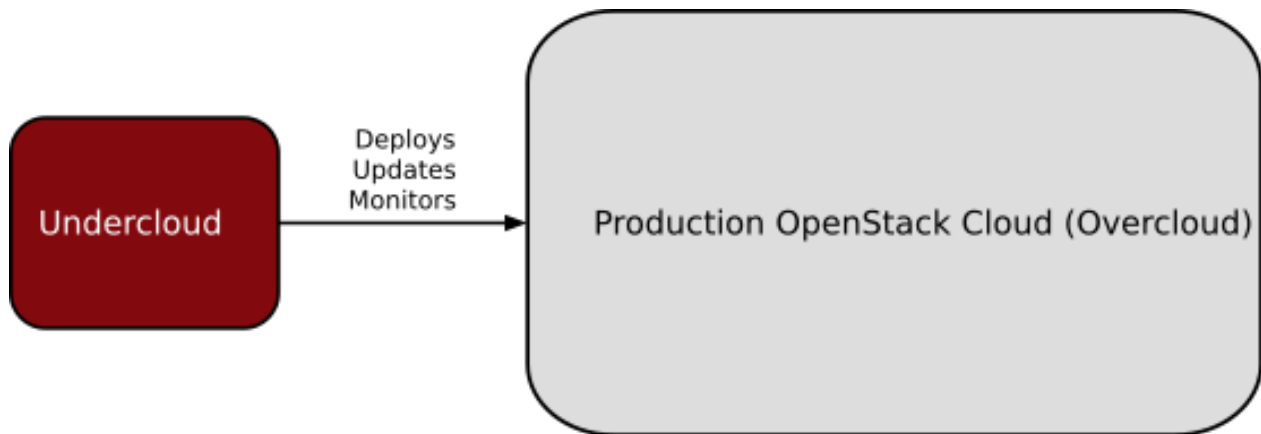
Heavy solutions

- Fuel: OpenStack cloud management system
- TripleO: OpenStack to manage OpenStack

OpenStack-on-OpenStack (1/5)

- Idea: use OpenStack to manager OpenStack
- Deployment cloud: Undercloud
- Workload cloud: Overcloud

OpenStack-on-OpenStack (2/5)

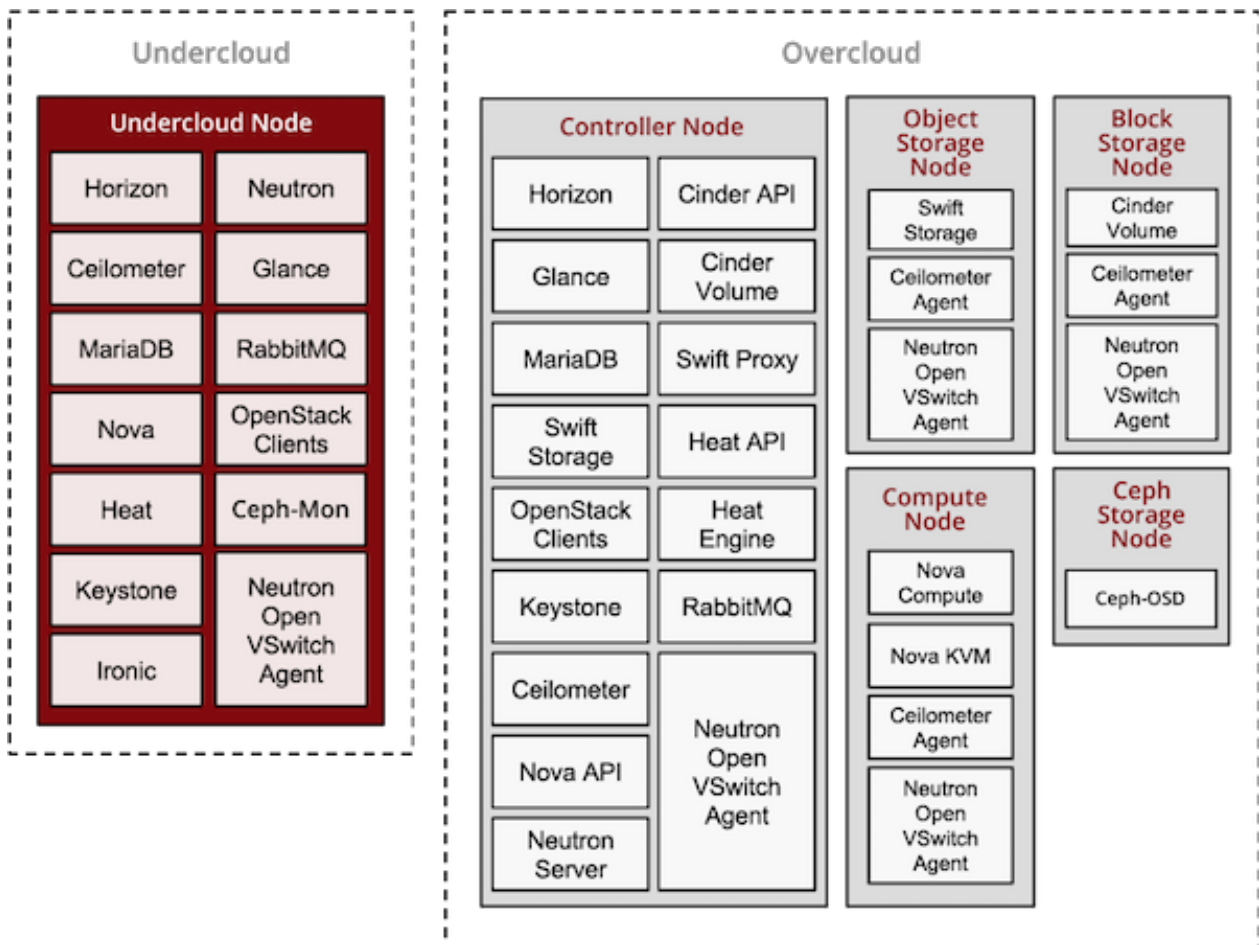


OpenStack-on-OpenStack (3/5)

Undercloud:

- Uses Nova, Neutron, Heat, Glance, Ironic
- Ironic for managing bare metal machines
- Heat templates for OC configuration

OpenStack-on-OpenStack (4/5)



OpenStack-on-OpenStack (5/5)

Benefits:

- Relies on stable software
- Well maintained core OpenStack components
- No need to learn new things
- Scaling OC is easy

Drawbacks:

- Complicated and heavy
- Difficult upgrades

Recap: using Heat (1/2)

- Figure out MUNI OpenStack access
- Install *openstackclient* into Python venv
- Create HOT with:
 - Network (port, security groups)
 - Instance

- Block storage

Recap: using Heat (2/2)

- Create *cloud-init* scripts
 - ...to install and configure software
- Instantiate Heat template
- Debug the deployment

Recap: cloud administration

- Relies on management software (TripleO, Fuel, ...)
- TripleO: undercloud -> overcloud
- Cloud administration is complicated

Questions?

<https://www.openstack.org/>