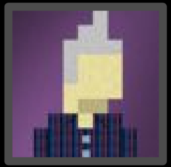
Taming OpenStack with Ansible

OpenStack Juno Summit - May 2014



John Dewey

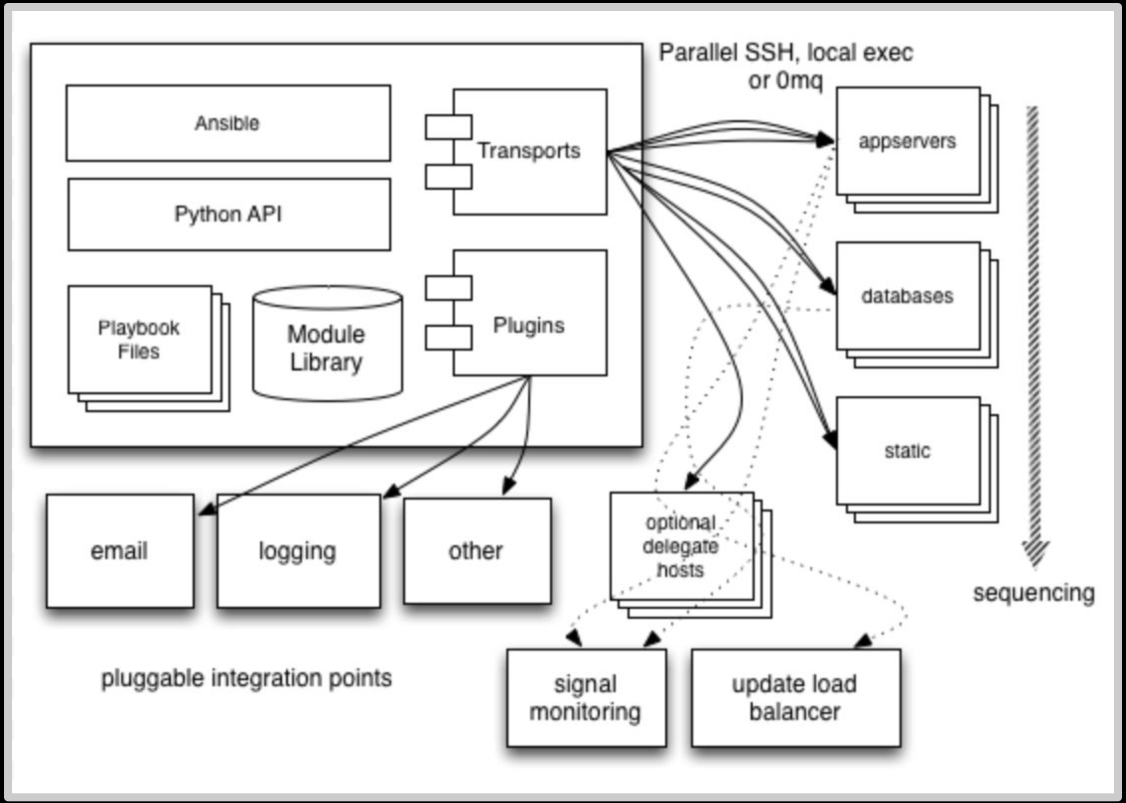
@retr0h

What is Ansible?

What is Ansible?

- Simple, multi-tier application deployment
- Deploys reliably and consistently
- Uses simple task descriptions vs custom code
- Capable of task orchestration

What is Ansible?



Overview

1. Simplicity in mind

Overview

1. Simplicity in mind
    a. Easy to Install

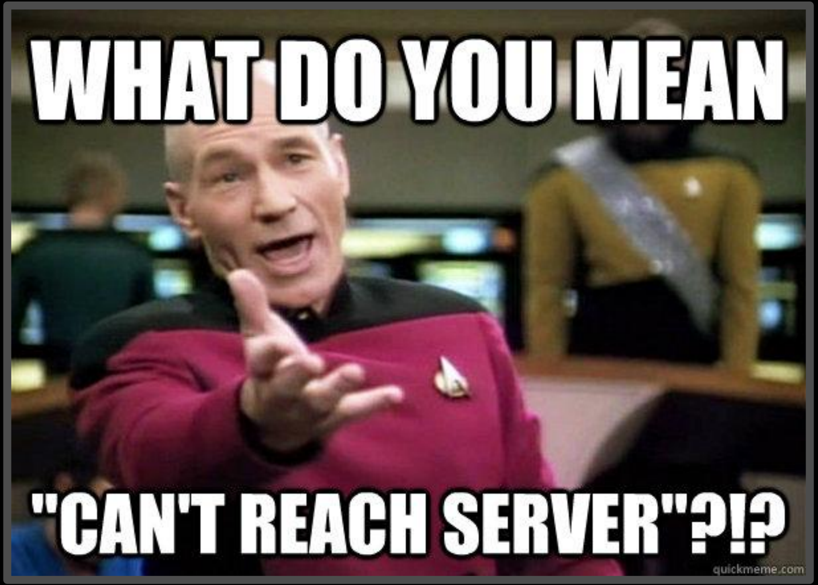
- `apt`
- `yum`
- `pip`
- `git clone`
- `tar`

Requirements

- `python 2.4+`
- `ssh`

Overview

1. Simplicity in mind
    a. Easy to Install
    b. Serverless



Overview

1. Simplicity in mind
   a. Easy to Install
   b. Serverless
   c. Agentless *

Overview

1. Simplicity in mind
   a. Easy to Install
   b. Serverless
   c. Agentless *
   d. YAML*

```
---
- name: add packages
  apt: pkg={{ item }}
  with_items:
    - smem
    - socat
    - pstack
```

Overview

1. Simplicity in mind
   a. Easy to Install
   b. Serverless
   c. Agentless *
   d. YAML*

Loops

- `with_items`
- `with_nested`
- `with_dict`
- `with_fileglob`
- `with_together`
- `with_subelements`
- `with_sequence`
- `with_random_choice`
- `with_first_found`
- `with_lines`
- `with_indexed_items`
- `with_flattened`

Overview

1. Simplicity in mind
   a. Easy to Install
   b. Serverless
   c. Agentless *
   d. YAML*

Conditionals

```
---
tasks:
  - command: /bin/false
    register: result
    ignore_errors: True
  - command: /bin/something
    when: result|failed
  - command: /bin/something_else
    when: result|success
  - command: /bin/still/something_else
    when: result|skipped
```

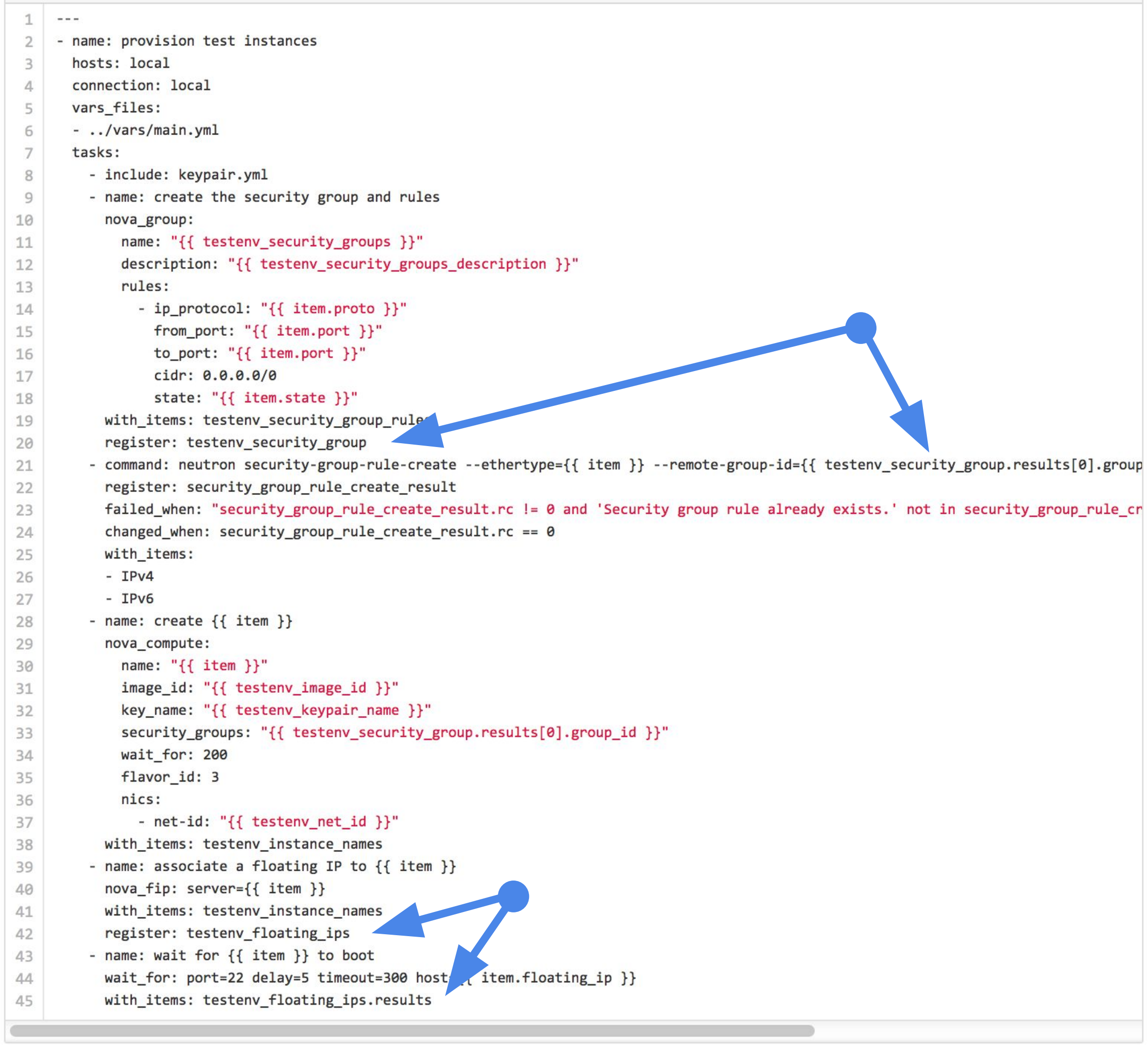
Overview

1. Simplicity in mind
2. Modules

```
$ pwd
/Users/jdewey/git/ansible/library
$ ls cloud/* | wc -l
62
$ find . -type f | wc -l
229
```

Overview

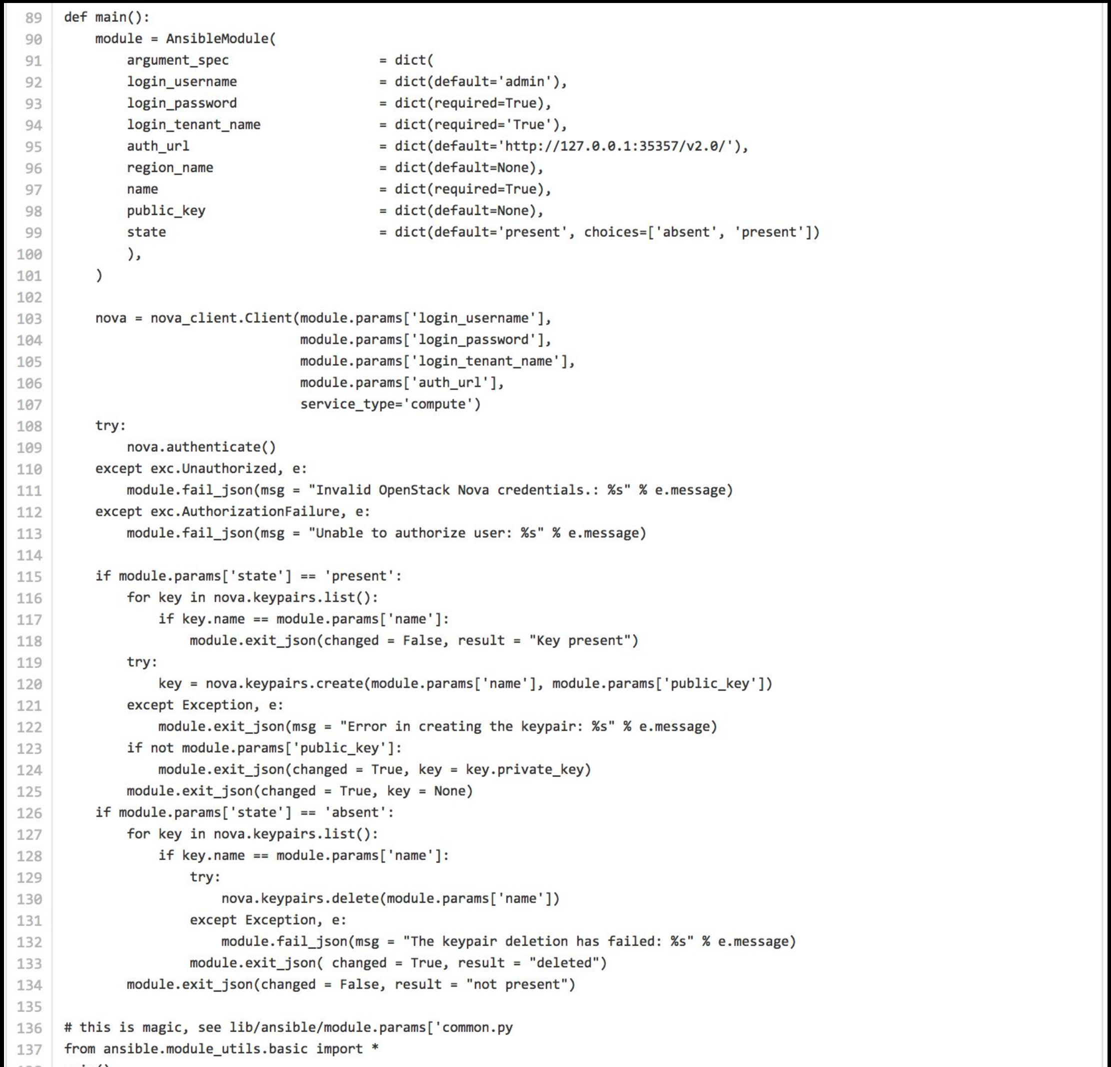
1. Simplicity in mind
2. Modules



```yaml
---
- name: provision test instances
  hosts: local
  connection: local
  vars_files:
  - ../vars/main.yml
  tasks:
    - include: keypair.yml
    - name: create the security group and rules
      nova_group:
        name: "{{ testenv_security_groups }}"
        description: "{{ testenv_security_groups_description }}"
        rules:
          - ip_protocol: "{{ item.proto }}"
            from_port: "{{ item.port }}"
            to_port: "{{ item.port }}"
            cidr: 0.0.0.0/0
            state: "{{ item.state }}"
      with_items: testenv_security_group_rules
      register: testenv_security_group
    - command: neutron security-group-rule-create --ethertype={{ item }} --remote-group-id={{ testenv_security_group.results[0].group
      register: security_group_rule_create_result
      failed_when: "security_group_rule_create_result.rc != 0 and 'Security group rule already exists.' not in security_group_rule_cr
      changed_when: security_group_rule_create_result.rc == 0
      with_items:
      - IPv4
      - IPv6
    - name: create {{ item }}
      nova_compute:
        name: "{{ item }}"
        image_id: "{{ testenv_image_id }}"
        key_name: "{{ testenv_keypair_name }}"
        security_groups: "{{ testenv_security_group.results[0].group_id }}"
        wait_for: 200
        flavor_id: 3
        nics:
          - net-id: "{{ testenv_net_id }}"
      with_items: testenv_instance_names
    - name: associate a floating IP to {{ item }}
      nova_fip: server={{ item }}
      with_items: testenv_instance_names
      register: testenv_floating_ips
    - name: wait for {{ item }} to boot
      wait_for: port=22 delay=5 timeout=300 host={{ item.floating_ip }}
      with_items: testenv_floating_ips.results
```

Overview

1. Simplicity in mind
2. Modules



```
def main():
    module = AnsibleModule(
        argument_spec                   = dict(
        login_username                  = dict(default='admin'),
        login_password                  = dict(required=True),
        login_tenant_name               = dict(required='True'),
        auth_url                        = dict(default='http://127.0.0.1:35357/v2.0/'),
        region_name                     = dict(default=None),
        name                            = dict(required=True),
        public_key                      = dict(default=None),
        state                           = dict(default='present', choices=['absent', 'present'])
        ),
    )

    nova = nova_client.Client(module.params['login_username'],
                              module.params['login_password'],
                              module.params['login_tenant_name'],
                              module.params['auth_url'],
                              service_type='compute')
    try:
        nova.authenticate()
    except exc.Unauthorized, e:
        module.fail_json(msg = "Invalid OpenStack Nova credentials.: %s" % e.message)
    except exc.AuthorizationFailure, e:
        module.fail_json(msg = "Unable to authorize user: %s" % e.message)

    if module.params['state'] == 'present':
        for key in nova.keypairs.list():
            if key.name == module.params['name']:
                module.exit_json(changed = False, result = "Key present")
        try:
            key = nova.keypairs.create(module.params['name'], module.params['public_key'])
        except Exception, e:
            module.exit_json(msg = "Error in creating the keypair: %s" % e.message)
        if not module.params['public_key']:
            module.exit_json(changed = True, key = key.private_key)
        module.exit_json(changed = True, key = None)
    if module.params['state'] == 'absent':
        for key in nova.keypairs.list():
            if key.name == module.params['name']:
                try:
                    nova.keypairs.delete(module.params['name'])
                except Exception, e:
                    module.fail_json(msg = "The keypair deletion has failed: %s" % e.message)
                module.exit_json( changed = True, result = "deleted")
        module.exit_json(changed = False, result = "not present")

# this is magic, see lib/ansible/module.params['common.py
from ansible.module_utils.basic import *
```

Overview

1. Simplicity in mind
2. Modules
3. Inventory

Hosts

```
[identity]
identity-1.example.com
identity-2.example.com
```

Group

Overview

1. Simplicity in mind
2. Modules
3. Inventory

Variables

```
---
identity:
  port: 5000
  admin_port: 35357
  auth_strategy: uuid
  ...
```

Overview

1. Simplicity in mind
2. Modules
3. Inventory
4. Playbooks

```
---
- hosts: identity
  roles:
    - base
    - memcached
    - apache2
    - { role: keystone,
identity.auth_strategy: pki }
```

Group Targeting

Overview

1. Simplicity in mind
2. Modules
3. Inventory
4. Playbooks
5. Roles

- Roles are tasks
- Roles are to be included in playbooks
- Roles are shareable
  https://galaxy.ansible.com

Overview

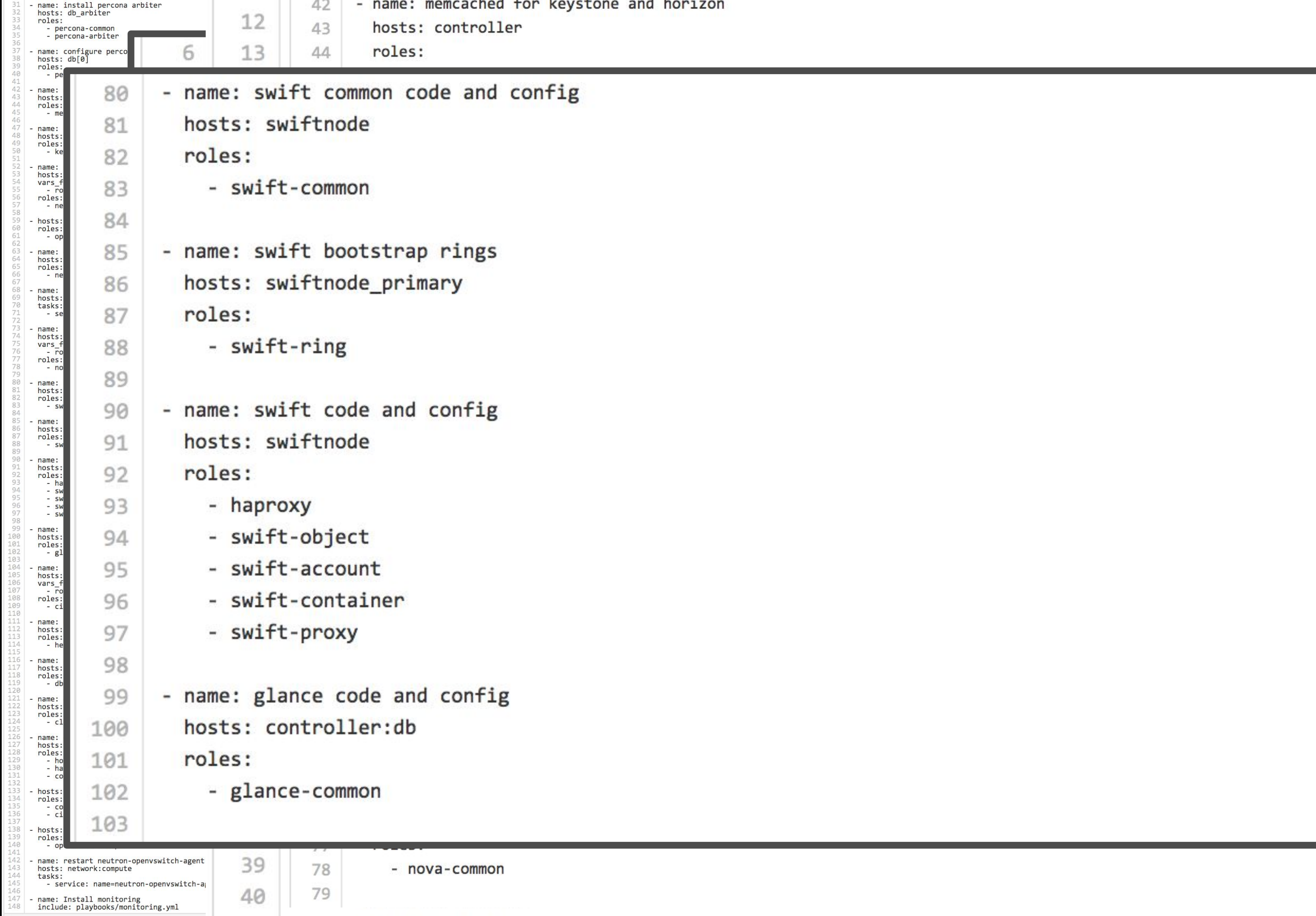
1. Simplicity in mind
2. Modules
3. Inventory
4. Playbooks
5. Roles

```
roles/
  identity/
    tasks/
    handlers/
    files/
    templates/
    defaults/
    vars/
    meta/
```

Overview

1. Simplicity in mind
2. Modules
3. Inventory
4. Playbooks
5. Roles
6. You already have everything you need

Orchestration



```
- name: install percona arbiter
  hosts: db_arbiter
  roles:
    - percona-common
    - percona-arbiter

- name: configure perco
  hosts: db[0]
  roles:
```

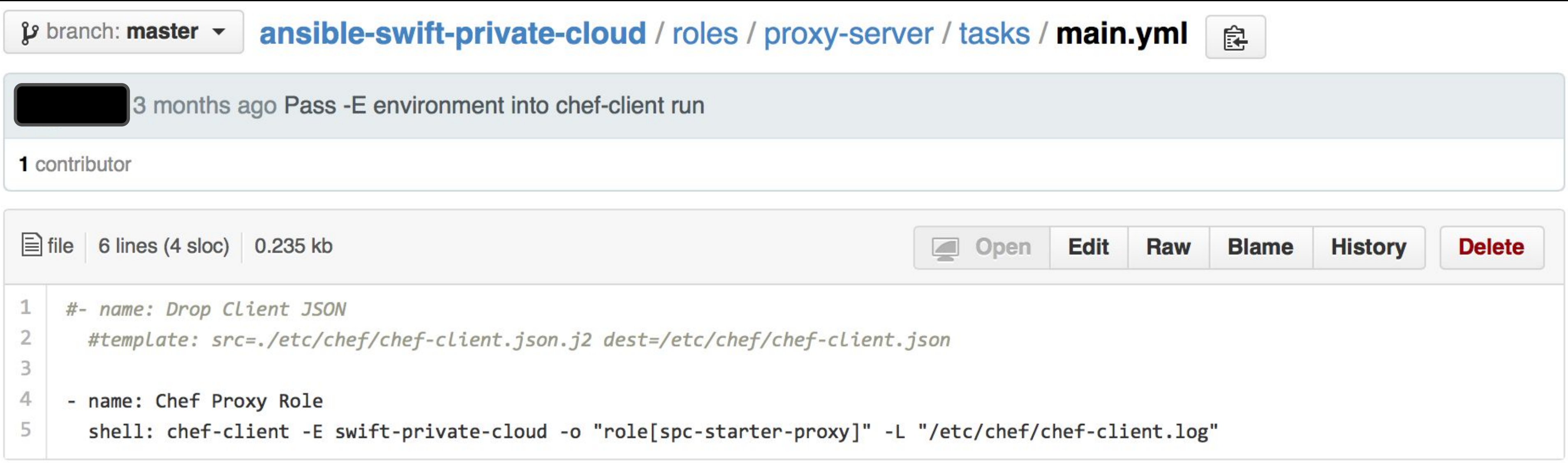
```
- name: memcached for keystone and horizon
  hosts: controller
  roles:
```

```
- name: swift common code and config
  hosts: swiftnode
  roles:
    - swift-common

- name: swift bootstrap rings
  hosts: swiftnode_primary
  roles:
    - swift-ring

- name: swift code and config
  hosts: swiftnode
  roles:
    - haproxy
    - swift-object
    - swift-account
    - swift-container
    - swift-proxy

- name: glance code and config
  hosts: controller:db
  roles:
    - glance-common
```

```
    - nova-common
```

```
- name: restart neutron-openvswitch-agent
  hosts: network:compute
  tasks:
    - service: name=neutron-openvswitch-a

- name: Install monitoring
  include: playbooks/monitoring.yml
```

Orchestration



branch: **master**

ansible-swift-private-cloud / roles / proxy-server / tasks / **main.yml**

[redacted] 3 months ago Pass -E environment into chef-client run

1 contributor

file | 6 lines (4 sloc) | 0.235 kb

```
#- name: Drop Client JSON
  #template: src=./etc/chef/chef-client.json.j2 dest=/etc/chef/chef-client.json

- name: Chef Proxy Role
  shell: chef-client -E swift-private-cloud -o "role[spc-starter-proxy]" -L "/etc/chef/chef-client.log"
```

Why?

```
@roles('mgmt')
def _create_default_network():
    with cd('/root'):
        run('. openrc && neutron net-create flat1004 --provider:network_type flat
--provider:physical_network physnet1')
```

- Not idempotent
- Everyone is rolling their own

When

```
---
- quantum_network: >
  name=flat1004 state=present
  provider_network_type=flat
  provider_physical_network=physnet1'
```

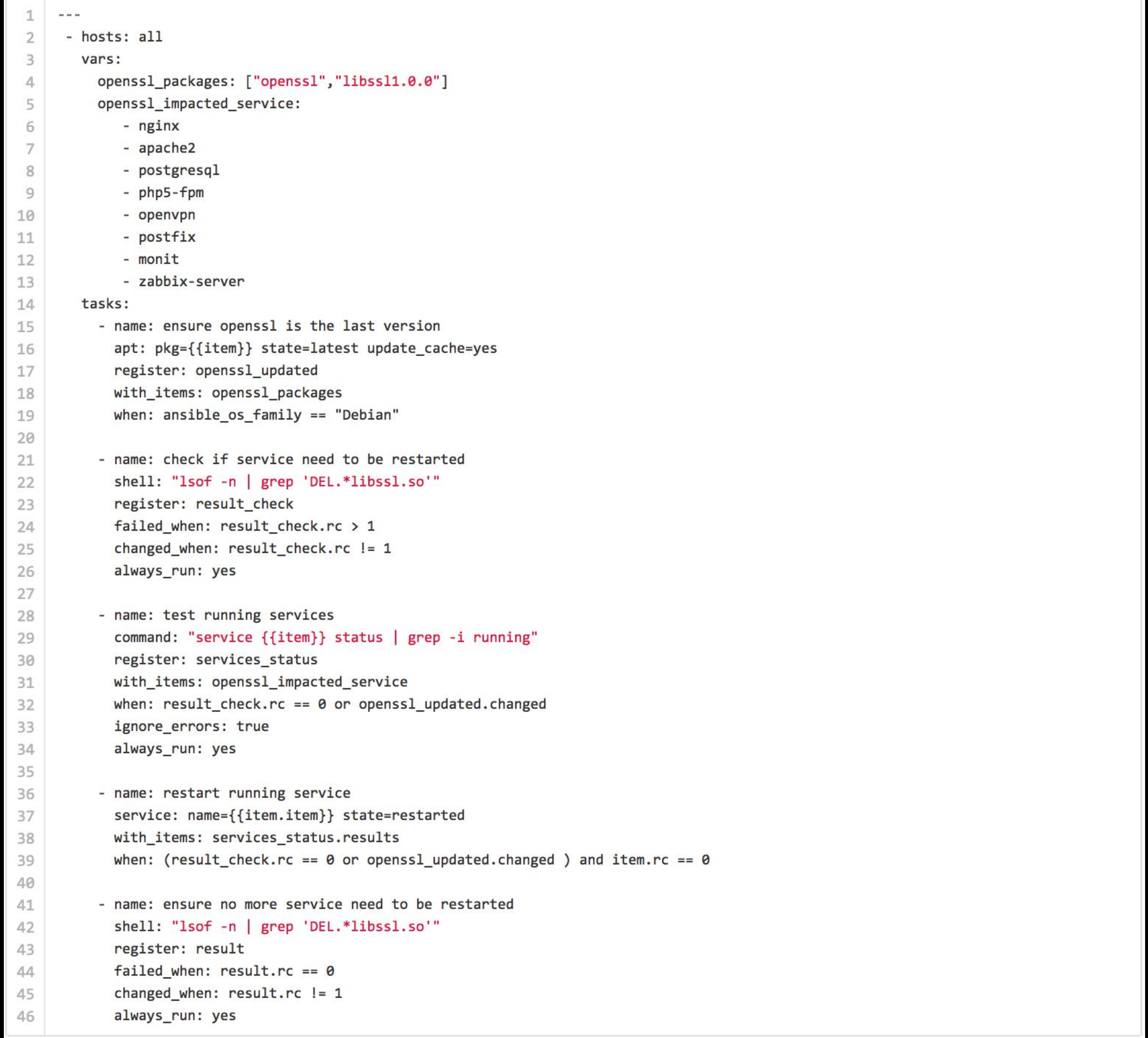
One Time Tasks

```
---
- name: cleanup expired keystone database tokens
  shell: mysql -e "delete from keystone.token where expires < date_sub(now(), interval 24 hours);"
```

One Time Tasks

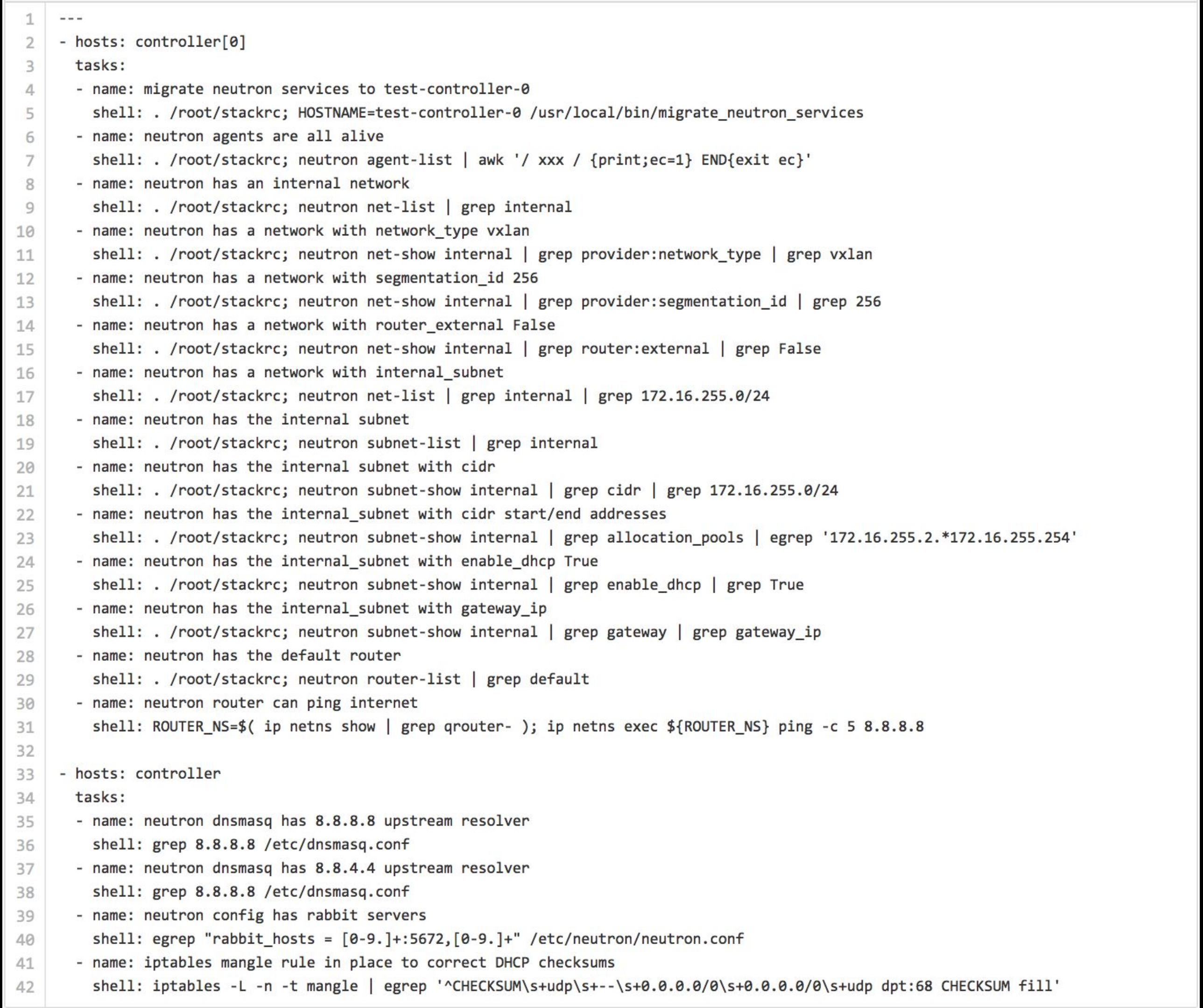
```
---
- name: is m1.tiny undersized?
  shell: mysql -e "select root_gb from nova.instance_types where name='m1.tiny';" | grep 10
  ignore_errors: True
  changed_when: False
  register: resize_tiny_flavor

- name: bump root disk size on m1.tiny
  shell: mysql -e "update nova.instance_types set root_gb=10 where name='m1.tiny';"
  when: resize_tiny_flavor.rc != 0
```

Operations



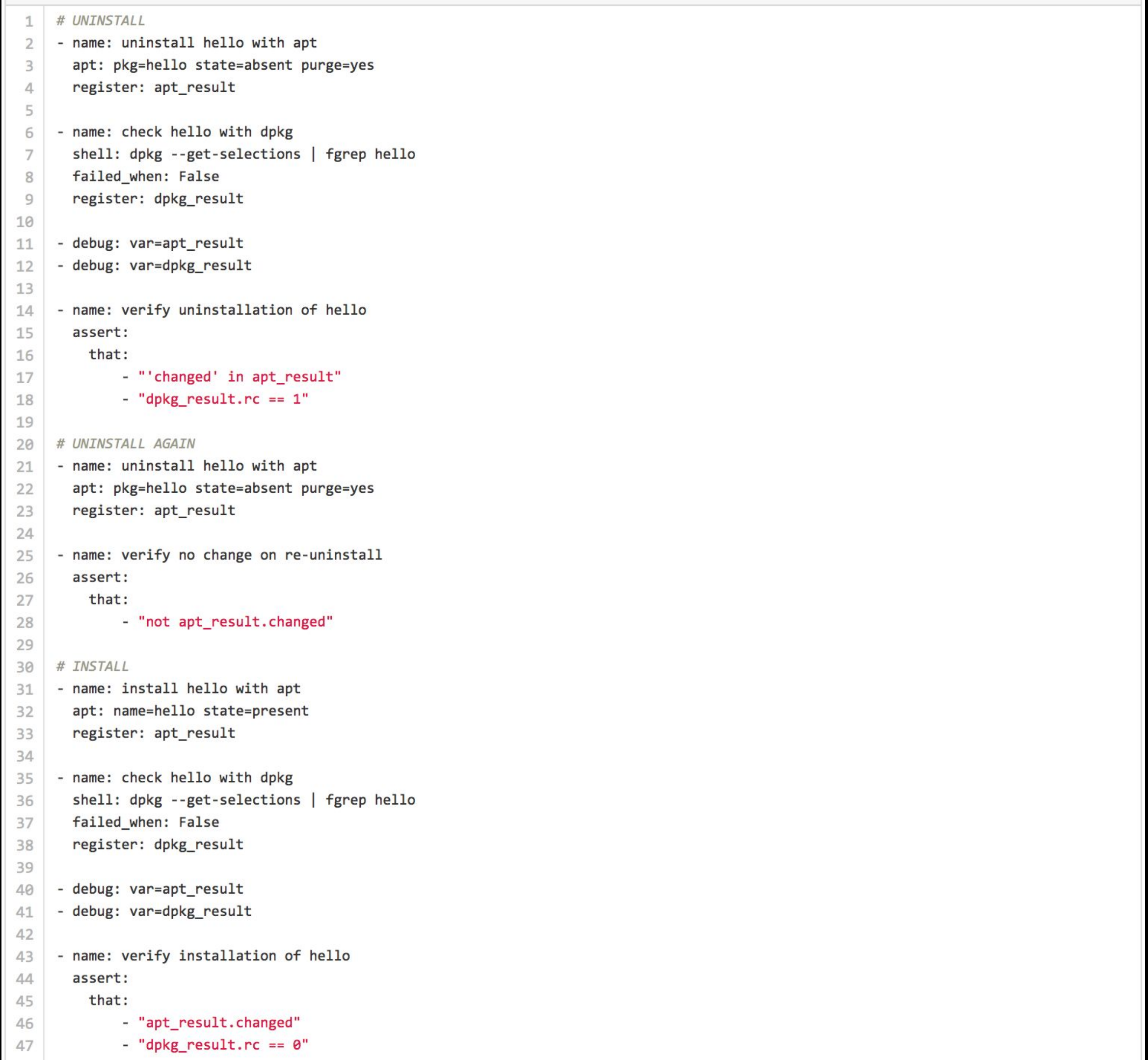
```yaml
---
 - hosts: all
   vars:
     openssl_packages: ["openssl","libssl1.0.0"]
     openssl_impacted_service:
         - nginx
         - apache2
         - postgresql
         - php5-fpm
         - openvpn
         - postfix
         - monit
         - zabbix-server
   tasks:
     - name: ensure openssl is the last version
       apt: pkg={{item}} state=latest update_cache=yes
       register: openssl_updated
       with_items: openssl_packages
       when: ansible_os_family == "Debian"

     - name: check if service need to be restarted
       shell: "lsof -n | grep 'DEL.*libssl.so'"
       register: result_check
       failed_when: result_check.rc > 1
       changed_when: result_check.rc != 1
       always_run: yes

     - name: test running services
       command: "service {{item}} status | grep -i running"
       register: services_status
       with_items: openssl_impacted_service
       when: result_check.rc == 0 or openssl_updated.changed
       ignore_errors: true
       always_run: yes

     - name: restart running service
       service: name={{item.item}} state=restarted
       with_items: services_status.results
       when: (result_check.rc == 0 or openssl_updated.changed ) and item.rc == 0

     - name: ensure no more service need to be restarted
       shell: "lsof -n | grep 'DEL.*libssl.so'"
       register: result
       failed_when: result.rc == 0
       changed_when: result.rc != 1
       always_run: yes
```

Integration Testing



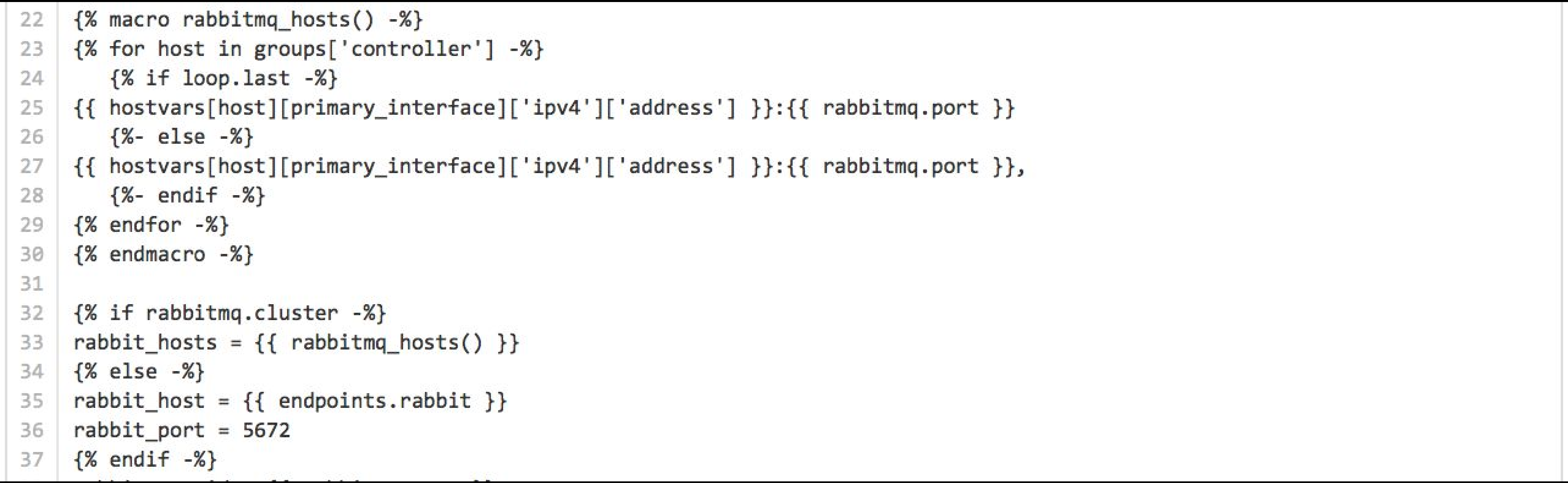
```
---
- hosts: controller[0]
  tasks:
  - name: migrate neutron services to test-controller-0
    shell: . /root/stackrc; HOSTNAME=test-controller-0 /usr/local/bin/migrate_neutron_services
  - name: neutron agents are all alive
    shell: . /root/stackrc; neutron agent-list | awk '/ xxx / {print;ec=1} END{exit ec}'
  - name: neutron has an internal network
    shell: . /root/stackrc; neutron net-list | grep internal
  - name: neutron has a network with network_type vxlan
    shell: . /root/stackrc; neutron net-show internal | grep provider:network_type | grep vxlan
  - name: neutron has a network with segmentation_id 256
    shell: . /root/stackrc; neutron net-show internal | grep provider:segmentation_id | grep 256
  - name: neutron has a network with router_external False
    shell: . /root/stackrc; neutron net-show internal | grep router:external | grep False
  - name: neutron has a network with internal_subnet
    shell: . /root/stackrc; neutron net-list | grep internal | grep 172.16.255.0/24
  - name: neutron has the internal subnet
    shell: . /root/stackrc; neutron subnet-list | grep internal
  - name: neutron has the internal subnet with cidr
    shell: . /root/stackrc; neutron subnet-show internal | grep cidr | grep 172.16.255.0/24
  - name: neutron has the internal_subnet with cidr start/end addresses
    shell: . /root/stackrc; neutron subnet-show internal | grep allocation_pools | egrep '172.16.255.2.*172.16.255.254'
  - name: neutron has the internal_subnet with enable_dhcp True
    shell: . /root/stackrc; neutron subnet-show internal | grep enable_dhcp | grep True
  - name: neutron has the internal_subnet with gateway_ip
    shell: . /root/stackrc; neutron subnet-show internal | grep gateway | grep gateway_ip
  - name: neutron has the default router
    shell: . /root/stackrc; neutron router-list | grep default
  - name: neutron router can ping internet
    shell: ROUTER_NS=$( ip netns show | grep qrouter- ); ip netns exec ${ROUTER_NS} ping -c 5 8.8.8.8

- hosts: controller
  tasks:
  - name: neutron dnsmasq has 8.8.8.8 upstream resolver
    shell: grep 8.8.8.8 /etc/dnsmasq.conf
  - name: neutron dnsmasq has 8.8.4.4 upstream resolver
    shell: grep 8.8.8.8 /etc/dnsmasq.conf
  - name: neutron config has rabbit servers
    shell: egrep "rabbit_hosts = [0-9.]+:5672,[0-9.]+" /etc/neutron/neutron.conf
  - name: iptables mangle rule in place to correct DHCP checksums
    shell: iptables -L -n -t mangle | egrep '^CHECKSUM\s+udp\s+--\s+0.0.0.0/0\s+0.0.0.0/0\s+udp dpt:68 CHECKSUM fill'
```

Integration Testing



```yaml
# UNINSTALL
- name: uninstall hello with apt
  apt: pkg=hello state=absent purge=yes
  register: apt_result

- name: check hello with dpkg
  shell: dpkg --get-selections | fgrep hello
  failed_when: False
  register: dpkg_result

- debug: var=apt_result
- debug: var=dpkg_result

- name: verify uninstallation of hello
  assert:
    that:
        - "'changed' in apt_result"
        - "dpkg_result.rc == 1"

# UNINSTALL AGAIN
- name: uninstall hello with apt
  apt: pkg=hello state=absent purge=yes
  register: apt_result

- name: verify no change on re-uninstall
  assert:
    that:
        - "not apt_result.changed"

# INSTALL
- name: install hello with apt
  apt: name=hello state=present
  register: apt_result

- name: check hello with dpkg
  shell: dpkg --get-selections | fgrep hello
  failed_when: False
  register: dpkg_result

- debug: var=apt_result
- debug: var=dpkg_result

- name: verify installation of hello
  assert:
    that:
        - "apt_result.changed"
        - "dpkg_result.rc == 0"
```

Common patterns



```
{% macro rabbitmq_hosts() -%}
{% for host in groups['controller'] -%}
   {% if loop.last -%}
{{ hostvars[host][primary_interface]['ipv4']['address'] }}:{{ rabbitmq.port }}
   {%- else -%}
{{ hostvars[host][primary_interface]['ipv4']['address'] }}:{{ rabbitmq.port }},
   {%- endif -%}
{% endfor -%}
{% endmacro -%}

{% if rabbitmq.cluster -%}
rabbit_hosts = {{ rabbitmq_hosts() }}
{% else -%}
rabbit_host = {{ endpoints.rabbit }}
rabbit_port = 5672
{% endif -%}
```

```
$ grep -R 'macro rabbitmq_hosts'  . | wc -l
       5
```

Variables can be tricky

```
$ cat memcached/defaults/main.yml
memcached:
  port 11211


# from site.yml
- name: openstack horizon service
  hosts: controller
  roles:
    - memcached # must be added to roles
    - horizon   # for horizon to reference {{ memcached.port }}
```

Variables can be tricky

```
$ cat group_vars/all.yml
rabbitmq:
  cluster: true

# from site.yml
- name: nova code and config
  hosts: controller
  vars_files:
    # don’t do this -- will reset the override from inventory
    - roles/rabbitmq/defaults/main.yml
  roles:
    - nova-common
```

Variables can be tricky

We ended up defining everything in inventory, and managing with a custom vars plugin.

```
$ grep vars_plugin ansible.cfg
vars_plugins = plugins/vars
```

DEVSTACK

References

- Ansible: https://github.com/ansible/ansible
- Ansible Docs: http://docs.ansible.com
- Ansible Swift: https://github.com/rcbops/ansible-swift-private-cloud
- Heartbleed: https://github.com/jdauphant/patch-openssl-CVE-2014-0160
- Ursula: https://github.com/blueboxgroup/ursula/

Success.