Infrastructure Definition Tools (OpenStack Heat) – Possible solutions

1. How does a Heat resource retrieve/make use of a value passed as a parameter tothe template? How does a Heat resource reference another Heat resource?

`{ getparam: PARAMETERNAME }` and `{ getresource: RESOURCENAME }`

2. Describe the main requirements for an Infrastructure Definition Tool.

Scriptable interface so we can automate tasks, Unattended mode meaning should not have to have user interaction and output should be parsable. Should support script characteristics: Idempotent, Pre- and Post-checks, Visible failure and Parameterized. And of course externalized configuration, meaning when you write your script it is easy to treat it as its own file which can be put in git, and not just as a file that is locked into the tool itself.

3. What is a “stack”? What properties/characteristics has a stack?

A collection of infrastructure elements that are defined as one unit. It is a stack if it can be created and destroyed as one unit.

4. What is a good rule-of-thumb (guiding principle) when splitting up your infrastructure into multiple stacks?

Scope of change. What changes will be made most often, who will be impacted, keep those elements together.

5. After you have completed the lab tutorial, modify the template in sucha way that the two servers can be different (e.g. an Ubuntu and a Windows instance). Also allow for a list of security groups to be added as parameters to each server. Delete the Heat stack from the previous exercise and launch it again but this time with an Ubuntu and a Windows instance (with the linux and windows security groups, respectively).

POSSIBLE SOLUTION:

```
# delete the demo stack:
openstack stack delete heat_demo

# modified two_servers_in_new_neutron_net.yaml:

heat_template_version: 2013-05-23

description: >

HOT template to create a new neutron network plus a router to the publicnetwork, and for
deploying two servers into the new network. The template alsoassigns floating IP
addresses to each server so they are routable from thepublic network.

parameters:
  key_name:
    type: string
    description: Name of keypair to assign to servers
  name_lin:
    type: string
    description: Name of linux server
  image_lin:
    type: string
    description: Name of image to use for servers
  flavor_lin:
    type: string
    description: Flavor to use for servers
  name_win:type: string
    description: Name of windows server
  image_win:
    type: string
    description: Name of image to use for servers
```

```
  flavor_win:
    type: string
    description: Flavor to use for servers
  public_net:
    type: string
    description: >
    ID or name of public network for which floating IP addresses will be allocated
  private_net_name:
    type: string
    description: Name of private network to be created
  private_net_cidr:
    type: string
    description: Private network address (CIDR notation)
  private_net_gateway:
    type: string
    description: Private network gateway address
  private_net_pool_start:
    type: string
    description: Start of private network IP address allocation pool


  private_net_pool_end:
    type: string
    description: End of private network IP address allocation pool
  sec_group_lin:
    type: comma_delimited_list
    description: Security groups
  sec_group_win:
    type: comma_delimited_list
    description: Security groups
  resources:
    private_net:
    type: OS::Neutron::Net
      properties:
        name: { get_param: private_net_name }
  private_subnet:
    type: OS::Neutron::Subnet
      properties:
        network_id: { get_resource: private_net }
        cidr: { get_param: private_net_cidr }
        gateway_ip: { get_param: private_net_gateway }
        allocation_pools:
          - start: { get_param: private_net_pool_start }
          end: { get_param: private_net_pool_end }
    router:type: OS::Neutron::Router
      properties:
        external_gateway_info:
          network: { get_param: public_net }
  router_interface:
    type: OS::Neutron::RouterInterface
      properties:
        router_id: { get_resource: router }
        subnet_id: { get_resource: private_subnet }
  server1:type: OS::Nova::Server
    properties:
      name: { get_param: name_lin }
      image: { get_param: image_lin }
      flavor: { get_param: flavor_lin }
      key_name: { get_param: key_name }
      networks:
        - port: { get_resource: server1_port }
```

```
server1_port:
    type: OS::Neutron::Port
    properties:
      network_id: { get_resource: private_net }
      security_groups: { get_param: sec_group_lin }
      fixed_ips:
        - subnet_id: { get_resource: private_subnet }
  server1_floating_ip:
    type: OS::Neutron::FloatingIP
    properties:floating_network: { get_param: public_net }
    port_id: { get_resource: server1_port }
  server2:
    type: OS::Nova::Server
    properties:
    name: { get_param: name_win }
    image: { get_param: image_win }
    flavor: { get_param: flavor_win }
    key_name: { get_param: key_name }
    networks:
      - port: { get_resource: server2_port }
  server2_port:
    type: OS::Neutron::Port
    properties:
      network_id: { get_resource: private_net }
      security_groups: { get_param: sec_group_win }
      fixed_ips:
        - subnet_id: { get_resource: private_subnet }
  server2_floating_ip:
    type: OS::Neutron::FloatingIP
    properties:floating_network: { get_param: public_net }
    port_id: { get_resource: server2_port }
outputs:
  server1_private_ip:
    description: IP address of server1 in private network
    value: { get_attr: [ server1, first_address ] }
  server1_public_ip:
    description: Floating IP address of server1 in public network
    value: { get_attr: [ server1_floating_ip, floating_ip_address ] }
  server2_private_ip:
    description: IP address of server2 in private network
    value: { get_attr: [ server2, first_address ] }
  server2_public_ip:
    description: Floating IP address of server2 in public network
    value: { get_attr: [ server2_floating_ip, floating_ip_address ] }
```

```
# create with:

openstack stack create -e heat_demo_mod_env.yaml -t \
two_servers_in_new_neutron_net.yaml heat_demo_mod

# with environment file heat_demo_mod_env.yaml:

parameters:
  key_name: KEY_NAME
  name_lin: ubuntu
  image_lin: Ubuntu Server 18.04 LTS (Bionic Beaver) amd64
  flavor_lin: m1.small
  name_win: windows
  image_win: Windows Server 2016 Std Eval
  flavor_win: m1.medium
  public_net: ntnu-internal
  private_net_name: net1
  private_net_cidr: 192.168.1XY.0/24
  private_net_gateway: 192.168.1XY.1
  private_net_pool_start: 192.168.1XY.200
  private_net_pool_end: 192.168.1XY.250
  sec_group_lin:
    - default
    - linux
  sec_group_win: default,windows
```