OpenStack Heat Template Composition

Paul Breaux, 2016-27-06

Heat Orchestration Templates (HOT) for OpenStack's Heat service can quickly grow in length as users need to pile in ever more resources to define their applications. A simple Nova instance requires a small volume at first. Soon it needs a private network, a public network, a software configuration deployment, a load balancer, a deluxe karaoke machine. This means the HOT files get bloated and difficult to read for mere humans. Let's think about why that is.... A template defines the set of resources necessary and sufficient to describe an application. It also describes the dependencies that exist between the resources, if any. That way, Heat can manage the life-cycle of the application without you having to worry about it. Often, the resources in a long template need to be visually grouped together to alert the reader that these things depend on one another and share some common goal (e.g. create a network, subnet, and router). When templates get to this state, we need to start thinking about composition.

Composition in HOT is done in a couple of ways. We will tackle the straight-forward approach first. Much of this material is presented in the official documentation for template composition.

The Parent and Child Templates

We'll often refer to a parent template and child template here. The parent template is the user's view into the Heat stack that defines our application. It's the entrypoint. A parent template contains all of the over-arching components of the application. The child on the other hand may describe a logically grouped chunk of that application. For example, one child template creates the Nova instances that launch BIG-IP devices while another child template creates all the network plumbing for those instances.

Let's look at an example of this. Below is the parent template:

heat_template_test.yaml

```
heat_template_version: 2015-04-30

description: Setup infrastructure for testing F5 Heat Templates.

parameters:
  ve_cluster_ready_image:
    type: string
    constraints:
      - custom_constraint: glance.image
  ve_standalone_image:
    type: string
    constraints:
      - custom_constraint: glance.image
  ve_flavor:
    type: string
    constraints:
      - custom_constraint: nova.flavor
  use_config_drive:
    type: boolean
    default: false
  ssh_key:
    type: string
    constraints:
      - custom_constraint: nova.keypair
```

```
  sec_group_name:
    type: string
  admin_password:
    type: string
    label: F5 VE Admin User Password
    description: Password used to perform image import services
  root_password:
    type: string
  license:
    type: string
  external_network:
    type: string
    constraints:
      - custom_constraint: neutron.network

resources:
  # Plumb the newtorking for the BIG-IP instances
  networking:
    type: heat_template_test_networks.yaml
    properties:
      external_network: { get_param: external_network }
      sec_group_name: { get_param: sec_group_name }
  # Wait for networking to come up and then launch two clusterable BIG-IPs
  two_bigip_devices:
    type: OS::Heat::ResourceGroup
    depends_on: networking
    properties:
      count: 2
      resource_def:
        # Reference a child template in the same directly where the heat_template_test.yaml is locate
        type: cluster_ready_ve_4_nic.yaml
        properties:
          ve_image: { get_param: ve_cluster_ready_image }
          ve_flavor: { get_param: ve_flavor }
          ssh_key: { get_param: ssh_key }
          use_config_drive: { get_param: use_config_drive }
          open_sec_group: { get_param: sec_group_name }
          admin_password: { get_param: admin_password }
          root_password: { get_param: root_password }
          license: { get_param: license }
          external_network: { get_param: external_network }
          mgmt_network: { get_attr: [networking, mgmt_network_name] }
          ha_network: { get_attr: [networking, ha_network_name] }
          network_1: { get_attr: [networking, client_data_network_name] }
          network_2: { get_attr: [networking, server_data_network_name] }
  # Wait for networking to come up and launch a standalone BIG-IP
  standalone_device:
    # Reference another child template in the local directory
    type: f5_ve_standalone_3_nic.yaml
    depends_on: networking
    properties:
      ve_image: { get_param: ve_standalone_image }
      ve_flavor: { get_param: ve_flavor }
      ssh_key: { get_param: ssh_key }
      use_config_drive: { get_param: use_config_drive }
      open_sec_group: { get_param: sec_group_name }
      admin_password: { get_param: admin_password }
      root_password: { get_param: root_password }
```

```
      license: { get_param: license }
      external_network: { get_param: external_network }
      mgmt_network: { get_attr: [networking, mgmt_network_name] }
      network_1: { get_attr: [networking, client_data_network_name] }
      network_2: { get_attr: [networking, server_data_network_name] }
```

Now that's still fairly verbose, but its doing some heavy lifting for us. It is creating almost all of the networking needed to launch a set of clusterable BIG-IP devices and a single standalone BIG-IP device. It takes in parameters from the user such as *ve_standalone_image* and *external_network* and passes them along to the child that requires them. The child stack then receives those parameters in the same way the template above does, by defining a parameter.

The parent template references the *heat_template_test_networks.yaml* template directly, expecting the file to be in the same local directory where the parent template is located. This is always created in the type field of a resource. In addition to relative paths, you can also reference another template with an absolute path, or URL.

You can also see the group of responsibilities here. One child stack (heat_template_test_networks.yaml) is building networks, another (cluster_ready_ve_4_nic.yaml) is launching a set of BIG-IP devices ready for clustering and yet another (f5_ve_standalone_4_nic.yaml) is launching a standalone BIG-IP device. Yet the dependencies are apparent in the depends_on property and the intrinsic functions called within that resource (more on that later). You will not successfully launch the standalone device without having the networking in place first, thus the *standalone_device* resource is dependent upon the networking resource. This means we can easily send data into the networking stack (as a parameter) and now we must get data out to be passed into the *standalone_device* stack. Let's look at the networking template and see what it defines as its outputs.

heat_template_test_networks.yaml

```
heat_template_version: 2015-04-30

description: >
    Create the four networks needed for the heat plugin tests along with their subnets and connect th

parameters:
  external_network:
    type: string
  sec_group_name:
    type: string

resources:
  # Four networks
  client_data_network:
    type: OS::Neutron::Net
    properties:
      name: client_data_network
  server_data_network:
    type: OS::Neutron::Net
    properties:
      name: server_data_network
  mgmt_network:
    type: OS::Neutron::Net
    properties:
      name: mgmt_network
  ha_network:
    type: OS::Neutron::Net
    properties:
      name: ha_network
```

```
# And four accompanying subnets
client_data_subnet:
  type: OS::Neutron::Subnet
  properties:
    cidr: 10.1.1.0/24
    dns_nameservers: [10.190.0.20]
    network: { get_resource: client_data_network }
server_data_subnet:
  type: OS::Neutron::Subnet
  properties:
    cidr: 10.1.2.0/24
    dns_nameservers: [10.190.0.20]
    network: {get_resource: server_data_network }
mgmt_subnet:
  type: OS::Neutron::Subnet
  properties:
    cidr: 10.1.3.0/24
    dns_nameservers: [10.190.0.20]
    network: { get_resource: mgmt_network }
ha_subnet:
  type: OS::Neutron::Subnet
  properties:
    cidr: 10.1.4.0/24
    dns_nameservers: [10.190.0.20]
    network: { get_resource: ha_network }
# Create router for testlab
testlab_router:
  type: OS::Neutron::Router
  properties:
    external_gateway_info:
      network: { get_param: external_network }
# Connect networks to router interface
client_data_router_interface:
  type: OS::Neutron::RouterInterface
  properties:
    router: { get_resource: testlab_router }
    subnet: { get_resource: client_data_subnet }
server_data_router_interface:
  type: OS::Neutron::RouterInterface
  properties:
    router: { get_resource: testlab_router }
    subnet: { get_resource: server_data_subnet }
mgmt_router_interface:
  type: OS::Neutron::RouterInterface
  properties:
    router: { get_resource: testlab_router }
    subnet: { get_resource: mgmt_subnet }
ha_router_interface:
  type: OS::Neutron::RouterInterface
  properties:
    router: { get_resource: testlab_router }
    subnet: { get_resource: ha_subnet }
open_sec_group:
  type: OS::Neutron::SecurityGroup
  properties:
    name: { get_param: sec_group_name }
    rules:
      - protocol: icmp
```

```
          protocol: icmp
          direction: ingress
        - protocol: icmp
          direction: egress
        - protocol: udp
          direction: ingress
        - protocol: udp
          direction: egress
        - protocol: tcp
          direction: ingress
          port_range_min: 1
          port_range_max: 65535
        - protocol: tcp
          direction: egress
          port_range_min: 1
          port_range_max: 65535

outputs:
  mgmt_network_name:
    value: { get_attr: [mgmt_network, name] }
  ha_network_name:
    value: { get_attr: [ha_network, name] }
  client_data_network_name:
    value: { get_attr: [client_data_network, name] }
  server_data_network_name:
    value: { get_attr: [server_data_network, name] }
```

You can see the logical grouping here, and this is where template composition shines. This simple template creates a security group, four networks, four subnets, and ties them together with a router. Even though the *heat_template_test.yaml* parent template uses this to build its networks for defining its application, another user may decide they want the same networking infrastucture, but they want four standalone devices and eight pairs of clusterable devices. Their only modification would be in the parent template, because the *heat_template_test_networks.yaml* template describes the set of networks those devices need to connect to. It is important to note that the above template is a fully functioning HOT template all by itself. You can launch it and it will build those four networks.

So how does the data get out of the networking template? The outputs section takes care of that. For attaching BIG-IP devices in the parent template to these networks, all we require is the network name, so the networking template kicks those back up to whomever is curious about such things. We saw the *get_param* function earlier, and now we can see the use of the *get_attr* function. In the *two_bigip_devices* resource, the parent template references the *networking* resource directly and then it accesses the *client_data_network_name* attribute (as seen below). This operation retrieves the network name for the *client_data_network* and passes it into the *cluster_ready_ve_4_nic.yaml* stack.

```
network_1: { get_attr: [networking, client_data_network_name] }
```

With that, we've successfully sent information into a child stack, retrieved it, then sent it into another child stack. This is very useful when working in large groups of users because my *heat_template_test_networks.yaml* template may benefit others. In time, you can build quite a collection of these concise HOT templates then use a parent template to orchestrate them in many complex ways. Keep in mind however, that HOT is declarative, meaning there are no looping constructs or if/else decisions to decide whether to create seven networks or four. For that, you would need to create two separate templates. As seen in the *OS::Heat::ResourceGroup* resource for *two_bigip_devices* however, we can toggle the number of instances launched by that resource at stack creation time. We can simply make the *count* property of that resource a parameter and the user can decide how many clusterable BIG-IP devices should be launched.

To launch the above template with the python-heatclient:

```
```

```
heat stack-create -f heat_template_test.yaml -P "ve_cluster_ready_image=cluster_ready-BIGIP-11.6.0.0.
```

New Resource Types in Environment Files

The second way to do template composition is to define a brand new resource type in an environment file. The environment in which a Heat stack is launched affects the behavior of that stack. To learn more about environment files, check out the official documentation. We will use them here to define a new resource type. The cool thing about environments is that the child templates looks exactly the same, and only one small change is needed in the parent template. It is the environment in which those stacks launch that changes. Below we are defining a Heat environment file and defining a new resource type in the *attribute_registry* section.

heat_template_test_environment.yaml

```
parameters:
  ve_cluster_ready_image: cluster_ready-BIG-IP-11.6.0.0.0.401.qcow2
  ve_standalone_image: standalone-BIG-IP-11.6.0.0.0.401.qcow2
  ve_flavor: m1.small
  ssh_key: my_key
  sec_group_name: bigip_sec_group
  admin_password: admin_password
  root_password: root_password
  license: XXXXX
  external_network: public_net

resource_registry:
  OS::Neutron::FourNetworks: heat_template_test_networks.yaml
  F5::BIGIP::ClusterableDevice: cluster_ready_ve_4_nic.yaml
  F5::BIGIP::StandaloneDevice: f5_ve_standalone_3_nic.yaml
```

The parent template will now reference the new resource types, which are merely mappings to the child templates. This means the parent template has three new resources to use which are not a part of the standard OpenStack Resource Types. The environment makes all this possible.

heat_template_test_with_environment.yaml

```
heat_template_version: 2015-04-30

description: Setup infrastructure for testing F5 Heat Templates.

parameters:
  ve_cluster_ready_image:
    type: string
    constraints:
      - custom_constraint: glance.image
  ve_standalone_image:
    type: string
    constraints:
      - custom_constraint: glance.image
  ve_flavor:
    type: string
    constraints:
      - custom_constraint: nova.flavor
  use_config_drive:
    type: boolean
    default: false
```

```
  ssh_key:
    type: string
    constraints:
      - custom_constraint: nova.keypair
  sec_group_name:
    type: string
  admin_password:
    type: string
    label: F5 VE Admin User Password
    description: Password used to perform image import services
  root_password:
    type: string
  license:
    type: string
  external_network:
    type: string
    constraints:
      - custom_constraint: neutron.network
  default_gateway:
    type: string
    default: None

resources:
  networking:
    # A new resource
    type: OS::Neutron::FourNetworks
    properties:
      external_network: { get_param: external_network }
      sec_group_name: { get_param: sec_group_name }
  two_bigip_devices:
    type: OS::Heat::ResourceGroup
    depends_on: networking
    properties:
      count: 2
      resource_def:
        # A new resource
        type: F5::BIGIP::ClusterableDevice
        properties:
          ve_image: { get_param: ve_cluster_ready_image }
          ve_flavor: { get_param: ve_flavor }
          ssh_key: { get_param: ssh_key }
          use_config_drive: { get_param: use_config_drive }
          open_sec_group: { get_param: sec_group_name }
          admin_password: { get_param: admin_password }
          root_password: { get_param: root_password }
          license: { get_param: license }
          external_network: { get_param: external_network }
          mgmt_network: { get_attr: [networking, mgmt_network_name] }
          ha_network: { get_attr: [networking, ha_network_name] }
          network_1: { get_attr: [networking, client_data_network_name] }
          network_2: { get_attr: [networking, server_data_network_name] }
  standalone_device:
    # A new resource
    type: F5::BIGIP::StandaloneDevice
    depends_on: networking
    properties:
      ve_image: { get_param: ve_standalone_image }
      ve_flavor: { get_param: ve_flavor }
```

```
      ve_flavor: { get_param: ve_flavor }
      ssh_key: { get_param: ssh_key }
      use_config_drive: { get_param: use_config_drive }
      open_sec_group: { get_param: sec_group_name }
      admin_password: { get_param: admin_password }
      root_password: { get_param: root_password }
      license: { get_param: license }
      external_network: { get_param: external_network }
      mgmt_network: { get_attr: [networking, mgmt_network_name] }
      network_1: { get_attr: [networking, client_data_network_name] }
      network_2: { get_attr: [networking, server_data_network_name] }
```

And here is how we utilize those new resources defined in our environment file. Note that we no longer define all the parameters in the cli call to the Heat client (with the -P flag) because it is set in our environment file.

```
heat stack-create -f heat_template_test_with_environment.yaml -e heat_template_test_environment.yaml
```

Resources:

For more information on Heat Resource Types, along with the possible inputs and outputs: http://docs.openstack.org/developer/heat/template_guide/openstack.html

For more examples of how to prepare a BIG-IP image for booting in OpenStack and for clustering those clusterable instances: https://github.com/F5Networks/f5-openstack-heat

Reference Templates:

Below is the two child templates used in the above examples. We developed these on OpenStack Kilo with a BIG-IP image prepared from our github repo above.

f5_ve_standalone_3_nic.yaml

```
heat_template_version: 2015-04-30

description: This template deploys a standard f5 standalone VE.

parameters:
  open_sec_group:
    type: string
    default: open_sec_group
  ve_image:
    type: string
    constraints:
      - custom_constraint: glance.image
  ve_flavor:
    type: string
    constraints:
      - custom_constraint: nova.flavor
  use_config_drive:
    type: boolean
    default: false
  ssh_key:
    type: string
    constraints:
      - custom_constraint: nova.keypair
  admin_password:
    type: string
    hidden: true
  root_password:
    type: string
    hidden: true
  license:
    type: string
    hidden: true
  external_network:
    type: string
    default: test
    constraints:
      - custom_constraint: neutron.network
  mgmt_network:
    type: string
    default: test
    constraints:
      - custom_constraint: neutron.network
  network_1:
    type: string
    default: test
    constraints:
      - custom_constraint: neutron.network
  network_1_name:
    type: string
    default: network-1.1
  network_2:
    type: string
    default: test
    constraints:
      - custom_constraint: neutron.network
```

```
  network_2_name:
    type: string
    default: network-1.2
  default_gateway:
    type: string
    default: None

resources:
  mgmt_port:
    type: OS::Neutron::Port
    properties:
      network: { get_param: mgmt_network }
      security_groups: [ { get_param: open_sec_group }]
  network_1_port:
    type: OS::Neutron::Port
    properties:
      network: { get_param: network_1 }
      security_groups: [ { get_param: open_sec_group }]
  floating_ip:
    type: OS::Neutron::FloatingIP
    properties:
      floating_network: { get_param: external_network }
      port_id: { get_resource: mgmt_port }
  network_2_port:
    type: OS::Neutron::Port
    properties:
      network: {get_param: network_2 }
      security_groups: [ { get_param: open_sec_group }]
  ve_instance:
    type: OS::Nova::Server
    properties:
      image: { get_param: ve_image }
      flavor: { get_param: ve_flavor }
      key_name: { get_param: ssh_key }
      config_drive: { get_param: use_config_drive }
      networks:
        - port: {get_resource: mgmt_port}
        - port: {get_resource: network_1_port}
        - port: {get_resource: network_2_port}
      user_data_format: RAW
      user_data:
        str_replace:
          params:
            __admin_password__: { get_param: admin_password }
            __root_password__: { get_param: root_password }
            __license__: { get_param: license }
            __default_gateway__: { get_param: default_gateway }
            __network_1__: { get_param: network_1 }
            __network_1_name__: { get_param: network_1_name }
            __network_2__: { get_param: network_2 }
            __network_2_name__: { get_param: network_2_name }
          template: |
            {
               "bigip": {
                   "f5_ve_os_ssh_key_inject": "true",
                   "change_passwords": "true",
                   "admin_password": "__admin_password__",
```

```
                    "root_password": "__root_password__",
                    "license": {
                        "basekey": "__license__",
                        "host": "None",
                        "port": "8080",
                        "proxyhost": "None",
                        "proxyport": "443",
                        "proxyscripturl": "None"
                    },
                    "modules": {
                        "auto_provision": "false",
                        "ltm": "nominal"
                    },
                    "network": {
                        "dhcp": "true",
                        "selfip_prefix": "selfip-",
                        "vlan_prefix": "network-",
                        "routes": [
                           {
                             "destination": "0.0.0.0/0.0.0.0",
                             "gateway": "__default_gateway__"
                           }
                        ],
                        "interfaces": {
                            "1.1": {
                                "dhcp": "true",
                                "selfip_allow_service": "default",
                                "selfip_name": "selfip.__network_1_name__",
                                "selfip_description": "Self IP address for BIG-IP __network_1_name__ n
                                "vlan_name": "__network_1_name__",
                                "vlan_description": "VLAN for BIG-IP __network_1_name__ network traffi
                                "is_failover": "false",
                                "is_sync": "false",
                                "is_mirror_primary": "false",
                                "is_mirror_secondary": "false"
                            },
                            "1.2": {
                                "dhcp": "true",
                                "selfip_allow_service": "default",
                                "selfip_name": "selfip.__network_2_name__",
                                "selfip_description": "Self IP address for BIG-IP __network_2_name__ n
                                "vlan_name": "__network_2_name__",
                                "vlan_description": "VLAN for BIG-IP __network_2_name__ network traffi
                                "is_failover": "false",
                                "is_sync": "false",
                                "is_mirror_primary": "false",
                                "is_mirror_secondary": "false"
                            }
                        }
                    }
                }
            }

outputs:
  ve_instance_name:
    description: Name of the instance
    value: { get_attr: [ve_instance, name] }
  ve_instance_id:
```

```
    description: ID of the instance
    value: { get_resource: ve_instance }
  floating_ip:
    description: The Floating IP address of the VE
    value: { get_attr: [floating_ip, floating_ip_address] }
  mgmt_ip:
    description: The mgmt IP address of f5 ve instance
    value: { get_attr: [mgmt_port, fixed_ips, 0, ip_address] }
  mgmt_mac:
    description: The mgmt MAC address of f5 VE instance
    value: { get_attr: [mgmt_port, mac_address] }
  mgmt_port:
    description: The mgmt port id of f5 VE instance
    value: { get_resource: mgmt_port }
  network_1_ip:
    description: The 1.1 Nonfloating SelfIP address of f5 ve instance
    value: { get_attr: [network_1_port, fixed_ips, 0, ip_address] }
  network_1_mac:
    description: The 1.1 MAC address of f5 VE instance
    value: { get_attr: [network_1_port, mac_address] }
  network_1_port:
    description: The 1.1 port id of f5 VE instance
    value: { get_resource: network_1_port }
  network_2_ip:
    description: The 1.2 Nonfloating SelfIP address of f5 ve instance
    value: { get_attr: [network_2_port, fixed_ips, 0, ip_address] }
  network_2_mac:
    description: The 1.2 MAC address of f5 VE instance
    value: { get_attr: [network_2_port, mac_address] }
  network_2_port:
    description: The 1.2 port id of f5 VE instance
    value: { get_resource: network_2_port }
```

cluster_ready_ve_4_nic.yaml

```
heat_template_version: 2015-04-30

description: This template deploys a standard f5 VE ready for clustering.

parameters:
  open_sec_group:
    type: string
    default: open_sec_group
  ve_image:
    type: string
    label: F5 VE Image
    description: The image to be used on the compute instance.
    constraints:
      - custom_constraint: glance.image
  ve_flavor:
    type: string
    label: F5 VE Flavor
    description: Type of instance (flavor) to be used for the VE.
    constraints:
      - custom_constraint: nova.flavor
  use_config_drive:
```

```
  type: boolean
  label: Use Config Drive
  description: Use config drive to provider meta and user data.
  default: false
ssh_key:
  type: string
  label: Root SSH Key Name
  description: Name of key-pair to be installed on the instances.
  constraints:
    - custom_constraint: nova.keypair
admin_password:
  type: string
  label: F5 VE Admin User Password
  description: Password used to perform image import services
root_password:
  type: string
  label: F5 VE Root User Password
  description: Password used to perform image import services
license:
  type: string
  label: Primary VE License Base Key
  description: F5 TMOS License Basekey
external_network:
  type: string
  label: External Network
  description: Network for Floating IPs
  default: test
  constraints:
    - custom_constraint: neutron.network
mgmt_network:
  type: string
  label: VE Management Network
  description: Management Interface Network.
  default: test
  constraints:
    - custom_constraint: neutron.network
ha_network:
  type: string
  label: VE HA Network
  description: HA Interface Network.
  default: test
  constraints:
    - custom_constraint: neutron.network
network_1:
  type: string
  label: VE Network for the 1.2 Interface
  description: 1.2 TMM network.
  default: test
  constraints:
    - custom_constraint: neutron.network
network_1_name:
  type: string
  label: VE Network Name for the 1.2 Interface
  description: TMM 1.2 network name.
  default: data1
network_2:
  type: string
  label: VE Network for the 1.2 Interface
```

```
    label: VE Network for the 1.3 Interface
    description: 1.3 TMM Network.
    default: test
    constraints:
      - custom_constraint: neutron.network
  network_2_name:
    type: string
    label: VE Network Name for the 1.3 Interface
    description: TMM 1.3 network name.
    default: data2
  default_gateway:
    type: string
    label: Default Gateway IP
    default: None
    description: Upstream Gateway IP Address for VE instances

resources:
  mgmt_port:
    type: OS::Neutron::Port
    properties:
      network: { get_param: mgmt_network }
      security_groups: [{ get_param: open_sec_group }]
  network_1_port:
    type: OS::Neutron::Port
    properties:
      network: {get_param: network_1 }
      security_groups: [{ get_param: open_sec_group }]
  floating_ip:
    type: OS::Neutron::FloatingIP
    properties:
      floating_network: { get_param: external_network }
      port_id: { get_resource: mgmt_port }
  network_2_port:
    type: OS::Neutron::Port
    properties:
      network: {get_param: network_2 }
      security_groups: [{ get_param: open_sec_group }]
  ha_port:
    type: OS::Neutron::Port
    properties:
      network: {get_param: ha_network}
      security_groups: [{ get_param: open_sec_group }]
  ve_instance:
    type: OS::Nova::Server
    properties:
      image: { get_param: ve_image }
      flavor: { get_param: ve_flavor }
      key_name: { get_param: ssh_key }
      config_drive: { get_param: use_config_drive }
      networks:
        - port: {get_resource: mgmt_port}
        - port: {get_resource: ha_port}
        - port: {get_resource: network_1_port}
        - port: {get_resource: network_2_port}
      user_data_format: RAW
      user_data:
        str_replace:
          params:
```

```
        __admin_password__: { get_param: admin_password }
        __root_password__: { get_param: root_password }
        __license__: { get_param: license }
        __default_gateway__: { get_param: default_gateway }
        __network_1_name__: { get_param: network_1_name }
        __network_2_name__: { get_param: network_2_name }
      template: |
        {
           "bigip": {
               "ssh_key_inject": "true",
               "change_passwords": "true",
               "admin_password": "__admin_password__",
               "root_password": "__root_password__",
               "license": {
                   "basekey": "__license__",
                   "host": "None",
                   "port": "8080",
                   "proxyhost": "None",
                   "proxyport": "443",
                   "proxyscripturl": "None"
               },
               "modules": {
                   "auto_provision": "false",
                   "ltm": "nominal"
               },
               "network": {
                   "dhcp": "true",
                   "selfip_prefix": "selfip-",
                   "routes": [
                      {
                        "destination": "0.0.0.0/0.0.0.0",
                        "gateway": "__default_gateway__"
                      }
                   ],
                   "interfaces": {
                       "1.1": {
                           "dhcp": "true",
                           "selfip_allow_service": "default",
                           "selfip_name": "selfip.HA",
                           "selfip_description": "Self IP address for BIG-IP Cluster HA subnet",
                           "vlan_name": "vlan.HA",
                           "vlan_description": "VLAN for BIG-IP Cluster HA traffic",
                           "is_failover": "true",
                           "is_sync": "true",
                           "is_mirror_primary": "true",
                           "is_mirror_secondary": "false"
                       },
                       "1.2": {
                           "dhcp": "true",
                           "selfip_allow_service": "default",
                           "selfip_name": "selfip.__network_1_name__",
                           "selfip_description": "Self IP address for BIG-IP __network_1_name__",
                           "vlan_name": "__network_1_name__",
                           "vlan_description": "VLAN for BIG-IP __network_1_name__ traffic",
                           "is_failover": "false",
                           "is_sync": "false",
                           "is_mirror_primary": "false",
                           "is_mirror_secondary": "false"
```

```
                                "is_mirror_secondary": "false"
                            },
                            "1.3": {
                                "dhcp": "true",
                                "selfip_allow_service": "default",
                                "selfip_name": "selfip.__network_2_name__",
                                "selfip_description": "Self IP address for BIG-IP __network_2_name__",
                                "vlan_name": "__network_2_name__",
                                "vlan_description": "VLAN for BIG-IP __network_2_name__ traffic",
                                "is_failover": "false",
                                "is_sync": "false",
                                "is_mirror_primary": "false",
                                "is_mirror_secondary": "false"
                            }
                        }
                    }
                }
            }
outputs:
  ve_instance_name:
    description: Name of the instance
    value: { get_attr: [ve_instance, name] }
  ve_instance_id:
    description: ID of the instance
    value: { get_resource: ve_instance }
  mgmt_ip:
    description: The mgmt IP address of f5 ve instance
    value: { get_attr: [mgmt_port, fixed_ips, 0, ip_address] }
  mgmt_mac:
    description: The mgmt MAC address of f5 VE instance
    value: { get_attr: [mgmt_port, mac_address] }
  mgmt_port:
    description: The mgmt port id of f5 VE instance
    value: { get_resource: mgmt_port }
  ha_ip:
    description: The HA IP address of f5 ve instance
    value: { get_attr: [ha_port, fixed_ips, 0, ip_address] }
  ha_mac:
    description: The HA MAC address of f5 VE instance
    value: { get_attr: [ha_port, mac_address] }
  ha_port:
    description: The ha port id of f5 VE instance
    value: { get_resource: ha_port }
  network_1_ip:
    description: The 1.2 Nonfloating SelfIP address of f5 ve instance
    value: { get_attr: [network_1_port, fixed_ips, 0, ip_address] }
  network_1_mac:
    description: The 1.2 MAC address of f5 VE instance
    value: { get_attr: [network_1_port, mac_address] }
  network_1_port:
    description: The 1.2 port id of f5 VE instance
    value: { get_resource: network_1_port }
  network_2_ip:
    description: The 1.3 Nonfloating SelfIP address of f5 ve instance
    value: { get_attr: [network_2_port, fixed_ips, 0, ip_address] }
  network_2_mac:
    description: The 1.3 MAC address of f5 VE instance
    value: { get_attr: [network_2_port, mac_address] }
```

```
  network_2_port:
    description: The 1.3 port id of f5 VE instance
    value: { get_resource: network_2_port }
  floating_ip:
    description: Floating IP address of VE servers
    value: { get_attr: [ floating_ip, floating_ip_address ] }
```

F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | f5.com

F5 Networks, Inc. Corporate Headquarters info@f5.com	F5 Networks Asia-Pacific apacinfo@f5.com	F5 Networks Ltd. Europe/Middle-East/Africa emeainfo@f5.com	F5 Networks Japan K.K. f5j-info@f5.com

©2017 F5 Networks, Inc. All rights reserved. F5, F5 Networks, and the F5 logo are trademarks of F5 Networks, Inc. in the U.S. and in certain other countries. Other F5 trademarks are identified at f5.com. Any other products, services, or company names referenced herein may be trademarks of their respective owners with no endorsement or affiliation, express or implied, claimed by F5. CS04-00015 0113