Anexos

Anexo 1: Guía de plantillas VNFD y VNFFGD

En este anexo se va a describir la estructura completa de los VNFD y VNFFGD según su definición en [1] y [2].

1) Guía de plantilla VNFD

```
tosca_definitions_version: tosca_simple_profile_for_nfv_1_0_0
description: Ejemplo de VNFD
metadata:
 template_name: sample-tosca-vnfd-template-guide
topology_template:
 node_templates:
   VDU:
     type: tosca.nodes.nfv.VDU.Tacker
     properties:
       image: Imagen a usar por la VNF
       flavor: Flavor a usar por la VNF
       availability_zone: zona de disponibilidad (nova)
       mgmt_driver: [default=noop]
       user_data: comandos a ejecutar en tiempo de creación
       user_data_format: formato de los comandos
       key_name: clave de usuario

   CP:
     type: tosca.nodes.nfv.CP.Tacker
     properties:
       management: [true, false]
       security_groups: lista de grupos de seguridad
(restricciones de tráfico)
     requirements:
       - virtualLink:
           node: VL asociado a este CP
       - virtualBinding:
           node: VDU asociado a este CP
   VL:
     type: tosca.nodes.nfv.VL
     properties:
       network_name: nombre de la red a conectar
       vendor: Tacker
   FIP:
     type: tosca.nodes.network.FloatingIP
     properties:
       floating_network: red externa donde crear la floating-ip
       floating_ip_address: floating-ip a asociar
     requirements:
       - link:
           node: CP asociado a la floating ip
```

[1] https://docs.openstack.org/tacker/latest/contributor/vnfd_template_description.html
[2] https://docs.openstack.org/tacker/latest/user/vnffg_usage_guide.html

Un **VNFD** está formado por cuatro secciones:

- *Virtual Deployment Unit* (VDU): sección donde se definen las características generales y de cómputo de la VNF como, por ejemplo, nombre, imagen del sistema operativo, RAM, memoria en disco, etc.
  El parámetro user_data hace referencia a los comandos de Shell que ejecutara la VNF en tiempo de creación.

  El parámetro mgmt_driver indica el driver de configuración de la VNF. Ciertos sistemas operativo soportan configuración como VNF a partir de paquetes de configuración en lugar de con comandos de Shell.
- *Connection Points* (CP): sección donde se define la asociación entre el VDU y el VL.
- *Floating IP* (FIP) (opcional): se define la asociación de una *floating-ip* a un CP para proporcionar conectividad con una red pública.

2) Guía de plantilla VNFFGD

```
tosca_definitions_version:
tosca_simple_profile_for_nfv_1_0_0

description: Sample VNFFG template

topology_template:

  node_templates:

    Forwarding_path2:
      type: tosca.nodes.nfv.FP.TackerV2
      description: creates path (CP1->CP2)
      properties:
        id: 52
        symmetrical: false
        policy:
          type: ACL
          criteria:
            - name: name
              classifier:
                network_src_port_id: Server port
                destination_port_range: port:port
                ip_proto: 4/6
                ip_dst_prefix: IP
        path:
          - forwarder: VNFD1
            capability: CP1
          - forwarder: VNFD2
            capability: CP2
                .
                .
                .

  groups:
    VNFFG1:
      type: tosca.groups.nfv.VNFFG
      description: SSH to Corporate Net
```

```
    properties:
      vendor: tacker
      version: 1.0
      number_of_endpoints: SF number
      dependent_virtual_link: [VL1,VL2, ...]
      connection_point: [CP1,CP2, ...]
      constituent_vnfs: [VNFD1,VNFD2, ...]
    members: [Forwarding_path2]
```

Un **VNFFGD** está formado por cuatro secciones:

- *Forwarding_path:* sección donde se definen las del clasificador de flujo para el tráfico que atravesara la cadena. En Path se definen los CP de los VNFD a los cuales tiene que dirigir el tráfico.
  El parámetro symetrical indica si la cadena de servicio es bidireccional o unidireccional
- *Groups:* sección donde se define el *Forwarding_path* que contiene las reglas de clasificación y el numero de SF que forman la cadena.
  Tambien para cada VNFD que forma la cadena, el CP y VL asociados.

Anexo 2: CLI de Tacker

En este anexo se van a detallar los comandos básicos de Tacker para su uso. La información completa se puede encontrar en [3]

1) Registro del VIM

```
tacker vim-register –config-file vim-config-file.yaml –is-default vim-name
```

Donde vim-config-file.yaml es el fichero que describe el VIM, contiene parámetros que hacen referencia a la instalación de Openstack y vim-name es el nombre que le das al VIM registrado.

El parámetro –is-default hace referencia a si se va a usar ese VIM por defecto para gestionar cualquier VNF

Ejemplo de vim-config-file.yaml

```
auth_url: 'https://192.168.122.3:5000/v3'
username: 'admin'
password: 'opnfv-secret-password'
project_name: 'admin'
project_domain_name: 'Default'
user_domain_name: 'Default'
cert_verify: 'False'
```

Donde; auth_url es la dirección del servicio de autenticación, en nuestro caso keystone, username y password son los datos de registro y Project_name, project_domain_name y user_domain_name: son datos por defecto.

2) Registro de los VNFD

Para crear un VNF es necesario registrar su VNFD en el catálogo de Tacker.

```
tacker vnfd-create –vnfd-file vnfd.yaml vnfd-name
```

Donde vnfd.yaml es el fichero que contiene la plantilla TOSCA que define la VNF. Información detallada en el Anexo 1: Guía de plantillas VNFD y VNFFGD.

3) Creación de las VNF

Las VNF se crean apartir de sus VNFD

```
Tacker vnf-create –vnfd-name vnfd-name vnf-name
```

Donde vnfd-name es el nombre del VNFD creado anteriormente y vnf-name es el nombre que le asignas a la VNF creada.

4) Registro de los VNFFGD

[3] https://docs.openstack.org/ocata/cli-reference/tacker.html

Para crear cadenas de servicio, se crean VNF *Forwarding Graph* (VNFFG), son usados para orquestar y gestionar el tráfico que atraviesan las VNF. Estos VNFFG se detallan en el anexo 2.

Para crear un VNFFG es necesario registrar su VNFFGD en el catálogo de Tacker

```
tacker vnffgd-create –vnffgd-file vnffgd.yaml vnffg-name
```

Donde vnffgd.yaml es el fichero que contiene la plantilla TOSCA que describe el VNF *Forwarding Graf*. Información detallada en el Anexo 1: Guía de plantillas VNFD y VNFFGD.

5) Creación del VNFFG

El VNFFG se crea a partir de su VNFFGD.

```
tacker vnffg-create –vnfggd-name vnffgd-name vnfg-name
```

Donde vnffgd-name es el nombre del VNFFGD creado anteriormente y vnffg-name es el nombre que le asignas al VNFFG creado.

Anexo 3: Despliegue del escenario 1

A continuación, se describe con detalle el procedimiento a seguir para realizar el despliegue específico y la puesta en marcha del escenario 1 descrito en el apartado 5.3.1 del proyecto.

Descargar la imagen de Ubuntu "Ubuntu 16.04 Xenial Xerus". Será el sistema operativo de las VNF.

```
wget https://cloud-images.ubuntu.com/xenial/current/xenial-server-cloudimg-amd64-disk1.img
```

Se establecen las variables de entorno necesarias para usar el CLI de Openstack.

```
source openrc
```

Se crea la imagen Openstack que se empleara para crear las instancias

```
openstack image create --disk-format qcow2 --container-format bare --public --file xenial-server-cloudimg-amd64-disk1.img Ubuntu-Image
```

--disk-format hace referencia al formato del fichero de la imagen de disco que se va a utilizar.

--container-format indica que la imagen se instalara sobre el disco. Similar a un servidor *baremetal.*

--public indica que la imagen será accesible para todos los usuarios de openstack.

Se definen las propiedades con las que se configurarán las VNF

```
openstack flavor create  --ram 512 --disk 8 --public Little-Flavor
```

--ram indica la memoria RAM que se le asigna a las VNF que se configuren mediante este *flavor.*

--disk indica el espacion de disco que se le asigna a las VNF que se configuren mediante este *flavor.*

--public indica que el flavor será accesible para todos los usuarios de openstack.

Se crea la clave RSA para acceder a VNF por SSH. Se crea a partir de la clave pública de la maquina *host*.

```
openstack keypair create --public-key ~/.ssh/id_rsa.pub ClaveSSH
```

Se crea la red y subred virtual externa, esta dispondrá de un *pool* de *floating-ip.*

```
openstack network create --external --provider-physical-network flat --provider-network-type flat PUBLIC
```

```
openstack subnet create --network PUBLIC --allocation-pool start=192.168.122.10,end=192.168.122.254 --subnet-range 192.168.122.0/24 --gateway 192.168.122.1 --no-dhcp Subnet-1
```

Se crea la red y subred privada donde estarán las VNF

```
openstack network create --internal Red-Privada

openstack subnet create --subnet-range 10.10.10.0/24 --dhcp --ip-version 4 --network Red-Privada Subred-Privada-1

openstack network create --internal Red-Privada-2

openstack subnet create --subnet-range 20.20.20.0/24 --dhcp --gateway 20.20.20.1 --ip-version 4 --network Red-Privada-2 Subred-Privada-2
```

--gateway 20.20.20.1, importante especificar que el *Gateway* se encuentra en esa dirección. Más adelante se creará la VNF correspondiente con el router con esa *fixed-ip.*

Se crea un *router* virtual llamado Router-Ext y se asocia a ambas redes. Es un componente lógico que reenvía paquetes de datos entre redes. También proporciona el reenvío de Capa 3 y NAT para proporcionar acceso a la red externa a las VNF mediante la asociación de *floating-ip.*

```
openstack router create --enable RouterExt

openstack router set --external-gateway PUBLIC --enable-snat RouterExt

openstack router add subnet RouterExt Subred-Privada-1
```

A continuación, se despliegan las VNF que van a estar en la “Red-Privada 1”.

Una vez creada la estructura de la red se crearán las VNF mediante el CLI de Tacker.

El primer paso para utilizar Tacker es registrar el VIM. Para ello, se ha de crear el fichero *vim-config.yaml* con el siguiente contenido y registra el VIM con dicho fichero.

vim-config.yaml

```
auth_url: 'https://192.168.122.3:5000/v3'
username: 'admin'
password: 'opnfv-secret-password'
project_name: 'admin'
project_domain_name: 'Default'
user_domain_name: 'Default'
cert_verify: 'False'
```

```
tacker vim-register --config-file vim-config.yaml –is-default vim1
```

A continuación, se crean plantillas TOSCA para cada VNF, se tratan de ficheros *.yaml* que describen la VNF, estos se emplearan para crear los VNFD. El formato de dichas plantillas puede consultarse en el Anexo 1: Guía de plantillas VNFD y VNFFGD. El detalle de los campos aquí empleados se describe en el Anexo 3.

Pc-Gestion.yaml

```
tosca_definitions_version: tosca_simple_profile_for_nfv_1_0_0
description: PC que gestiona los demas
metadata:
  template_name: vnfd-PG

topology_template:
  node_templates:
    VDU1:
      type: tosca.nodes.nfv.VDU.Tacker
      properties:
        name: PC_Gestion
        image: Ubuntu-Image
        flavor: Little-Flavor
        key_name: ClaveSSH
        availability_zone: nova
        mgmt_driver: noop
        user_data_format: RAW
        user_data: |
          #!/bin/bash
          sudo chmod 777 /etc/resolv.conf
          sudo echo "nameserver 8.8.8.8" > /etc/resolvCP.conf
          while read line; do echo "$line">>resolvCP.conf; done <
/etc/resolv.conf
          sudo rm /etc/resolv.conf
          mv /etc/resolvCP.conf /etc/resolv.conf
          #sudo apt-get --yes --force-yes install python
          sudo echo "-----BEGIN RSA PRIVATE KEY-----

MIIEowIBAAKCAQEAyzh0YN3Z+tX6w9qm5ZWn9wuu+AIXfB9tHa4VzB6HjfLVgvVJ

gK9osjkiS7ONk4Ow/qIfhlTl/kgtJf7eoY6Kyhevmm+wmAJ6Igw5YXg2oAyga+6f

g1LLCyDyluWOTbmt5O1B36vy/3L06pafqGeZRA9o9zfa0i7M7c83aYXIU6OpUVyj

fKIR02C2qR1CcZ1rPN++5ihR+l448ujT15V5h2N72x+dkbBJ1QLTmLxh5MSx6jbL

peaGHRqZ01oEVbumcjH7VJoVU+cwVin8/jFk1SF1DTYwRhAAL0K78GWydxoggKi6

34xdeli4KOWyN7ksaxaWsH2QBo2vKn8OySVzaQIDAQABAoIBAQCcGZhjk+ZNpDJE

YV9T+ToVLRdQkCLECEvDwaYirczIP6C8QHgebh7Iz9RFO/3jMwQxBA2dHPex9HUh

u339sUVw6PLNeT+39CcmoictdK7ZD/nmDlze/ijTOpK9UEWX9KnbsFeqXSWCDSaO

T/cB39o+CmxBgdkHHEXz4fkZiFcRzsUfEg1+Y1DTvFgBS7T/2+KzQuLaoUUBUZT5

1lM5OB3C2shh8qiXJhyHYhjQRwv2qF6O4P1chlG7vL7AgqlgO1x7Y4OPrqDM0MOb

b8HetLbnIhNgXqvjN/zy1srtlhWfxf94yQp7H/orvpMJ5jw+3qUZCBWo9/TBw1aO
```

```
Z/RMRx8VAoGBAPPUA0Q+v0v1sM6x5BMvxDxwsyex3WhPCUVNFZj+jvsbHGDBl0Nl
Rwj2632CgICzsRn0hkaN4aChfxH6ftwJ9AfXr40tiqGEUe2zVAwQs/jH9XWjQtgO
O499xf/8VUr3Cut1EqzPTImL7jk8RfiwoKED5cTPtYBVDjRoGw5+M/B7AoGBANVd
f7ZkRVH4p1lN1TiTYw7liBxLajmVARqApmCG9jx2IV3TWEh3pmK5G4fZkRfVv2PJ
xmfoFNpTSzXtKN8vWBB6+Go9sGZ2+T2XJdXCcy1ZqlMYeS2+/j7p/IeIMGzsAeM0
CDv2dEqk9weYTDu0zFYMsQMZPE2K2/bvFoXSZNBrAoGAIcF7Rkptj2WPOb5U9fcJ
8tgjZV5xaYxvmyGF2O18+/SP5DFI8LKnt+z/Qxv7EFZQMWix4jioQOW6wtFsGKhk
GWXZzvC8HYpFEWRfQWBamhmMuOGGUoT95+qgq8TSRhOXdmt5z2TPksdFjrTydvB0
/HWerLWYyhB8a/Lxs/ry09sCgYBs7xOpV4Bc3YrzeV2HSRimHmJjr81IIN8zXMOV
PNKzA8z9Tk4gkZYNiVyY+2est6DDnd0CZ+ddoHEh0zeu20knAOGbvGs3pT6TR3w1
qtWLLeBcFH6p8H12OZiIyeuPcN911LbvhmDRS3AkHImvYhuBQ3GMl3HGVUMAzAi/
wi9eiQKBgENkmdT+F2s+r2SEB4t+uPB4/uzra0zvIbK1yH+C+VEQv2g4yd9QUr0u
tbi5byOZa5XHfxT6KDpStbyCZmuIYAUkql5mEMN8ceNlRHxS6jDe8gzHg8dFyHTj
          yXTZ31LouOhBuya95hV8tGSNrpPAqBeebqUfket3iWKspdZqxzBc
          -----END RSA PRIVATE KEY-----" > /home/ubuntu/id_rsa
    CP1:
      type: tosca.nodes.nfv.CP.Tacker
      properties:
        type: vnic
        management: true
        anti_spoofing_protection: false
      requirements:
        - virtualLink:
            node: VL1
        - virtualBinding:
            node: VDU1
    VL1:
      type: tosca.nodes.nfv.VL
      properties:
        network_name: Red-Privada
        vendor: Tacker
    FIP1:
      type: tosca.nodes.network.FloatingIP
      properties:
        floating_network: public
        floating_ip_address: 192.168.122.10
      requirements:
        - link:
            node: CP1
```

DNS.yaml

```
tosca_definitions_version: tosca_simple_profile_for_nfv_1_0_0
description: Servidor DNS autoritativo de Red-Privada-2
metadata:
  template_name: vnfd-DNS

topology_template:
  node_templates:
    VDU2:
      type: tosca.nodes.nfv.VDU.Tacker
      properties:
        name: DNS-server
        image: Ubuntu-Image
        flavor: Little-Flavor
        key_name: ClaveSSH
        availability_zone: nova
        mgmt_driver: noop
        user_data_format: RAW
        user_data: |
          #!/bin/bash
          sudo chmod 777 /etc/resolv.conf
          sudo echo "nameserver 8.8.8.8" > /etc/resolvCP.conf
          while read line; do echo "$line">>resolvCP.conf; done <
/etc/resolv.conf
          sudo rm /etc/resolv.conf
          mv /etc/resolvCP.conf /etc/resolv.conf
          sudo apt-get --yes install bind9
          sudo mv /etc/bind/named.conf /etc/bind/named.conf.Orig
          sudo echo -e "include \"/etc/bind/rndc.key\";
          options {
          directory \"/var/cache/bind\";
          min-roots 1;
          match-clients {any;};
          };
          logging {
          category lame-servers { null; };
          //category cname { null; };
          };
          zone \"lan2.com\" {
          type master;
          file \"/etc/bind/db.lan2.com\";
          };" > /etc/bind/named.conf
          sudo echo "\$TTL  604800 ; default ttl
          lan2.com. IN  SOA DNS.lan1.com. root.localhost. (
          1         ; Serial
          604800       ; Refresh
          86400        ; Retry
          2419200     ; Expire
          604800       ; Negative Cache TTL
          )
          lan2.com.       84600   IN      NS      DNS.lan1.com.
          DNS.lan1.com. 84600   IN  A   10.10.10.10
          WEB.lan2.com.    86400   IN  A   20.20.20.5" >
/etc/bind/db.lan2.com
          sudo service bind9 start
    CP2:
      type: tosca.nodes.nfv.CP.Tacker
      properties:
```

```
        type: vnic
        management: true
        ip_address: 10.10.10.10
        anti_spoofing_protection: false
      requirements:
        - virtualLink:
            node: VL2
        - virtualBinding:
            node: VDU2
    VL2:
      type: tosca.nodes.nfv.VL
      properties:
        network_name: Red-Privada
        vendor: Tacker
    FIP2:
      type: tosca.nodes.network.FloatingIP
      properties:
        floating_network: PUBLIC
        floating_ip_address: 192.168.122.11
      requirements:
        - link:
            node: CP2
```

Router-Ubuntu.yaml

```
tosca_definitions_version: tosca_simple_profile_for_nfv_1_0_0
description: Router frontera
metadata:
  template_name: vnfd-Router

topology_template:
  node_templates:
    VDU3:
      type: tosca.nodes.nfv.VDU.Tacker
      properties:
        name: Router-Ubuntu
        image: Ubuntu-Image
        flavor: Little-Flavor
        key_name: ClaveSSH
        availability_zone: nova
        mgmt_driver: noop
        user_data_format: RAW
        user_data: |
          #!/bin/bash
          sudo chmod 777 /etc/resolv.conf
          sudo echo "nameserver 8.8.8.8" > /etc/resolvCP.conf
          while read line; do echo "$line">>resolvCP.conf; done <
/etc/resolv.conf
          sudo rm /etc/resolv.conf
          mv /etc/resolvCP.conf /etc/resolv.conf
          sudo echo 1 > /proc/sys/net/ipv4/ip_forward
          sudo echo "auto ens4" >> /etc/network/interfaces
          sudo echo "iface ens4 inet static" >>
/etc/network/interfaces.d/50-cloud-init.cfg
          sudo echo "address 20.20.20.1" >>
/etc/network/interfaces.d/50-cloud-init.cfg
          sudo echo "netmask 255.255.255.0" >>
/etc/network/interfaces.d/50-cloud-init.cfg
          sudo echo "network 20.20.20.0" >>
/etc/network/interfaces.d/50-cloud-init.cfg
```

```
          sudo echo "broadcast 255.255.255.0" >>
/etc/network/interfaces.d/50-cloud-init.cfg
          sudo service networking restart

    CP3:
      type: tosca.nodes.nfv.CP.Tacker
      properties:
        type: vnic
        management: true
        anti_spoofing_protection: false
      requirements:
        - virtualLink:
            node: VL3
        - virtualBinding:
            node: VDU3
    VL3:
      type: tosca.nodes.nfv.VL
      properties:
        network_name: Red-Privada
        vendor: Tacker
    CP4:
      type: tosca.nodes.nfv.CP.Tacker
      properties:
        ip_address: 20.20.20.1
        type: vnic
        anti_spoofing_protection: false
      requirements:
        - virtualLink:
            node: VL3
        - virtualBinding:
            node: VDU3
    VL4:
      type: tosca.nodes.nfv.VL
      properties:
        network_name: Red-Privada-2
        vendor: Tacker
    FIP3:
      type: tosca.nodes.network.FloatingIP
      properties:
        floating_network: PUBLIC
        floating_ip_address: 192.168.122.13
      requirements:
        - link:
            node: CP3
```

WEB.yaml

```
tosca_definitions_version: tosca_simple_profile_for_nfv_1_0_0
description: Servidor Web
metadata:
  template_name: vnfd-WEB

topology_template:
  node_templates:
    VDU4:
      type: tosca.nodes.nfv.VDU.Tacker
      properties:
        name: WEB
        image: Ubuntu-Image
        flavor: Little-Flavor
        key_name: ClaveSSH
        availability_zone: nova
        mgmt_driver: noop
        user_data_format: RAW
        user_data: |
          #!/bin/bash
          sudo chmod 777 /etc/resolv.conf
          sudo echo "nameserver 8.8.8.8" > /etc/resolvCP.conf
          while read line; do echo "$line">>resolvCP.conf; done <
/etc/resolv.conf
          sudo rm /etc/resolv.conf
          mv /etc/resolvCP.conf /etc/resolv.conf

    CP5:
      type: tosca.nodes.nfv.CP.Tacker
      properties:
        type: vnic
        ip_address: 20.20.20.5
        management: true
        anti_spoofing_protection: false
      requirements:
        - virtualLink:
            node: VL5
        - virtualBinding:
            node: VDU4
    VL5:
      type: tosca.nodes.nfv.VL
      properties:
        network_name: Red-Privada-2
        vendor: Tacker
```

Una vez definidas las plantillas TOSCA, se procede a registrar los VNFD con las plantillas creadas.

```
tacker vnfd-create –vnfd-file Pc-Gestion.yaml vnfd-PG

tacker vnfd-create –vnfd-file DNS.yaml vnfd-DNS

tacker vnfd-create –vnfd-file Router.yaml vnfd-Router

tacker vnfd-create –vnfd-file PC-1.yaml vnfd-WEB
```

Con los VNFD registrados, se crean las VNF a partir de los VNFD.

```
tacker vnf-create –vnfd-name vnfd-PG vnf-PG

tacker vnf-create –vnfd-name vnfd-DNS vnf-DNS

tacker vnf-create –vnfd-name vnfd- Router vnf- Router
```

Tras estas operaciones ya tenemos el escenario desplegado, a falta de crear el servidor WEB. A continuación, se agrega una ruta en el *router* virtual “Router-Ext” para saber alcanzar la “Red-Privada-2” (20.20.20.0)

```
openstack router set --route destination=20.20.20.0/24,gateway=10.10.10.9 RouterExt
```

Como se ha comentado, una vez creado “Router”y la ruta para alcanzar la “Red-Privada-2” en “RouterExt”, creamos el servidor WEB.

```
tacker vnf-create –vnfd-name vnfd- PC1 vnf- WEB
```

Anexo 4: Despliegue del escenario 2

A continuación, se describe con detalle el procedimiento a seguir para realizar el despliegue específico y la puesta en marcha del escenario 2 descrito en el apartado 5.3.2.

Descargar la imagen de Ubuntu "Ubuntu 16.04 Xenial Xerus". Será el sistema operativo de las VNF.

```
wget https://cloud-images.ubuntu.com/xenial/current/xenial-server-cloudimg-amd64-disk1.img
```

Se establecen las variables de entorno necesarias para usar el CLI de Openstack.

```
source openrc
```

Se crea la imagen Openstack que se empleara para crear las instancias

```
openstack image create --disk-format qcow2 --container-format bare --public --file xenial-server-cloudimg-amd64-disk1.img Ubuntu-Image
```

--disk-format hace referencia al formato del fichero de la imagen de disco que se va a utilizar.

--container-format indica que la imagen se instalara sobre el disco. Similar a un servidor *baremetal.*

--public indica que la imagen será accesible para todos los usuarios de openstack.

Se definen las propiedades con las que se configurarán las VNF

```
openstack flavor create  --ram 512 --disk 8 --public Little-Flavor
```

--ram indica la memoria RAM que se le asigna a las VNF que se configuren mediante este *flavor.*

--disk indica el espacio de disco que se le asigna a las VNF que se configuren mediante este *flavor.*

--public indica que el flavor será accesible para todos los usuarios de openstack.

Se crea la clave RSA para acceder a VNF por SSH. Se crea a partir de la clave pública de la maquina *host*.

```
openstack keypair create --public-key ~/.ssh/id_rsa.pub ClaveSSH
```

Se crea la red y subred virtual externa. Los términos red y subred en Openstack hacen referencia a un conjunto de direcciones IP y sus características que pueden ser asignadas a las VNF bajo las características de las redes tradicionales.

```
openstack network create --external --provider-physical-network flat --provider-network-type flat PUBLIC

openstack subnet create --network PUBLIC --allocation-pool start=192.168.122.10,end=192.168.122.254 --subnet-range 192.168.122.0/24 --gateway 192.168.122.1 --no-dhcp Subnet-1
```

El parámetro –external hace referencia que la red es externa, dispone de *floating-ip*.

Se crea la red y subred privada donde estarán las VNF

```
openstack network create --internal Red-Privada

openstack subnet create --subnet-range 10.10.10.0/24 --dhcp --ip-version 4 --network Red-Privada Subred-Privada-1
```

Se crea un *router* virtual y se asocia a ambas redes. Es un componente lógico que reenvía paquetes de datos entre redes. También proporciona el reenvío de Capa 3 y NAT para proporcionar acceso a la red externa a las VNF mediante la asociación de *floating-ip.*

```
openstack router create --enable RouterExt

openstack router set --external-gateway PUBLIC --enable-snat RouterExt

openstack router add subnet RouterExt Subred-Privada-1
```

Una vez creada la estructura de la red se crearán las VNF mediante el CLI de Tacker.

El primer paso para utilizar Tacker es registrar el VIM. Para ello, se ha de crear el fichero *vim-config.yaml* con el siguiente contenido y registra el VIM con dicho fichero.

vim-config.yaml

```
auth_url: 'https://192.168.122.3:5000/v3'
username: 'admin'
password: 'opnfv-secret-password'
project_name: 'admin'
project_domain_name: 'Default'
user_domain_name: 'Default'
cert_verify: 'False'
```

```
tacker vim-register --config-file vim-config.yaml –is-default vim1
```

A continuación, se crean plantillas TOSCA para cada VNF. El formato de dichas plantillas puede consultarse en [69]. El detalle de los campos aquí empleados se describe en el Anexo 1.

Cliente.yaml

```
tosca_definitions_version: tosca_simple_profile_for_nfv_1_0_0
description: Cliente-SFC
metadata:
  template_name: Cliente

topology_template:
  node_templates:
    VDU1:
      type: tosca.nodes.nfv.VDU.Tacker
      properties:
        name: Cliente
        image: Ubuntu-Image
        flavor: Little-Flavor
        key_name: ClaveSSH
        availability_zone: nova
        mgmt_driver: noop
        user_data_format: RAW
        user_data: |
          #!/bin/sh
          sudo chmod 777 /etc/resolv.conf
          sudo echo "nameserver 8.8.8.8" > /etc/resolvCP.conf
          while read line; do echo "$line">>resolvCP.conf; done <
/etc/resolv.conf
          sudo rm /etc/resolv.conf
          mv /etc/resolvCP.conf /etc/resolv.conf
          sudo apt-get --yes --force-yes install python
    CP1:
      type: tosca.nodes.nfv.CP.Tacker
      properties:
        type: vnic
        management: true
        anti_spoofing_protection: false
      requirements:
        - virtualLink:
            node: VL1
        - virtualBinding:
            node: VDU1
    VL1:
      type: tosca.nodes.nfv.VL
      properties:
        network_name: Red-Privada
        vendor: Tacker
    FIP1:
      type: tosca.nodes.network.FloatingIP
      properties:
        floating_network: public
        floating_ip_address: 192.168.122.10
      requirements:
        - link:
            node: CP1
```

ParentalControl.yaml

```
tosca_definitions_version: tosca_simple_profile_for_nfv_1_0_0
description: VNF-SFC
metadata:
  template_name: ParentalControl

topology_template:
  node_templates:
    VDU2:
      type: tosca.nodes.nfv.VDU.Tacker
      properties:
        name: ParentalControl
        image: Ubuntu-Image
        flavor: Little-Flavor
        key_name: ClaveSSH
        availability_zone: nova
        mgmt_driver: noop
        user_data_format: RAW
        user_data: |
          #!/bin/sh
          sudo chmod 777 /etc/resolv.conf
          sudo echo "nameserver 8.8.8.8" > /etc/resolvCP.conf
          while read line; do echo "$line">>resolvCP.conf; done <
/etc/resolv.conf
          sudo rm /etc/resolv.conf
          mv /etc/resolvCP.conf /etc/resolv.conf
          sudo apt-get --yes --force-yes install python
    CP2:
      type: tosca.nodes.nfv.CP.Tacker
      properties:
        type: vnic
        management: true
        anti_spoofing_protection: false
      requirements:
        - virtualLink:
            node: VL2
        - virtualBinding:
            node: VDU2
    VL2:
      type: tosca.nodes.nfv.VL
      properties:
        network_name: Red-Privada
        vendor: Tacker
    FIP2:
      type: tosca.nodes.network.FloatingIP
      properties:
        floating_network: public
        floating_ip_address: 192.168.122.11
      requirements:
        - link:
            node: CP2
```

VNF2.yaml

```
tosca_definitions_version: tosca_simple_profile_for_nfv_1_0_0
description: VNF-SFC
metadata:
  template_name: vnfd2
topology_template:
  node_templates:
    VDU3:
      type: tosca.nodes.nfv.VDU.Tacker
      properties:
        name: VNF-2
        image: Ubuntu-Image
        flavor: Little-Flavor
        key_name: ClaveSSH
        availability_zone: nova
        mgmt_driver: noop
        user_data_format: RAW
        user_data: |
          #!/bin/sh
          sudo chmod 777 /etc/resolv.conf
          sudo echo "nameserver 8.8.8.8" > /etc/resolvCP.conf
          while read line; do echo "$line">>resolvCP.conf; done <
/etc/resolv.conf
          sudo rm /etc/resolv.conf
          mv /etc/resolvCP.conf /etc/resolv.conf
          sudo apt-get --yes --force-yes install python
    CP3:
      type: tosca.nodes.nfv.CP.Tacker
      properties:
        type: vnic
        management: true
        anti_spoofing_protection: false
      requirements:
        - virtualLink:
            node: VL3
        - virtualBinding:
            node: VDU3
    VL3:
      type: tosca.nodes.nfv.VL
      properties:
        network_name: Red-Privada
        vendor: Tacker
    FIP3:
      type: tosca.nodes.network.FloatingIP
      properties:
        floating_network: public
        floating_ip_address: 192.168.122.12
      requirements:
        - link:
            node: CP3
```

VNF3.yaml

```
tosca_definitions_version: tosca_simple_profile_for_nfv_1_0_0
description: VNF-SFC
metadata:
  template_name: vnfd3

topology_template:
  node_templates:
    VDU4:
      type: tosca.nodes.nfv.VDU.Tacker
      properties:
        name: VNF-3
        image: Ubuntu-Image
        flavor: Little-Flavor
        key_name: ClaveSSH
        availability_zone: nova
        mgmt_driver: noop
        user_data_format: RAW
        user_data: |
          #!/bin/sh
          sudo chmod 777 /etc/resolv.conf
          sudo echo "nameserver 8.8.8.8" > /etc/resolvCP.conf
          while read line; do echo "$line">>resolvCP.conf; done <
/etc/resolv.conf
          sudo rm /etc/resolv.conf
          mv /etc/resolvCP.conf /etc/resolv.conf
          sudo apt-get --yes --force-yes install python
    CP4:
      type: tosca.nodes.nfv.CP.Tacker
      properties:
        type: vnic
        management: true
        anti_spoofing_protection: false
      requirements:
        - virtualLink:
            node: VL4
        - virtualBinding:
            node: VDU4
    VL4:
      type: tosca.nodes.nfv.VL
      properties:
        network_name: Red-Privada
        vendor: Tacker
    FIP4:
      type: tosca.nodes.network.FloatingIP
      properties:
        floating_network: public
        floating_ip_address: 192.168.122.13
      requirements:
        - link:
            node: CP4
```

Servidor.yaml:

```
tosca_definitions_version: tosca_simple_profile_for_nfv_1_0_0
description: Servidor-SFC
metadata:
  template_name: vnfd-servidor

topology_template:
  node_templates:
    VDU5:
      type: tosca.nodes.nfv.VDU.Tacker
      properties:
        name: Servidor
        image: Ubuntu-Image
        flavor: Little-Flavor
        key_name: ClaveSSH
        availability_zone: nova
        mgmt_driver: noop
        user_data_format: RAW
        user_data: |
          #!/bin/sh
          sudo chmod 777 /etc/resolv.conf
          sudo echo "nameserver 8.8.8.8" > /etc/resolvCP.conf
          while read line; do echo "$line">>resolvCP.conf; done <
/etc/resolv.conf
          sudo rm /etc/resolv.conf
          mv /etc/resolvCP.conf /etc/resolv.conf
          sudo apt-get --yes --force-yes install python
          sudo python -m SimpleHTTPServer 80 > /dev/null 2>&1 &
    CP5:
      type: tosca.nodes.nfv.CP.Tacker
      properties:
        type: vnic
        management: true
        anti_spoofing_protection: false
      requirements:
        - virtualLink:
            node: VL5
        - virtualBinding:
            node: VDU5
    VL5:
      type: tosca.nodes.nfv.VL
      properties:
        network_name: Red-Privada
        vendor: Tacker
    FIP5:
      type: tosca.nodes.network.FloatingIP
      properties:
        floating_network: public
        floating_ip_address: 192.168.122.14
      requirements:
        - link:
            node: CP5
```

Una vez definidas las plantillas TOSCA, se procede a registrar los VNFD con las plantillas creadas.

```
tacker vnfd-create –vnfd-file Cliente.yaml vnfd-Cliente
```

```
tacker vnfd-create –vnfd-file ParentalControl.yaml vnfd-PC

tacker vnfd-create –vnfd-file VNF2.yaml vnfd-VNF2

tacker vnfd-create –vnfd-file VNF3.yaml vnfd-VNF3

tacker vnfd-create –vnfd-file Servidor.yaml vnfd-Servidor
```

Con los VNFD registrados, se crean las VNF a partir de los VNFD.

```
tacker vnf-create –vnfd-name vnfd-Cliente vnf-Cliente

tacker vnf-create –vnfd-name vnfd-PC vnf-PC

tacker vnf-create –vnfd-name vnfd-VNF2 vnf-VNF2

tacker vnf-create –vnfd-name vnfd-VNF3 vnf-VNF3

tacker vnf-create –vnfd-name vnfd-Servidor vnf-Servidor
```

Tras estas operaciones ya tenemos el escenario desplegado, a continuación, se añade el funcionamiento SFC.

Lo primero se crean las plantillas TOSCA que definen la clasificación y encaminamiento de los flujos. Se van a crear dos, una para tráfico TCP y otra para trafico UDP.

vnffgd-SFC-HTTP.yaml

```
ca_definitions_version: tosca_simple_profile_for_nfv_1_0_0

description: Vnffgd para SFC

topology_template:

  node_templates:

    Forwarding_path1:
      type: tosca.nodes.nfv.FP.TackerV2
      description: creates path (CP1->CP2)
      properties:
        id: 51
        policy:
          type: ACL
          criteria:
            - name: HttpControl
              classifier:
                network_src_port_id: 1dc1eaa7-12c9-437a-ad51-
37b527d7dd41
                destination_port_range: 80-80
                ip_proto: 6
        path:
          - forwarder: vnfd-PControl
            capability: CP2

  groups:
    VNFFG1:
      type: tosca.groups.nfv.VNFFG
      description: HTTP to Corporate Net
```

```
      properties:
        vendor: tacker
        version: 1.0
        number_of_endpoints: 1
        dependent_virtual_link: [VL2]
        connection_point: [CP2]
        constituent_vnfs: [vnfd-PControl]
      members: [Forwarding_path1]
```

vnffgd-SFC-UDP.yaml

```
tosca_definitions_version: tosca_simple_profile_for_nfv_1_0_0

description: Vnffgd para SFC

topology_template:

  node_templates:

    Forwarding_path3:
      type: tosca.nodes.nfv.FP.TackerV2
      description: creates path (CP1->CP5)
      properties:
        id: 53
        policy:
          type: ACL
          criteria:
            - name: manage_udp
              classifier:
                network_src_port_id: 1dc1eaa7-12c9-437a-ad51-
37b527d7dd41
                ip_proto: 17
        path:
          - forwarder: vnfd-VNF2
            capability: CP3
          - forwarder: vnfd-VNF3
            capability: CP4

  groups:
    VNFFG3:
      type: tosca.groups.nfv.VNFFG
      description: UDP to Corporate Net
      properties:
        vendor: tacker
        version: 1.0
        number_of_endpoints: 2
        dependent_virtual_link: [VL3, VL4]
        connection_point: [CP3,CP4]
        constituent_vnfs: [vnfd-VNF2,vnfd-VNF3]
      members: [Forwarding_path3]
```