1 Design the NFV solution

Each experiment can be seen as a description of a network service and a number of instance of this description. In the following sections and subsections, we will refer to the first as Network Service Descriptor and to the seconds as Network Service Records, following in this way the ETSI NFV specification v1.1.1 [1] naming conventions.

1.1 VNF design

Before going directly to implement the code of the network service, is important to understand how the combination of network functions will work not only in a cloud, but also in a NFV environment.

This improvement of the environment changes radically also how the classic network functions need to be implemented. In addition, the SoftFIRE infrastructure abstracts even more the management and orchestration capabilities, offering on the top SFA FIRE APIs [2]. The enablement of these APIs allows the experimenter to use some kind of tool, such as jFed [3], that helps to delineate the experiment by composing the involved nodes (network functions), defining in that way the experiment (Network Service Descriptor).

Hence, to define a Network Service Descriptor, the experimenter has to define each network functions, that are now Virtual Network Functions. In particular, the network function has to be installed, configured and started automatically by the SoftFIRE platform, thus the experimenter needs to implement such lifecycle management procedures, that will be invoked by the SoftFIRE MANO layer.

Alongside these lifecycle procedure, there are other resources to conceive: the description of the VNF, the description of the VNF package and of course the Virtual Network Function (VNF) core code. The VNF descriptor, the VNF Package descriptor and the lifecycle management procedures will create the VNF Package itself that will be uploaded in the Softfire Software Portal (SSP) like the VNF image.

The VNF image is supposed to contain the VNF code, dependencies included. The VNF code can be public or not, but in both cases it is suggested to prepare an image starting from one of the available ones that can be found in the Redmine wiki page containing it. This process will be explained in details in the following sections.

To summarize, the VNF is described by the VNF Package and the experimenter, in order to be able to use a VNF in its experiment, has to upload the VNF Package and related image in the SoftFIRE platform. The VNF Package is composed by:

- VNF descriptor (VNFD)
- Package descriptor (Metadata.yaml)
- Lifecycle management procedures (scripts)

The VNF Descriptor is represented with the Open Baton [4] model [5] [6]. Examples can be also found in the Redmine wiki page. The following tables explains the fields that compose the VNFD and that has to be created by the experimenter.

Table 1. VNF Descriptor

Field name	Description	Type
name	The name of the VNF. Unique in a NSD	String

vendor	The provider of the VNF	String
version	The version of the VNF	String
type	The VNF type. This field is relevant for the dependencies configurations. Must be unique and cannot contain special chars only letter and digits	String
endpoint	The VNFM that will manage the VNF. Must be "generic"	String
vdu	The NFV requirement of the VNF. See Table 2	List
configurations	The configuration specific for this VNF containg a name and a list of ConfigurationParameters composed by key, name	Json object
virtual_link	The network to use. If not specified differently by the SoftFIRE use { "name":"softfire-internal"}	Json object
deployment_flavour	The flavor key[1] to use: possibilities are: • m1.tiny • m1.small • m1.medium	Json object
lifecycle_event	The lifecycle management procedures. See Table 3	List
requires	The logical requirement that the VNF has. This is explained deeper later on.	Json object

Table 2. Virtual Deployment Unit

Field name	description	Type
name	The vdu name	String
vm_image	The list of image names to use. This name will be available once the VNF image is uploaded to the SSP	List
scale_in_out	Number of maximum VNF component. One is minimum	Int
vnfc	The VNF Components that are reflected as VMs. It contains the list of Connection Points that are the network that this component must be connected to. Use { "floatingIp":"random",	List

[1] Please consider that the physical infrastructure is shared among experimenters. If the experimenter has any specific requirements, please contact the SoftFIRE Team

	"virtual_link_reference":"softfire-internal" }[2]	
vimInstanceName	A list of Point of Presence. If more than one is provided the NFVO will chose randomly. Please specifiy the PoP where deploy the VM in between: • vim-instance-fokus, in order to deploy in Fokus testbed • vim-instance-er, in order to deploy in Ericsson testbed • vim-instance-uos, in order to deploy in Surrey testbed • vim-instance-ti, in order to deploy in Telecom Italia testbed.	list

Table 3. Lifecycle Event

Field name	description	type
event	The lifecycle event code in which these procedures must be executed: • INSTANTIATE • CONFIGURE • START	String
lifecycle_events	List of scripts names (procedures) that has to be called during this lifecycle event	List

This describe the VNF that the experimenter wants to upload.

A network service has usually more than one VNF and more than one dependencies. The way the dependencies are defined is through the **requires** field in the VNFD. This field looks like the following:

The parameter names could be:

```
"requires":{
        "type_from_which_the_vnf_requires":{
            "parameters":[ "list of parameter names that this VNF requires" ]
        }
}
```

- The network name (virtual link) for obtaining the private ip

[2] If the experimenter has any specific requirements, should contact the SoftFIRE Team

- The network name (virtual link) followed by "_floatingIp" for the public ip
- Configuration parameter keys

These parameters are automatically injected as environment variables in the lifecycle script. The dependencies scripts **must** start with the foreign type followed by underscore ("foreigntype_"), if not the script will not be executed.

More detailed information can be found here [7]

As stated above, the VNFs are connected each others through dependencies. If some dependencies are not satisfied, the onboard of the NS Descriptor is not allowed and an error will be thrown.

The SoftFIRE infrastructure brings to the VNF developer, the experimenter, some more thoughts. The VNFs will be on the same datacenter or will they be spread over multiple? Which VMs I really need to reach from within the SoftFIRE VPN?

If all the VNF will be placed in the same datacenter, then the experimenter needs "public" IPs only for the virtual resources to which he wants to be able to log in. This can be achieved by specifying the "floatingIp" fields in the VNF Descriptor (see Table 2). The floating IP to choose is always "random"[3].

To conclude the VNF Package we need to describe the package itself using a Metadata.yaml file as follows:

name: the name of the package
description*: "a text description of the package"*
provider*: the provider*
shared*: false*
image*:*
  upload*: false*
  names*:*
    - The name of the image. This is known only after uploading he image to the SSP
vim_types*:*
  - openstack

In order to create a package, all these resources have to be packed in a .tar file together with the scripts representing the lifecycle events in a folder called scripts [8].

The VNF must run on top of a Virtual Machine that is deployed based on an image. Already provided images can be used or it is also possible to create a specific one and upload it in the SoftFIRE infrastructure.

1.1.1.1.1. Image design

A VNF usually have dependencies. While most of the datacenters provides internet connectivity inside the VM, some not, thus in some cases it may be necessary to prepare an image in advance containing the preinstalled dependencies (and binary code of the VNF). This also help the deployment phase because it accelerates the installation (INSTANTIATE, see Table 3) since all the code is already installed and does not need to be downloaded.

The SoftFIRE platform already provides some preconfigured images. In case the VNF does not have special requirements, it is then possible to use one of the following images just by putting the name in the VNFD→VDU (see Table 2).

[3] There are special cases where this is not valid, for instance in case of necessity of usage of RAN. In this case contact the SoftFIRE Team for instructions.

Name	Description
Ubuntu 14.04 Cloud based	This image contains the ubuntu 14.04 OS without any further dependencies
ob-ems-mongo-iperf	This image contains the EMS agent and some preinstalled tools such as mongodb and iperf
OpenIMSCore_preConf_ems-16	This image contains the preinstalled code of OpenIMS.

In case the VNF needs a particular software installed or there is the need to reduce the deployment phase time, starting from one of this images is possible to create a new one.

The process is the following:

1. Download the image called “Ubuntu 14.04 Cloud based”
2. Upload it to a local openstack [9].
3. Launch a VM using this image
4. Log in to this image using Secure Shell (SSH) and the private key selected while launching
5. Install all the needed software
6. Then log out
7. Shutdown the instance
8. Create a snapshot and download it choosing a unique name

1.1.2. Upload experiment resources

The snapshot created can be directly uploaded to the SoftFIRE Software Portal (SSP). For accessing the SSP, the same credentials of the SoftFIRE Web Portal are valid.

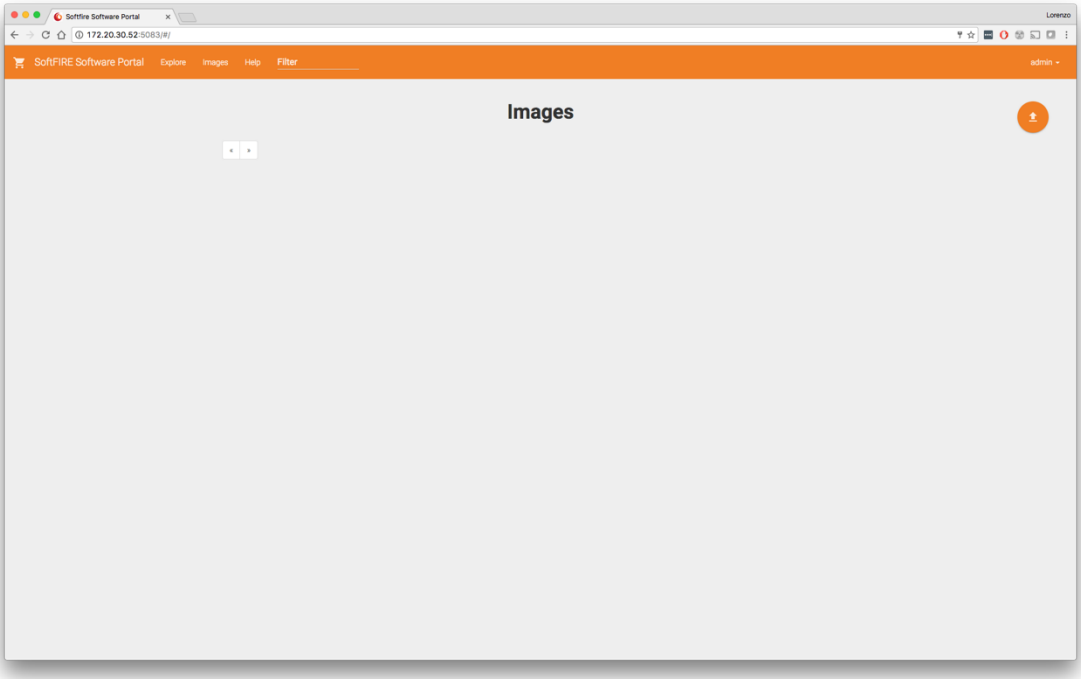

Figure 1. SSP upload image

When uploaded, the image will be then transferred to the datacenters and after some minutes it will be available for being used by a VNF. Only now it is possible to upload the VNF Package in the SSP, specifying the image name just uploaded in the VNF Package Metadata.yaml.

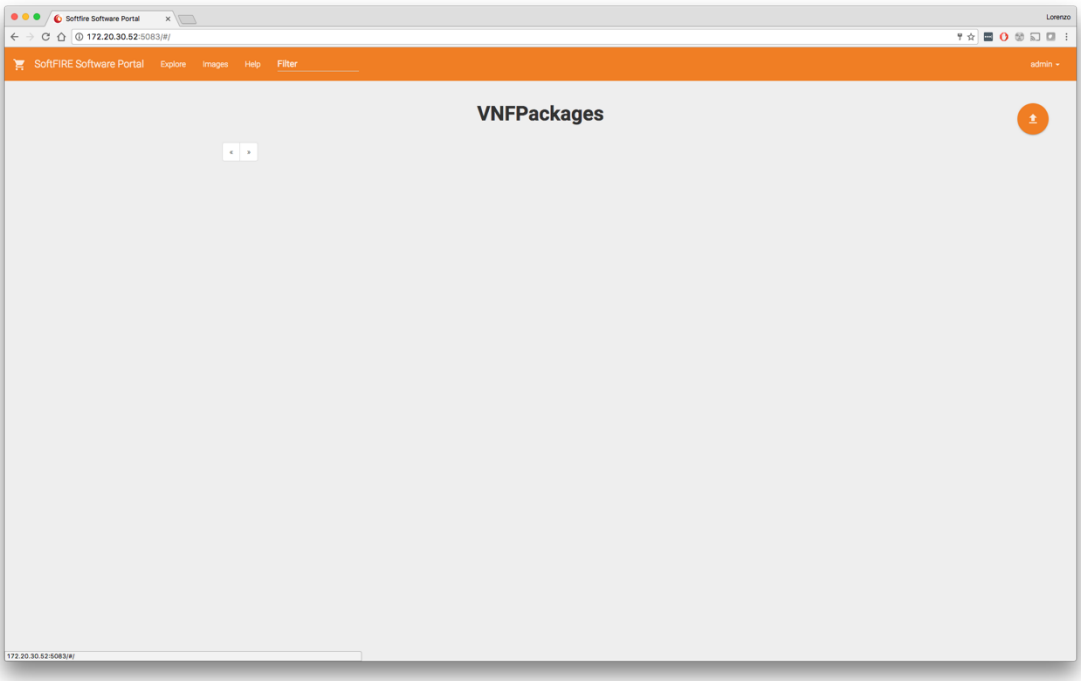

Figure 2. SSP upload VNF Package

2 References

[1] ETSI, «ETSI GS NFV-MAN 001,» etsi, 1 12 2014. [En línea]. Available: http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf.

[2] GENI, «SFA,» [En línea]. Available: http://groups.geni.net/geni/wiki/SliceFedArch.

[3] iMinds, [En línea]. Available: http://jfed.iminds.be/.

[4] Open Baton, «openbaton,» [En línea]. Available: http://openbaton.github.io/.

[5] Open Baton, «Open Baton Descriptor Model,» [En línea]. Available: http://openbaton.github.io/documentation/ns-descriptor/.

[6] Open Baton, «Open Baton Virtual Network Function Descriptor,» [En línea]. Available: http://openbaton.github.io/documentation/vnf-descriptor/.

[7] Open Baton, «Open Baton dependency management between VNF,» [En línea]. Available: http://openbaton.github.io/documentation/vnfm-generic/.

[8] Open Baton, «Open Baton VNF Package how to,» [En línea]. Available: http://openbaton.github.io/documentation/vnf-package/.

[9] OpenStack, «Open source software for creating private and public clouds.,» [En línea]. Available: https://www.openstack.org/.