OPNFV FUNCTEST configuration/installation guide

Release arno.2015.1.0 (fe179a4)

OPNFV

February 24, 2016

CONTENTS

CHAPTER

ONE

INTRODUCTION

** DOCUMENT IS IN PROGRESS FOR BRAHMAPUTRA **

This document describes how to install and configure Functest in OPNFV.

CHAPTER

TWO

PREREQUISITES

The OPNFV deployment is out of the scope of this document but it can be found in XXX. The OPNFV platform is considered as the System Under Test (SUT) in this document.

Several prerequisites are needed for functest:

1. A Jumphost to run Functest on
2. Docker daemon shall be installed on the Jumphost
3. A public/external network created on the SUT
4. Connectivity from the Jumphost to the SUT public/external network
5. Connectivity from the Jumphost to the SUT management network

NOTE: "Jumphost" refers to any server which meets the previous requirements. Normally it is the same server from where the OPNFV deployment has been triggered.

2.1 Docker installation

Log on your jumphost then install docker (e.g. for Ubuntu):

```
curl -sSL https://get.docker.com/ | sh
```

Add your user to docker group to be able to run commands without sudo:

```
sudo usermod -aG docker <your_user>
```

References:

- Ubuntu
- RHEL

2.2 Connectivity to OPNFV management network

Some of the Functest tools need to have access to the OpenStack management network of the controllers [1].

For this reason, an interface shall be configured in the Jumphost in the OpenStack management network range.

Example:

```
The OPNFV Fuel installation uses VLAN tagged 300 and subnet 192.168.1.0/24 as
Openstack Management network.
.
Supposing that eth1 is the physical interface with access to that subnet:
$ ip link add name eth1.300 link eth1 type vlan id 300
$ ip link set eth1.300 up
$ ip addr add 192.168.1.66/24 dev eth1.300
```

2.3 External network on SUT

Some of the tests against the VIM (Virtual Infrastructure Manager) need an existing public network to succeed. This is needed, for example, to create floating IPs to access instances from the public network (i.e. Jumphost).

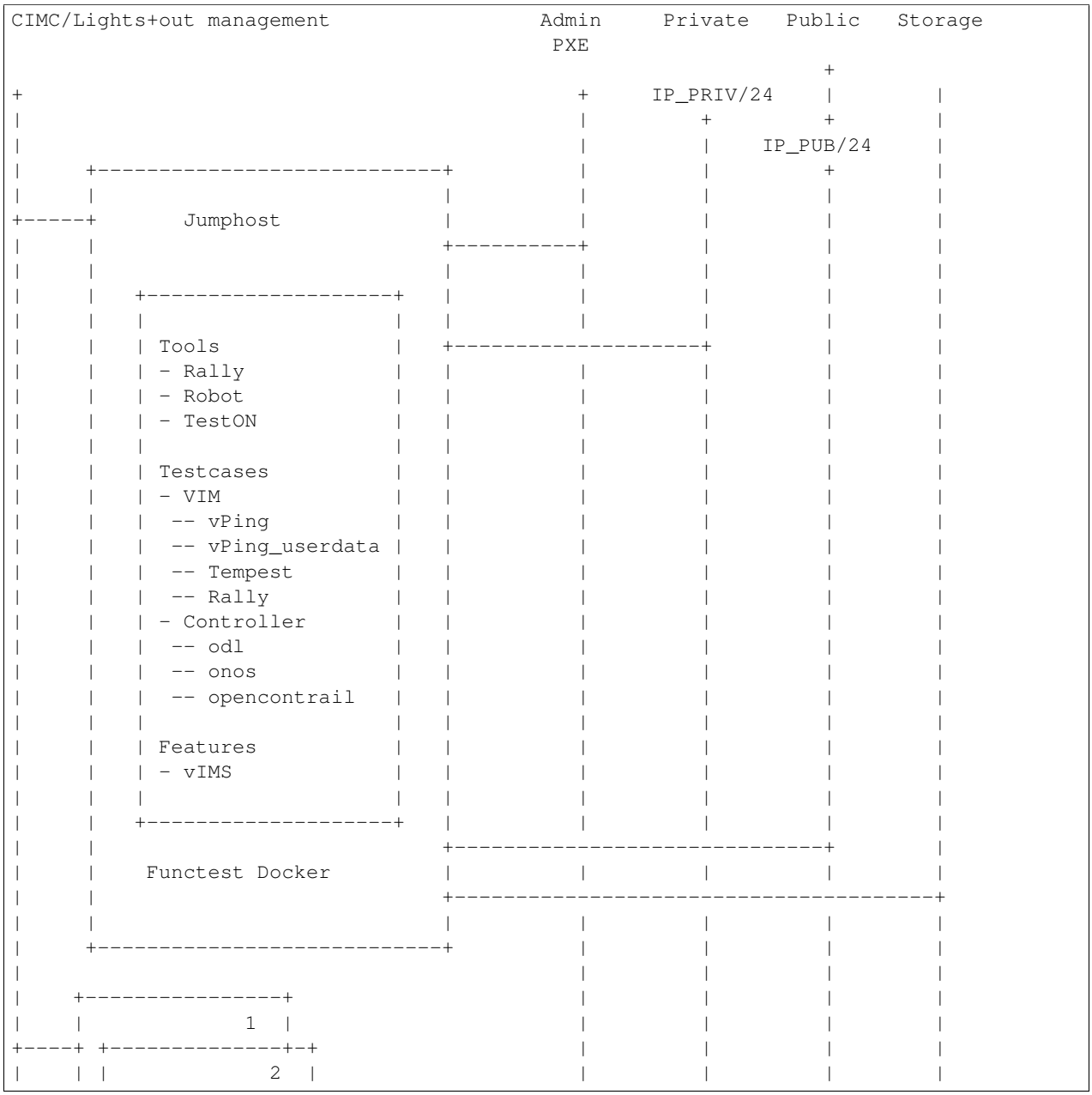
By default, any of the four OPNFV installers provide a fresh installation with an external network created along with a router.

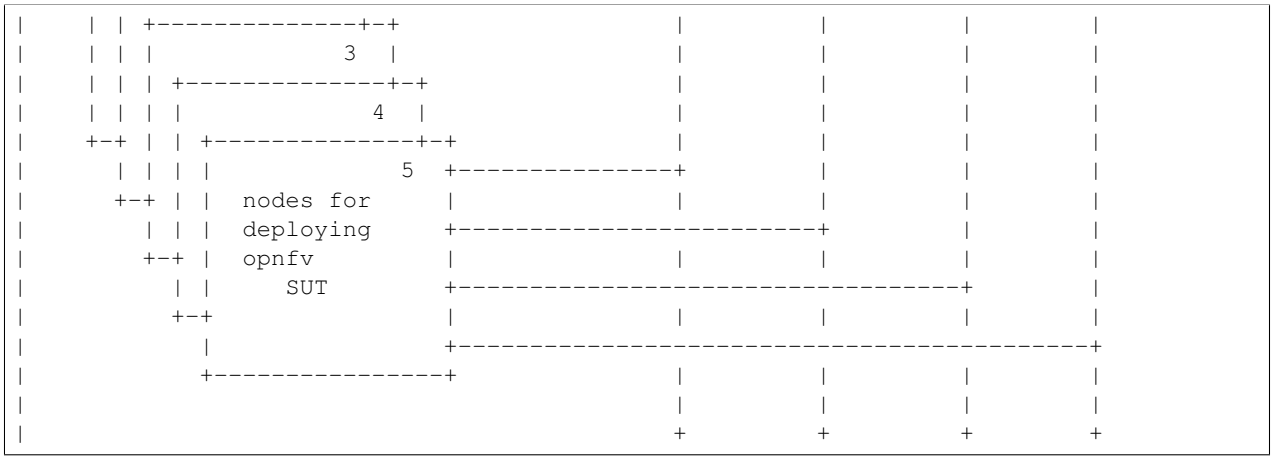
CHAPTER

THREE

HIGH LEVEL ARCHITECTURE

The high level architecture of Functest within OPNFV can be described as follow:





All the libraries and dependencies needed by all the Functest tools are pre-installed in the Docker image. This allows running Functest on any platform with any Operating System.

The Docker image will:

- retrieve OpenStack credentials
- prepare the environment according to the SUT
- perform the appropriate tests
- push the results into the OPNFV test result database

This Docker image can be integrated into CI or deployed **independently** of the CI. A description of the Brahmaputra testcases can be retrieved in the Functest user guide [2]

Please note that the Functest container has been designed for OPNFV, however, it would be possible to adapt it to any VIM+controller environment as most of the testcases are upstream testcases.

CHAPTER

FOUR

FUNCTIONAL TESTING INSTALLATION

Pull the Functest Docker image from the Docker hub:

```
$ docker pull opnfv/functest:brahmaputra.1.0
```

Check that the image is available:

```
$ docker images
```

Run the docker container giving the environment variables:

```
- INSTALLER_TYPE. Possible values are "apex", "compass", "fuel" or "joid".
- INSTALLER_IP. each installer has its installation strategy.
```

Functest may need to know the IP of the installer to retrieve the credentials (e.g. usually “10.20.0.2” for fuel, not neede for joid...).

The minimum command to create the Functest docker file can be described as follows:

```
docker run -it -e "INSTALLER_IP=10.20.0.2" -e "INSTALLER_TYPE=fuel" opnfv/functest:brahmaputra.1.0 /b
```

Optionally, it is possible to precise the container name through the option –name:

```
docker run --name "CONTAINER_NAME" -it -e "INSTALLER_IP=10.20.0.2" -e "INSTALLER_TYPE=fuel" opnfv/fun
```

It is also possible to to indicate the path of the OpenStack creds using -v:

```
docker run  -it -e "INSTALLER_IP=10.20.0.2" -e "INSTALLER_TYPE=fuel" -v <path_to_your_local_creds_fil
```

The local file will be mounted in the container under /home/opnfv/functest/conf/openstack.creds

After the run command the prompt appears which means that we are inside the container and ready to run Functest.

Inside the container, the following directory structure should be in place:

```
`-- home
    `-- opnfv
      |-- functest
      |   |-- conf
      |   |-- data
      |   `-- results
      `-- repos
          |-- bgpvpn
          |-- functest
          |-- odl_integration
          |-- rally
          |-- releng
          `-- vims-test
```

Basically the container includes:

- Functest directory to store the configuration (the OpenStack creds are paste in /home/opngb/functest/conf), the data (images neede for test for offline testing), results (some temporary artifacts may be stored here)
- Repositories: the functest repository will be used to prepare the environment, run the tests. Other repositories are used for the installation of the tooling (e.g. rally) and/or the retrieval of feature projects scenarios (e.g. bgpvpn)

The arborescence under the functest repo can be described as follow:

```
.
  |-- INFO
  |-- LICENSE
  |-- commons
  |   |-- ims
  |   |-- mobile
  |   `-- traffic-profile-guidelines.rst
  |-- docker
  |   |-- Dockerfile
  |   |-- common.sh
  |   |-- prepare_env.sh
  |   |-- requirements.pip
  |   `-- run_tests.sh
  |-- docs
  |   |-- configguide
  |   |-- functest.rst
  |   |-- images
  |   `-- userguide
  `-- testcases
      |-- Controllers
      |-- VIM
      |-- __init__.py
      |-- config_functest.py
      |-- config_functest.yaml
      |-- functest_utils.py
      |-- functest_utils.pyc
      |-- vIMS
      `-- vPing
```

We may distinguish 4 different folders:

- commons: it is a folder dedicated to store traffic profile or any test inputs that could be reused by any test project
- docker: this folder includes the scripts that will be used to setup the environment and run the tests
- docs: this folder includes the user and installation/configuration guide
- testcases: this folder includes the scripts required by Functest internal test cases

Firstly run the script to install functest environment:

```
$ ${repos_dir}/functest/docker/prepare_env.sh
```

NOTE: ${repos_dir} is a default environment variable inside the docker container, which points to /home/opnfv/repos

Run the script to start the tests:

```
$ ${repos_dir}/functest/docker/run_tests.sh
```

NOTE: This will run ALL the tests by default, see [2] for details

4.1 Focus on the OpenStack credentials

The OpenStack credentials are needed to test the VIM. There are 3 ways to provide them to Functest:

- using the -v option when running the Docker container
- create an empty file in /home/opnfv/functest/conf/openstack.creds and paste the needed info in it.
- **automatically retrieved using the following script::** $repos_dir/releng/utils/fetch_os_creds.sh

Once the credentials are there, they shall be sourced before running the tests:

```
source /home/opnfv/functest/conf/openstack.creds
```

4.2 Additional Options

In case you need to provide different configuration parameters to Functest (e.g. commit IDs or branches for the repositories, ...) copy the config_functest.yaml from the repository to your current directory and run docker with a volume:

```
$ wget https://git.opnfv.org/cgit/functest/plain/testcases/config_functest.yaml
$ cmd1 = "/home/opnfv/repos/functest/docker/prepare_env.sh"
$ cmd2 = "/home/opnfv/repos/functest/docker/run_tests.sh"
$ docker run -t -e "INSTALLER_TYPE=fuel" -e "INSTALLER_IP=10.20.0.2" opnfv/functest \
    -v $(pwd)/config_functest.yaml:/home/opnfv/functest/conf/config_functest.yaml "${cmd1} && ${cmd2}"
```

CHAPTER

FIVE

INTEGRATION IN CI

In CI we use the docker file and execute commande within the container from Jenkins.

Docker creation in set-functest-env builder [3]:

```
envs="INSTALLER_TYPE=${INSTALLER_TYPE} -e INSTALLER_IP=${INSTALLER_IP} -e NODE_NAME=${NODE_NAME}"
[...]
docker pull opnfv/functest:latest_stable
cmd="docker run -id -e $envs ${labconfig} ${sshkey} ${res_volume} opnfv/functest:latest_stable /bin/b
echo "Functest: Running docker run command: ${cmd}"
${cmd}
docker ps -a
sleep 5
container_id=$(docker ps | grep 'opnfv/functest:latest_stable' | awk '{print $1}' | head -1)
echo "Container ID=${container_id}"
if [ -z ${container_id} ]; then
    echo "Cannot find opnfv/functest container ID ${container_id}. Please check if it is existing."
    docker ps -a
    exit 1
fi
echo "Starting the container: docker start ${container_id}"
docker start ${container_id}
sleep 5
docker ps
if [ $(docker ps | grep 'opnfv/functest:latest_stable' | wc -l) == 0 ]; then
    echo "The container opnfv/functest with ID=${container_id} has not been properly started. Exiting
    exit 1
fi
cmd="${FUNCTEST_REPO_DIR}/docker/prepare_env.sh"
echo "Executing command inside the docker: ${cmd}"
docker exec ${container_id} ${cmd}
```

Test execution in functest-all builder [3]:

```
echo "Functest: run $FUNCTEST_SUITE_NAME"
cmd="${FUNCTEST_REPO_DIR}/docker/run_tests.sh --test $FUNCTEST_SUITE_NAME ${flag}"
container_id=$(docker ps -a | grep opnfv/functest | awk '{print $1}' | head -1)
docker exec $container_id $cmd
```

Docker clean in functest-cleanup builder [3]:

```
echo "Cleaning up docker containers/images..."
# Remove previous running containers if exist
if [[ ! -z $(docker ps -a | grep opnfv/functest) ]]; then
echo "Removing existing opnfv/functest containers..."
docker ps | grep opnfv/functest | awk '{print $1}' | xargs docker stop
docker ps -a | grep opnfv/functest | awk '{print $1}' | xargs docker rm
```

```
fi

# Remove existing images if exist
if [[ ! -z $(docker images | grep opnfv/functest) ]]; then
echo "Docker images to remove:"
docker images | head -1 && docker images | grep opnfv/functest
image_tags=($(docker images | grep opnfv/functest | awk '{print $2}'))
for tag in "${image_tags[@]}"; do
    echo "Removing docker image opnfv/functest:$tag..."
    docker rmi opnfv/functest:$tag
done
fi
```

CHAPTER

SIX

CONFIGURATION

Everything is preconfigured in the docker file. It is however possible to customize the list of tests, see *[2]* for details.

CHAPTER

SEVEN

ERRORS

CHAPTER

EIGHT

REFERENCES

OPNFV main site: opnfvmain.

OPNFV functional test page: opnfvfunctest.

IRC support channel: #opnfv-testperf