OPNFV FUNCTEST Configuration Guide

Release arno.2015.1.0 (9c3f0aa)

OPNFV

April 11, 2016

CONTENTS

CHAPTER

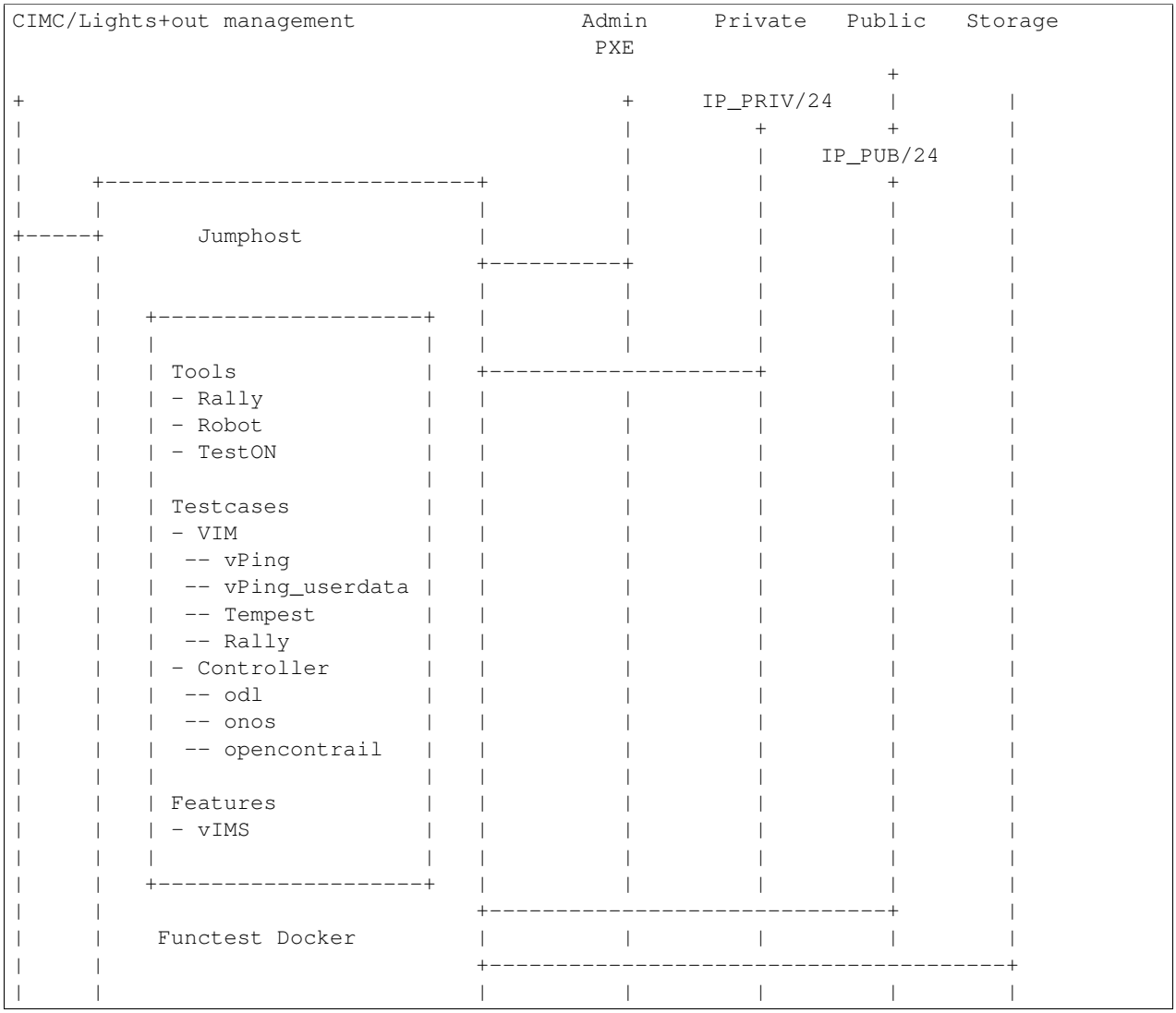
ONE

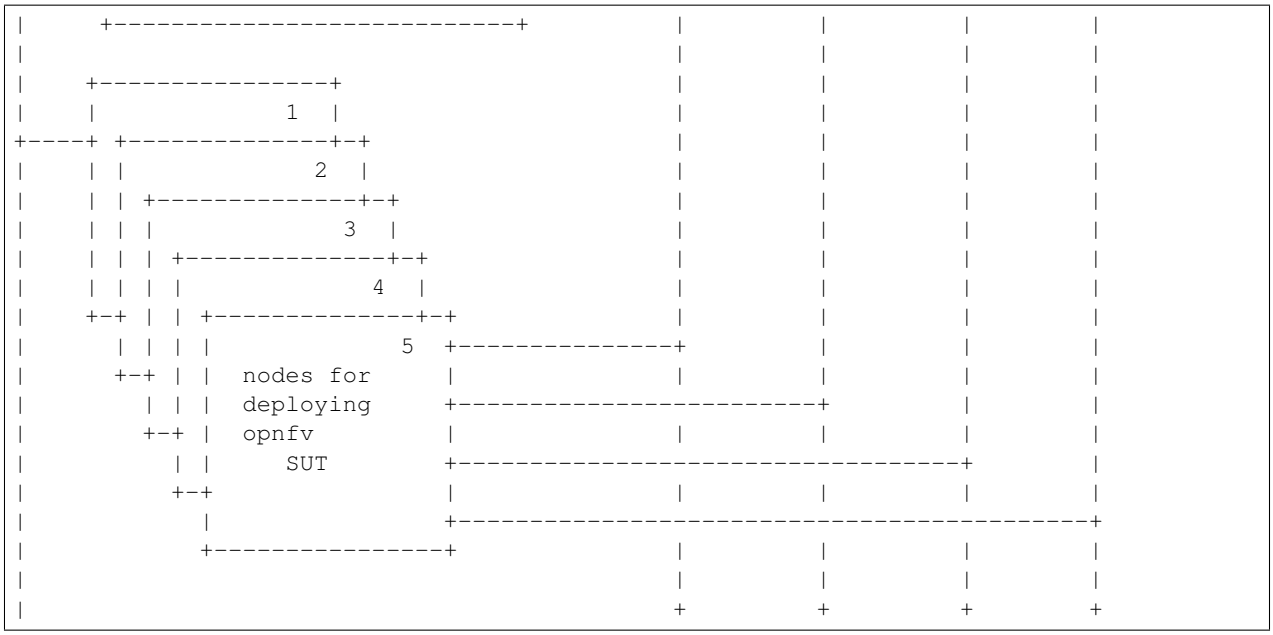
INTRODUCTION

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

1.1 High level architecture

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





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 on any Operating System.

The automated mechanisms inside the Functest Docker container 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.

Please note that the Functest container has been designed for OPNFV, however, it would be possible to adapt it to any VIM+controller environment since most of the test cases are integrated from upstream communities.

The test cases are described in the Functest User Guide [2]

CHAPTER

TWO

PREREQUISITES

The OPNFV deployment is out of the scope of this document but it can be found in [4]. 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 previously.

2.1 Docker installation

Log on your Jumphost and 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 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. Docker container).

By default, the four OPNFV installers provide a fresh installation with an external network created along with a router. Make sure that subnet is reachable from the Jumphost

2.3 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.

For example, if the management network is on VLAN 300 and subnet 192.168.1.0/24 and assuming 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
```

This is just an example about how to configure an interface with vlan, but it might differ depending on the deployment settings on each installer. Check the corresponding installer instructions for more info.

CHAPTER

THREE

INSTALLATION AND CONFIGURATION

3.1 Preparing the Docker container

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** : IP of the installer node/VM.

Functest may need to know the IP of the installer to retrieve automatically the credentials from the installer node/VM or even from the actual controllers.

The minimum command to create the Functest Docker container can be described as follows:

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

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 credentials 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*

If the intention is to run Functest against any of the supported OPNFV scenarios, it is recommended to include also the environment variable **DEPLOY_SCENARIO**, for example:

```
docker run -it -e "INSTALLER_IP=10.20.0.2" -e "INSTALLER_TYPE=fuel" -e "DEPLOY_SCENARIO=os-odl_l2-no
```

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

```
`-- home
    `-- opnfv
      |-- functest
      |   |-- conf
      |   |-- data
      |   `-- results
      `-- repos
```

```
        |-- bgpvpn
        |-- doctor
        |-- functest
        |-- odl_integration
        |-- onos
        |-- promise
        |-- rally
        |-- releng
        `-- vims-test
```

Basically the container includes:

- Functest directory to store the configuration (the OpenStack creds are stored in /home/opngb/functest/conf/openstack.creds), the data (cirros image needed for some tests), results (some temporary result logs may be stored here)
- Repositories: the functest repository will be used to prepare the environment and run the tests. Other repositories are used for the installation of the needed tooling (e.g. rally) and/or the retrieval of feature projects scenarios (e.g. promise)

The structure under the Functest repository can be described as follows:

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

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 and other feature projects test cases.

After the *run* command, a new prompt appears which means that we are inside the container and ready to move to the next step.

3.2 Useful Docker commands

When typing **exit** in the container prompt, this will cause exiting the container and probably stopping it. When stopping a running Docker container all the changes will be lost, there is a keyboard shortcut to quit the container without stopping it: CTRL+P+Q. To reconnect to the running container **DO NOT** use the *run* command again (since it will create a new container), use *exec* instead:

```
docker ps
<copy the container ID>
docker exec -ti <CONTAINER_ID> /bin/bash
```

or simply:

```
docker exec -ti $(docker ps|grep functest|awk '{print $1}') /bin/bash
```

There are other useful Docker commands that might be needed to manage possible issues with the containers.

List the running containers:

```
docker ps
```

List all the containers including the stopped ones:

```
docker ps -a
```

It is useful sometimes to remove a container if there are some problems:

```
docker rm <CONTAINER_ID>
```

Use the *-f* option if the container is still running, it will force to destroy it:

```
docker -f rm <CONTAINER_ID>
```

The Docker image is called **opnfv/functest** and it is stored in the public Docker registry under the OPNFV account: dockerhub. The are many different tags that have been created automatically by the CI mechanisms, but the one that this document refers to is **brahmaputra.1.0**. Pulling other tags might cause some problems while running the tests.

Check the Docker documentation dockerdocs for more information.

3.3 Preparing the Functest environment

Once the docker container is up and running, execute the following command in the prompt:

```
${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/*

This script will make sure that the requirements to run the tests are met and will install the needed libraries and tools by all Functest test cases. It must be run only once every time the docker is started from sratch.

3.4 Focus on the OpenStack credentials

The OpenStack credentials are needed to run the tests against 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 credentials 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
```

or simply using the environment variable **creds**:

```
. $creds
```

After this, try to run any OpenStack command to see if you get any output, for instance:

```
openstack user list
```

This will return a list of the actual users in the OpenStack deployment. In any other case, check that the credentials are sourced:

```
env|grep OS_
```

This command must show a set of environment variables starting with *OS_*, for example:

```
OS_REGION_NAME=RegionOne
OS_DEFAULT_DOMAIN=default
OS_PROJECT_NAME=admin
OS_PASSWORD=admin
OS_AUTH_STRATEGY=keystone
OS_AUTH_URL=http://172.30.10.3:5000/v2.0
OS_USERNAME=admin
OS_TENANT_NAME=admin
OS_ENDPOINT_TYPE=internalURL
OS_NO_CACHE=true
```

If still the OpenStack command does not show anything or complains about connectivity issues, it could be due to an incorrect url given to the OS_AUTH_URL environment variable. Check the deployment settings.

3.5 SSL Support

If you need to connect to a server that is TLS-enabled (the auth URL begins with 'https') and it uses a certificate from a private CA or a self-signed certificate you will need to specify the path to an appropriate CA certificate to use to validate the server certificate with the environment variable OS_CACERT:

```
echo $OS_CACERT
/etc/ssl/certs/ca.crt
```

However, this certificate does not exist in the container by default. It has to be copied manually from the OpenStack deployment. This can be done in 2 ways:

1. Create manually that file and copy the contents from the OpenStack controller.
2. (recommended) Add the file using a Docker volume when starting the container:

```
    -v <path_to_your_cert_file>:/etc/ssl/certs/ca.cert
```

You might need to export OS_CACERT environment variable inside the container:

```
export OS_CACERT=/etc/ssl/certs/ca.crt
```

Certificate verification can be turned off using OS_INSECURE=true. For example, Fuel uses self-signed cacerts by default, so an pre step would be:

```
export OS_INSECURE=true
```

3.6 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 the container with a volume:

```
wget https://git.opnfv.org/cgit/functest/plain/testcases/config_functest.yaml

<modify the file accordingly>

docker run -ti -e \
"INSTALLER_TYPE=fuel" -e "INSTALLER_IP=10.20.0.2" \
opnfv/functest:brahmaputra.1.0 \
-v $(pwd)/config_functest.yaml:/home/opnfv/functest/conf/config_functest.yaml \
/bin/bash\
```

However, this is not recommended since most of the test cases rely on static parameters read from this file, and changing them might cause problems.

3.7 Proxy support

Functest needs internet access to download some resources for some test cases. For example to install the Rally environment. This might not work properly if the Jumphost is running through a Proxy.

If that is the case, make sure the resolv.conf and the needed proxy environment variables are properly set:

```
export http_proxy=<your http proxy settings>
export https_proxy=<your https proxy settings>
```

Or refer to the official Docker documentation for Proxy settings.

Before running **prepare_env.sh** make sure you can ping http and https sites inside the container. For example:

```
nc -v google.com 80
Connection to google.com 80 port [tcp/http] succeeded!

nc -v google.com 443
Connection to google.com 443 port [tcp/https] succeeded!
```

CHAPTER

FOUR

INTEGRATION IN CI

In CI we use the Docker image and execute the appropriate commands 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

FIVE

REFERENCES

OPNFV main site: opnfvmain.

OPNFV functional test page: opnfvfunctest.

IRC support channel: #opnfv-testperf