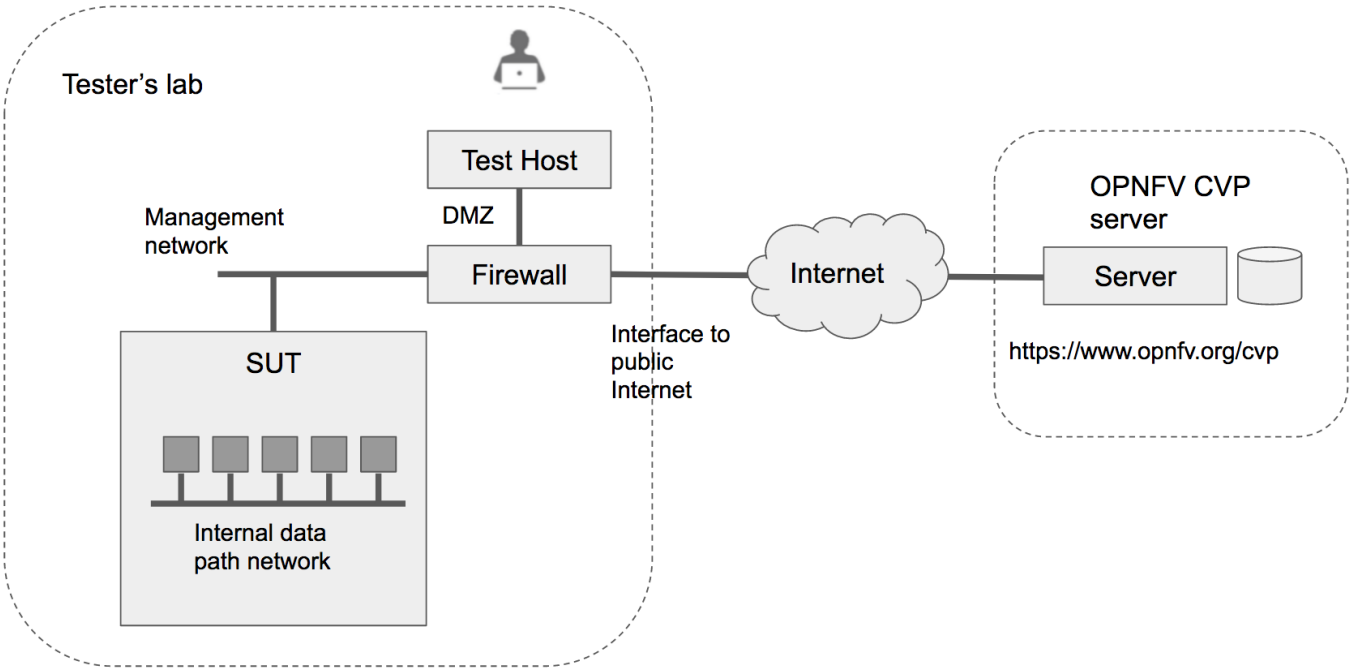
Compliance Verification Program Testing User Guide

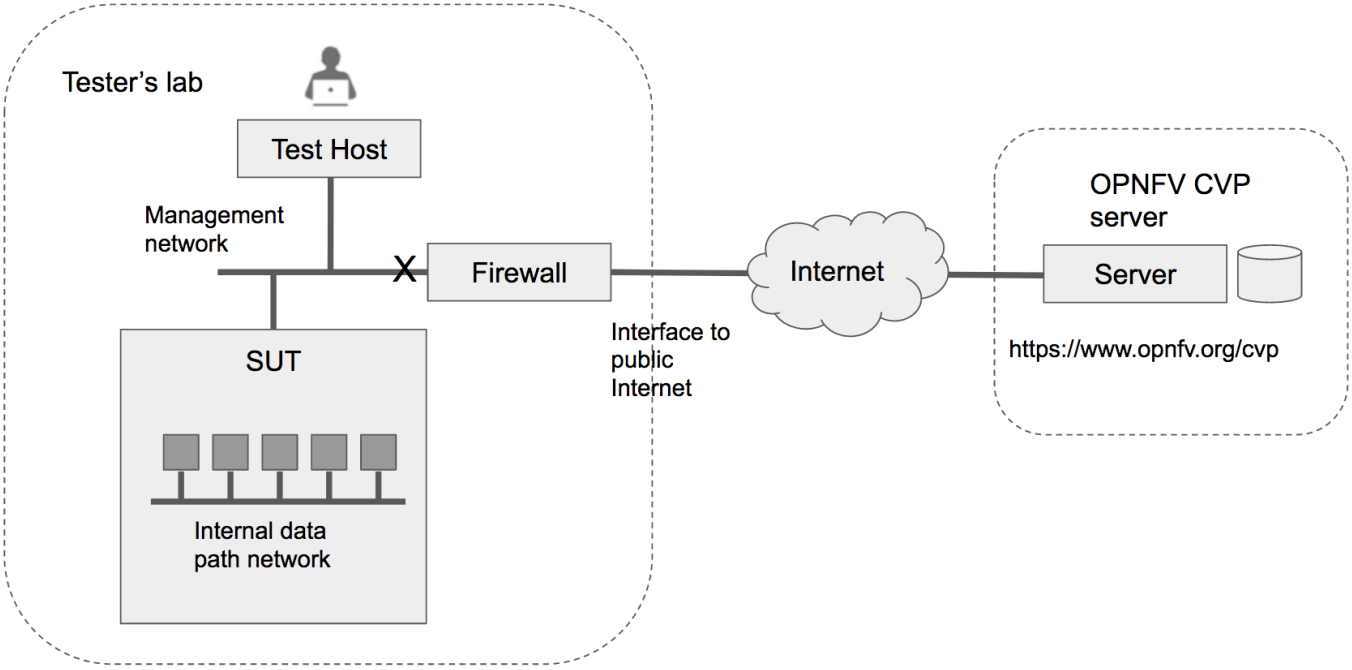
Conducting CVP Testing with Dovetail

Overview

The Dovetail testing framework for CVP consists of two major parts: the testing client that executes all test cases in a lab (vendor self-testing or a third party lab), and the server system that is hosted by the CVP administrator to store and view test results based on a web API. The following diagram illustrates this overall framework.



Within the tester's lab, the Test Host is the machine where Dovetail executes all automated test cases. It is a part of the Test Harness, and must not be part of the System Under Test (SUT) itself. The OPNFV community customarily refers this machine as the *Jumphost*, but we will avoid that naming convention here because it can be loaded with other meanings, e.g. it does not imply that it serves an intermediary role for secure access. The above diagram assumes that the tester's Test Host is situated in a DMZ which has network access to both the SUT and the OPNFV server via the public Internet. This arrangement may not be practical or convenient in some company's labs for security reasons. Dovetail also supports an offline mode of testing that is illustrated in the next diagram.



In the offline mode, the Test Host only needs to have access to the SUT's API via the internal management network, but not to the public Internet. This user guide will highlight differences between the online and offline modes of the Test Host. While it is possible to run the Test Host as a virtual machine, this user guide assumes it is a physical machine for simplicty.

Dovetail client tool (or just Dovetail tool or Dovetail for short) can be installed in the Test Host either directly as a Python script, or as a Docker container. The Docker container method is simpler and supports both online and offline modes. If a tester wishes to experiment further with Dovetail or extend or modify its behavior for testing other than just CVP, then she or he will need to install Dovetail as Python scripts. This guide will describe both methods of installation. Once installed, and properly configured, the remaining test process is mostly identical for the two options. We first go over the steps of running the test suite. Then, we discuss how to view test results and make sense of them, for example, what the tester may do in case of unexpected test failures. Finally, we describe additional Dovetail features that are not absolutely necessary in CVP testing but users may find useful for other purposes. One example is to run Dovetail for in-house testing as preparation before official CVP testing; another example is to run Dovetail experimental test suites other than the CVP test suite. Experimental tests may be made available by the community for experimenting less mature test cases or functionalities for the purpose of getting feedbacks for improvement.

Installing Dovetail

Before taking this step, a tester should check the hardware and networking requirements of the SUT, and the Test Host in particular, to make sure they are compliant with Pharos specification.

In this section, we describe the procedure to install Dovetail client tool on the Test Host. The Test Host must have network access to the management network with access rights to the Virtual Infrastructure Manager's API.

Checking the Test Host Readiness

The OPNFV community does not have definitive requirements on a specific operating system or version, other than that it runs a reasonably up to date version of Linux that can support the Dovetail tool chain. For example,

- Ubuntu 16.04.2 LTS (Xenial) or 14.04 LTS (Trusty)
- CentOS-7-1611
- Red Hat Enterprise Linux 7.3

- Fedora 24 or 25 Server

Non-Linux operating systems, such as Windows, Mac OS, have not been tested and are not supported.

If online mode is used, the tester should also validate that the Test Host can reach the public Internet. For example,

```
$ ping www.opnfv.org
PING www.opnfv.org (50.56.49.117): 56 data bytes
64 bytes from 50.56.49.117: icmp_seq=0 ttl=48 time=52.952 ms
64 bytes from 50.56.49.117: icmp_seq=1 ttl=48 time=53.805 ms
64 bytes from 50.56.49.117: icmp_seq=2 ttl=48 time=53.349 ms
...
```

Or, if the lab environment does not allow ping, try validating it using HTTPS instead.

```
$ curl https://www.opnfv.org
<!doctype html>


<html lang="en-US" class="no-js">
<head>
...
```

The following are all the ports that should be open in the firewall to the public Internet in order to support the Dovetail online mode testing.

```
$ echo 'List open ports to support online mode.'
```

Configuring the Test Host Environment

The Test Host needs a few environment variables set correctly in order to access the Openstack API required to drive the Dovetail tests. For convenience and as a convention, we will also create a home directory:

```
$ cd $HOME
$ mkdir dovetail
```

At this point, you will need to consult your SUT (Openstack) administrator to correctly set the configurations in a file. The Openstack settings need to be configured such that the Dovetail client has all the necessary credentials and privileges to execute all test operations. If the SUT uses terms somewhat differently from the standard Openstack naming, you will need to adjust this file accordingly.

In our example, we will use the file '$HOME/dovetail/env_config.sh'.

```
$ echo 'Current version has a bug that the home dir must be /home/opnfv.'

$ cat ~/dovetail/env_config.sh

# Project-level authentication scope (name or ID), recommend admin project.
export OS_PROJECT_NAME=admin

# Authentication username, belongs to the project above, recommend admin user.
export OS_USERNAME=admin

# Authentication password. Use your own password
export OS_PASSWORD=xxxxxxxx

# Authentication URL, one of the endpoints of keystone service. If this is v3 version,
# there need some extra variables as follows.
export OS_AUTH_URL='http://xxx.xxx.xxx.xxx:5000/v3'
```

```
# Default is 2.0. If use keystone v3 API, this should be set as 3.
export OS_IDENTITY_API_VERSION=3

# Domain name or ID containing the user above.
# Command to check the domain: openstack user show <OS_USERNAME>
export OS_USER_DOMAIN_NAME=default

# Domain name or ID containing the project above.
# Command to check the domain: openstack project show <OS_PROJECT_NAME>
export OS_PROJECT_DOMAIN_NAME=default

# Home directory for dovetail
export DOVETAIL_HOME=$HOME/dovetail
```

Export all these variables into environment by,

```
$ sudo source $HOME/dovetail/env_config.sh
```

You can validate that the Openstack environmental settings are correct by,

```
$ openstack service list
```

Installing Prerequisite on the Test Host

The main prerequisite software for Dovetail are Python and Docker. Note that regardless of whether Dovetail is run as a container or as Python scripts, Docker is always required.

In the CVP test suite for the Danube release, Dovetail requires Python 2.7. Python 3.x is not supported at this time.

Use the following steps to check if the right version of python is already installed, and if not, install it.

```
$ python --version
Python 2.7.6

$ echo "Please add Python installation or upgrade steps"
```

Dovetail requires at least Docker 1.12.3. We have validated these Docker versions in the OPNFV lab: '1.12.3, 1.12.5, 1.13, 17.03.1-ce, 17.04.0-ce, 17.05.0-ce'. Use the following step to check if the right version of Docker is already installed, and if not, install it.

```
$ docker --version
Docker version 17.04.0-ce, build 4845c56
```

If your system does not have Docker already installed, or Docker is older than 1.12.3, you will need to upgrade in order to run Dovetail. The Docker installation process can be more complex, you should refer to the official Docker installation guide that is relevant to your environment.

The following simple script will install the latest version of Docker. If you do not intend to upgrade your system's Docker version to the latest, you should not use this simple method.

```
$ echo 'Revise to use exact versions once the supported versions are finalized'
$ wget -qO- https://get.docker.com/ | sh
```

Please consult the official documentation in https://docs.docker.com/engine/installation/ to install or upgrade a specific version of Docker. You may also `wget -qO- https://get.docker.com/` and edit the version string to the version you choose.

The above installation steps assume that the Test Host is in the online mode. For offline testing, use the following offline installation steps instead.

In order to install or upgrade Python offline,

```
$ echo "Please add offline python installation or upgrade"
```

In order to install Docker offline, download Docker static binaries and copy the tar file to the Test Host, then you may follow these instructions to install.

```
$ tar /path/to/the-file.tar.gz
```

Copy the executables to a directory on $PATH, for example, /usr/bin/.

```
$ sudo cp docker/* /usr/bin/
```

Start the Docker daemon.

```
$ sudo dockerd &
```

Installing Dovetail on the Test Host

The Dovetail project maintains a Docker image that has Dovetail test tools preinstalled. This Docker image is tagged with versions. Before pulling the Dovetail image, check the OPNFV's CVP web page first to determine the right tag for CVP testing.

If the Test Host is online, you can directly pull.

```
$ sudo docker pull opnfv/dovetail:latest
latest: Pulling from opnfv/dovetail
30d541b48fc0: Pull complete
8ecd7f80d390: Pull complete
46ec9927bb81: Pull complete
2e67a4d67b44: Pull complete
7d9dd9155488: Pull complete
cc79be29f08e: Pull complete
e102eed9bf6a: Pull complete
952b8a9d2150: Pull complete
bfbb639d1f38: Pull complete
bf7c644692de: Pull complete
cdc345e3f363: Pull complete
Digest: sha256:d571b1073b2fdada79562e8cc67f63018e8d89268ff7faabee3380202c05edee
Status: Downloaded newer image for opnfv/dovetail:latest

$ echo 'The correct <tag> will replace *latest* once it is released.'
```

An example of the <tag> is *latest*.

```
$ echo 'Until the official tag is released, we will use *latest* in the examples.'
```

If the Test Host is offline, you will need to first pull the Dovetail Docker image, and all the dependent images that Dovetail uses, to a host that is online. The reason that you need to pull all dependent images is because Dovetail normally does dependency checking at run-time and automatically pull images as needed, if the Test Host is online. If the Test Host is offline, then all these dependencies will also need to be manually copied.

```
$ sudo docker pull opnfv/dovetail:latest
$ sudo docker pull opnfv/functest:latest
```

```
$ sudo docker pull opnfv/yardstick:latest
```

Once all these images are pulled, save the images, copy to the Test Host, and then load the Dovetail and all dependent images at the Test Host.

At the online host, save images.

```
$ sudo docker save -o dovetail.tar opnfv/dovetail:latest opnfv/functest:latest opnfv/yardsti
```

Copy dovetail.tar file to the Test Host, and then load the images on the Test Host.

```
$ sudo docker load --input dovetail.tar
```

Now check to see that the Dovetail image has been pulled or loaded properly.

```
$ sudo docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
opnfv/functest      latest              9eaeaea5f203        8 days ago          1.53GB
opnfv/dovetail      latest              5d25b289451c        8 days ago          516MB
opnfv/yardstick     latest              574596b6ea12        8 days ago          1.2GB
```

Regardless of whether you pulled down the Dovetail image directly online, or loaded from a static image tar file, you are ready to run Dovetail.

```
$ echo 'Current version has a bug that the home dir must be /home/opnfv/.'

$ sudo docker run --privileged=true -it \
          -v $DOVETAIL_HOME/env_config.sh:$DOVETAIL_HOME/env_config.sh \
          -v /home/opnfv/dovetail/results:/home/opnfv/dovetail/results \
          -v /var/run/docker.sock:/var/run/docker.sock \
          opnfv/dovetail:<tag> /bin/bash
```

The `-v` options map files in the host to files in the container.

Running the CVP Test Suite

Now you should be in the Dovetail container's prompt and ready to execute test suites.

The Dovetail client CLI allows the tester to specify which test suite to run. By default the results are stored in a local file `$DOVETAIL_HOME/results`.

```
$ dovetail run --testsuite <test-suite-name> --openrc <path-to-env-config-file>
```

<path-to-env-config-file> should be `/home/opnfv/dovetail/env_config.sh` as specified in the `-v` option when you run the docker image.

Multiple test suites may be available. For the purpose of running CVP test suite, the test suite name follows the following format, `CVP_<major>_<minor>_<patch>` For example, CVP_1_0_0.

```
$ dovetail run --testsuite CVP_1_0_0 --openrc /home/opnfv/dovetail/env_config.sh
```

If you are not running the entire test suite, you can choose to run an individual test area instead.

```
$ dovetail run --testsuite CVP_1_0_0 --testarea ipv6\
     --openrc /home/opnfv/dovetail/env_config.sh
```

Until the official test suite is approved and released, you can use the *proposed_tests* for your trial runs, like this.

```
$ dovetail run --testsuite proposed_tests --testarea ipv6\
    --openrc /home/opnfv/dovetail/env_config.sh
Executing command: 'sudo rm -rf /home/opnfv/dovetail/results/*'
2017-05-23 05:01:49,488 - run - INFO - ================================================
2017-05-23 05:01:49,488 - run - INFO - Dovetail compliance: proposed_tests!
2017-05-23 05:01:49,488 - run - INFO - ================================================
2017-05-23 05:01:49,488 - run - INFO - Build tag: daily-master-4bdde6b8-afa6-40bb-8fc9-5d568
2017-05-23 05:01:49,536 - run - INFO -
docker version:
client:17.05.0-ce
server:17.04.0-ce
2017-05-23 05:01:49,537 - run - WARNING -

Version mismatch, may cause unpredictable errors, you can update or install the lastest docke
wget -qO- https://get.docker.com/ | sh

Client:17.05.0-ce
Server:17.04.0-ce
2017-05-23 05:01:49,710 - run - INFO - >>[testcase]: dovetail.ipv6.tc001
2017-05-23 05:08:22,532 - run - INFO - Results have been stored with file /home/opnfv/dovetai
2017-05-23 05:08:22,538 - run - INFO - >>[testcase]: dovetail.ipv6.tc002
...
```

Making Sense of CVP Test Results

Updating Dovetail or a Test Suite

Additional Dovetail Usages

Installing Dovetail Source

You can also choose to install Dovetail as source rather than as a container. It may give you more flexibility in some situations.

The first step is to update and install all dependent packages.

a. Ubuntu

```
$ sudo apt-get update
$ sudo apt-get -y install gcc git vim python-dev python-pip --no-install-recommends
```

b. CentOS and RedHat

```
$ sudo yum -y update
$ sudo yum -y install epel-release
$ sudo yum -y install gcc git vim-enhanced python-devel python-pip
```

c. Fedora

```
$ sudo dnf -y update
$ sudo dnf -y install gcc git vim-enhanced python-devel python-pip redhat-rpm-config
```

Now we are ready to install Dovetail source.

```
$ cd $DOVETAIL_HOME
$ sudo git clone https://git.opnfv.org/dovetail
$ cd $DOVETAIL_HOME/dovetail
$ sudo pip install -e ./
```

You can verify that the installation is successful by,

```
$ dovetail -h
```

Running Dovetail Locally

Running Dovetail with Experimental Test Cases

Running Individual Test Cases or Special Cases

Dovetail Command Line Interface Reference