Example Documentation table of contents

Release draft (0ad43fd)

OPNFV

February 09, 2016

CONTENTS

Contents:

CHAPTER

ONE

HOW TO CREATE DOCUMENTATION FOR YOUR OPNFV PROJECT

this is the directory structure of the docs/ directory that can be found in the root of your project directory

```
./etc
./etc/opnfv-logo.png
./etc/conf.py
./how-to-use-docs
./how-to-use-docs/documentation-example.rst
./how-to-use-docs/index.rst
```

To create your own documentation, Create any number of directories (depending on your need) and place in each of them an index.rst. This index file must refence your other rst files.

- Here is an example index.rst

```
Example Documentation table of contents
=======================================

Contents:

.. toctree::
   :numbered:
   :maxdepth: 4

   documentation-example.rst

Indices and tables
==================

* :ref:`search`

Revision:

Build date: |today|
```

CHAPTER

TWO

THE SPHINX BUILD

When you push documentation changes to gerrit a jenkins job will create html documentation.

- Verify Jobs

For verify jobs a link to the documentation will show up as a comment in gerrit for you to see the result.

- Merge jobs

Once you are happy with the look of your documentation you can submit the patchset the merge job will copy the output of each documentation directory to http://artifacts.opnfv.org/$project/docs/$name_of_your_folder/index.html

Here are some quick examples of how to use rst markup

This is a headline:

```
here is some code, note that it is indented
```

links are easy to add: Here is a link to sphinx, the tool that we are using to generate documetation http://sphinx-doc.org/

- Bulleted Items

  this will be bold

```bash
echo "Heres is a code block with bash syntax highlighting"
```

Leave these at the bottom of each of your documents they are used internally

Revision:

Build date: February 09, 2016

image:: ../etc/opnfv-logo.png

height 40

width 200

alt OPNFV

align left

CHAPTER

THREE

INTRODUCTION

Welcome to QTIP's documentation !

QTIP is an OPNFV Project.

QTIP aims to benchmark OPNFV platforms through a "Bottom up" approach, testing bare-metal components first.

The overall problem this project tries to solve is the general characterization of an OPNFV platform. It will focus on general performance questions that are common to the platform itself, or applicable to multiple OPNFV use cases. QTIP will provide the capability to quantify a platform's performance behavior in a standardized, rigorous, and open way, and a well-documented methodology to reproduce the results by anyone interested.

The chapter `02-methodology` describes the methodology implemented by the QTIP Project for NFVI performance benchmarking. The chapter `03-list-of-testcases` includes a list of available Yardstick test cases.

The *QTIP* framework is deployed in the Dell OPNFV community lab. It is infrastructure and application independent.

See also:

Pharos for information on OPNFV community labs.

3.1 Contact QTIP

Feedback? Contact us

CHAPTER

FOUR

GUIDE TO RUN QTIP:

This guide will serve as a first step to familiarize the user with how to run QTIP the first time when the user clones QTIP on to their host machine. In order to clone QTIP please follow the instructions in the installation.rst located in docs/userguide/installation.rst.

4.1 QTIP Directory structure:

The QTIP directory has been sectioned off into multiple folders to facilitate segmenting information into relevant categories. The folders that concern the end user are *test_cases/* and *test_list/*.

4.2 test_cases/:

This folder is used to store all the config files which are used to setup the environment prior to a test. This folder is further divided into opnfv pods which run QTIP. Inside each pod there are folders which contain the config files segmented based on test cases. Namely, these include, *Compute*, *Network* and *Storage*. The default folder is there for the end user who is interested in testing their infrastructure but arent part of a opnfv pod.

The structure of the directory for the user appears as follows

```
test_cases/default/compute
test_cases/default/network
test_cases/default/storage
```

The benchmarks that are part of the QTIP framework are listed under these folders. An example of the compute folder is shown below. Their naming convention is <BENCHMARK>_<VM/BM>.yaml

```
dhrystone_bm.yaml
dhrystone_vm.yaml
whetstone_vm.yaml
whetstone_bm.yaml
ssl_vm.yaml
ssl_bm.yaml
ramspeed_vm.yaml
ramspeed_bm.yaml
dpi_vm.yaml
dpi_bm.yaml
```

The above listed files are used to configure the environment. The VM/BM tag distinguishes between a test to be run on the Virtual Machine or the compute node itself, respectively.

4.3 test_list/:

This folder contains three files, namely *compute*, *network* and *storage*. These files list the benchmarks are to be run by the QTIP framework. Sample compute test file is shown below

```
dhrystone_vm.yaml
dhrystone_bm.yaml
whetstone_vm.yaml
ssl_bm.yaml
```

The compute file will now run all the benchmarks listed above one after another on the environment. *NOTE: Please ensure there are no blank lines in this file as that has been known to throw an exception.*

4.4 Preparing a config file for test:

We will be using dhrystone as a example to list out the changes that the user will need to do in order to run the benchmark. Dhrystone on Compute Nodes: ^^^^^^^^^^^^^^^^^^^^^^^^^^^^

QTIP framework can run benchmarks on the actual compute nodes as well. In order to run dhrystone on the compute nodes we will be editing the dhrystone_bm.yaml file.

```
Scenario:
  benchmark: dhrystone
  host: machine_1, machine_2
  server:
```

The *Scenario* field is used by to specify the name of the benchmark to run as done by *benchmark: dhrystone*. The *host* and *server* tag are not used for the compute benchmarks but are included here to help the user *IF* they wish to control the execution. By default both machine_1 and machine_2 will have dhrystone run on them in parallel but the user can change this so that machine_1 run dhrystone before machine_2. This will be elaborated in the *Context* tag.

```
Context:
  Host_Machines:
    machine_1:
      ip: 10.20.0.6
      pw:
      role: host
    machine_2:
      ip: 10.20.0.5
      pw:
      role: host

   Virtual_Machines:
```

The *Context* tag helps the user list the number of compute nodes they want to run dhrystone on. The user can list all the compute nodes under the *Host_Machines* tag. All the machines under test must be listed under the *Host_Machines* and naming it incrementally higher. The *ip:* tag is used to specify the IP of the particular compute node. The *pw:* tag can be left blank because QTIP uses its own key for ssh. In order to run dhrystone on one compute node at a time the user needs to edit the *role:* tag. *role: host* for machine_1 and *role: server* for machine_2 will allow for dhrystone to be run on machine_1 and then run on machine_2.

```
Test_Description:
  Test_category: "Compute"
  Benchmark: "dhrystone"
  Overview: >
```

```
    ''' This test will run the dhrystone benchmark in parallel  on
    machine_1 and machine_2.
```

The above field is purely for a description purpose to explain to the user the working of the test and is not fed to the framework.

4.5 Sample dhrystone_bm.yaml file:

```
Scenario:
  benchmark: dhrystone
  host: machine_1, machine_2
  server:

Context:
  Host_Machines:
    machine_1:
      ip: 10.20.0.6
      pw:
      role: host
    machine_2:
      ip: 10.20.0.5
      pw:
      role: host

  Virtual_Machines:


Test_Description:
  Test_category: "Compute"
  Benchmark: "dhrystone"
  Overview: >
      ''' This test will run the dhrystone benchmark in parallel  on
      machine_1 and machine_2.\n
```

4.5.1 Dhrystone on Virtual Machine:

To run dhrystone on the VMs we will be editing dhrystone_vm.yaml file. Snippets on the file are given below.

```
Scenario:
benchmark: dhrystone
host: virtualmachine_1, virtualmachine_2
server:
```

The *Scenario* field is used by to specify the name of the benchmark to run as done by *benchmark: dhrystone*. The *host* and *server* tag are not used for the compute benchmarks but are included here to help the user *IF* they wish to control the execution. By default both virtualmachine_1 and virtualmachine_2 will have dhrystone run on them in parallel but the user can change this so that virtualmachine_1 run dhrystone before virtualmachine_2. This will be elaborated in the *Context* tag.

```
Context:
  Host_Machines:

  Virtual_Machines:
    virtualmachine_1:
      availability_zone: compute1
      public_network: 'net04_ext'
      OS_image: QTIP_CentOS
      flavor: m1.large
      role: host
    virtualmachine_2:
      availability_zone: compute2
      public_network: 'net04_ext'
      OS_image: QTIP_CentOS
      flavor: m1.large
      role: host
```

The *Context* tag helps the user list the number of VMs and their characteristic. The user can list all the VMs they want to bring up under the *Virtual_Machines:* tag. In the above example we will be bringing up two VMs. One on Compute1 and the other on Compute2. The user can change this as desired *NOTE: Please ensure you have the necessary compute nodes before listing under the 'availability_zone:' tag*. The rest of the options do not need to be modified by the user.

4.5.2 Running dhrystone sequentially (Optional):

In order to run dhrystone on one VM at a time the user needs to edit the *role:* tag. *role: host* for virtualmachine_1 and *role: server* for virtualmachine_2 will allow for dhrystone to be run on virtualmachine_1 and then run on virtualmachine_2.

```
Test_Description:
  Test_category: "Compute"
  Benchmark: "dhrystone"
  Overview:
  This test will run the dhrystone benchmark in parallel on
  virtualmachine_1 and virtualmachine_2
```

The above field is purely for a decription purpose to explain to the user the working of the test and is not fed to the framework.

4.6 Sample dhrystone_vm.yaml file:

```
Scenario:
benchmark: dhrystone
host: virtualmachine_1, virtualmachine_2
server:

Context:
  Host_Machines:

  Virtual_Machines:
    virtualmachine_1:
      availability_zone: compute1
      public_network: 'net04_ext'
      OS_image: QTIP_CentOS
      flavor: m1.large
      role: host
    virtualmachine_2:
      availability_zone: compute2
      public_network: 'net04_ext'
      OS_image: QTIP_CentOS
      flavor: m1.large
      role: host

Test_Description:
  Test_category: "Compute"
  Benchmark: "dhrystone"
  Overview: >
  This test will run the dhrystone benchmark in parallel on
  machine_1 and machine_2.\n
```

CHAPTER

FIVE

COMMANDS TO RUN THE FRAMEWORK:

In order to start QTIP on the default lab please use the following commands (asssuming you have prepared the config files in the test_cases/default/ directory and listed the intended suite in the test_list/<RELEVANT-SUITE-FILE>):

First step is to export the necessary information to the environment.

```
source get_env_info.sh -n <INSTALLER_TYPE> -i <INSTALLER_IP>
```

for running qtip on an openstack deployed using FUEL with the Installer IP 10.20.0.2

```
source get_env_info.sh -n fuel -i 10.20.0.2
```

This will generate the *opnfv-creds.sh* file needed to use the python clients for keystone, glance, nova, and neutron.

```
source opnfv-creds.sh
```

Running QTIP on the using *default* as the pod name and for the *compute* suite

```
python qtip.py -l default -f compute
```

Running QTIP on the using *default* as the pod name and for the *network* suite

```
python qtip.py -l default -f network
```

Running QTIP on the using *default* as the pod name and for the *storage* suite

```
python qtip.py -l default -f network
```

CHAPTER

SIX

RESULTS:

QTIP generates results in the *results/* directory are listed down under the particularly benchmark name. So all the results for dhrystone would be listed and time stamped.

CHAPTER SEVEN

INDICES AND TABLES

- search

Revision:

Build date: February 09, 2016