Get started as a SampleVNF developer

Introduction

This project provides a placeholder for various sample VNF (Virtual Network Function) development which includes example reference architecture and optimization methods related to VNF/Network service for high performance VNFs. This project provides benefits to other OPNFV projects like Functest, Models, yardstick etc to perform real life use-case based testing and NFVi characterization for the same.

The sample VNFs are Open Source approximations* of Telco grade VNF's using optimized VNF + NFVi Infrastructure libraries, with Performance Characterization of Sample† Traffic Flows.
• * Not a commercial product. Encourage the community to contribute and close the feature gaps.
• † No Vendor/Proprietary Workloads

It helps to facilitate deterministic & repeatable bench-marking on Industry standard high volume Servers. It augments well with a Test Infrastructure to help facilitate

Therefore there are many ways to contribute to samplevnf.

You can:

- Develop new test cases in samplevnf
- Review codes
- Develop/contribute to existing VNFs or new VNFs
- Write samplevnf documentation

This page describes how, as a developer, you may interact with the samplevnf project.

Where can I find some help to start?

You can also directly contact us by mail with [SampleVNF] prefix in the title at opnfv-tech-discuss@lists.opnfv.org or on the IRC chan #opnfv-samplevnf.

How TOs

How can I contribute to SampleVNF?

If you are already a contributor of any OPNFV project, you can contribute to samplevnf.

If you are totally new to OPNFV, you must first create your Linux Foundation account, then contact (helpdesk@rt.linuxfoundation.org; helpdesk@opnfv.org) in order to declare you in the repository database.

We distinguish 2 levels of contributors:

- the standard contributor can push patch and vote +1/0/-1 on any samplevnf patch
- The commitor can vote -2/-1/0/+1/+2 and merge

SampleVNF commitors are promoted by the samplevnf contributors.

Gerrit & JIRA

OPNFV uses **Gerrit** for web based code review and repository management for the Git Version Control System. You can access OPNFV Gerrit from this link.

Please note that you need to have Linux Foundation ID in order to use OPNFV Gerrit. You can get one from this link.

OPNFV uses **JIRA** for issue management. An important principle of change management is to have two-way trace-ability between issue management (i.e. JIRA) and the code repository (via Gerrit).

In this way, individual commits can be traced to JIRA issues and we also know which commits were used to resolve a JIRA issue.

If you want to contribute to samplevnf, you can pick a issue from SampleVNF's JIRA dashboard or you can create you own issue and submit it to JIRA.

Submitting code to Gerrit

Installing and configuring Git and Git-Review is necessary in order to submit code to Gerrit. The **Getting to the code** page will provide you with some help for that.

Comitting the code with Git

Open a terminal window and set the project's directory to the working directory using the cd command. In this case "/home/opnfv/samplevnf" is the path to the samplevnf project folder on my computer. Replace this with the path of your own project.

```
cd /home/opnfv/samplevnf
```

Tell Git which files you would like to take into account for the next commit. This is called 'staging' the files, by placing them into the staging area, us

```
git add samplevnf/samples/sample.yaml
...
```

Alternatively, you can choose to stage all files that have been modified (that is the files you have worked on) since the last time you generated a commit, by using the -a argument.

```
git add -a
```

Git won't let you push (upload) any code to Gerrit if you haven't pulled the latest changes first. So the next step is to pull (download) the latest changes made to the project by other collaborators using the 'pull' command.

```
git pull
```

Now that you have the latest version of the project and you have staged the files you wish to push, it is time to actually commit your work to your local Git repository.

```
git commit --signoff -m "Title of change

Test of change that describes in high level what
was done. There is a lot of documentation in code
so you do not need to repeat it here.

JIRA: SAMPLEVNF-XXX"
```

The message that is required for the commit should follow a specific set of rules. This practice allows to standardize the description messages attached to the commits, and eventually navigate among the latter more easily.

This document happened to be very clear and useful to get started with that.

Verify your patch locally before submitting

Once you finish a patch, you can submit it to Gerrit for code review. A developer sends a new patch to Gerrit will trigger patch verify job on Jenkins CI.

Pushing the code to Gerrit for review

Now that the code has been comitted into your local Git repository the following step is to push it online to Gerrit for it to be reviewed. The command we will use is 'git review'.

```
git review
```

This will automatically push your local commit into Gerrit.

Code review

You can add Samplevnf committers and contributors to review your codes.

Modifying the code under review in Gerrit

At the same time the code is being reviewed in Gerrit, you may need to edit it to make some changes and then send it back for review. The following steps go through the procedure.

Once you have modified/edited your code files under your IDE, you will have to stage them. The 'status' command is very helpful at this point as it provides an overview of Git's current state.

```
git status
```

The output of the command provides us with the files that have been modified after the latest commit.

You can now stage the files that have been modified as part of the Gerrit code review edition/modification/improvement using git add command.

It is now time to commit the newly modified files, but the objective here is not to create a new commit, we simply want to inject the new changes into the previous commit. You can achieve that with the '--amend' option on the 'commit' command:

```
git commit --amend
```

If the commit was successful, the 'status' command should not return the updated files as about to be commited.

The final step consists in pushing the newly modified commit to Gerrit.

```
git review
```

References

[1]: http://artifacts.opnfv.org/samplevnf/docs/testing_user_userguide_vACL/index.html