KVM4NFV CICD Jobs

Background:

- **Build packages**: Kernel rpm, Kernel deb, Qemu rpm, Qemu deb packages. The scripts for generating this will be placed under opnfv kvmfornfv tree [https://gerrit.opnfv.org/gerrit/#/admin/projects/kvmfornfv](https://gerrit.opnfv.org/gerrit/#/admin/projects/kvmfornfv).
- **As part of the CICD process**, any change in the kernel/qemu builds will trigger the releng jobs. These jobs will verify the changed scripts by building the rpm and debian packages inside docker containers, testing them by launching the host-guest and performing the cyclic tests. The jobs will also execute the merge process to the parent repository, if the changed rpm and debian builds are deployed successfully on the host-guest and give the expected results.

CICD Workflow Umbrella:

It broadly includes the following:

1. **Build package scripts**- This section talks about the scripts that will be mentioned inside kvmfornfv tree, for creating Kernel rpm, Kernel debian, Qemu rpm and Qemu debian builds.
2. **High level Jobs**- This section lists the sequential order of the main processes that the releng JJB will trigger. Depending upon the type of Job (verify/merge/daily), activities like compiling the builds, deploying it on the host, spawning a guest and running the cyclic tests on the guest will be executed.
3. **Detailing the Jobs**- This area enlists all the low-level tasks that will be executed as part of the verify, merge and daily jobs. Each task will have the following elements:
   - (a) What- what the sub-tasks are? building, deploying, booting, triggering, testing, etc
   - (b) Where- where this sub-task will be running? on the host, on the guest, on the jenkins slave, inside the dockers, etc
   - (c) How- how this sub-task should work? it should be triggered by JJB, script on host/slave, new build scripts, etc
4. **Future Work**- This section details the tasks planned in the pipeline, to automate some scenarios for RT-Low Latency.

After running the cyclic tests on the guest as part of the CICD process is attained, some of features that can be implemented are- SR-IOV, OVSDPDK and Live migration scenario tests using L2 Forwarding packetgen tool; Inter VM communication, etc.

Which OPNFV components to be used for implementing these scenarios has to be discussed also, like VSPerf, Functest, Yardstick, etc.

1. **Build package scripts**
   - 1.1. Kernel rpm build script already available
   - 1.2. Work on Kernel debian builds creation
   - 1.3. Work on Qemu rpm builds creation
   - 1.4. Work on Qemu debian builds creation

2. **High level jobs execution**

   - 2.1. The kernel/qemu rpm/deb packages pass the compilation.
   - 2.2. The new built packages are uploaded to artifact repository.
   - 2.3. These artifacts are then downloaded to Intel Pod1 Jump server.
   - 2.4. On the above Intel Pod1 jump server, an ubuntu container with yardstick is available. Here, Yardstick will run the cyclic test to check latency after performing automated host and guest configuration. (guest environment is defined by a preconfigured image currently, present on Intel Pod1 Node1.

3. **Detailing the jobs:**

   In the job template file, below are proposed testing scenarios:

   3.1. **Verify Job**- It includes the test for the patch verification and merge under separate setup, execute and teardown categories.
      - 3.1.1. Setup():
      - 3.1.1.1. Clone the changed kernel/qemu rpm/debian repository. Run the build scripts created in Step (1) on this changed repo and creates new rpm/deb packages for kernel and qemu respectively.

   3.2. **Merge Job**- It includes the test for the merge under separate setup, execute and teardown categories.
      - 3.2.1. Setup():
      - 3.2.1.1. Clone the merged kernel/qemu rpm/debian repository. Run the build scripts created in Step (1) on this merged repo and creates new rpm/deb packages for kernel and qemu respectively.
3.1.1.2. Store the new created built packages on artifacts.opnfv.org and download onto Intel Pod1
3.1.1.3. Yardstick runs cyclictest to automate the process of applying the built package (eg: *.rpm/*.deb) on the host, and performing the host and guest configuration tasks. Some of the available scripts are-
   a) host-setup0.sh
   b) host-setup1.sh
   c) host-run-qemu.sh
   d) guest-setup0.sh
   e) guest-setup1.sh

   Note: If guest booting or configuration setup scripts fail, the test fails.
3.1.2. Execute()-
   3.1.2.1. Run cyclictest on the guest and check the result.

   Note: If result exceeds the base latency, the test fails.
3.1.3. Teardown()-
   3.1.3.1. At any point of time, if the test/scripts fail in steps (3.1.1) OR (3.1.2)-
   a) Logs will be maintained to record the associated issues: TBD
   b) Report Verified+1 against this change on Gerrit in order to prevent the merge and delete the recently tested built package from artifacts.opnfv.org
   3.1.3.2. If the test/scripts execute successfully in steps (3.1.1) and (3.1.2), report Verified1 against this change on Gerrit.

   Maintain this recently tested built package on artifacts.opnfv.org
   3.1.3.3. Octopus support for reinstalling the previous kernel/qemu version on the test host should be made possible: TBD

3.2. Merge Job- Scripts are defined in Releng for automating the merge tasks.
3.3. Daily Job- This testing could be integrated/executed by Yardstick and run in scope of the daily build/deploy/test CI loop.
   3.3.1. Testcases defined in (3.1.2.3) to be run as part of Yardstick daily Jenkins job
   3.3.2. [KVM] to support proper troubleshooting if/when the test case fails: TBD

4. Future Work:

The intention is to automate some scenarios for RT-Low Latency.

After the test scenario of automatically implementing Cyclicttest as part of the CICD process (on the host environment) is achieved, work will be carried out on the following scenarios as well-

- 4.1. SR-IOV scenario tests for L2 Forwarding using packetgen tool
- 4.2. OVS-DPDK scenario tests L2 Forwarding using packetgen tool
- 4.3. Live migration scenario tests for L2 Forwarding using packetgen tool
- 4.4. Inter VM communication (after implementation is available)

Note: Add 4.1 & 4.2 to VSPerf git repo, 4.3 & 4.4 to FuncTest or Yardstick based on study & discussion.

Note: VMs should be launched with CPU isolation, Core pinning and interrupt free.