LINUX FOUNDATION
COLLABORATIVE PROJECTS

Leveraging OPNFV test tools beyond the NFV domain

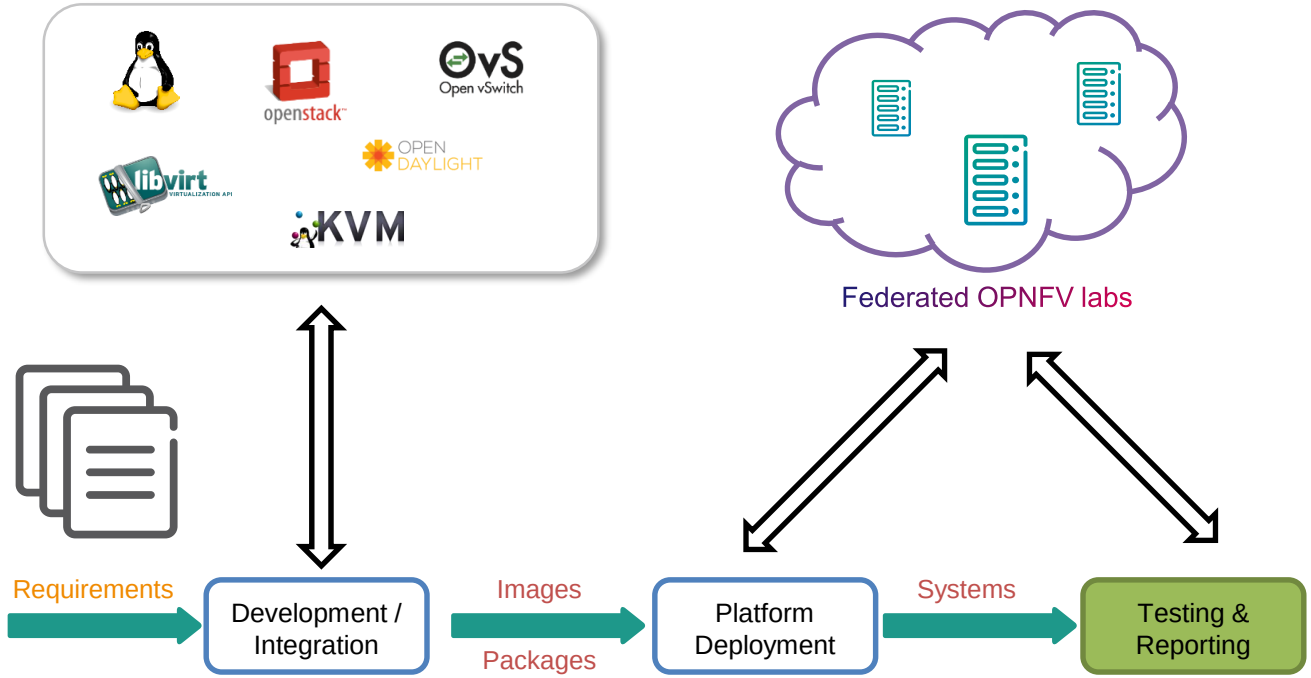
Georg Kunz, Emma Foley & the OPNFV testing community

Goals of this talk

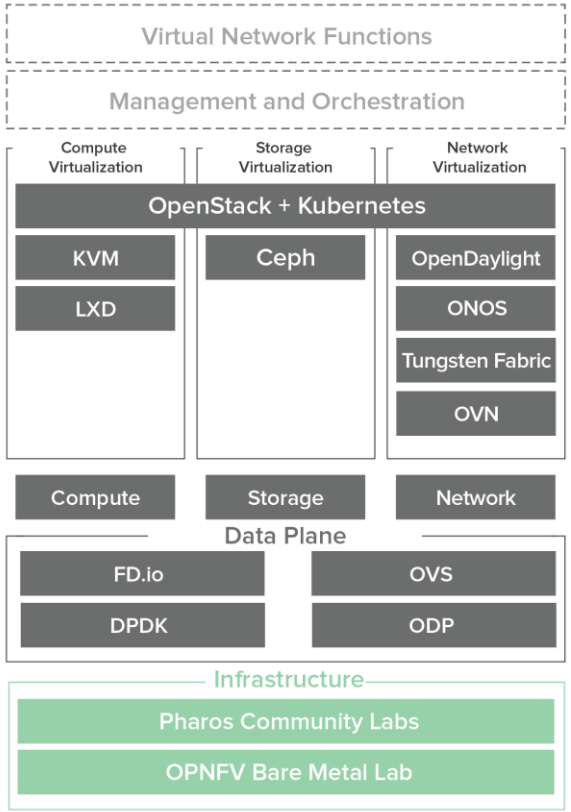
1. Create awareness for OPNFV test tools
   - Targeting users outside of NFV domain and telcos not active in OPNFV
   - Beneficial for most cloud operators and developers
   - Leverage the extensive tooling OPNFV has built over 4 years

2. Trigger a discussion about the evolution of the OPNFV test tools
   - How to evolve the test tools to address emerging use cases?
   - Learn from people outside of NFV domain about their needs

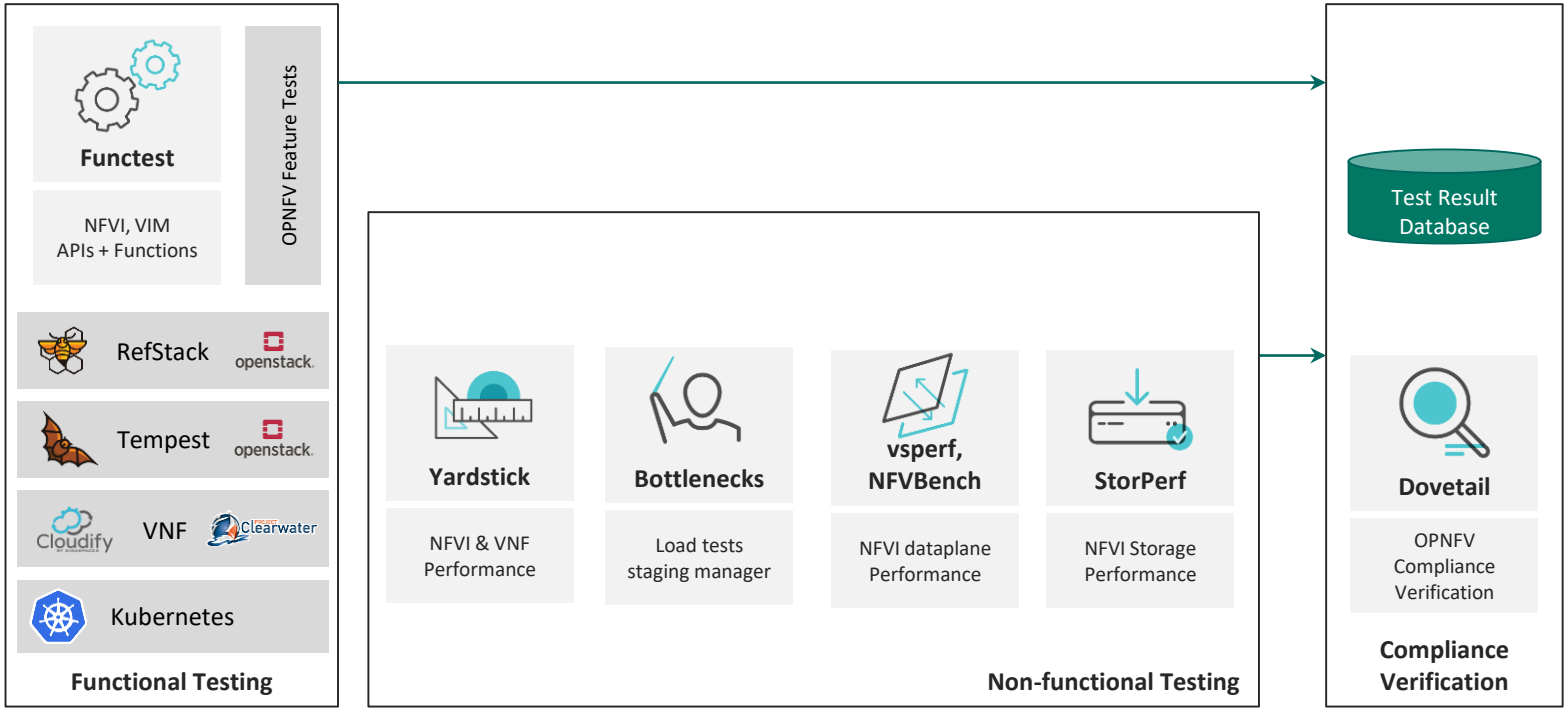
What does OPNFV do?



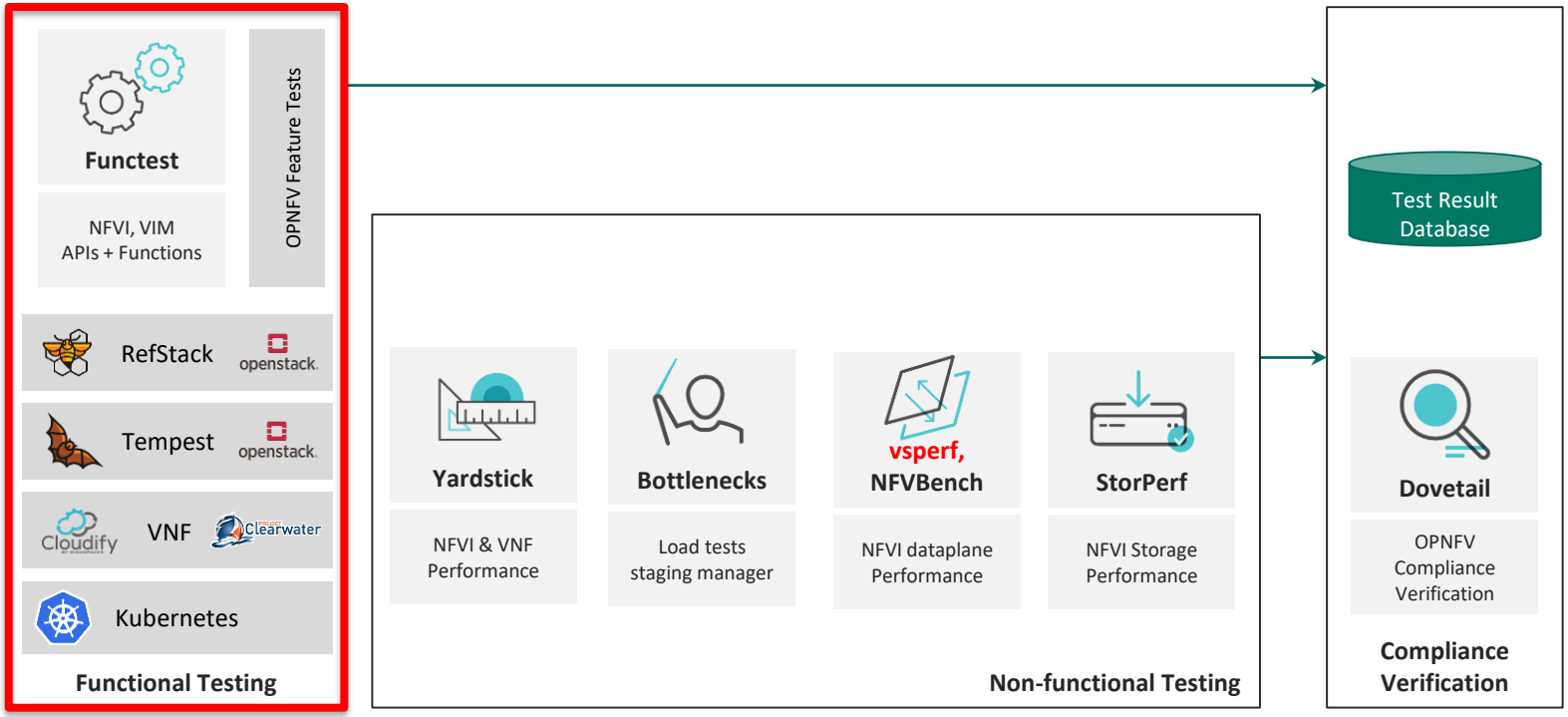
What does OPNFV do?



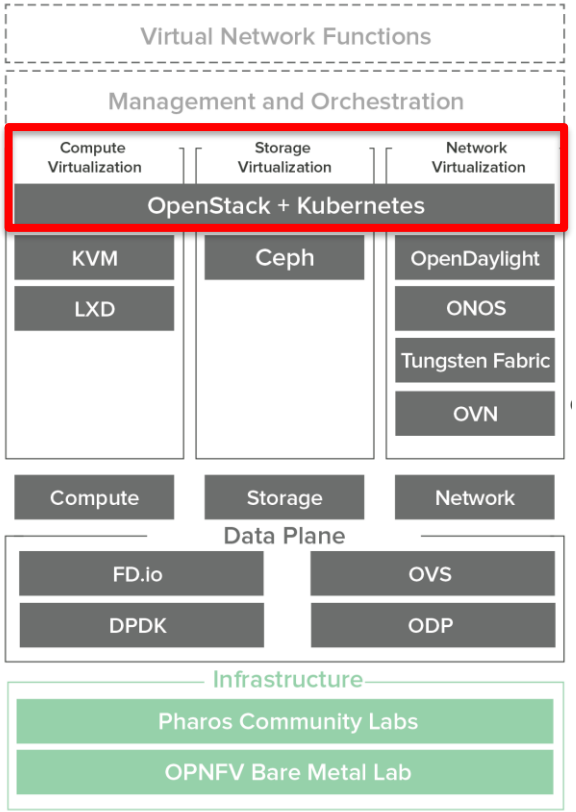
OPNFV Test Ecosystem



OPNFV Test Ecosystem



Functest



Description

Functional verification of OpenStack and K8s

Components tested

Cloud infrastructure control plane

Stage deployed

From patch set verification to release gating

Collected metrics

Pass / fail

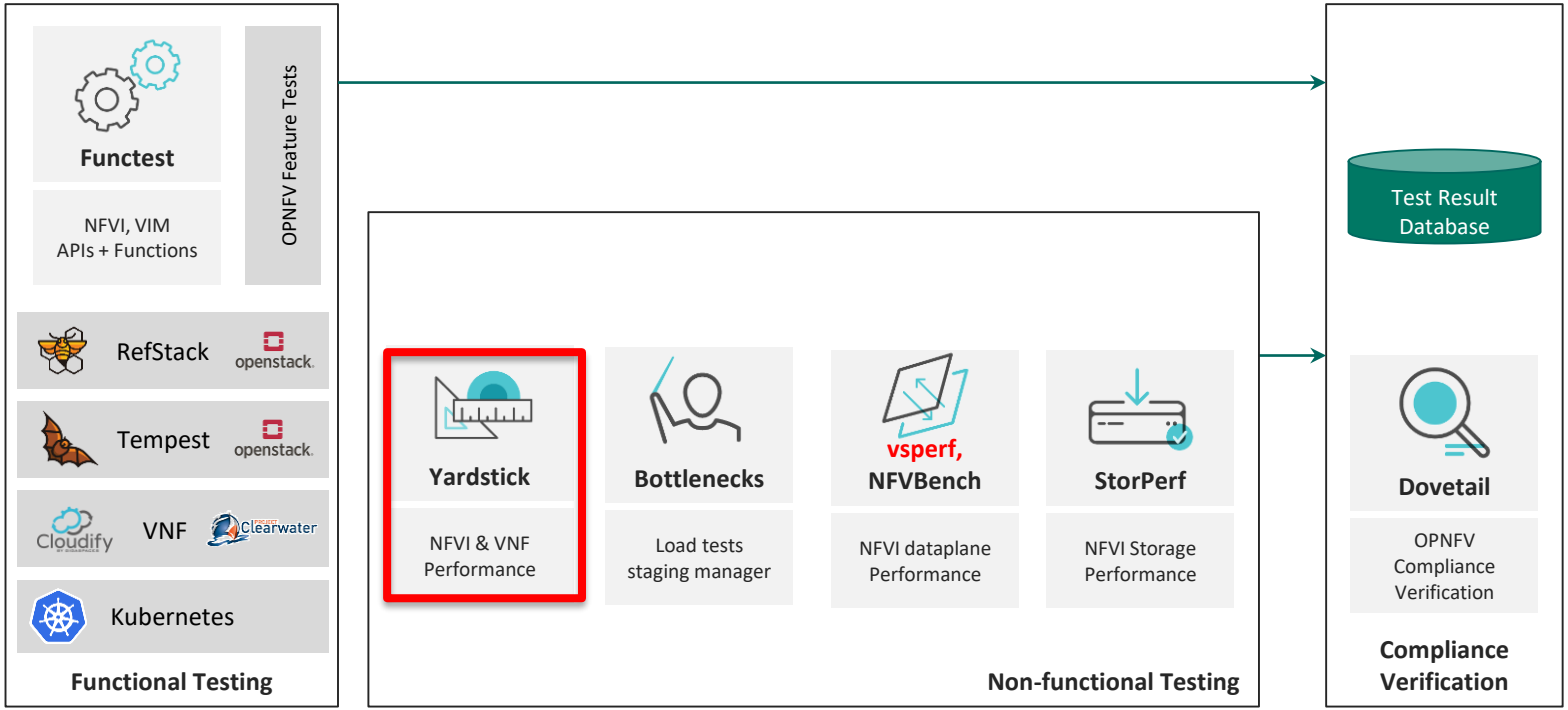
Project packaging/release

Multiple docker containers

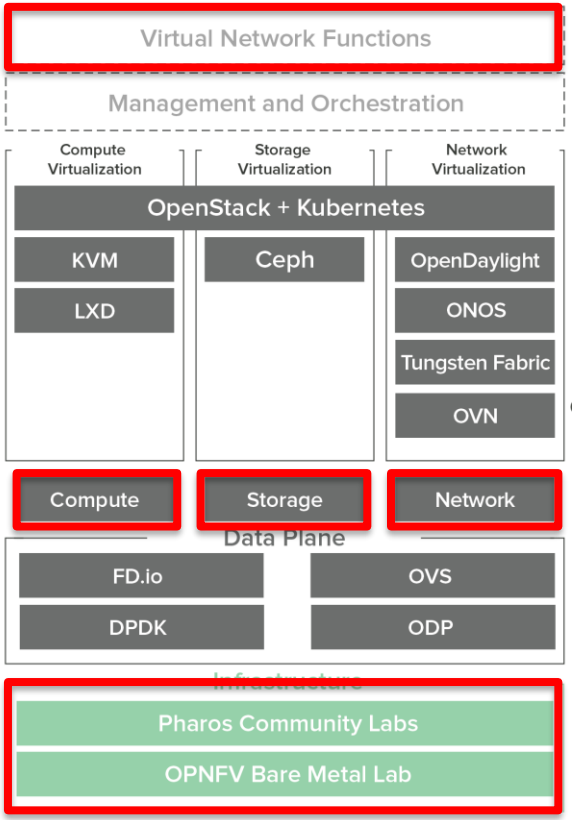
Extensibility

Build with extensibility in mind: based on Xtesting

OPNFV Test Ecosystem



Yardstick



Description

Infrastructure Verification and NFVI/VNF characterisation

Components tested

Cloud infrastructure resources

Stage deployed

CI and pre-production verification

Collected metrics

Performance metrics and pass/fail metrics (HA tests)

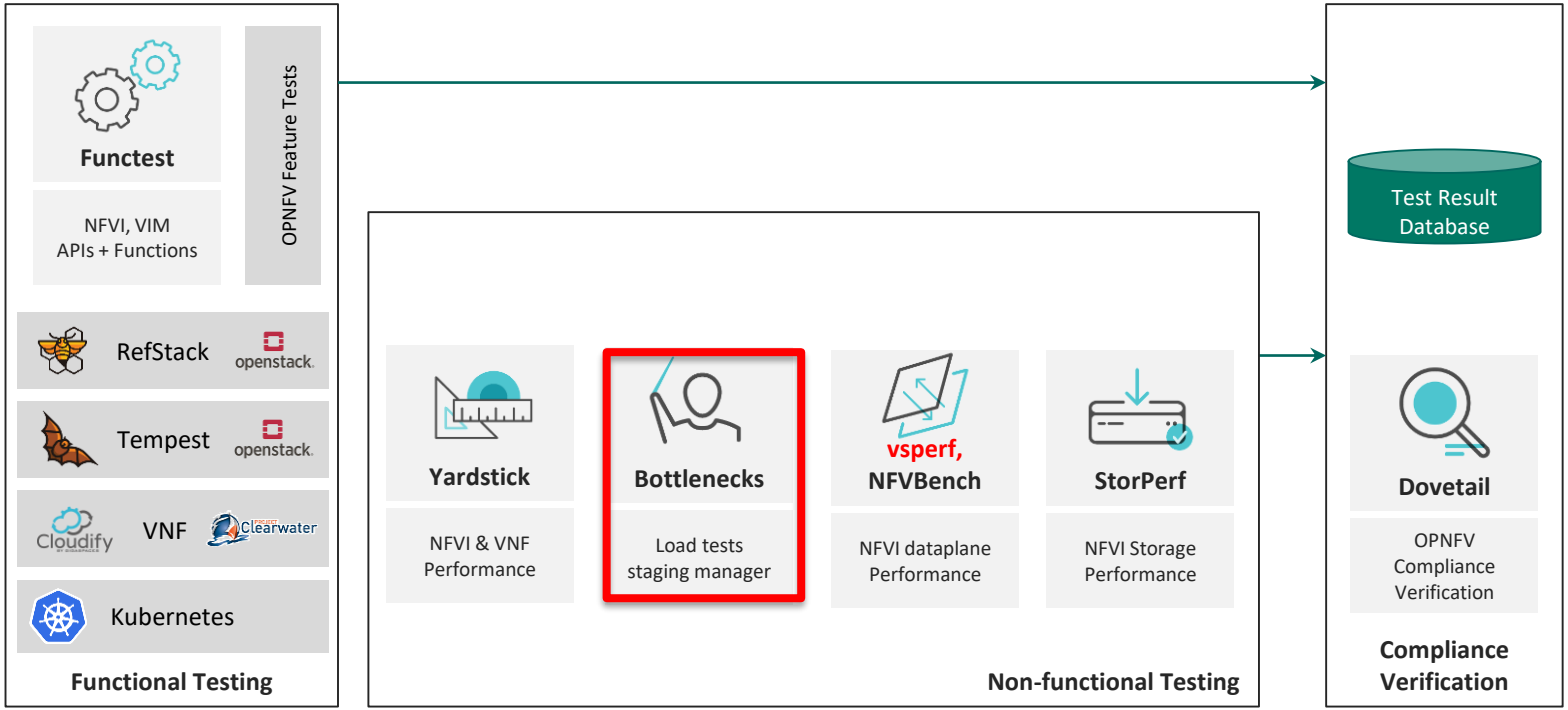
Project packaging/release

Docker container

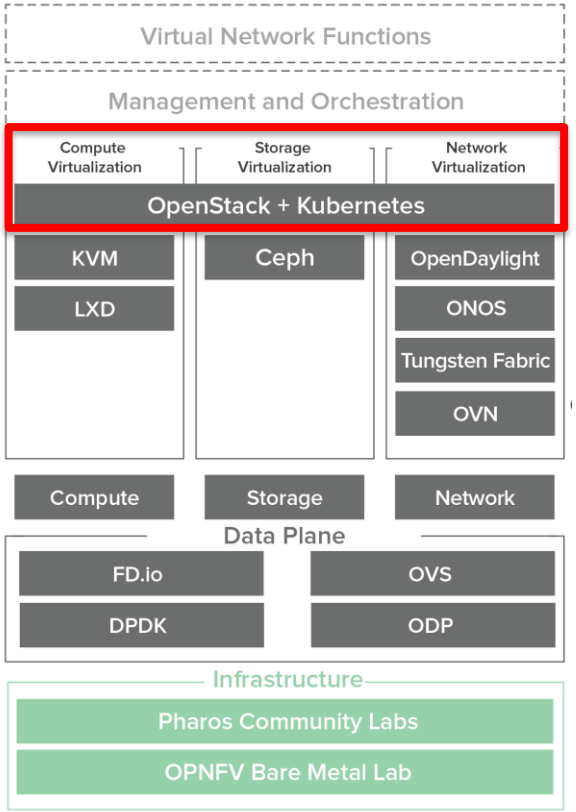
Extensibility

Test cases integrated as scenarios

OPNFV Test Ecosystem



Bottlenecks



Description

Simulates extreme or long term product usage

Components tested

Cloud infrastructure control plane

Stage deployed

CI and performance tuning of infrastructure

Collected metrics

pass/fail metrics

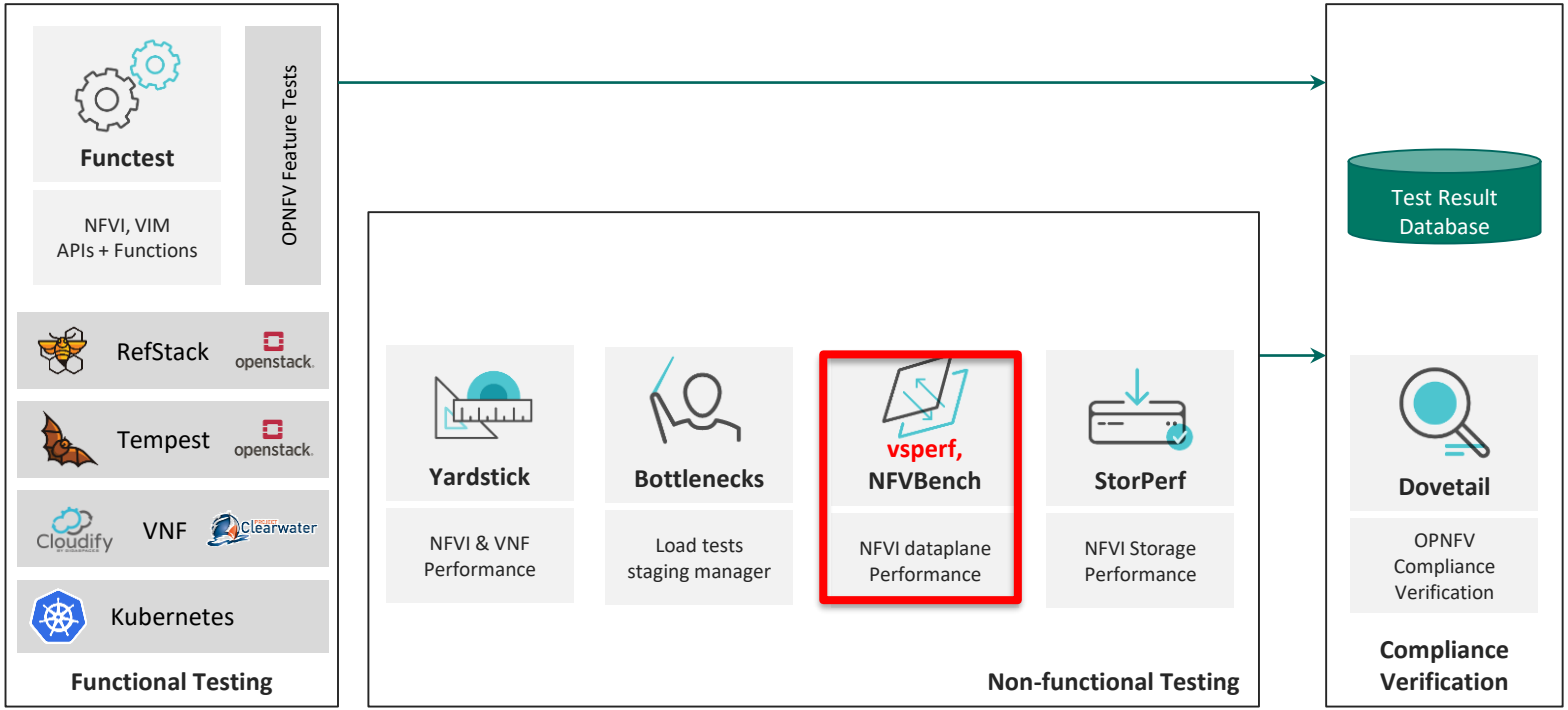
Project packaging/release

Docker container

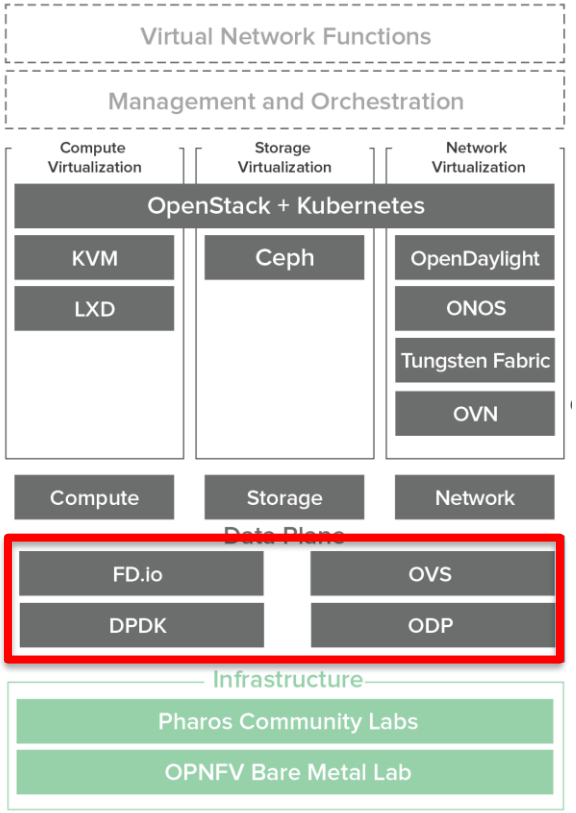
Extensibility

Test scheduler for other OPNFV tools, e.g. Yardstick, StorPerf

OPNFV Test Ecosystem



vsperf



Description

Optimizing switching technologies and NFVI data path components

Components tested

Virtual switch and packet processing components

Stage deployed

Pre-deployment evaluation

Collected metrics

Performance metrics as reported by traffic generators

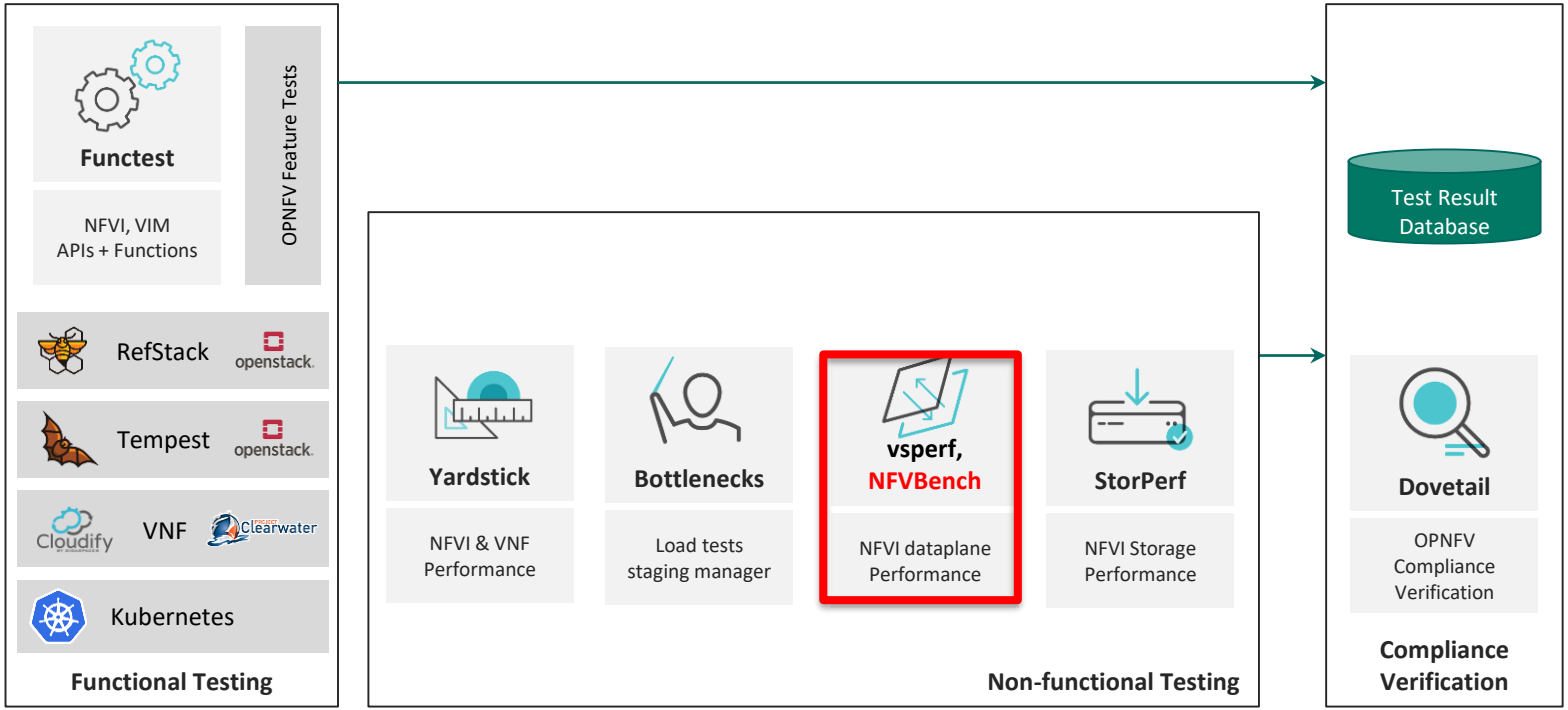
Project packaging/release

Source code package

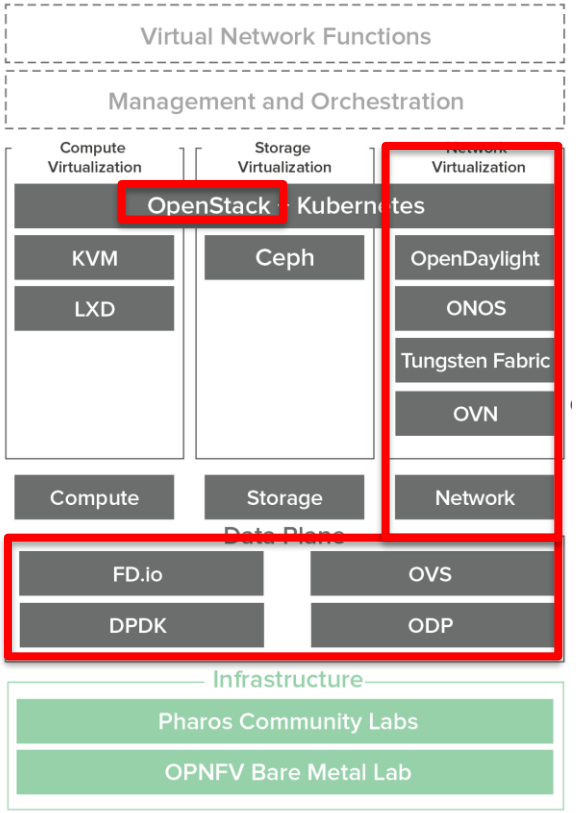
Extensibility

Integration of custom tests possible

OPNFV Test Ecosystem



NFVBench



Description

Full stack data plane performance measurements

Components tested

Full data plane stack: packet forwarding and virtualization components

Stage deployed

Pre-production, performance tuning and monitoring

Collected metrics

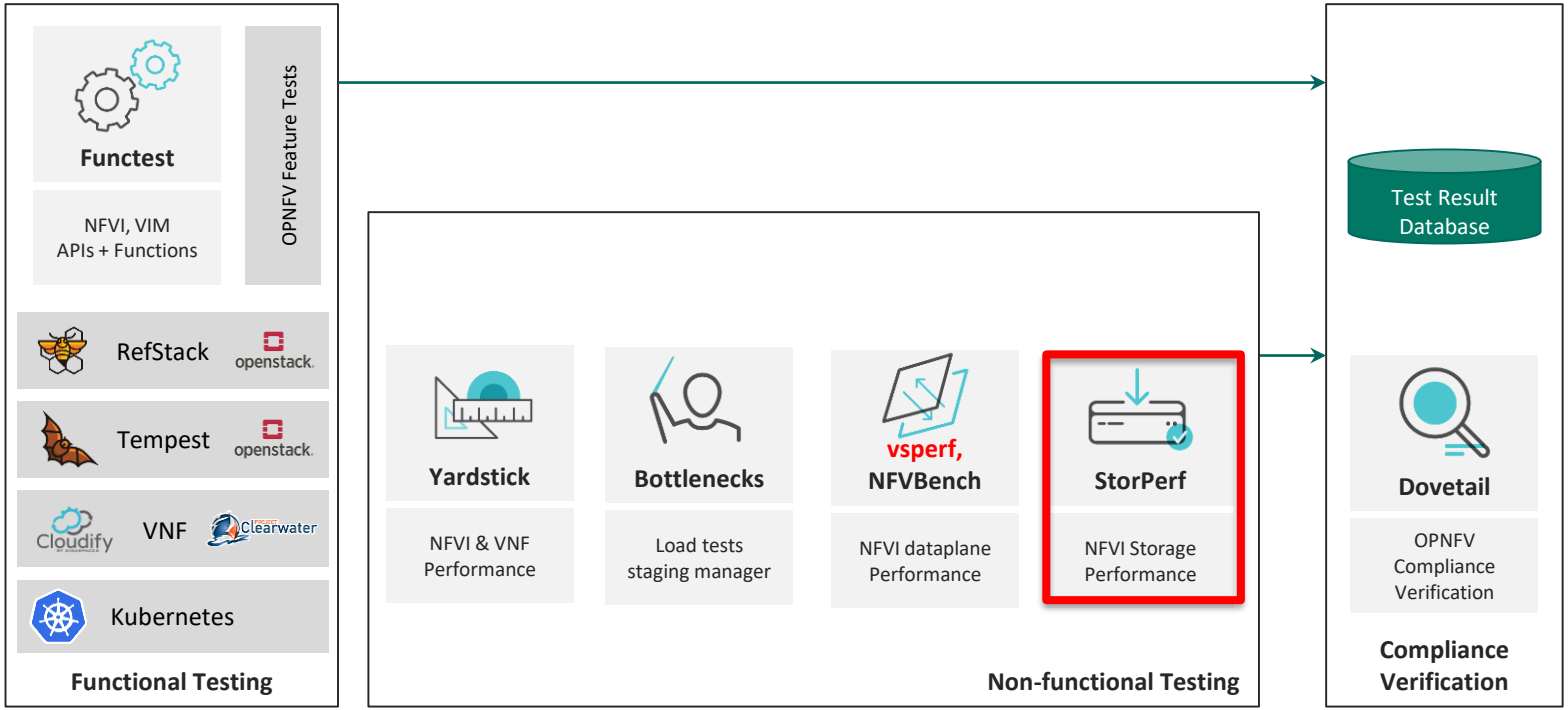
Metrics reported by T-Rex **Project packaging/release**

Single self-contained Docker container

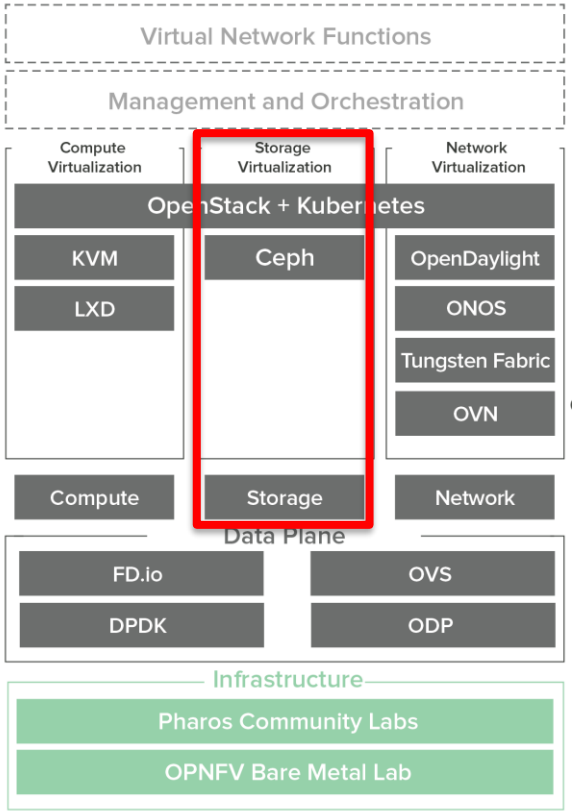
Extensibility

Wide range of parameters in PVP, PVVP, SR-IOV etc. scenarios

OPNFV Test Ecosystem



StorPerf



Description

Performance measurements of block & ephemeral storage at the VM level

Components tested

Storage subsystem

Stage deployed

Pre-production and lab environment

Collected metrics

Performance metrics in steady state, test failed if no stabilization

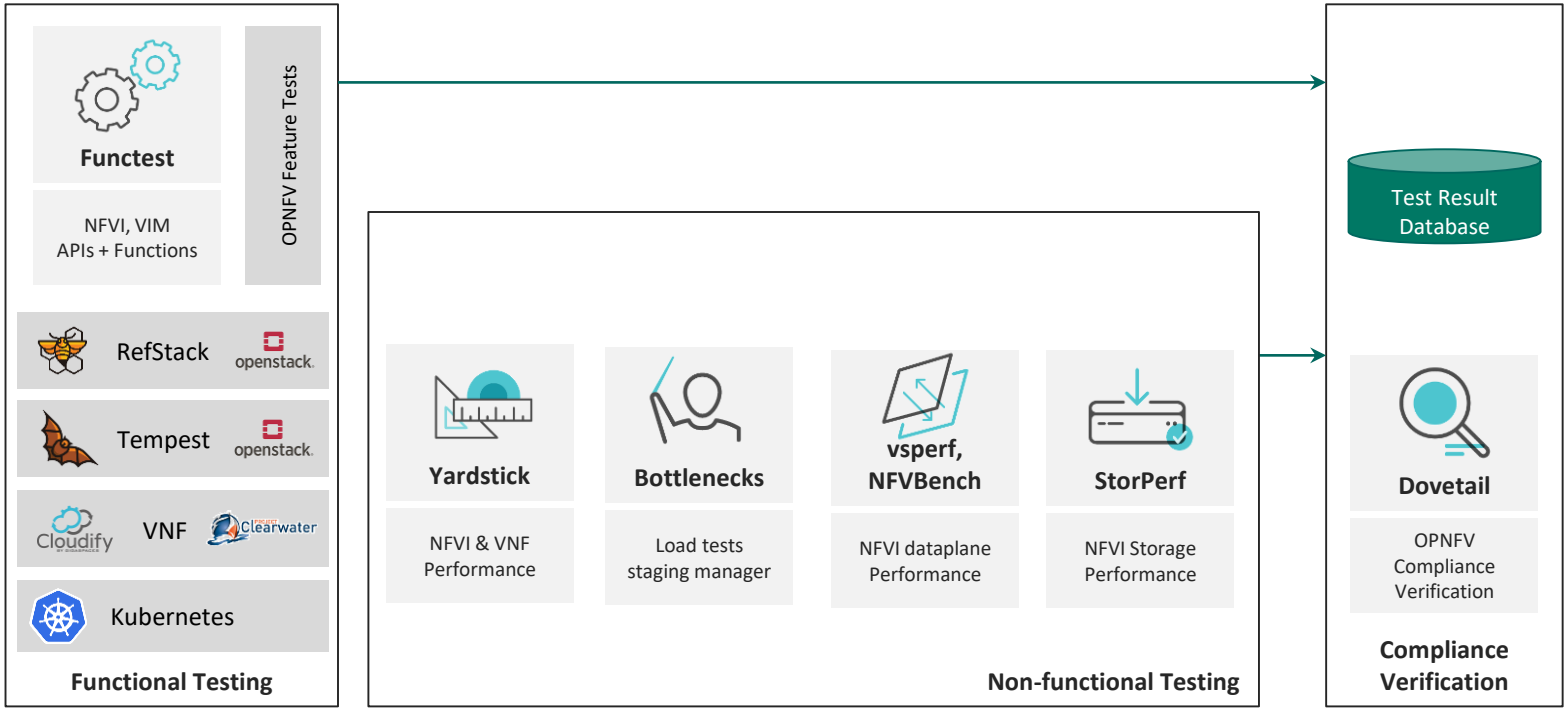
Project packaging/release

Docker container

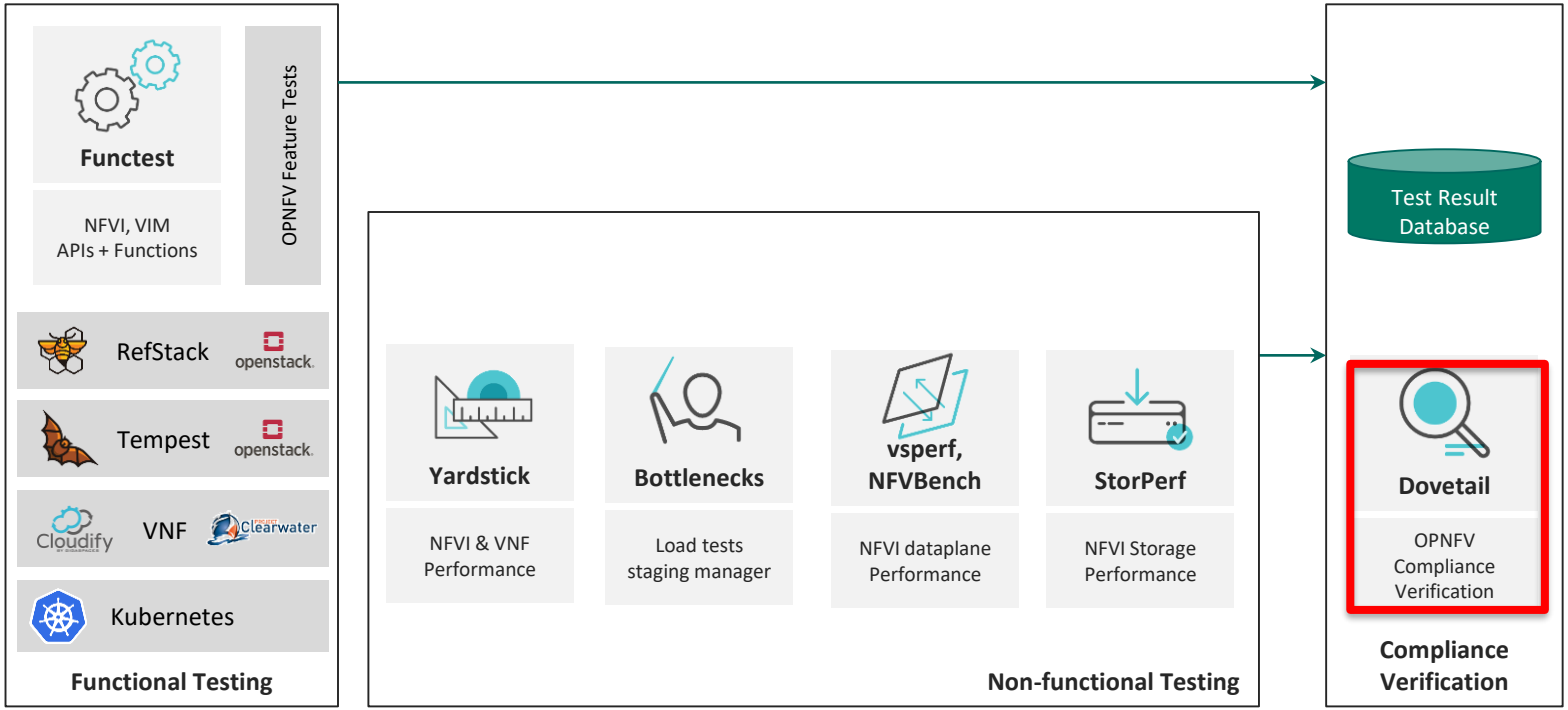
Extensibility

Wide array of parameter: e.g. nr of VMs, queue depth, I/O access pattern

OPNFV Test Ecosystem



OPNFV Test Ecosystem



OPNFV Compliance Program

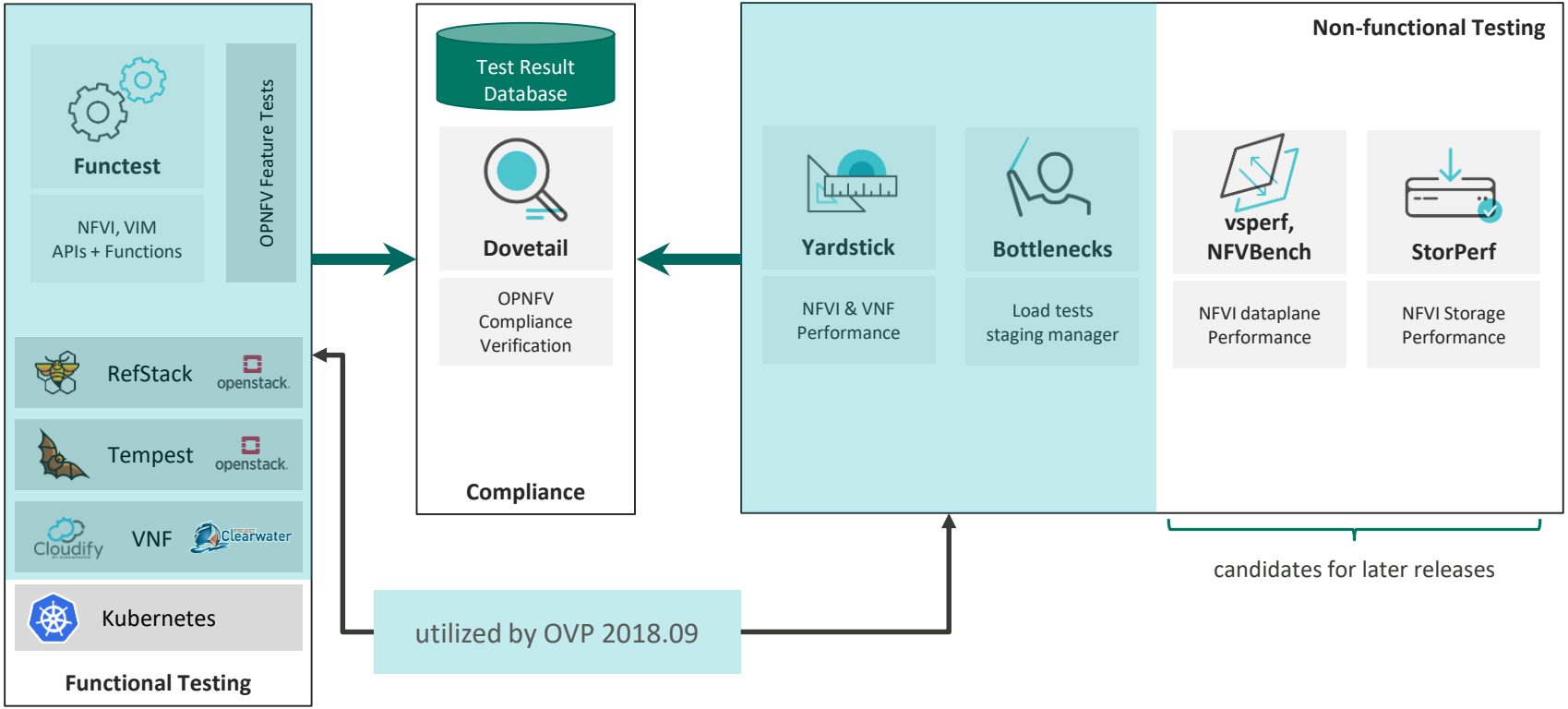
- OPNFV Verified Program (OVP) verifies that a commercial cloud platform exposes the same
  - key APIs,
  - behaviors, and
  - characteristics

  as a reference platform **defined through a specific selection of test cases**

- Main objective: Reduce vendor selection and application onboarding cost
  - Establish industry-accepted technical baseline
  - Simplify RFIs and RFPs

- Main components of OVP
  1. OPNFV test frameworks providing the actual OPNFV and upstream test cases
  2. Dovetail: Wrapper for OPNFV test tools and reporting tool

OPNFV Compliance Program



LINUX FOUNDATION
COLLABORATIVE PROJECTS

Addressing emerging use cases

Addressing emerging use cases

- OPNFV traditionally focused on NFVi data center scenarios
  - Medium to large scale deployments in centralized data centers
  - VNFs = legacy Network Functions in VMs

- Emerging use cases impose new requirements on test tools
  - Edge computing
  - Cloud native computing

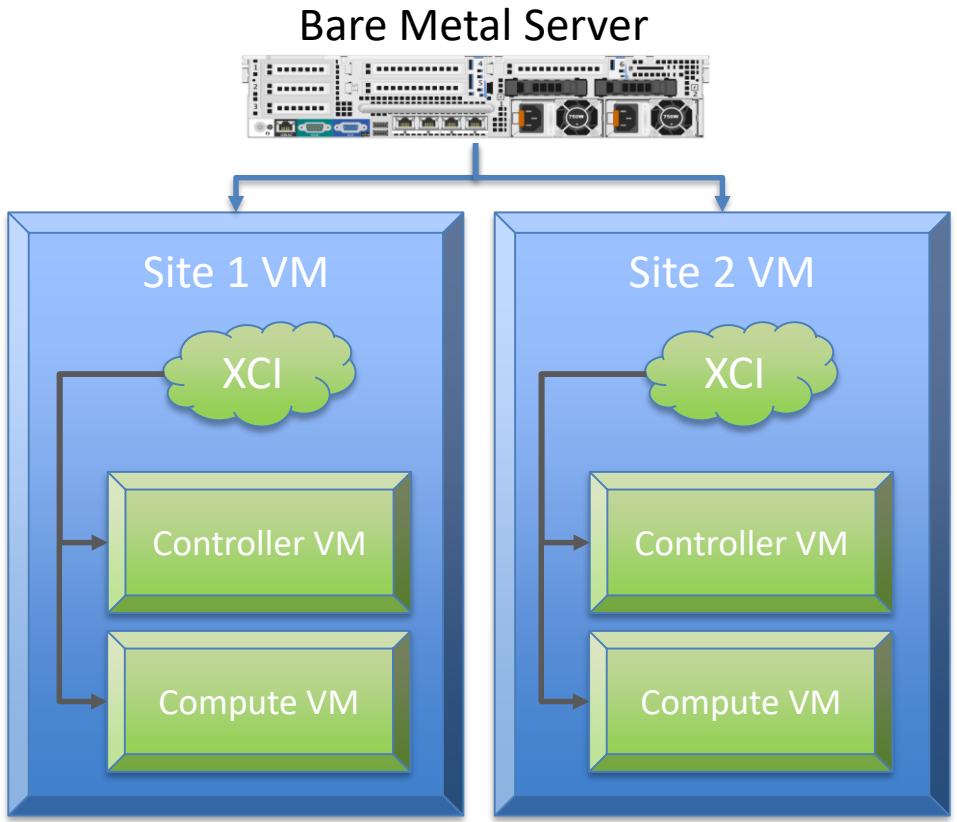
⇨ How to address those requirements?

Edge Computing

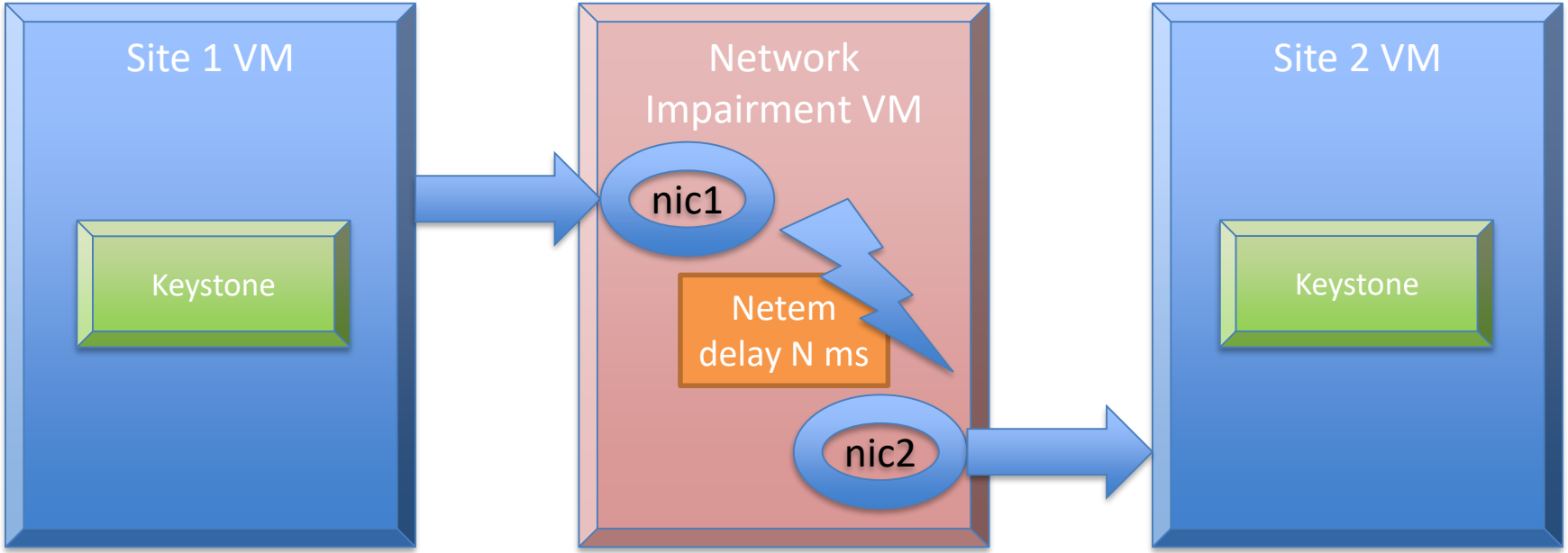
- Impact of edge computing on test tools and methods
  - Test topology
    - Automatic deployment of multiple sites
    - Inter-site connectivity
  - Consideration of networking effects
    - Control and data plane latency
    - Limited bandwidth, jitter, packet drops
  - Hardware resources
    - Limited resources in the edge: 1-4 servers

Virtual Edge in a Box

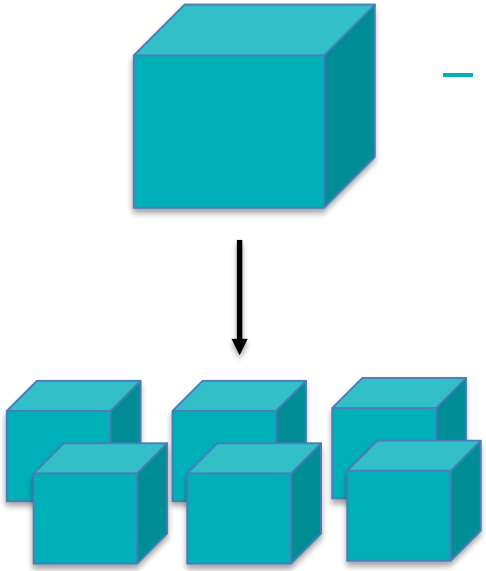
- OPNFV XCI
  - Mini flavor installs OpenStack from master in VMs
  - Can itself be in a VM
  - 2 full OpenStack environments in 1 server



Modeling of Edge Networking Environment



Cloud Native Computing

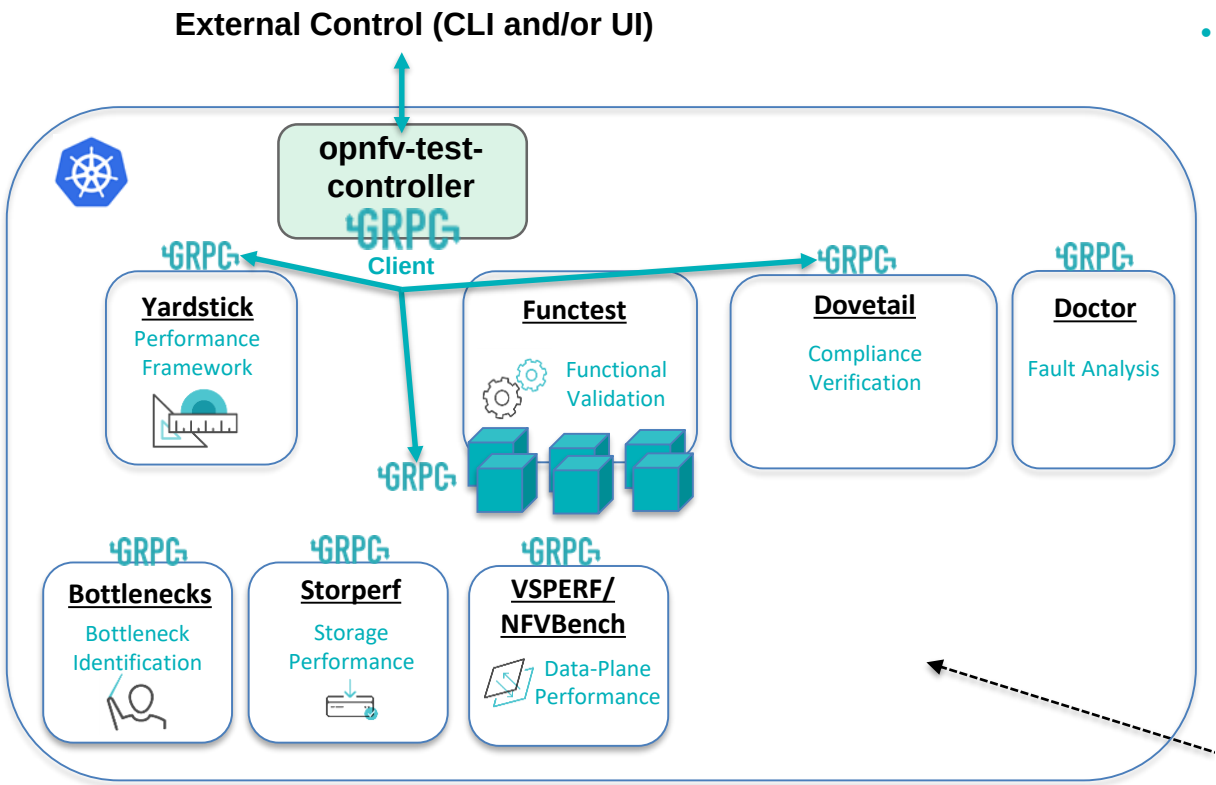


– Monolithic App

– Break down into smaller chunks

- Microservice architecture puts functionality into separate services:
  - Iterative development
  - Division of labor
  - Reduce single point of failure
  - Language/deployment flexibility
  - Build different apps using subsets of services

Cloud Native & OPNFV Test Projects



- **Consider cloud native for OPNFV test projects**
  - Package as micro-services
  - Many are already containerized
    - Functest divided into 8+
  - Add GRPC or REST server interfaces
  - Make actions more atomic within each
  - Orchestrate system level tests using different combinations of services/actions
  - Deploy all OPNFV test services in a single manifest potentially
  - Use tool-chains such as Spinnaker for CI/CD
  - Installer projects are also considering cloud native for some services

Summary

- Join us!
  - OPNFV test working group
    - https://wiki.opnfv.org/display/testing/TestPerf
  - OPNFV
    - https://wiki.opnfv.org/, https://www.opnfv.org/
  - OPNFV Verified
    - https://www.opnfv.org/
- Provide feedback and input!

Questions

opnfv-users@lists.opnfv.org

#functest

#yardstick #nsb

#bottlenecks

#nfvbench #vsperf

#dovetail

LINUX FOUNDATION
COLLABORATIVE PROJECTS

Backup slides

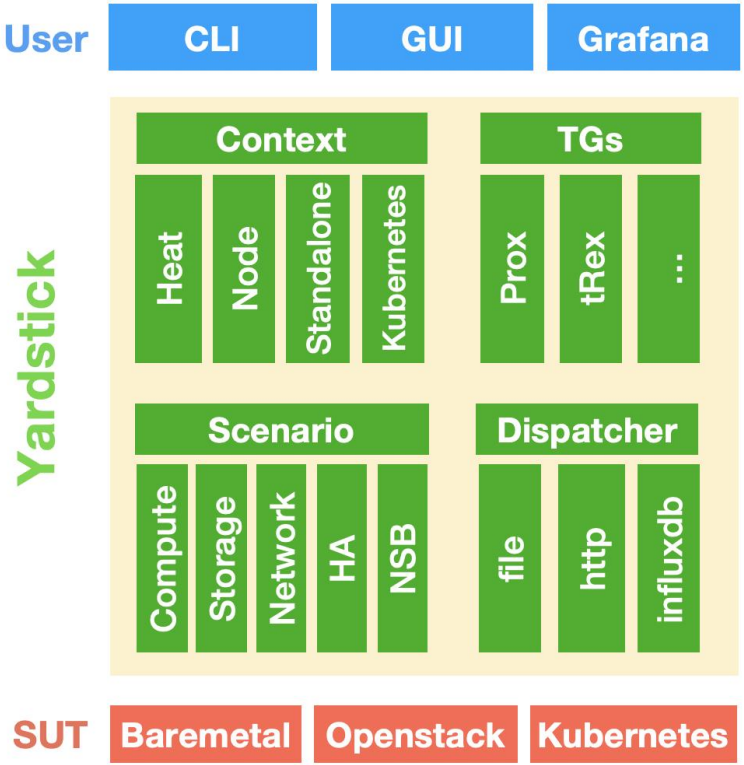
Functest in a nutshell

- Verify any kind of OpenStack and Kubernetes deployments (OPNFV model) or production environments
- Conform with upstream rules (OpenStack gate jobs and Kubernetes conformance tests)
- Ensure that the platforms meet Network Functions Virtualization requirements

Functest suites

- All functional tests as defined by the upstream communities (e.g. Tempest, neutron-tempest-api, Barbican, Patrole...)
- Upstream API and dataplane benchmarking tools (Rally, Vmtp and Shaker)
- Virtual Network Function deployments and testing (vIMS, vRouter and vEPC)

Yardstick

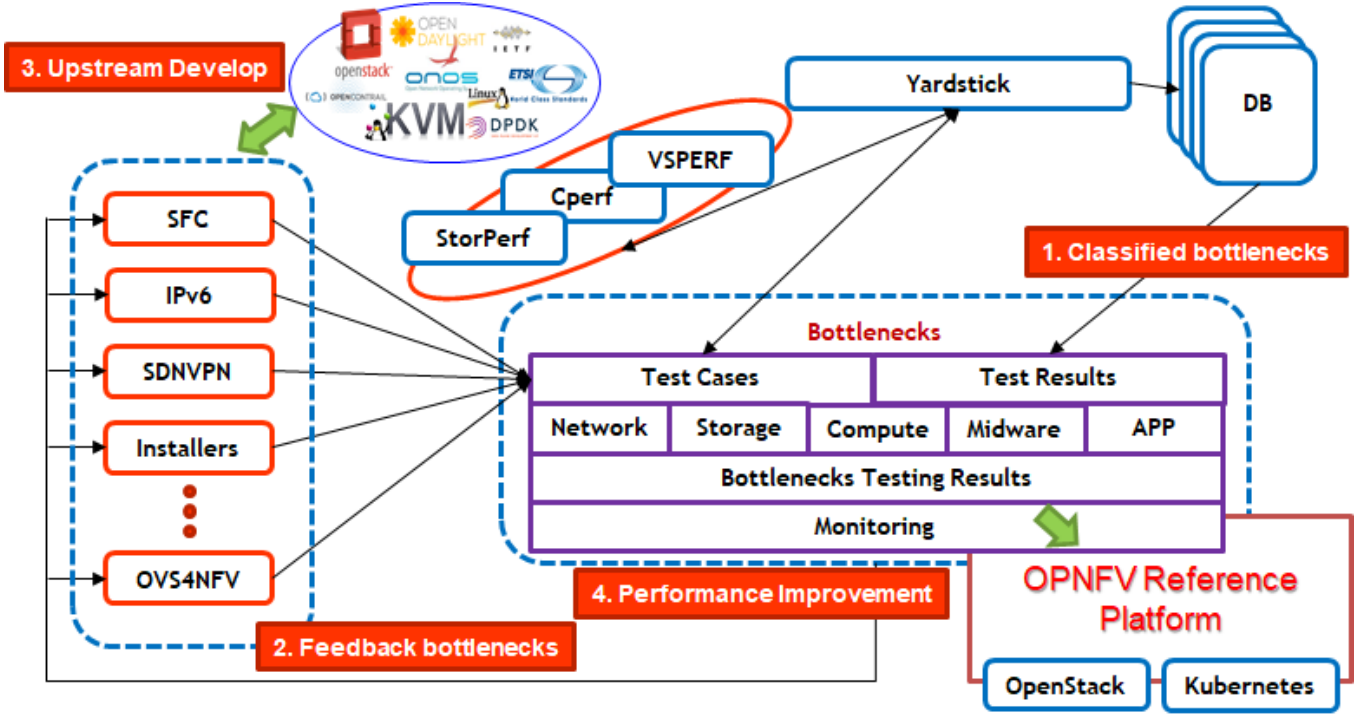


rify infrastructure compliance from
ual Network Function (VNF).

development of a **testing framework**,
uli to enable NFVI verification.

NSB (Network services benchmarking).

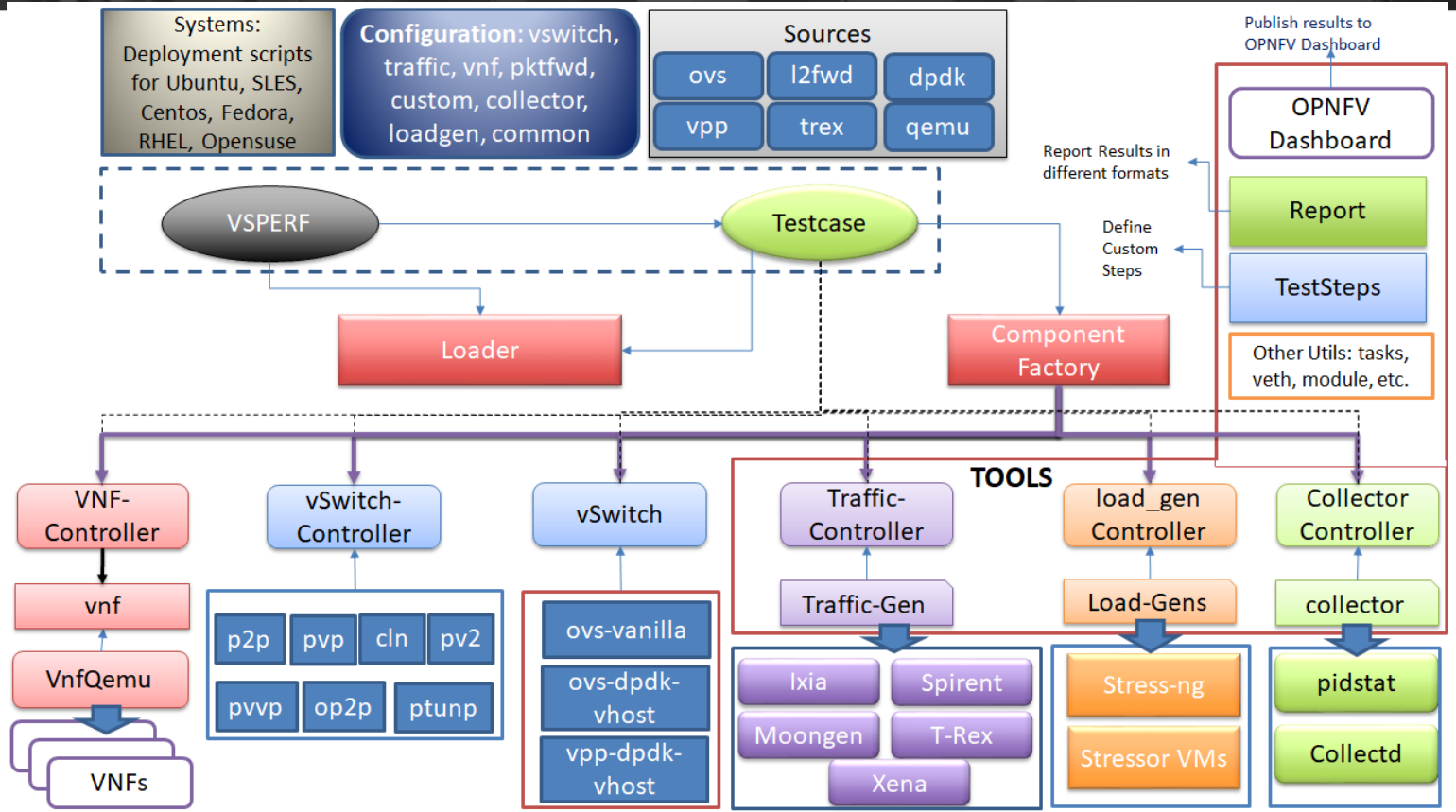
Bottlenecks



VSPerf

- Automated Framework for dataplane performance benchmarking,
  - Switching Technologies with Physical and Virtual Interfaces
- Configuration and control of topology, vswitch, VNF, traffic-generator and other software components are performed by VSPERF.
  - VSPERF provides the user the ability to choose the vswitch, Traffic-generator, VNF, etc.
- VSPERF is used as a tool for optimizing switching technologies, qualifying packet processing components and for pre-deployment evaluation of the NFV platform datapath.
- Virtual Switches:
  - OVS, VPP
- Traffic Generators
  - T-Rex, Spirent, Ixia, Xena, Moongen
- Deployment Scenarios
  - Phy2Phy, PVP, PVVP, Custom.
- VSPERF tests are defined and driven by Level Test Design (LTD) Specification.
  - VSPERF supports designing and implementing custom tests through its 'integration-tests' feature.
- VSPERF supports multiple modes:
  - Ex: Trafficgen-off mode: VSPERF will do setup of DUT, but no control the traffic-generator.

VSPerf


Systems: Deployment scripts for Ubuntu, SLES, Centos, Fedora, RHEL, Opensuse
Configuration: vswitch, traffic, vnf, pktfwd, custom, collector, loadgen, common
Sources
ovs
l2fwd
dpdk
vpp
trex
qemu
Publish results to OPNFV Dashboard
OPNFV Dashboard
Report Results in different formats
Report
Define Custom Steps
TestSteps
Other Utils: tasks, veth, module, etc.
VSPERF
Testcase
Loader
Component Factory
TOOLS
VNF-Controller
vnf
VnfQemu
VNFs
vSwitch-Controller
p2p
pvp
cln
pv2
pvvp
op2p
ptunp
vSwitch
ovs-vanilla
ovs-dpdk-vhost
vpp-dpdk-vhost
Traffic-Controller
Traffic-Gen
Ixia
Spirent
Moongen
T-Rex
Xena
load_gen Controller
Load-Gens
Stress-ng
Stressor VMs
Collector Controller
collector
pidstat
Collectd

NFVBench

- Tool that provides an automated way to measure the network performance for the most common data plane packet flows on any OpenStack system.
- Designed to be easy to install and easy to use by non-experts
  - there is no need to be an expert in traffic generators and data plane performance testing.
- The tool is built around the open source T-Rex traffic generator and is useful for testing a full NFVI subsystem that includes ToR switches.
- The key areas of strength for NFVbench are in its automation of the traffic generator, ability to test a full subsystem, and to perform this testing on a production cloud.