OVP 2.0 Technical Ideas

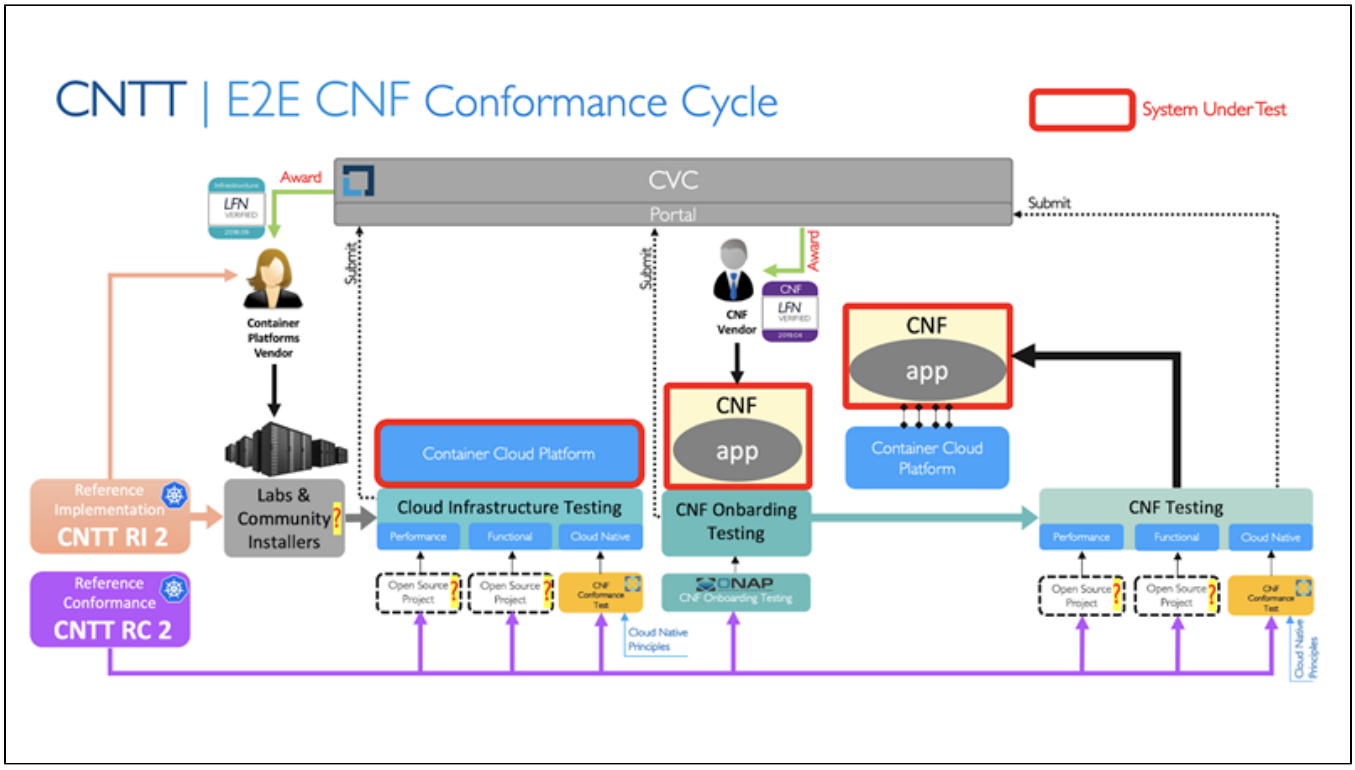
All members are encouraged to contribute directly into this working draft. May want to break this apart into one component per page...Buck

- E2E CNF Conformance Cycle
  - Test Categories (NOT for MVP)
  - Adding Kubernetes to ETSI NFV/MANO
- Evolving architecture from VNFs to CNFs
- Compliance & Verification Program
- Verification and Certification Process
- Cloud Native Network Function (CNF) Best Practices Criteria

E2E CNF Conformance Cycle

(Below is from Jim's mail to OVP members (3/4/2020). Will be discussed on next meeting (3/9/2020).

In the below diagram the "?" indicate processes that need definition and refinement for the differing execution environment of OVP Phase 2.



Test Categories (NOT for MVP)

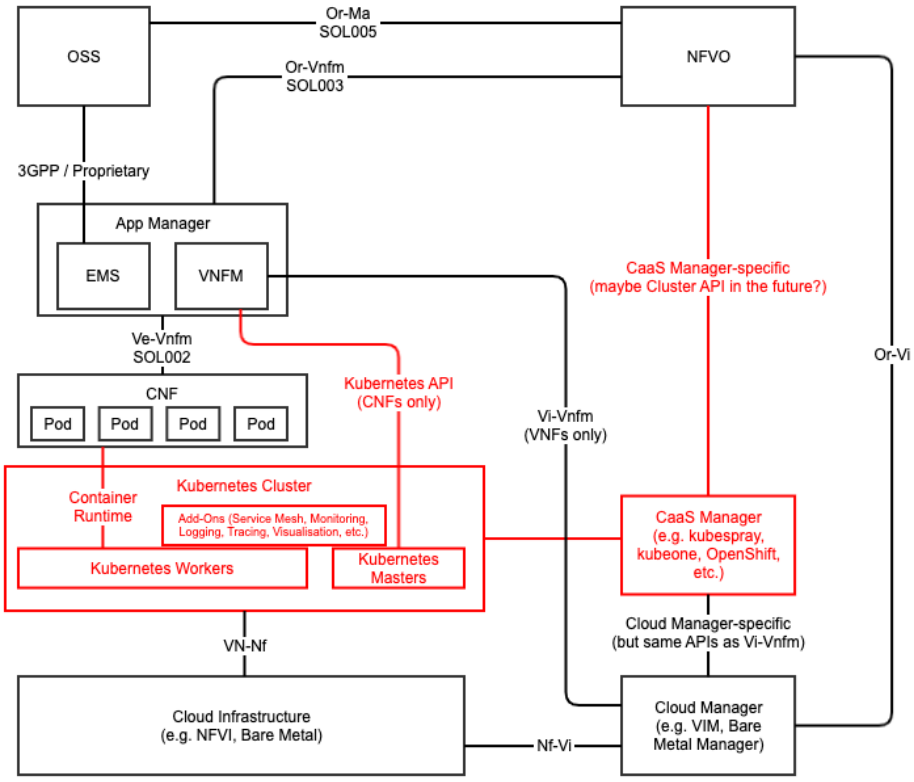
- **Cloud Infrastructure Validation Testing**
  - Cloud Native ready post-install
  - Security patches
  - Functional
  - Manifest validation
  - Performance
- **CNF Compliance Testing**
  - Packaging: (Helm v3.0, TOSCA, HEAT)
  - Cloud Native.
  - On-Boarding and Lifecycle Management.
- **CNF Validation Testing:**
  - API/Interfaces Testing
  - Major subsytem connectivity
  - Functional Testing
  - Performance Testing
  - Interoperability (?)

- **Platform Testing**
  - Management Stack (K8s, ONAP) testing.
  - integration with the infrastructure.

Adding Kubernetes to ETSI NFV/MANO

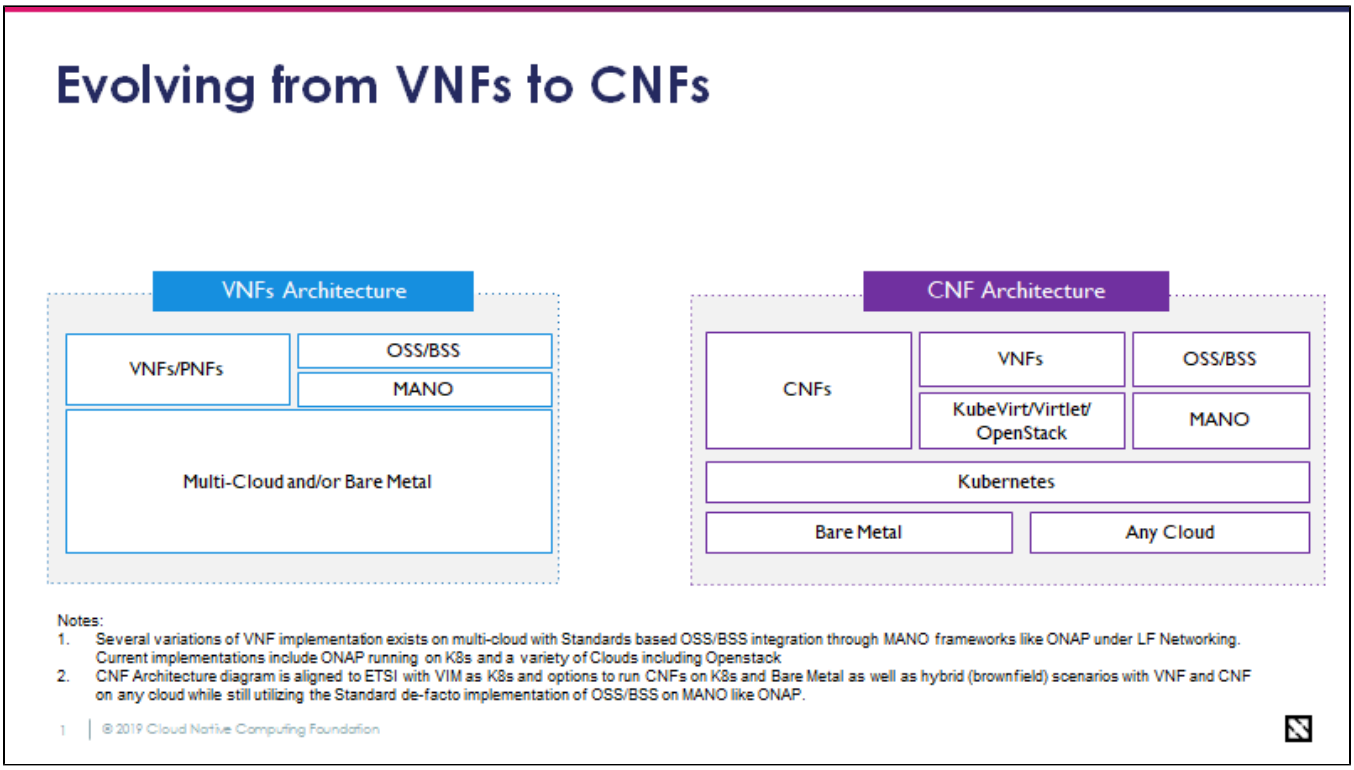
Thoughts from Tom Kivlinas to how Kubernetes and the Kubernetes LCM fits into the existing NFV/MANO architecture, to help identify the interfaces and capabiltiies we do want to test in OVP2.0.

We should probably look to add key components such as SDN controllers and other functions that are required for the end-to-end functionality of a network service, if that's what we want to be testing (as per Catherine Lefevre's comments in the call on 09 Mar 2020 .



Evolving architecture from VNFs to CNFs

(Olivier - Add short description and I would like to gain clarity around CNF orchestration. Specifically, we should clarify that CNFs are to be orchestrated by Kubernetes and that MANO is utilized specifically for VNFs. If this is not the case, document why? Add clarity to the CNF Architecutre below.



Compliance & Verification Program

OVP2.0 E2E Compliance and Verification Program adds additional collaboration with the CNCF in the testing and verification of Cloud Native Network Functions.

Catherine - comments and questions regarding the E2E Compliance & Verification Slide (below):

#1 CNF Conformance has been removed from the CNF boxes in alignment with the CNF Conformance definition - https://github.com/cncf/cnf-conformance

#2 VIM (Openstack) and VIM (K8S) are part of CNTT NFVI boxes in alignment with https://github.com/cntt-n/CNTT/tree/master/doc/ref_arch

#3 NFVi, PFNi NFVI, PFNI – spelling aligned with ETSI

#4 Remove "for CNFs and" from "Hybrid (PNF/VNF & CNF) Compliance and Verification with OVP Ph2 in partnership with CNCF Conformance tests ~~for CNFs and~~ for K8s in CNCF'

Additional changes to be made:

#5 Picture under "Today" should include PNF, not only VNF

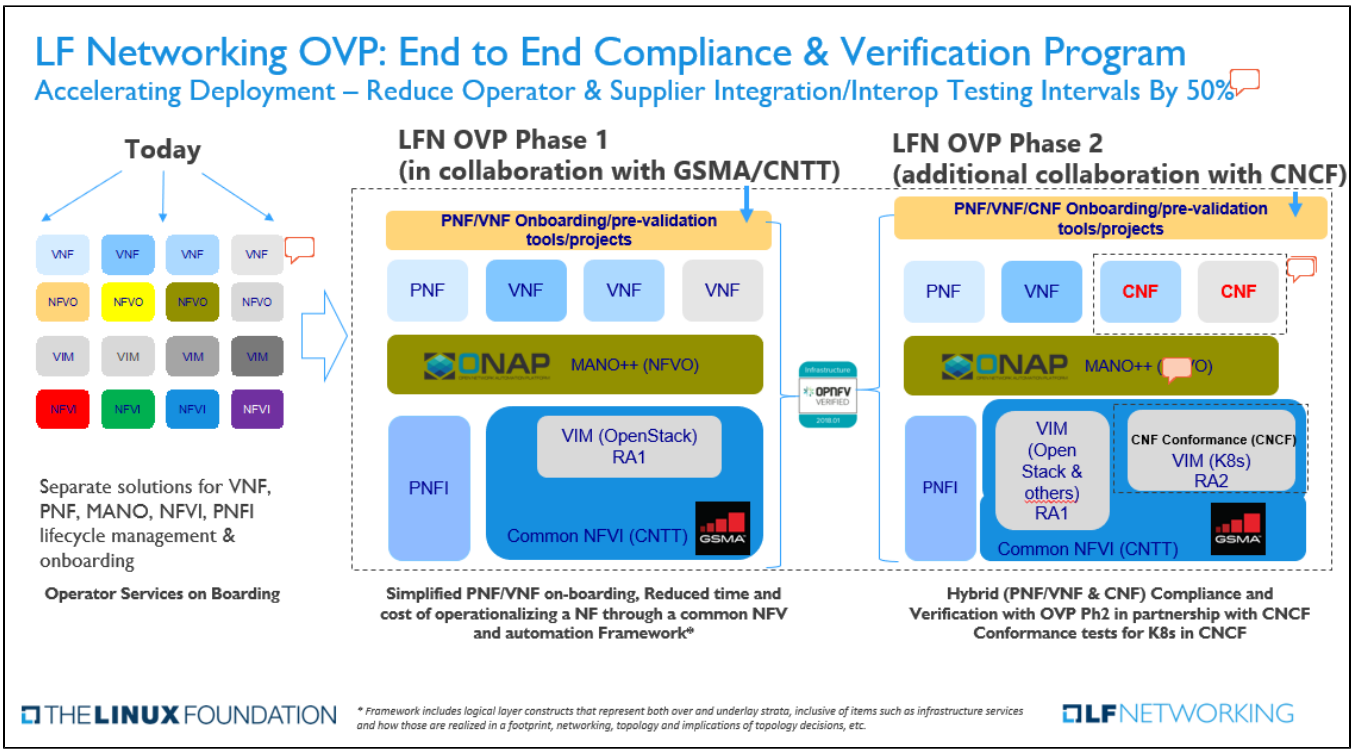
Questions:

#6 Reduce Testing Intervals by 50% - How "50%" have been determined?

it was an early estimate based on discussions with Industry analysts and consultants. They looked at the POC/Interop phase for several carriers and the fact that VNF/CNF would be tested with NFVI across at least 2 vendors gives that number. BUT we would love to see actual numbers as a test case if possible.

#7 MANO++ - is it to confirm that ONAP is Mano++ meaning that ONAP is providing additional capabilities than Mano functions i.e. Control Loops, etc.?

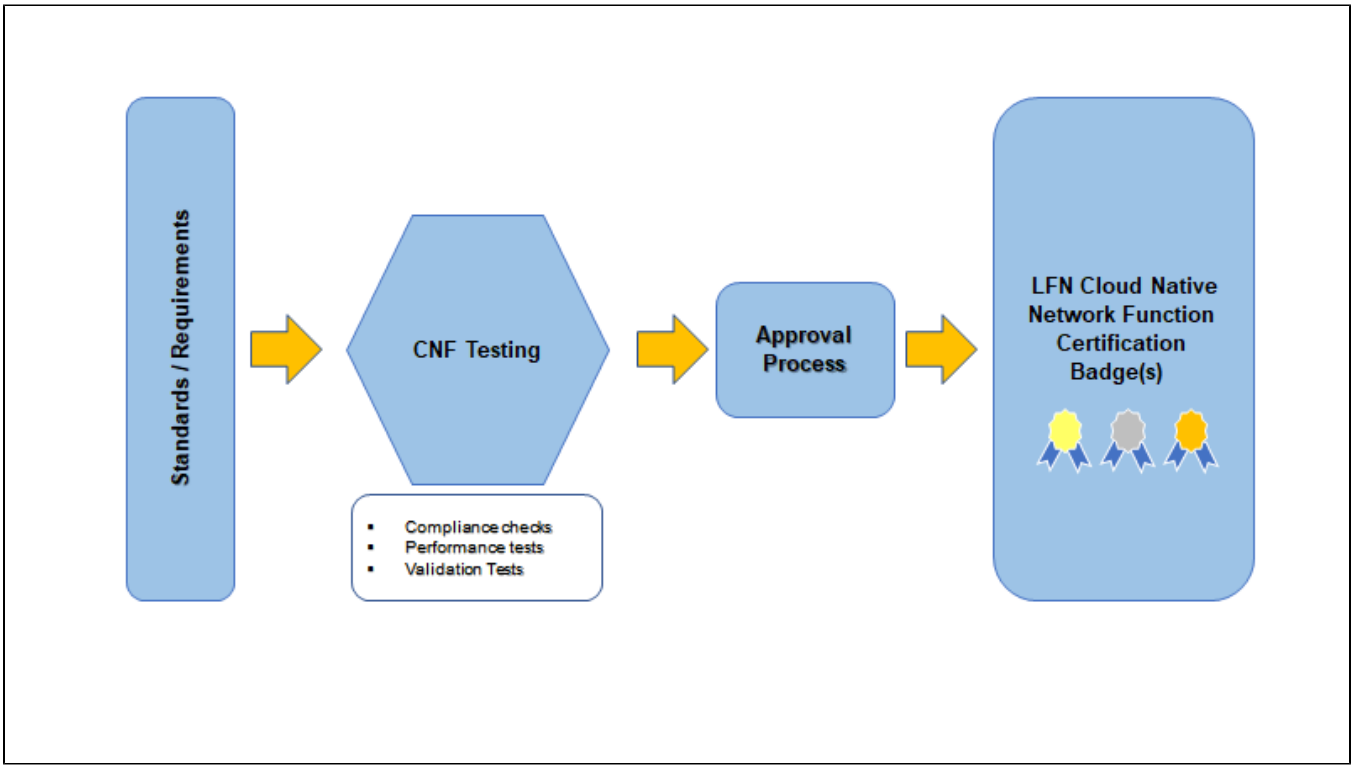
Yes, That was the assumption. Closed loop control, Analytics etc..



Verification and Certification Process

OPV 2.0 will provide an open source test suite which enables both open and closed source CNFs to demonstrate conformance and implementation of cloud-native practices. Compliance to the verification process will result in a LFN Cloud Native CNF badge(s). The LFN CNF badges should provide value for both operators and commercial vendors.

(We should outline the types of testing as well as clarify where these tests belong. Workload testing, Platform testing, etc.)



Cloud Native Network Function (CNF) Best Practices Criteria

Propose that the below should be aligned, agreed, and later decomposed into testable items. We should sort with mandatory first and optional (but still verifiable) criteria at the end. Organize into levels of Gold, Silver, Bronze if this makes sense. What will the value of a Bronze be for operators and vendors?

	CNF Best Practice Criteria
1	CNFs are decomposed into loosely coupled collection of services (e.g. use micro-services design).
2	Micro-services should, where possible, be independent of the host operating system.
3	Micro-services should utilize well defined declarative APIs
4	Micro-services should be designed with a clean separation between stateful and stateless services (have separation of state storage and business logic)
5	Clear separation should exist between micro-services (e.g. use of containers)
6	Micro-services should provide a mechanism to verify container content. Containers should not require root level permissions. It should be possible to secure API access to a micro-service using common methods.
7	Micro-services should be resilient (self-healing and distributed) • A strategy for self-healing accompanies the micro-service • Is compatible with declarative configuration and container orchestration
8	Micro-services should be elastic (vertical and horizontal scaling in response to load) • A strategy for auto-scaling accompanies the micro-service • Is compatible with a declarative configuration and container orchestration
9	Micro-services should be dynamic - where possible, have a small footprint and fast startup time enabling efficient use of resources and rapid scaling.
10	Micro-services may utilize a service mesh for service-to-service communication over a network
11	Micro-service/Container life-cycle is managed by an orchestrator (e.g. K8s)
12	Micro-services utilize immutable infrastructure:
A	• Infra is easily produced and repeatable (automated)
B	• Infra is consistent (infrastructure elements are identical each time they are implemented)
C	• Infra is disposable (create, destroy, replace, resize, etc.)
D	• Configuration of infra is protected from changes after deployment
E	• Deployment process of infra is versioned, automated, and all dependencies packaged together with the application during build phase and then used in deployment
F	• Observable (infrastructure elements externalize their internal states to allow metrics, tracing, and logging)