**Project Name:**

* Proposed name for the project: Octopus
* Proposed name for the repository: TBD
* Project Categories:
  + Documentation (developers user guide)
  + Requirements ( guideline and high level requirement for integration)
  + Integration & Testing (main part)
  + Collaborative Development (affecting all upstream projects)

**Project description:**

**Problem Statement:**

OPNFV will use many upstream open source projects to create the reference platform. All these projects are developed and tested independently and in many cases, not have use cases of other projects in mind. Therefore it is to be expected that integration of these projects probably will unveil some gaps in functionality, since testing the OPNFV use cases needs the interworking of many upstream projects. Thus this integration work will bring major benefit to the community.

Therefore the goal of the CI project – Octopus – is to quickly provide prototype integration of a first set of upstream projects. Step by step this later will be evolved to a full blown development environment with automated test and verification as a continuous integration environment, supporting both, the parallel evolutionary work in the upstream projects, and the improvement of NFV support in this reference platform.

**Summary**

The CI project provides the starting point for all OPNFV development activities. It starts by integrating stable versions of basic upstream projects, and from there creates a full development environment for OPNFV including automatic builds and basic verification. This is a very complex task and therefore needs a step by step approach. At the same time it is urgent to have a basic environment in place very soon.

The following list shows all necessary tasks for creation of the final environment and gives some idea how the work will be approached. More details need to be worked out during the project lifetime; the appendix provides some more details that are already available.

* **Select a basic set of upstream projects to be integrated in the first step**   
  *we will start with the list from the F2F (see below) and adjust if necessary*
* **Select a basic hardware definition for the build servers and for the basic verification**   
  *Linux Foundation provides the initial environment*
* **Select the tools for code repository, build processes and automation**   
  *Linux Foundation provides the initial tools and environment*
* **Define the image formats for code deployment in the test environment**   
  *Start with a quick solution and later decide for a format allowing easy deployment for developers and automated verification*
* **Define the branching rules for the code repositories**   
  *this will be defined by TSC in the development process and used here*
* **Provide the code repositories**   
  *part of Linux Foundation support*
* **Integrate the basic set of upstream projects**   
  *Main part of the first step of the project*
* **Provide necessary OPNFV glue code to integrate different upstream project codes**   
  *necessary in main part of the project*
* **Implement build process for developers**   
  *start off with basic build tools and adjust step by step. If necessary, a separate project for build environment can be established*
* **Implement automatic build process on central servers**   
  *as before, start with basic tools and adjust step by step or via separate project*
* **Provide or implement basic test objects** (VNFs and VNFM, NFVO)   
  *will be done in project “VNF deployment test cases”*
* **Implement automatic process for basic verification of the build images on central servers**   
  *as above, start with basic tools and adjust step by step or via separate project*
* **Implement the process for continuous upgrade to newer versions of upstream projects**   
  *cover by project “Simultaneous Release” and development process*
* **Implement the process for adding more upstream projects to the automated build and verification**   
  *part of later stages of the project*
* **Execute the continuous automated builds and basic verification**   
  *part of later stages of the project*

The above topics interlink to each other very closely.

**Basic set of upstream projects**

The initial set was already identified during face-to-face on October 2, 2014:

* Open Stack - subprojects to be selected
* KVM / QEMU / libvirt (Support under Nova)
* Open Daylight (Support along with Neutron)
* Linux Kernel (Supported Compute nodes)
* OVS (accelerated version using DPDK user space patches)   
  *note: OVS is known to be too slow, so need to enhance and provide alternate to meet the SDN/DP latency & scale requirements*
* CEPH - block, data, image…   
  *can start with Cinder and progress to Ceph as it evolves; some concerns, but use as start point and add alternate solutions later*

The initial set is meant to be as small as possible, so the initial environment can be provided early.

Of course together with any of those projects, also all dependencies, e.g. MySQL (or some other SQL DBs) as the database, RabbitMQ (or some other MQs) as the message bus, etc.. must be included.

**Scope:**

Octopus will create the complete basic development and build environment for OPNFV and start the periodic execution of the automatic build and basic verification. It will provide the processes for development, for upgrade to newer versions of upstream projects and for adding more components or upstream projects.

Octopus will not decide when and which upstream project to be added to OPNFV.

**Testability:**

Octopus will provide the basic test and verification environment. Details will be specified later.

**Documentation:**

The whole environment as created by Octopus will be well documented with user guides for all developers. For any specifics in the upstream projects, it will refer to the documentation in the upstream project.

Require Management API for OPNFV Platform and Tools for OPNFV CLI/GUI.

**Dependencies:**

The first step of integration has no dependency on other OPNFV projects.

Test and verification in later project stages is dependent on the test objects provided by VNF deployment test cases project.

Continuous integration in later project stages is dependent on the Simultaneous Release project.

**Committers and Contributors:**

**Companies/affiliations committed to the project and willing to provide substantial resources:**

* Huawei (10), contacts: Uli Kleber (ulrich.kleber ad huawei.com), Prakash Ramchandran (p.ramchandran ad huawei.com)
* Cisco, contact: Ian Wells (iawells ad cisco.com)
* RedHat, contact: Dave Neary (dneary ad redhat.com
* HP, contact: Dave Lenrow (david.lenrow ad hp.com)
* ..

**Maintainers:**

**Committers:**

**Planned deliverables:**

See project description

The first step will be to generate a build of all the upstream components.

**Proposed Release Schedule:**

* 1st Alfa release 6 months from start date
* to be dependent on start version like Juno on Openstack in Nov 1st week as one to launch.

**Appendix:**

The following information is preliminary and may be used during the project lifetime.

**Basic set of upstream projects**

The initial set was already identified during face-to-face on October 2, 2014:

* Open Stack - augmenting and starting on top, subprojects to be selected, e.g.:.
  + Electron(NFV Open Platform Management Module) – New CLI
  + Keystone (Identity Module)
  + Nova (Compute Module)
  + Glance (Image Catalog)
  + Neutron (Network module) / with SDN option through ODL like project.
  + Cinder (Block Storage) /with CEPH option for integration and
  + Swift (Object Storage)
  + Horizon (Dashboard/GUI) – [Electron Dashboard as Horizon Extension]
  + Heat (Orchestration of Stack)
  + Cielometer (Telemetry Module)
* KVM / QEMU / libvirt (Support under Nova)
* Open Daylight (Support along with Neutron)
* Linux Kernel ( Supported Compute nodes)
* OVS (DPDK/ODP) note: OVS is known to be too slow, so need to enhance and provide alternate to meet the SDN/DP latency & scale requirements
* CEPH - block, data, image… (can start with Cinder and progress to Ceph as it evolves; some concerns, but use as start point and add alternate solutions later)

The initial set is meant to be as small as possible, so the initial environment can be provided early.

Of course together with any of those projects, also all dependencies, e.g. MySQL (or some other SQL DBs) as the database, RabbitMQ (or some other MQs) as the message bus, etc.. must be included. This includes selection of some alternatives or providing multiple options in later steps.

Additional upstream projects will be added when the initial process is established. Examples of upstream projects that may be added later:

* Docker style environments
* DPI solutions

Guidelines for selection of upstream code:

1. The Source code meets the requirements of SWA-1-5 as per NFV specs to a larger extent and there is larger community acceptance of the same.
2. The Source code is Open source and with no License or IPR restrictions with proper releases from the Community and or Vendors associated.
3. The Source code is interafacable with minimum efforts and fit the OPNFV framework.
4. The Community or Vendor is willing to contribute and develop the source in OPNFV community to make it acceptable for alignment with upstream releases.
5. The timelines of upstream code availability is in accordance with OPNFV time lines.
6. Technical Steering Committee will have final say in accepting or rejecting the Contribution and upstream codes to ensure fair process. The objective is to ensure that Stack’s build for OPNFV 1.0 are consistent with Reference Models in Normative documents of NFV.

**Basic Hardware selection**

* Central build servers for the automatic builds
* Recommendations for developers to build their hardware and software environment

**Tools for Code Repository, Build Processes and Automation**

Octopus will create a hierarchical build system that uses the build tools as defined by each of the upstream projects and combines them via some scripting. Therefore the necessary tools will be a superset of the tools used by the upstream projects.

In some cases however, OPNFV may also just use complete binaries from an upstream project. In that case, OPNFV doesn’t need to use the build process of upstream projects again, but then patches for the upstream project cannot be developed or verified in OPNFV environment.

Additional tools need to be selected for the automated (periodic) build and basic verification, as well as deployment of the build images on target servers in the test environments.

Examples:

* Code repository: git /Github or mercurial or svn
* Scripting: python/bash and Orchestration tools like Salt/Ansible for VNF lifecycle
* Automation: Jenkins
* Development: DevStack
* Deployment: Pacstack/Stackforge & Sleeted Vendor Offerings for reference implementation
* Verification: Tempest
* NFVI Tools – Vagrant, Cobbler and Baremetal builders like Ironic
* Service Orchestration : Java based for North Bound if taken up in NFVOPS 1.0
* …

**Image formats for code deployment in the test environment**

Select the format to match all build tools and deployment requirements e.g. qcow2 with metatdata in container formats (OVF) and other qemu image converters if IaaS platform lacks any

**Define the branching rules for the code repositories**

Clearly define when branches in the repositories shall be created or merged.

The branching strategy needs to define both:

* Usage of branching by OPNFV development (6 monthly cycle with few snapshots)
* Usage of branches of upstream projects and their relation to branching in OPNFV repository (e.g. periodic merge from upstream branches after their successful builds or only use stable branches), also how to use branching when patches to upstream projects are contributed to the upstream project but already active in OPNFV, etc.

**Provide the code repositories**

Provision and make available to all developers. (All via Github)

Maintain the repository.

**Integrate the basic set of upstream projects**

Develop the changes on the basic set of upstream project that are necessary to make them run together as a basis for an NFVI + VIM layer as defined in the NFV architectural framework.

**Implement build process for developers**

Develop a simple build system that can be used by the developers of OPNFV. (e.g. Can use DevStack extensions for OPNFV)

Best practice would be to minimize differences to central build.

**Implement automatic build process on central servers**

Develop automation of the build process.

Decide on frequency (e.g. start biweekly and bring it down to daily) and execute the builds.

Tagging of the builds.

**Provide or implement basic test objects (VNFs, and VNFM, NFVO)**

Select from open source or implement basic test objects to be used for verification. (like Tempest Integration test Suite in Openstack)

**Implement automatic process for basic verification of the build images on central servers**

Develop automation of the basic verification using the test objects.

**Implement the process for continuous upgrade to newer versions of upstream projects**

(controller Sourcing based on Gerrit Approvals)

While all previous steps just create a single version of the OPNFV platform and development like in other open source projects, this step provides the environment to synchronize the OPNFV build process with upstream projects as they move to newer versions.

Note that Octopus will not be responsible for the decision when to synchronize, but provide the necessary mechanisms in the automated build and verification.

**Implement the process for adding more upstream projects to the automated build and verification**

(TSC Approval Basis)

Adding more upstream projects in many cases will need adjustments in the tool chain.

**Execute the continuous automated builds and basic verification**

(CI with Release testing)

Octopus covers both the initial creation of the development and build environment, as well as maintaining the environment and executing the automated builds and tests.