Pharos "lab-as-a-Service (LaaS)" project

Community labs provide physical and virtual resources for project development and testing as well as production resources (along with the Linux Foundation lab). The number of OPNFV projects needing access to bare-metal resources are expected to increase as well as need more scaled and diverse deployment environments. This means that community lab resources will likely continue to be in heavy demand from approved projects and the OPNFV production pipe-line.

The OPNFV LaaS project will use some form of Cloud environment to provide individual developers with an easy way of getting started ... i.e. to "try-out" an OPNFV deployment and begin to develop and test features with minimal overhead or knowledge about configuring and deploying an OPNFV instance.

Pharos community is collecting requirements for this project ... please add your edits/comment to this Wiki page or use the mailing list ... opnfv-tech-discuss@lists.opnfv.org with [pharos] included in the subject line.

Requirements for OPNFV LaaS include ...

- Automated reservation and access (i.e. does not depend on emails and providing credentials manually)
- See what resources are booked and what are free
- Admin dash-board
  - Usage statistics (including who accessed and how long)
  - Control access manually if needed
- On-line documentation (overview, get started, searchable help, FAQ, etc.)
- OPNFV "hello world" application activated with a button or simple command
- Graphical user interface
  - Compute storage and network configuration of the virtual configured environment
  - Easy for user to recreate default setup
- ???

Proposed design of the IOL Lab at UNH Durham

Goals / Deliverables Phase 1

- Set of scripts to provision / create a virtual instances of a Pharos Pos, consisting of 6 VMs (Jump Host & 5 nodes)
- Integration of script with resource request / dashboard / Jenkins, allowing for full automation
- Working system for 6 pods, available to community developers through the dashboard

Design / Architecture Phase 1

Static Setup – Virtual Machines and network are pre-configured, to function as a “Virtual Pharos Pod.” A fixed number of “Virtual Pods” would be operated over the set of hardware, which access / assignments handles in a similar fashion to existing infrastructure. Each “Virtual Pod” would be longer lived, i.e. not torn down after usage, but could be “re-initialized” to a “known state” from a previously saved image / snapshot. This would be in contrast to a difficult to engineer and maintain dynamic setup.

- Simple setup and maintenance
- With a static setup 6 identical virtual machines can be run per server
  - Jump Host runs as one of the node, running either CentOS or Ubuntu with KVM installed
  - Networks established using either Linux bridging or OVS
  - Availability of ISO for each installer to the Jump Host
- Establish a proposed time limit for the resource, approximately 1 week, and allowing extensions of the time.
  - This might be able to be linked to the Pharos booking tool that is currently being developed.
  - Enhance the booking tool to “setup” the environment/handle extensions of the service.

Phase 2 Changelog:

Terminology new to this release:

Pod: a collection of hardware configurations that can be reused, embodied by what would be put into an OPNFV PDF file.

Config: a software configuration descriptor that applies to a specific POD. This includes operating systems and (in the future) software installation information for OPNFV and other LFN projects

Booking: a Pod + a Config + some metadata, such as how long it will last and who/what it’s for. Since both pods and configs are reusable, a booking makes them into a real single use thing that embodies what a Booking was in v1.0. If your needs aren’t too novel, you should be able to reuse an existing standard Pod and matching Config to book some hardware and get to work in a flash.
At-a-Glance:

- Users can now create Pods
- Pods can have more than one machine
- Users can specify network layout for their pods with a GUI tool
- Pods can be configured with standalone Configs
- Users can view deployment status
- Users can now create snapshots of certain machines that can be substituted for the standard images so they can bring their custom environments with them across bookings and machines
- Workflows allow users to seamlessly create a pod, a config, and a booking with them all in one go
- Users can now add collaborators to their booking
- PDFs are automatically generated based on how a user defines their Pod and Config for a Booking

Technical Coolness:

- Heavily restructured codebase to better model an OPNFV installation and allow for more extensibility
- Logging is now properly implemented, allowing for faster issue resolution and more efficient debugging
- Tests for all major components have been implemented to aid further development and avoid regressions
- New API supports multiple labs
- Heavy use of templates simplifies user interaction and allows for more flexibility for the labs
- Support for outages on a per host and per lab basis for routine maintenance and emergencies
- Flexible linear workflow: a workflow format has been created in such a way that additional "workflow extensions" and steps can be easily added with minimal fuss
- Analytics and statistical data on bookings are now being generated
- The foundation for automatic OPNFV installation has been laid

- If a user gets interrupted during completing a workflow, they can pick up right where they left off
- Users can view their bookings in detail, including the status of various subtasks, overall progress, and any messages or info sent by labs to them that are specific to a given subtask
- Removed Jenkins slave views. This will become a separate app if popular demand requires it be brought back.
- Added proper homepage
- Secured API

Virtual Hardware Requirements

**Minimum virtual pod requirements** – Nodes do not need to meet the exact Pharos requirements when virtual, utilizing around 8GB RAM for 6 nodes per server. Setup with 4 virtual NICs.

Per node:

- RAM: 8GB
- CPU: 4 cores (Largest amount per openstack deployment)
- Storage Space: 100Gb (Largest requirement among openstack deployments)
- Network: 4 NICs (Users would be required to setup VLANs for additional networks)

Hypervisor

**KVM** – Use KVM, with a template to create the Virtual Machines with automated scripts. KVM will allow for a completely FOSS testing environment as well.