

# ONAP Beijing Lesson Learned and Casablanca Process Improvement

*Work in Progress*

## Lesson Learned

### Retrospective

- Identification of issues - earlier than later by tracking Risks
- Pair-wise testing performed by the project teams
- 72h stability test run

Need for continuous integration testing:

- Some experience is based on HPA testing on vCPE use case. vCPE use case, even though, it was tested on Amsterdam successfully, it took a while to make vCPE work on Beijing components. It could be that there was some regression. Since HPA depends on vCPE use case, it could not be tested till the end. This can be avoided if previously tested use cases are verified continuously. Automation of use case testing & continuous integration verifications are key to add new functionalities in future releases.

### Lesson Learned Areas

- Communication – Wiki, Mailing Lists, IRC

1. When you are a mailing list moderator **do not** approve subscription requests from accounts with obviously pornographic domain names. (yes, this really happened. recently.)
2. Lack of transparency on LF IT ticket raised by the community. The LF RT tools has poor capability to email properly the whole set of information. The ONAP community has no visibility on what LF IT is currently working on.
3. Feedback compiled by [Kenny Paul](#) from ONS-NA: (See [TSC 2018-04-05](#) Agenda slides)
  - a. Onboarding an engineer is very hard; The wiki isn't laid out well and is confusing, needs to be cleaned up
  - b. Responsiveness from existing Community members answering new contributor questions can be a challenge
  - c. All approved projects should be required to post meeting minutes to the wiki
  - d. The volume of untagged emails makes filtering almost impossible
4. Need absolutely clear and unambiguous validation that **RocketChat** can be used via https from China and most companies without requiring a VPN, or the use of an alternative device/network -kenny:. Must be secured if setup by LF (simply b/c of well known pwd).
5. Would like the Community to decide whether the use of IRC as the "official" minute mechanism is a practice worth continuing or not; very few people are using it, many can't use it and unlike other communities, virtually no one contributes to the minutes. Compare that with the fact that everyone can use zoom chat and there is always a great deal of contribution being performed there. -kenny
6. WIKI Help: things are hard to find unless you type the right keywords or have the proper URL. Some information no more accurate. Matter of **organization**, not matter of quantity of information. Recommendation to use ReadTheDocs to start versus the wiki? Another approach may be to make Wiki searches more deficient by publishing and enforcing guidelines for usage of page names and labels.
7. WIKI: Onboarding new community members is the challenge.
8. Is there a good enough flow of information-communication between sub-committees and projects? Overall: yes. Sub-committee feedback of to the TSC is not systematic(as we wish).
9. Expectation from sub-sub-committee back to the team and then driving the execution. Use-case owner is missing. Use-case team to fill a checklist? We may defined what that role is.

- Labs & Test Environment

1. After M4 code freeze, the community is spending a lot of time to get the Health Check sanity test to pass. HealthCheck is the kind of automated test that MUST pass everyday all along the release lifecycle.
2. Despite the effort made by some ONAP partners in providing Labs, we have reached the limit on current labs infrastructure. This is preventing further needed testing (like test job during the verify phase).
3. Need to develop a full Agile CI-CD pipeline. Full Chain to run automatically the sanity HC, CSIT, E2E. Everything running continuously.
4. How to make better usage of XCI-CD environment (OPNFV,...)
5. CSIT tests under integration repo must branch early to support the other component branches and their test automation (at least must branch on code freeze deadline to provide enough time to fix the CSIT tests in the release branches).
6. Supporting HEAT and OOM based deployments is getting harder with duplicate maintenance of the config values (it may help if we can somehow abstract such duplication of config values).
7. More ONS-NA Feedback: The amount of time to setup the dev environment is a barrier to engagement - Developers don't want to invest a lot of time on that.
8. Improve CSIT coverage to cover all features that the project delivers and to reduce manual testing (if any)

- Release Management

1. Progress check between M2 and M4 of intermediate development by release manager might help to improve the quality of the final product (for example scheduling the demos of each component to show the current progress at M3 deadline might be right choice too). This avoids any misunderstanding of the features or requirements by the dev team and can receive feedback from the usecase or architecture teams to improve them when there is still time until code freeze.
2. Lack of formality to freeze model and scope increasing risks to put any milestone in jeopardy

3. Reset expectation on Milestone meaning (enforcement?).
4. Grouping of Projects (staggered): to address Release dependency.
  - a. Idea of 3 months project (push back at TSC)
  - b. Defining another milestone (no that good idea?)
  - c. Start early pair-wise testing?

- JIRA Management

1. Often lack of updated information in "Status" field, that prevent to know if someone is working on a defect. This issue varies depending on project and people.
2. Adopt a single JIRA workflow. Currently 2 are in place (1 from openecomp, 1 simple from Open-o).
3. PTLs are owner of the scope of iteration.
4. Review the 2 JIRA workflows and define 1 workflow for ALL.

- Code Review

1. Requires active committers and more active code reviewers (right now only few committers or reviewers are always reviewing and merging the code).

- Development Infrastructure

1. The LF toolchain that is currently in place, do allow to merge in master code that has not been thoroughly tested. This leads to massive disruption in the testing.
2. Nexus 3 slowness. This has been impacting Integration Team tremendously as it tooks 3-4 more times just to download Docker images.
3. Current 1/3 party scan security tool (Nexus IQ) do not work for Python and JavaScript packages and thus prevent the community to have an holistic view on vulnerabilities.
4. For the record, it was decided that the community would not account for JavaScript in using Sonar (code coverage) in Beijing Release.
5. Slowness of the full IT chain (jenkins, nexus, wiki,...)
6. Local testing: how easy to setup an env for a developer to perform its own testing before submitting code. Need reference VNF, AAI, ... (too much time spend to setup environment). Idea of lab reference to be used as a model for configuration. (currently SB-07 serves that purpose).
7. Pair wise testing for a great value added in Beijing release.
8. 72 hours helps to identify defects. Docker Upgrades went fine.
9. Backup and restore capacity for SB 04-07 in Windriver? Have we ever asked Windriver?
10. Feature parity on LABS (do not over tax Windriver).
11. Nexus IQ scan performed during the verify. If error then block the build.
12. Idea of parallel on a single job. Currently atomic at build level. To investigate-optimize Jenkins Jobs. (time to build the VMs, ...).

- PTL Role

1. More ONS-NA feedback:
  - a. *"Either you commit or you forfeit. There really isn't any middle ground here."*
  - b. If you can't attend a meeting, assign a delegate. If you send a delegate more often than you attend, you shouldn't be a PTL.
2. PTL-cross PTL discussion: feeling of not involvement in some decision. Having a place for PTLs to speak up.

- Architecture

- Need for simplifying ONAP Micro-Service architecture by leveraging best practices from Industry :
  - Certificate enrollment (for Mutual TLS communication among services) in Beijing was manual, time consuming and error prone. Need automation of certificate enrollment is needed.
  - Scale-out of ONAP is removed from the scope in Beijing release. Services that attempt do it has some learnings. Scale-Out/Load balancing/Circuit-breaking/DB-sync related functionality should be removed from actual micro services to sidecars/side-kicks to avoid each developer spending time and getting it right.
- Need share as much as possible the building blocks with regarding to the implementation of new ONAP Micro-services
  - While the community focus on more about the micro-service arch. the diversity of building blocks for each micro-services complicates the debug/maintain effort for everyone willing to exercise/develop existing/new use cases. e.g. While java and python language are prevalent used for each most of micro-services, some other language binding(e.g. golang) was introduced as well. so whoever want to debug those microservices will have to either seek the help from community (with no promise to get timely support) or have to learn those language/framework,etc. This apparently hinder the adoption of some ONAP components, even they complies to micro-services architecture. I suggest each team should evaluate/review carefully what kind of building blocks are utilized instead of the choice to the contributors only. The principle is : whenever it is possible, the existing building blocks should be considered as preferred choices.
- New project introduction
  - New projects that are in "incubation" stage should not be a dependency for the rest of the mature projects. This will avoid introducing new complexities to the process of deploying and running ONAP.
  - Only when a project graduates from their incubation stage should other projects create such dependencies.

- Use cases:

- Increase the scope of vCPE use case in deploying multiple virtual Gateways (Multi-Customer support)

- Open Source Values

1. One critical aspect of Open Source is transparency (but opposition to Silos). We still see some corporation submitting huge pile of code days prior to major milestones. This approach is preventing the whole community to collaborate effectively.
2. More ONS-NA feedback:
  - a. internal company staff meetings/decisions != project team meetings/decisions
3. Maturity of the community to understand the release is time-boxed and thus we have to adjust on scope and quality. Process adherence.

- Others

1. Commit process: code review. Dialog necessary.

- 1) Concerns on big code merge. 2) That comes late. In case big code should come in, define an upper timeframe limit.
3. Use Case: more details functional requirements. more focus on cross system integration (architecture). Lack of defined type, flow and content of messages (need additional details).


## Process Improvement

### Goals:

1. Don't merge code that has not been tested
2. Speed up the verify and the merge
3. More automated testing

### High Level Categories of Process Improvements

1. CI-CD: tools and processes
  - a. Self-commit:
    - i. automate Docker release version
    - ii. When **time is critical (after RC0)**, allow **PTLs** to merge their change. This should allow be applicable to LF IT (CI-Management).
    - iii. Option: Follow OpenStack process loosely, I.E. no one merges code except Jenkins. When a Gerrit patch gets at least a +2 from a committer and a +1 from someone else we could have Jenkins merge the patch or do some variation of this to ensure faster patch merges but still ensure code review.
  - b. Increase CI-Management committers list
  - c. CD: use of OPNFV Clover. Helen/Integration
  - d. Packaging: Improve Docker images build: incremental versus rebuild everything, optimize docker package (size, time to download) (LF+Gary-Michael). Fact: 13 docker templates in Beijing. Focus first on the most downloaded docker images (sdc, so aai,...)
  - e. Standardize the build jobs (benefit: ease to add dependency): LF.
  - f. Daily Build: get back to daily build. Build only image on code changes. (once a week one full build).
  - g. Partner with OPNFV: Clover project.
2. Testing
  - a. Improve time to get the full ONAP installed and started (currently 12 hours to debug): 1 hour for HC + 1 another hour for instantiate => 3 hours. Every project team effort. Better knowledge (training) of OOM + K8S + Helm
  - b. Look at how to architect for testability, and debugability : Brian
    - i. the idea is how could we modify the helm chart setup so that we can enable a project to pull a new merge or verify docker without the risk that they will lead to a broken helm install from failed terminating pods, pv,pvc, port conflicts etc.
    - ii. helm upgrade --set enabled=true/false has led to broken installs and the need to purge
    - iii. Would be nice to be able to have projects in testing labs segmented so that project A's make project/make onap; docker stop /enabled=false/enabled=true doesnt cause other projects to be affected
    - iv. Right now the teams use HEAT base for preliminary testing because that isolation maintains a more stable debugging environment particularly since developers dont run into the read only file system issues for various configuration files
    - v. While it is not the right solution - in some sense the Amsterdam per project namespace was actually more forgiving - although in fairness we didnt use the Amsterdam OOM as much as we did the Beijing OOM
  - c. Love to plug a vFW-vDNS automated testing in the verify jobs: Not realistic for Casablanca. ( Gary to try)
  - d. Manual way to build docker off on Verify job
3. Release milestones
  - a. Enforcement (respect and act upon automated results): (naming and shaming if criteria not met) on TLabs starting at M2 for Casablanca. (increase lab coverage over time, over release)
    - i. HealthCheck: must not be broken for more than 48 hours
    - ii. Use case (VFW vDNS are automated): must not be broken for more than 48 hours
    - iii. CSIT: must not be broken for more than 48 hours
    - iv. Daily Build: must not be broken for more than 48 hours
    - v. What for new projects?: HealthCheck after M4
    - vi. Perform testing Staging and Prod env: Gary.
4. Labs and Infrastructure
  - a. Resource (hardware): Get data from Stephen Gooch : Michael
 

 **OPENLABS-294** - Get size of windriver lab - RAM - it looks like 1.8 TB for 10.3 TB - to pass to beijing conference talk CLOSED

WindRiver current config: 500 VMs, 2.8 TB RAM, 9.8TB Disk.
  - b. Increase uplink speed on windriver network (slicing 10 GB):
  - c. jenkins + nexus (currently 8GB is minimum): make it faster: LF. Jenkins fails cost minimum 1/2 days (for US). To bring this to the TSCG Board.
5. Packaging
  - a. :
6. Code Coverage and License scan
7. Security
8. SLA with LF: Idea Use JIRA to prioritize tasks.

LF to fix Staging Nexus.