POC definition

PROPOSAL - following feedback from the ONAP TSC 2019-02-07 the POC definition for Dublin

Proof of concept

POC DEFINTION:

Proof of concept (**PoC**) is a realization of a certain method or idea in order to demonstrate its feasibility,[1] or a demonstration in principle with the aim of verifying that some concept or theory has practical potential. A proof of concept is usually small and may or may not be complete.(source: Wikipedia)

POC GUIDELINES:

1. **SPONSOR** - Every POC shall have a named sponsor for the feature whom acts as spokesperson at all checkpoints/meetings.
2. **NON BLOCKING** - POC ~~(only)~~ related defects shall not block the current release and not create any backward incompatibility
3. **COMPLIANCE** - POC should be fully compliant with ONAP development guidelines, including development procedures.
    a. **License/Vulnerabilities** - POC development that includes entry points in the release version must comply with license scans and vulnerability fixes.
    b. **Proprietary Components** - PoC *may* include proprietary software or 3rd party software. It should be explicitly stated that the PoC is not contributing its software otherwise it is subject to CLAs. Note: The 3rd party software should comply with security guidelines otherwise there could be a backdoor vulnerability into ONAP.
    c. **Security** - PoC development shall be compliant to security guidelines and incorporate the latest security fixes so as not to create a security vulnerability.
    d. **Performance** - PoC should not impact the performance of the main code; this should be implied as it is kept in a separate Repo. The PoC development team should have an eye on ONAP performance if in the future it might be evolved to be incorporated into the ONAP platform.
    e. **Back Doors** - A POC should not introduce a back-door to the stable features so it must comply with security, code coverage, and license scans requirements.
    f. **Resources** - The PoC should perform an assessment to see that adequate resources are available such as PoC toolchains (Jenkins, Sonar, NexusID, LF resources, Test coverage resources etc).
    g. **Approval for inclusion** - The TSC or Use Case S/C shall approve the PoC to proceed as part of the current release.
4. **INDEPENDENT OF MAIN RELEASE** - PoC are not part of the (current) ONAP release product.
    a. **Documentation** - The PoC may have documentation in a wiki developed by the community; but will not be incorporated into the official release documentation.
    b. **Independence** - The PoC shall be kept independent of the official (current) release. The PoC shall not introduce any new requirements in the current release.
    c. **Integration/Testing responsibility** - There is no responsibility from the integration or test teams to test the PoC S/W. The testing, integration and use of the PoC shall be in the purview of the PoC development team.
    d. **PoC Release Notes** - PoC release notes may be developed by the PoC team, they may include details of functionality available, known issues, and known limitations/incomplete features to keep of track development issues as they arise and serve as a communication vehicle for those evolving the PoC in the future.
5. **CODE INDEPENDENCE** - The PoC software shall be kept separate and independent of the main ONAP branch/repository.
    a. **Common Code Usage** - PoC software using common code, does not change the treatment of the common code with regards to completeness, S3P, documentation and other common code expectations.
    b. **Separate Branch** - PoC should be kept in separate repository branch. There should not be PoC S/W introduced into the main ONAP software branch. This prevents complications in testing and integration of the ONAP software in the main branch. The PoC should not modify any existing (current release) core ONAP components.
    c. **Evolutionary Platform Software** - EXCEPTION CASE: Under the blessing of a PTL for the platform component, there might be a situation where a PoC is demonstrating something to evolve the Platform component (e.g. SO, VID, VNF-SDK etc) and some of the PoC software may be included with in the *CURRENT* release. Experience has shown that it was important to merge PoC code in the current release to make the task manageable. Note: it is expect this would be most useful in large, multi-release PoCs.
6. **PROCESS** - The PoC *may* follow a process during development.
    a. **Milestones** - The PoC team *may* follow some sort of development process. It is suggested that they use the Mx milestones/gates as the release progresses or some similar process. Note: it is expected that using a process would facilitate formal inclusion of the PoC in the subsequent ONAP release(s).
    b. **Final Approval** - The final approval of a PoC shall ultimately be determined by the sponsor/leader of the PoC (see Point #1). It is suggested that a "report" of how the PoC went to socialize and inform the TSC or other relevant teams.
    c. **PoC Overlap** - It is recommended that the PoC team socialize their PoC so that other teams working on Use cases and PoCs are aware of potential cross-interactions that might occur and overlap in scope and functionality.
7. **SCOPE** - A PoC can be used to demonstrate functionality related to a project concept, a Use Case, functional or non-functional requirements, project specific extensions.
    a. **Definition** - It is the responsibility of the PoC team to define the scope and extent of the PoC.
    b. **Project Review** - the PoC team shall review the PoC with impacted sub-committees (such a security and modeling S/C) and projects PTLs (such as A&AI and SDC).
8. **POC INCORPORATION** - The point of a PoC is to demonstrate something (hopefully) useful. PoCs teams that wish to incorporate the PoC code into the main repository should follow the standard procedures for introducing new functionality and use cases in subsequent ONAP releases
    a. **Feature Proposals** - There exists a standard procedure new functionality introduction leading up to M0 shall be followed by a PoC in subsequent releases by the PoC team if they wish to incorporate PoC Software into the main repository.
    b. **Partial PoC Introduction** - The original PoC scope may be altered from what is introduced in a future release. An ideal case might be that a PoC software might serve as seed code, but it does not have to. Lessons learned can guide what would get incorporated or proposed in the future release.

c. **Merge Responsibility** - It will be the responsibility of the PoC Team to *properly* merge the PoC code into the main-line branch (S/W repo) if it becomes part of the future release. Merging should follow compliance principles (described in point #3 above).
d. **Proprietary Components** - If the PoC used proprietary components or software, for formal including into ONAP it would need to be excluded in the final merge because ONAP is an open source initiative. Note: A PoC using proprietary components are "on" or "with" ONAP; a contributed PoC (without proprietary components) are said to be "in" ONAP.

==-=-=-=-=-=-=-=-=-=-=-=-=-CUT LINE, Previous content preserved for continuity... Delete after TSC review-=

POC is equivalent to "Tech Preview" or "Experimental"

Guidelines

1. Every POC has a named sponsor for the feature whom acts as spokesperson at all checkpoints/meetings
2. remove POC feature development is second priority to committed feature/tech debt development efforts.
3. Dependencies on other projects for POC functionality should be clearly communicated as POC development
4. POC (only) related defects cannot block the release
5. POC release notes should include details of functionality available, known issues, and known limitations/incomplete features
6. Integration and test resources are primarily for the committed features and should not be used for POC until either
    a. primary functionality is complete
    b. testing on the primary functionality is blocked - and cycles are available while blocker gets resolved
7. POC issues are secondary topics at PTL or TSC meetings
8. POC development can waive "product completeness" requirements - like S3P, Localization/I18N
    a. Basic documentation is expected so others can access the feature - full commercial grade documentation is not expected
9. POC development must comply with license scans and vulnerability fixes
10. POC may be promoted to fully supported feature prior to the end of the release with approval of the TSC
11. POC features may be withdrawn from the release at any time with notification to the TSC from the sponsor
12. POC functionality is targeted for full feature in a specific future release

ISSUES:

- How to treat "use case" POCs that use the common code base? Defects in a use case could impact other functionality...