R2 proposals for Non-functional requirements

This will include e.g. credentials usage, performance, resiliency, testing requirements etc.

- ## Support ONAP platform upgrade
  - How can we update a running ONAP instance in the field. Should this be part of the OOM scope with help of the architecture sub committee? The OOM/Kubernetes Rolling Update capability should be considered for this.
- ## Support ONAP reliability

  - We have no well defined approach to fault tolerance neither on the component nor the system level. Should this be part of the OOM scope with help of the architecture sub committee? The OOM for Carrier Grade Deployments page describes a comprehensive approach to this.
  - An instance of the complete "ONAP platform shall be configurable to be part of an "N+1" group of instances that operate in a specific, coordinated, controlled fail-over manner. This coordinated fail-over manner enables the operator to select an "the idle +1" instance in the group to functionally replace any of the operating, healthy, "N" instances of the group on demand, within a specified amount of time. This approach shall be used as a common method for surviving disaster, and also, the common approach to software upgrade distribution.

    NOTE: Failover doesn't necessarily need to be always controlled. For example, if a component fails in the middle of a workflow, ONAP should be able to switch to another available instance of that component so as to terminate operations correctly, without involving humans in the loop.

- ## Support ONAP scalability
  - How to scale ONAP? Should this be part of the OOM scope with help of the architecture sub committee? The OOM /Kubernetes Ingress Resources, Replica Sets should be considered for this.
  - Answer to above question below.
  -
  - **Asynchronous Push Model for Latest Data** enabled by DMaaP pub-sub active-sync or lazy replication across WAN is key to scale
    - Synchronous Poll model for volumetric data does not scale and often lands up using stale data
  - Further details on **Infrastructure Telemetry** at scale are described below
- ## Support ONAP monitoring
  - Common logging formats and approaches need to be supported and automated cross components monitoring tools should be developed/provided
  - The ability to monitor and detect issues with the delivery of feeds and topics in DMaaP should be provided
  - OOM already has the ability to deploy a 3 way clustered, centralized Consul server and one or more Consul agents to monitor the health of the ONAP components in real-time. Health monitors for many of the ONAP components are already available.
- ## Support ONAP stability
  - Besides logging, standardized Data Model for various common Objects including hierarchies should be provided
  - For example, objects such as DC, Tenant, Cluster etc. are managed by Multi Cloud and also used by Policy, DCAE, OF etc.
- ## Support a Common ONAP security framework for authorization and authentication
  - See: https://lists.onap.org/pipermail/onap-tsc/2017-September/001684.html
  - The architecture & security sub committee should define a common security architecture which the projects need to agree to adopt.
- **Secure all Secrets and Keys (Identity or otherwise) while they are in persistent memory or while they are in use**
  - Secrets such as passwords and keys are in clear current ONAP infrastructure components. Security breaches are a possibility if these secrets are not protected well. In Openstack and K8S, secret/key server is used to store the secrets. Services that require secrets get them on needed based from secret server and use them.   Secrets in secret server are protected using local HSM or TEE. Secrets in services are also can be secured using TEE for better security (never expose secrets in clear even when they are in use).   It is needed to define the security architecture similar to barbican, HashiCorp vault and enhance ONAP services use this secret server to get the secrets on demand basis. And it is highly recommended that ONAP services also secure the secrets when they are in use.
- **Support for ensuring that all ONAP infrastructure VMs/containers are brought up with intended Software**
  - ONAP infrastructure itself is set of multiple services. At last count, there are more than 30 services. In addition, some of these services can also be run in various sites for scalability and availability. For example, some DCAE components may be run in various sites. If underlying operating system/system-software and ONAP software is tampered, that could result in misbehavior and in the worst case hackers taking control of the network.   It is good practice to ensure that the ONAP servers and services (containers or VMs) are brought up on servers with intended firmware, OS, utilities etc.. TPM based software attestation is one popular method and same can be considered here.
- **Secure communication among the ONAP Services**
  - Mutual-TLS (Certificate based authentication)
  - Auto certificate provisioning on each ONAP micro-service when it comes up.
  - Local CA support to provide certificates to ONAP services.
- **CA Service for VNFs certificate enrolment**
  - Many VNFs need to communicate with other VNFs securely over TLS. Since Mutual TLS requires certificates, there would be a need for CA service for VNFs for VNFs to get the certificate enrolled.
  - CA Service to provide X.509V3 certificates as one of the ONAP services.
  - CA Service to provide OCSP service.
  - Ability for VNFs (as part of VNF SDK?) to request certificate from the CA service by providing PKCS10 CSR.
  - Ability for VNFs (as part of VNF SDK?) to check the certificate validity.
- **VNF Package Security**
  - Provides authenticity and integrity for VNF package
  - Manifest file based security as specified in ETSI GS NFV-SOL004 standard
  - Provide optional secrecy for sensitive VNF package artifacts as specified in the same ETSI standard
- **PKCS11 support for private keys**

        - Many modern servers support some kind of TEEs (TPM, SGX, TrustZone etc...). TEEs provide ability to keep the private keys and perform the crypto operations such as RSA/DSA/EC-DSA sign and RSA decrypt operations within TEE without ever exposing the keys in application memory. PKCS11 is one common method used by cryptography libraries. Many TEE vendors also provide PKCS11 plugins to talk to their TEEs. It is strongly recommended to use these schemes across ONAP services that use HTTPS transactions. Also, this facility can also be used to store private keys of VNFs. Hence, it is also recommended for VNFs to take advantage of TEEs to store private keys and perform cryptography operations in TEE via PKCS11 interface.
    - **Support VNF CSAR validation**
        - In R1 vendors need to use ONAP SDC to validate their VNF CSAR file. A tool (VNF SDK? or online tool) should be provided to vendors to test their template without installing ONAP instance by themselves
    - **API Versioning**
        - What are the detailed rules of API depreciation. We did commit at the NJ F2F that APIs can't just change between releases but we never agreed on the actual process/timeline surrounding API changes etc.
        - APIs for the current release, and in addition, the most recent two prior releases shall be supported, and if changed by comparison to the two prior releases, shall support deprecated, unchanged functions of such two prior releases.
    - **Support ONAP Portal Platform enhancements**
        - Convenient Portal's SDK adaption for ONAP developers - improve support of latest UI technologies and spring boot based apps.
        - Enhance Platform features - Widgets features, Interactive user guide feature in Portal, Q&A section for users linked from Portal.
        - Platform High-Availability - support Kubernetes compatibility.
        - Meet LF standards - JUnits, Sonar issues, ONAP package migration, Improved logging
        - On-boarding new apps - The new applications that need to be on-boarded onto Portal Platform - collaborate with use case requirements.
        - Security Enhancements - OAUTH, AAF support - Adapting centralized role management by all applications that are now provided by the Portal platform - collaborate with ONAP security sub-committee to review the security best practices and standards expected from the Platform.
    - **Software Quality Maturity**
        - The ONAP platform is composed of many components, each component exhibiting varying degrees of "software maturity."   A proposed requirement here is that prior to accepting software updates for an particular component as part of the complete ONAP solution, it must be shown that the software for such a component meets or exceeds specific quantitative measures of software reliability. AKA "Software Releasability Engineering" analysis data should be computed and disclosed for each component. This data is computed by collecting time-series data from the testing process (testing the published use-cases) and fitting test failure counts to a family of curves known to track defect density over the software lifetime. Information on this type of quantitative anallysis is here: https://drive.google.com/open?id=0By_UqQM0rEuBei10TVdTOU5CalU   A particular tool that may be used to compute this "SRE" (Software Releasability Engineering) is open sourced and available entirely in a Docker container here: https://cloud.docker.com/swarm/ehwest/repository/docker/ehwest/sre/general   Numerous papers are in the literature explaining the use of Markov Chain Monte Carlo methods to fit test data to the target curve.
    - **Documentation and some toolkit to use ONAP and adapt it**
    - **Support for a common resiliency platform**
        - Making ONAP resilient is challenging broadly because of three reasons – (1) ONAP services need to be replicated both within and across sites, with efficient failover mechanisms. This is a challenging problem because of WAN latencies and frequent network partitions. (2) The clients of ONAP are spread across the world, and may have locality and high throughput needs. Hence, the load needs to be distributed and or federated across the different replicas of the ONAP services to satisfy such needs. (3) ONAP services often have a diverse range of requirements in terms of replication and resiliency. While some components need to carefully manage state across replicas, others may be stateless. Similarly, some of the ONAP services have strict requirements in terms of how the load should be shared across replicas.

          To address these complex resiliency challenges, different ONAP teams are building their own carefully reasoned, handcrafted solutions leading to much wastage of resources. A proposed requirement here is to systematically understand and model the common resiliency concerns across ONAP services and provide a resiliency platform with the necessary building blocks to address these concerns. Each individual ONAP service can use this common resiliency platform to create design patterns for resiliency as required.
    - **Consistent Component Configuration**
        - Each of the ONAP components have their own method for specifying configuration values (typically property files, environment variables, chef recipes, etc.) which make globalization of common parameters (such as OpenStack Keystone IP addresses) difficult. OOM/Helm provides mechanisms for such global values but the component teams need to adopt common methods.
    - Provide powerful test and debug environment/tools
    - **Consistent API pattern**
        - API is very important because it's hard to make significant changes to your API once it's released and we want to get as much right as possible at first, It seems that there's no consistent way for Restful API design for ONAP projects now and some of the API definition are not very appropriate.
    - **Unified Design Environment**
        - Unify all design tools (DG, Policy, DCAE Template, CLAMP, Holmes, A&AI schema, etc) and make them as integral part of service design.
    - **Programmable level of platform quality**
        - There should be a possibility to configure platform quality level (e.g. stability level, performance level, security level etc.) as required by the specific Service Provider. The reason is that high platform quality requirements may be very expensive in terms of resources and support, therefore it is up to customer to define what platform quality is desired.
        - OOM deployment specifications will be parameterized to allow multiple deployments types - minimal for development to fully redundant, geo-redundant deployments for production.
    - **Infrastrcture Telemetry - key new requirements**
        - **Aggregate Data (Tenant, Cluster etc.)** is key to hierarchical multi-site VNF placement solutions driven by OF, DCAE etc.
            - Data at atomic level (VM, Host etc.) does not scale
        - **Asynchronous Push Model for Latest Aggregate Data** enabled by DMaaP pub-sub active-sync or lazy replication across WAN is key to scale
            - Synchronous Poll model for volumetric data does not scale and often lands up using stale data