Multi OS

- Requirements:
  - StarlingX shall support multiple operating systems to enable broader use of the software by end users, operators and customers (who likely have strong OS preferences)
  - Intel would like one of those operating system to be Clear Linux
- Supporting Multiple OS's impacts the project in many ways across many sub-projects
- We are looking to "crawl, walk, run":
  - Crawl: Run the services on a non-CentOS 7x Host Linux OS (with RPMs)
  - Walk: Enable StarlingX to build & support more than one OS/package format
  - Run: Install, run & update StarlingX on multiple OSes
  - Fly: Run StarlingX on any version of Linux (OS-Independance)

Multi OS so far - DevStack enablement

- Our first step was show that it is possible to enable several STX services in DevStack, and to patch DevStack to enable Clear.
- We can now run DevStack on Clear Linux including a couple STX services while adding new packages to Clear for missing dependencies.
  - The services don’t actually do anything in DevStack except not die
  - There are STX service reviews pending for 2+ months for DevStack plug-ins
- We would like to finish this work to enable all STX services in DevStack such that the services start, run and respond to API requests
- Would like further work on STX DevStack to include contributions from the service teams

MultiOS Plan / Resourcing - Crawling step

- Options for Crawling: (not exclusive)
  A. Complete DevStack integration to pathfinding for multiOS but more importantly to enable devs and automated testing.
     - Goal: Pass Sanity testing against STX DevStack. Enable Zuul based service testing
  B. Stand up the services independently outside of DevStack and outside the standard StarlingX installation on various operating systems (demo quality) in simplex configuration (Intel MVD)
     - Use StarlingX configuration tools/scripts (puppet), RPM packaging, etc…
     - Dependent on work to enable OpenStack on Clear Linux (in progress)
     - Will help highlight future work to make the services more independent
     - Goal: Demo quality initially, longer term pass Sanity
  C. Start planning the work to replace the current build with refactored build tools
     - E.g. the existing specification for makefile / spec file improvements
     - Begin planning specifications to refactor the build into more modular components
- Which (if any) are the best approach to Crawling?

MultiOS Open Issues - how do we Walk/Run?

- Which OS's / versions do we want to support?
  - **Recommendation**: CentOS 7, CentOS 8, Clear Linux, Ubuntu LTS. RHEL? Others?
- How do we handle the patches against OS components across Multi OSes?
  - Maintaining patched components across multiple OSes will impact rebasing and support.
  - **Recommendation**: Upstream and/or remove as many as possible.
    - Analysis shows that ~ 50% of STX kernel patches are present in newer kernels.
  - The value proposition for StarlingX is 1) it allows you to easily deploy a very high performance Edge stack. Some OS patches are needed for performance. Is there value in creating a version of StarlingX that includes limited patches and sacrifices some performance that can be more easily supported on multiple OS's?
    - Should we optimize for broad adoption or highest performance?
  - Documenting exactly how to patch the OS to gain max performance could enable users to run any Linux OS (once we separate out the services, installation, configuration, etc…)

MultiOS Open Issues - how do we Walk/Run?

- How do we validate multiple OSes and OS versions?
  - Each OS/version increases validation effort
  - May be value in supporting some in parallel e.g. enabling CentOS 8 while still supporting 7.x
  - **Recommendation**: Define one OS version per release as the reference OS. Fully validate on that OS. Sanity test the software on any other supported OS. Add basic tests to cover package file handling differences to the Sanity test suite.
- How do we support multiple OSes and OS versions?
  - Each OS/version we support means build breaks, rebasing, CVE updates, bug fixes and other on-going support costs.
  - **Recommendation**: Define support Tiers. Tier 1 OS's are fully supported by the project and limited in number. Tier 2 OSes get CVE updates and best effort support.

MultiOS Open Issues - how do we Walk/Run?

- Do we keep the current StarlingX install paradigm or adopt OS installers?
  - E.g. “`boot stx.iso`” vs “`$ sudo apt get StarlingX`” vs “`conjure-up starlingx`”
  - If we support OS-package based installation, how do we configure the software?
  - How do we plan to support ZTP? Does that impact this choice?
- Is there a good way to order the huge amount of work needed to reach OS and Package Independence?
  - **Recommendation**: Start with the build system, refactor for multi-OS builds and multiple package file format support. Work our way up the stack. Meanwhile work on the services to refactor for independence, OS and package agnosticism.