Recap: Why LTS?

- Users are often broadly categorised into two categories, their focus is either
  - Latest functionality, or
  - Stability

- Two types of DPDK releases for these users
  - Master releases: Latest features
  - LTS releases: Fixes only

- Since DPDK v18.11 release
  - Master branch: 3013 commits, features and fixes
  - 18.11 stable branch: 909 commits, fixes only

- LTS release motto: "Don't make it worse"
How old is the code being fixed?

Fixes since DPDK 18.11 - Time since bug introduced

Fixes since DPDK 18.11 - Normalised with code size

Fixes

Fixes per KLOC

Months

Months
A year in the life of stable/LTS

- Branches maintained
  - 16.11 EOL, 17.11 continued, 18.11 started

- Short Term Stable Releases no longer by default
  - (STS: one release ~ 3 months after master release to add early fixes)
  - Not enough interest/validation resources
  - Better value testing Master and LTS releases
  - Can still be done for a particular version if enough community support

- Scheduling
  - More LTS usage -> higher priority (for maintainers and validation teams)
  - -> more predictable releases
A year in the life of stable/LTS

● Release notes
  - Known issues / Fixes not backported
  - Added summary of validations

● Tools
  - Travis, openSUSE Build Service
  - Queued commits/rebases on github with commit link in emails

● The following companies/projects reported validation on LTS releases in the last year
  - IBM, Intel, Mellanox, Microsoft, Red Hat
  - OVS, SPDK
LTS Fixes

![Bar chart showing commits for LTS versions 16.11, 17.11, and 18.11.]
16.11 LTS is dead, long live the LTS
16.11 LTS is dead, long live the LTS
Verschlimmbesserung

- Wouldn't it be great to have new features in LTS too?
  - "Cake and eat it" releases
  - (some latest features from master, some stability from stable)
  - But actually it may also be called...
  - "No cake and hungry" releases
  - (Not all latest features, not fully stable)

- If DPDK LTS is becoming more popular - Why?
  - Stability is its USP
  - Adding invasive features removes part of that
  - What would a user just requiring stability use then?

- Bugfix: Fixes existing functionality/features that don't do what they should
  - Wishful thinking does not count as existing functionality/features
  - Can be grey areas
Balancing stability

- On scale of bad ---> not as bad
  - Change ABI/API
  - Change common components used by multi-vendors (e.g. membuf lib)
  - Change components for a single vendor (e.g. a vendor PMD)
  - Add code that will **obviously** not impact existing functionality
    - e.g. An additional define used in a case statement
  - Add a new pmd without changing any of the above
  - Add a new PCI ID

- Key factors
  - Invasiveness
  - Commitment to validate and support

- "Don't make it worse"
LTS/Stable: How you can help

- If using LTS, please test the RCs with your use cases
- Please share the test result summary for the Release Notes
- Please use both "Cc: stable@dpdk.org" and "Fixes:" tags
  - Every patch missing a tag requires manual intervention
  - Reading through mailing list / emailing author / checking the code
- Would like to add another maintainer so there is more cover
- What can we do to make LTS better?
- Is the Master/LTS release cadence correct?
16.11 LTS is dead, long live the LTS

16.11 LTS commits per release

LoC percentage over total vs folder, diffstat 16.11 to 16.11.9