ReleaseStrategyProposal

release strategy proposal

Problems to address

- Instability of stable releases
    - +1: covener
- Conservatively managed distributions are drifting farther and farther away from HEAD
    - +1: covener

Things working well

- New modules can get into users' hands pretty easily.
    - +1: covener
- Few streams to worry about
    - +1: covener
- Long lifecycle of a release
    - +1: covener

Proposal 1

This is a WIP. Please feel free to edit if you preserve the spirit, or fork it into a new proposal if you don't.

The philosophy here is to have 1 or more conservatively managed releases but to also always have 1 or more more liberally managed releases where slightly more disruptive things are tolerated. But the latter is neither trunk nor a "development" release.

Some things that characterize a more conservatively managed release:

- Behavior changes tend to be opt-in.
- Refactoring is limited.
- New function, new directives, and new modules are acceptable if their enablement doesn't put the stability of existing function at risk.
    - For example, mod_md on its own would have been OK, but the changes to mod_ssl to accommodate it would have needed to be (at best) guarded differently.

1. Establish a litmus test ("rules") for what can go into early maintenance levels of a release
2. Establish rules for what can go into later maintenance levels of a release
3. Establish rules for how a major.minor graduates from "early" to "late"
    - What does it mean for the previous 1 or 2 major.minor?
    - We owe special handling to 2.4 because it didn't start this way.
4. Formally document the above

How this would work over time:

- 2.6 is released with a few new/small things
- 2.4 is stabilized
- 2.6.$small continues to get the kinds of things we're doing in 2.4 today
- Eventually something big comes along and we do a 2.7 or 2.8
- 2.6 is stabilized when 2.7/2.8 is released
    - 2.4 sticks around but maybe we pick an EOL. For 2.4, we pick it farther out then we normally would since the policy is post-GA.

Problems

- Do we need to only pick a subset of major.minor's to be eventually-LTS so we don't end up with different distributions on arbitrary major.minors? This helps cap the # of streams in service AND avoids distributions picking different ones and causing more work on all sides.