**2023-11-08 - Officially Supported Technologies**

**Date**
08 Nov 2023

**Attendees**
- Craig McNally
- Ingolf Kuss
- Florian Gleixner
- Jenn Colt
- Julian Ladisch
- Maccabee Levine
- Zak_Burke

**Discussion items**

<table>
<thead>
<tr>
<th>Time</th>
<th>Item</th>
<th>Who</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 min</td>
<td>Scribe</td>
<td>All</td>
<td>Jakub Skoczen is next, followed by Taras Spashchenko</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Both are absent, so Jenn Colt will take notes today.</td>
</tr>
<tr>
<td>5-10 min</td>
<td>Officially Supported</td>
<td>All</td>
<td>Postgres 12 EOL Fall 2024...</td>
</tr>
<tr>
<td></td>
<td>Technologies</td>
<td></td>
<td>Handle in Quesnelia page Quesnelia - Technical Council - FOLIO Wiki</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Typescript needs to be addressed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Open question: Timelines</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Want to give people more lead time before the Poppy release</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Today:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Check in on progress</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Florian Gleixner (postgres 13? 14? 15? DR-000038 - PostgreSQL Upgrade to 16)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Wait for TC to talk about on Monday</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Jeremy Huff / Zak_Burke (typescript)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Lombok?</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MinIO/S3?</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Need to bring Quesnelia page up to date</td>
</tr>
</tbody>
</table>
Officially Supported Technologies - Upkeep

- See 2023-06-21 Meeting notes for the most recent discussion notes on this the upkeep process.

- Options:
  1. Capture/document the ad-hoc process which we've been following in lieu of a having a formally defined process
  2. Go back to the previous meeting notes and try to resurrect the process we started to define (see link to meeting notes above)
  3. Do #1 now (short-term/stop-gap solution) and continue to try to make progress on #2 (longer-term solution)

- Considerations:
  - Relevant release milestones
  - Expectations for the TCR process / Which list should we be looking at?
  - How do we ensure this isn't a moving target for developers (both internal and external to the project) throughout the release cycle
  - Do we need notification mechanisms for announcing changes to the OST list for the release in development?
  - Implications on planning for the next release cycle

Notes from today:
Focus on upkeep:

- How do statuses transition?
- What is active?
- Can be up to 4 releases
- Right now - Poppy, Orchid, Nolana. But Q is also "being developed" Or is it up to GA?
- Period of time where there is often 2 in dev. We wouldn't want to make Q have to comply with P
- What milestones should inform statuses? Where in the dev process can we nail things down? Statuses in line with dev processes?
- Milestones are in the release cycle but toward the end. Don't know until part way through the release cycle what version of RMB, etc
- Maybe changes need to be valid all the way down to active
- Changes happen right up to the last moment, and even during support periods
- Want to be able to make the decisions we can early but that won't be all of them
- Accepted would be nice but maybe not practical, too linear. Leave for now.
- Do the kinds of changes matter? Like only for security vs for features
  - Discuss more later
- If we accept changes all along does the page make sense at all? Can people still use it for planning if it is always a moving target?
  - Maybe for active only take security changes? But basic technologies stay the same
  - Freeze earlier in the cycle? Tell people early on if the version will change, like what Stripes does, RMB process is different? Stripes bumps version number early on then aggregates changes in the cycle to that release.
  - Ahead of cycle determine okapi/stripes/rmb versions?
    - Making decision on stuff we own different from external
    - Have to plan far ahead
  - Lists was Orchid version of stripes but it was going to be in Poppy. If new modules have to have released versions and next version will have a breaking change, they have to refactor
  - So if know going to break stripes do early in the cycle but then can they use snapshot versions?
    - challenge to say they have to use version that doesn't exist yet
    - snapshot versions interferes with reproducible builds
  - Distinguish between internal released tools and external ones?
    - Limit OST to things not internal?
    - Might be easier to talk about external separate from internal?
  - Historical policy around 1st party releases, could be talked about in the future
  - Ahead of cycle determine okapi/stripes/rmb versions?
    - Making decision on stuff we own different from external
    - Have to plan far ahead
  - Lists was Orchid version of stripes but it was going to be in Poppy. If new modules have to have released versions and next version will have a breaking change, they have to refactor
  - So if know going to break stripes do early in the cycle but then can they use snapshot versions?
    - challenge to say they have to use version that doesn't exist yet
    - snapshot versions interferes with reproducible builds
  - Distinguish between internal released tools and external ones?
    - Limit OST to things not internal?
    - Might be easier to talk about external separate from internal?
  - Historical policy around 1st party releases, could be talked about in the future
  - Ahead of cycle determine okapi/stripes/rmb versions?
    - Making decision on stuff we own different from external
    - Have to plan far ahead
  - Lists was Orchid version of stripes but it was going to be in Poppy. If new modules have to have released versions and next version will have a breaking change, they have to refactor
  - So if know going to break stripes do early in the cycle but then can they use snapshot versions?
    - challenge to say they have to use version that doesn't exist yet
    - snapshot versions interferes with reproducible builds
  - Distinguish between internal released tools and external ones?
    - Limit OST to things not internal?
    - Might be easier to talk about external separate from internal?
  - Historical policy around 1st party releases, could be talked about in the future
  - Ahead of cycle determine okapi/stripes/rmb versions?
    - Making decision on stuff we own different from external
    - Have to plan far ahead

- End up updating change column to clarify what can/is likely to change in statuses
  - Probably difficult to change release process so this makes sense to distinguish, but doesn't really help TCR problem
  - Might need to relax OST restrictions for 1st party items

- Do we still need accepted?
  - Might make sense for third party
  - Need to see if actually works practically and then come back to it maybe

- Upkeep activities
  - Drafting starts at GA for release that is two releases later (R starts at GA of P)
  - Move to accepted before release scope refinement deadline for given release
  - Move to active when development begins on that release, which is around time of feature freeze of the release prior to the given release
  - Archived when it is out of support, tied to GA of future release (2 out?)
  - TC should plan when the release schedule is published

Action Items