# Sysadministrivia

## Linux, Lagers, and Late Nights

---

# S5E1: "The High Costs of Technical Debt"

**Posted** 2020-03-01 23:59
**Modified** 2020-02-29 03:51
**Comments** 0

**Navigation**

| Previous Episode | Next Episode |
|---|---|
| S5E0: "Frameworks Without Much Bite" | S5E2: "I Am the Keymaster; Are You the Gatekeeper?" |

**Log**

| Recorded (UTC) | Aired (UTC) | Editor |
|---|---|---|
| 2020-02-20 03:20:37 | 2020-02-29 07:31:31 | "Edita" |

**Verification**

| Format | SHA256 | GPG | Audio File |
|---|---|---|---|
| MP3 | `1a8824fe3253234715ca6b0f51c3c23139e3968f459f2007e5d19644c3458c37` | click | click |
| OGG | `1784815f0552e8f515306f7f3d17cbe623b93e995e8ea447ca749055e966c932` | click | click |

Quicklisten:

We talk about technical debt – what it is, significant causes, potential solutions.

- Just the Tip
- Notes
- 15 Clams
- Errata
- Music

## Just the Tip

- Paden talks about notekeeping/notetaking.
  - He mentions Microsoft's OneNote?.
    - Both he and Jthan love it.
  - Also mentioned:
    - Etherpad
    - Evernote
- I talk about using my Android as a slide remote in LibreOffice Impress. You can see that talk here.

## Notes

Starts at **12m04s**.

I was drinking water. Paden was drinking Dr. Pepper and water. Jthan was drinking Tension Tamer tea.

- Technical debt
  - You pay in wasted time (and effort, etc.) instead of money
  - It is when it costs you more to maintain a system or system of systems – be it time, effort, sometimes money, etc.
    - "Anything you do that is ultimately going to make you do more work later, that is (incurred) technical debt"
  - Causes:
    - "Improper" staffing
      - Understaffing
      - Overstaffing
      - Misappropriation of roles/duties
      - Poor allocation of workloads
      - etc.
    - I'm fairly certain the leading cause is **not doing things the right way the first time**
      - Because it takes longer, and the higher-ups can get impatient…
      - If they lack vision/foresight…
      - Because humans are fucking dumb and are notoriously terrible at planning, generally speaking. (But also, see the "impatient" thing.)

    - Other causes include:
        - NIH (when it would make sense to use an outside product/solution already developed)
        - Not understanding scope of the problem (and therefore not understanding the scope needed for the solution), or not understanding the capabilities of existing possible solutions, etc.
        - Demands being made without understanding of the technology behind them.

- Jthan mentions our segment we did on HackOps
- Technical debt looks like:
    - e.g. rebooting a server every day instead of finding root cause
    - If you're spending more time and/or repeatedly doing something mundane rather than fixing it/automating it
    - Applying multiple patches to a codebase instead of taking advantage of plugins/APIs/hooks, etc.
- Fixing a single technical debt has benefits beyond just fixing that particular issue.
- Our goal in Operations is to **minimize** downtime and **increase** availability/reliability, rather than completely eradicating downtime (because that's impossible).
    - Yes, Virginia, sometimes this means rebooting to patch your kernels – because fixing security vulnerabilities and other bugs will **increase the reliability** of the machine. Your uptime clock means shit if you're running a 2.x Linux kernel.
- Potential solutions
    - Documentation
    - Delegation
    - Do it the Right Way the first time. Best practices are called as such for a REASON – they've been time-tested.
    - While you can't know the best approach before applying one sometimes, it's important to have a deep understanding of your environment **before** developing a solution.
        - Which is why we all hate outside vendors – they know nothing about your environment, and expect you to bend your environment to their tool rather than being able to bend their tool to your environment.
    - Fight NIH by simply asking "*is this (self-developed thing) a nice-to-have or a necessity?*"
    - "Is this something we're planning to keep deployed rather than just a temporary project? If so, we want to do it the best way possible for the long run."
        - For **purely temporary** (~1 week or less) **and** single-use needs (i.e. BOTH conditions), it's better to have a partially-functioning thing than no thing at all.
        - "Deploy/release early, deploy/release often" is TERRIBLE advice and a fundamentally **broken** theory for production/deployment.
            - This is why you don't trust a developer with deploy access for production systems, and why the vast majority of "DevOps" concepts are broken by nature.
    - Sometimes you have to let go of your sentimentality/attachment to a project/solution/approach when it no longer fits/scales/works.
    - Use configuration management/system orchestration.
        - They help you solve your existing technical debt…
        - But also allow for you to **expand your existing functionality and processes** easily. A "force multiplier".
        - It supports being "future-minded" and gives you a "golden road forward to avoid future technical debt".
    - If you're not good at future-planning, that's okay – but you ABSOLUTELY should not be making these decisions on your own then.
- The entire reason technical debt is bad is because **it keeps you away from new deployments/features/etc.**.
    - Ironic, because a leading cause OF technical debt is **deploying new things *before you are ready for them***.
- Paden mentions The Phoenix Project but take it with a grain of salt; a vast majority of the theory presented in it is only applicable for HUGE deployments. Don't fool yourself into thinking you need to support millions of servers when you have less than 10k, let alone 50 or so.

# 15 Clams

In this segment, Jthan shares with you a little slice of life. The title is a reference to this video. (2m16s in)

Starts at **48m01s**.

Jthan talks about an interaction we had on Twitter. Essentially, someone asked for a recommendation for a turnkey ZFS solution. Like many (all?) turnkey solutions, there are none that are good. This person took offense, and accused some of us of "gatekeeping" – which is dumb. If you don't know what you're doing, DON'T DO IT.

Jthan, who LITERALLY is a storage admin and deals with ZFS every day and has many a fight with FreeNAS, should know why FreeNAS is terrible. Right? You'd think, but apparently it's bad for him to suggest people not use it and learn how to properly administer ZFS in the first place, according to some stranger on the Internet who was complaining about trying to fix a FreeNAS box in the first place (because the original users couldn't).

"It's very appealing to look at [turnkey solutions], but … they only lead to technical debt." Who fixes the turnkey solution when it breaks? Not the people that are enticed by the apparent *ease* of a turnkey (because they have no idea of the internals), and not the people that have the knowledge/ability/experience to turn up their own solution in the first place (because the turnkey solution is **too limiting** to them). Ergo, turnkey solutions are by nature **useless**.

# Errata

- Jthan's body is ready.
- Paden is a little confused about what technical debt actually is. It's nothing to do with overpurchasing of hardware, and isn't really a fiscal thing necessarily.
- AI **really is** just a bunch of if statements, despite what people trying to sell you AI-backed things or developing AI-backed things are saying. Be wary of them, as they only wish to justify their pricetag.
- "Grok" is in both the original Hacker's Dictionary (also referred to as the Jargon File), and the new one.

# Music

**Music Credits**

| Track | Title | Artist | Link | Copyright/License |
|---|---|---|---|---|
| Intro | Into tha future | Smooth Genestar | click | CC-BY-NC-ND 4.0 |
| Outro | Bossa Manao | Xcel/Word | click | CC-BY-NC-ND 4.0 |

(All music is royalty-free, properly licensed for use, used under fair use, or public domain.)

**Author** r00t^2
**Categories** Season Five

# Comments

There are currently no comments on this article.

---