Tales about updating a software appliance over 15 years

IACBOX

about::me

Lukas Ruetz

Coding since 20 years

First 10 years self employed

Second 10 years for Asteas

- Based in Landeck and Innsbruck

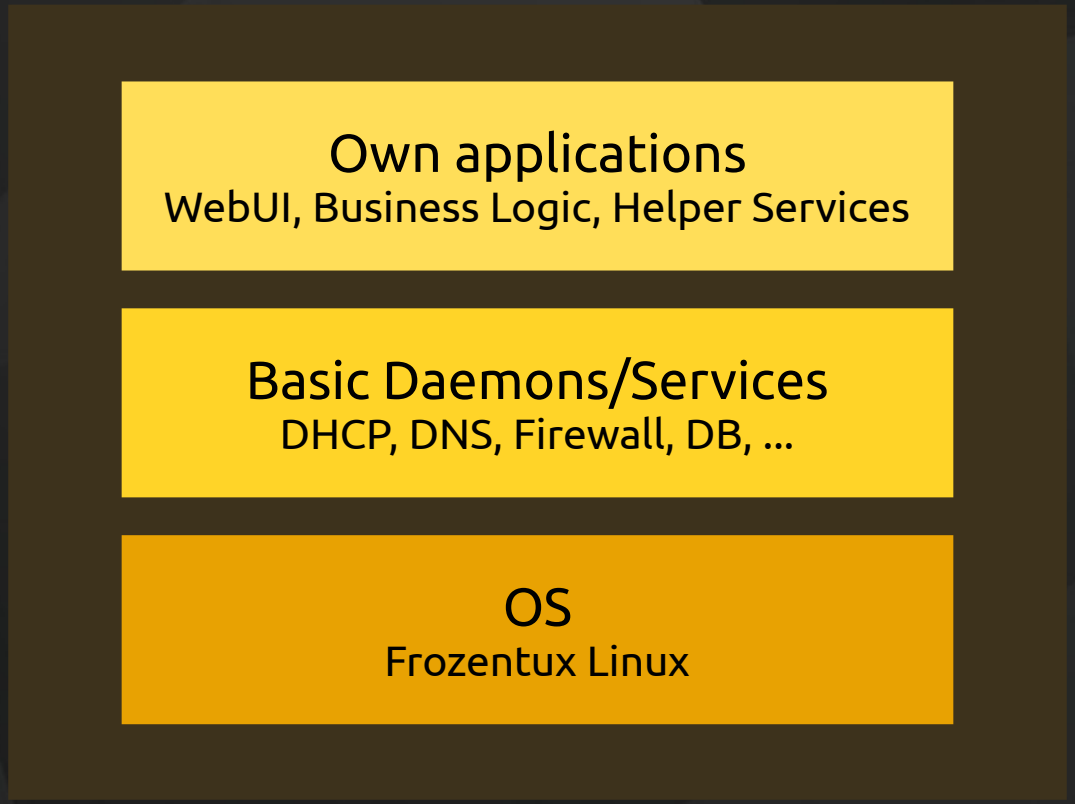
IACBOX

Updated for 17 years ...

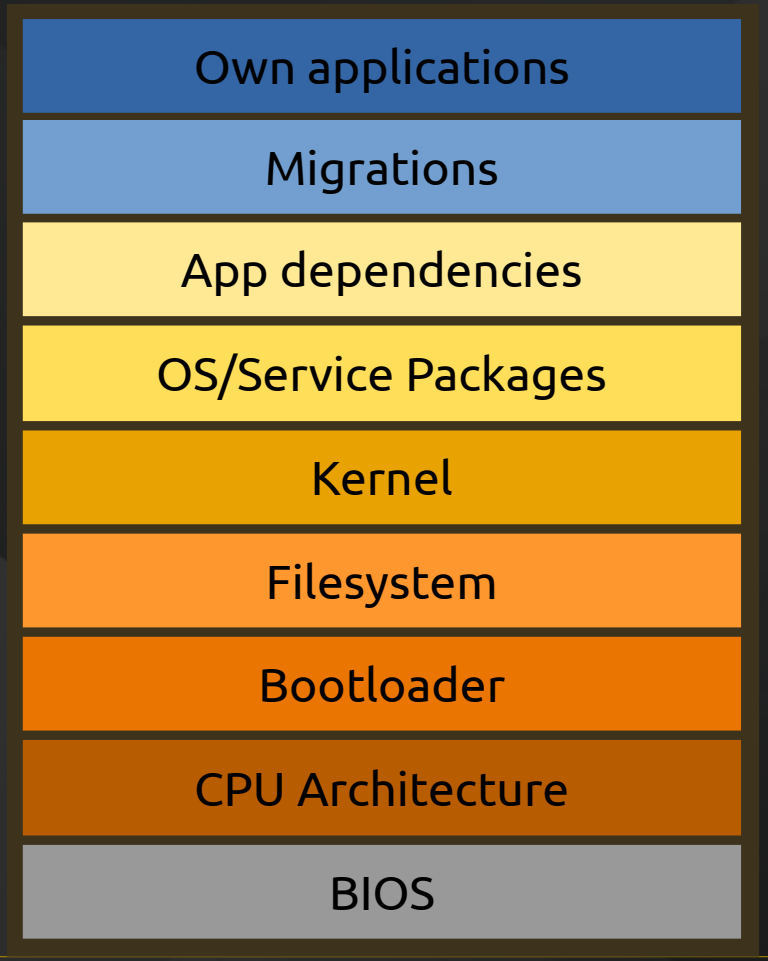
IACBOX

- Internet Access Control
- Captive Portal
- Manage and control access to your (WiFi) network
- Running in Hotels, Hospitals, Airports, Schools, Cruise ships, ...

Software appliance



Software appliance



App code

Migrate state (DB schema evolution, files on disk, …)

App dependencies and transitive dependencies

Linux default packages. Migrations maybe needed too

Linux kernel (incl. custom patches and modules)

ReiserFS → ext4

Lilo → Grub2

32bit → 64bit

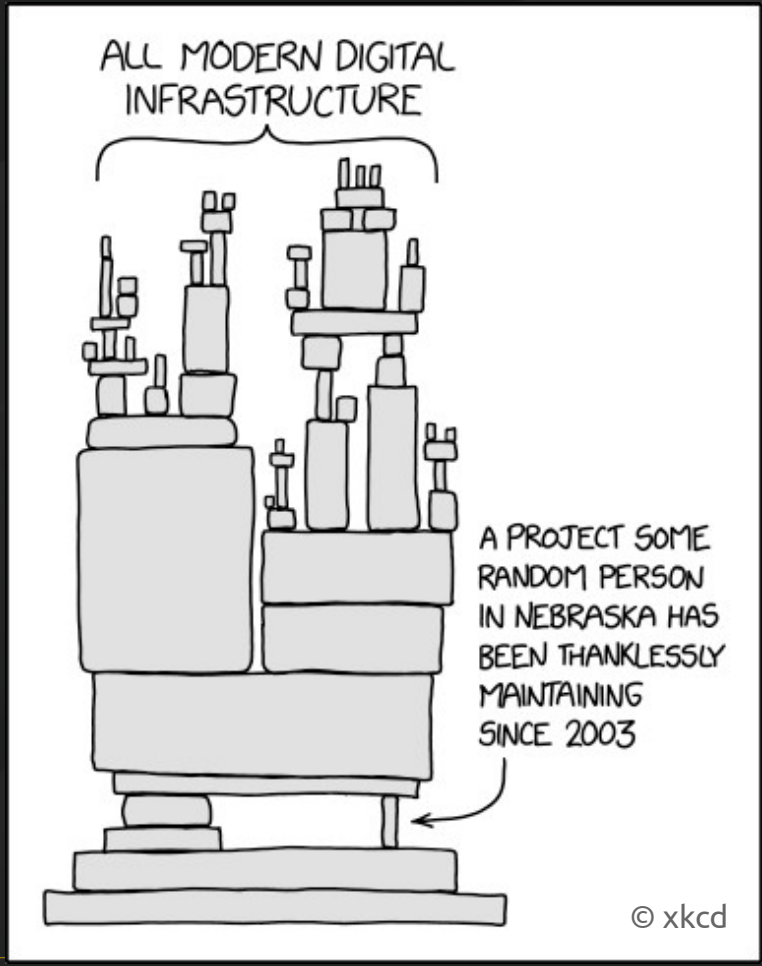
Legacy BIOS → EFI (reinstall needed)

Why Updates?

- Add new features
- Fix old bugs
- Fix new bugs (introduced in the prev. update)
  - Worst: re-introduce an older bug (regressions)
- Remove features
  - Data-driven decision (Telemetry)
    - But take care of needed „Marketing Features“
  - Communication needed
  - Think twice before introduce new features
- But … most of your stack is not your code!

Why Updates?

- Dependency hell
- State of a dependency – still maintained? EoL? Incompatible update path?
- Breaking (external) API changes
- Breaking changes on the client side (Browser, iOS and Android updates, ...)



Create an Update

In the old days…

- Lot’s of manual plumbing, 10 Bash Scripts, archive creation
- Error prone workflow, especially with many iterations

Now

- Tooling to automate 95%

Next major

- Full automated image and patch update generation

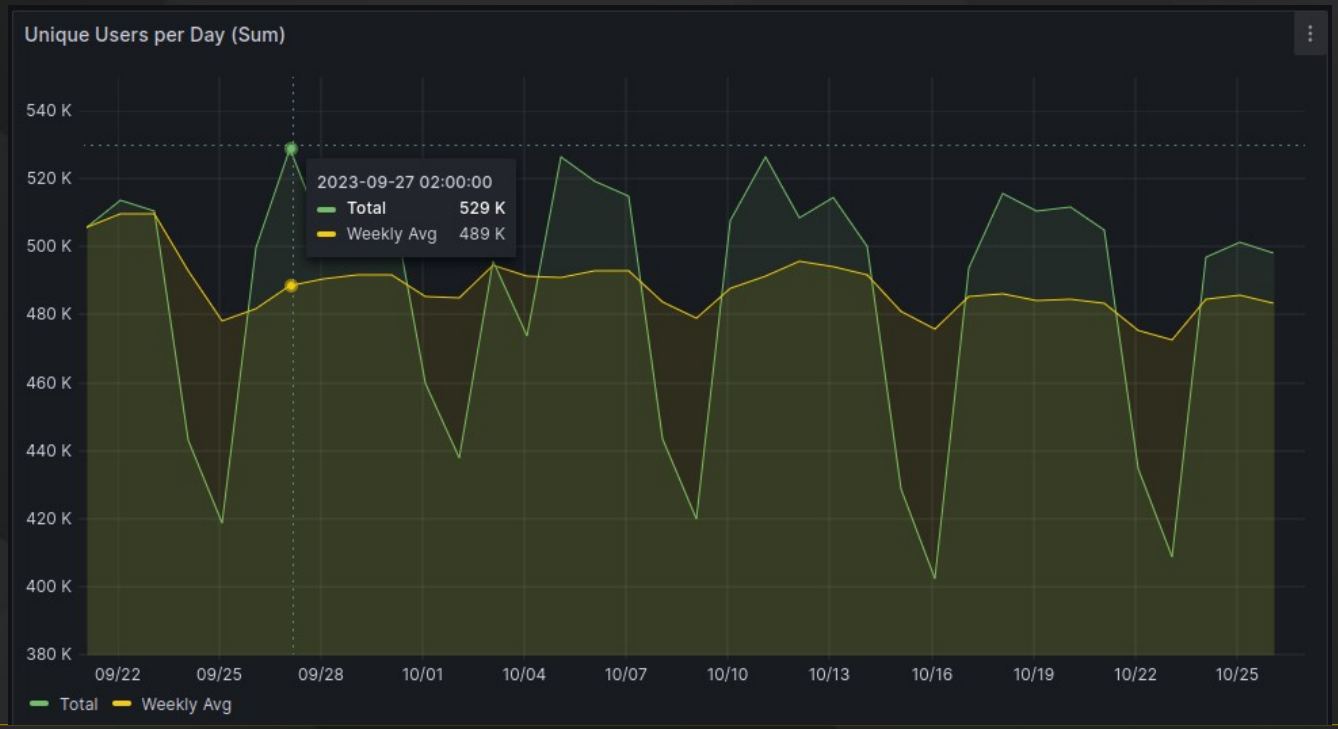
Update process

- Check for updates (till now once per week, once per day with next version)
- Check for valid software maintenance
- Download, install, migrate, (and most of the time) reboot
- Pray (on a major kernel upgrade) and monitor
- Update special parts (DNS filter lists, custom themes, ...)
- Sequential updates only, no per-version updates
  - In the past: Download and udpate one-by-one (incl. all kernels reboots)
  - Now: Download all updates, only the newest kernel, only one reboot
  - Improvement with >20 Updates: 45min → 4min

Update Release Strategies

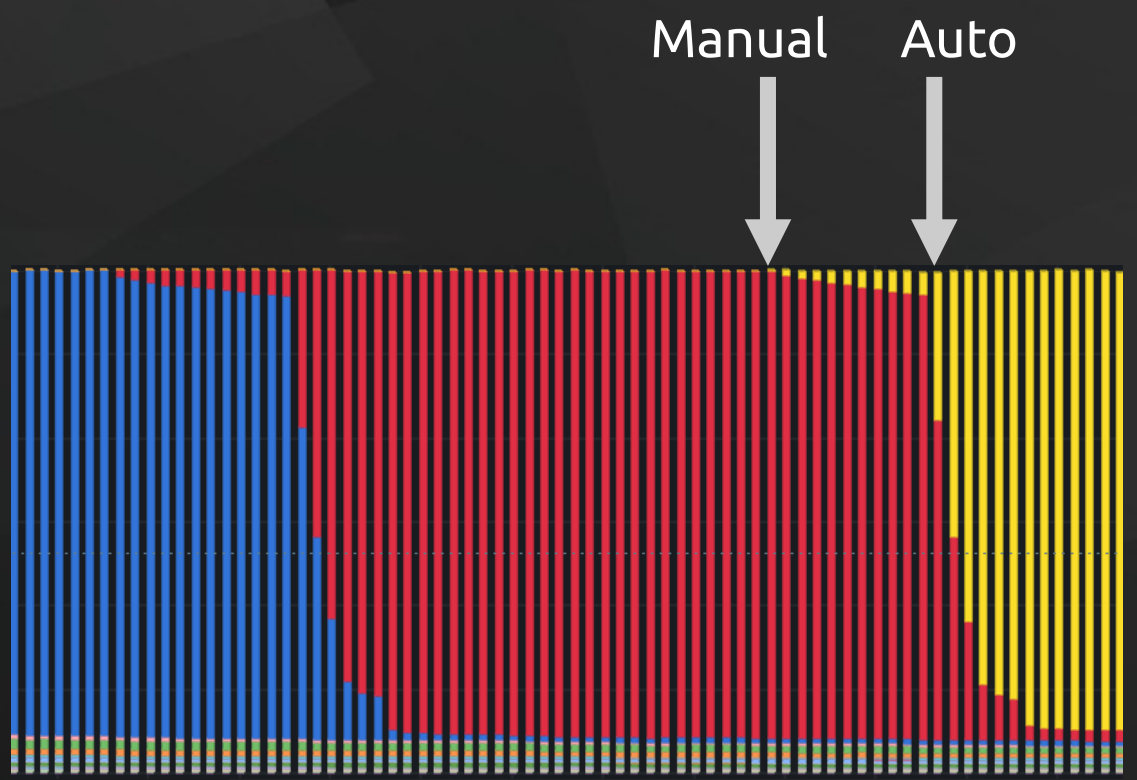
When to update *which* systems (out of thousands)

- Update all systems at once?



Update Release Strategies

- Our approach: 2 Phases + random distribution
  1) Manual phase
  2) Auto phase



Update Release Strategies

- Distribution of Auto-Updates
  - Random pick by a certain propability like 5%/day
  - Geographic location
  - Per Feature/Tag (any business property)
  - Limit number of updates per customer (felt quality)
  - We use a combination of different „selectors“ above
- Server-side logic that has the final decision
  - Custom forced-upgrade/deny-upgrade list

Security

An online update feature is like a
vendor built-in
remote code execution
functionality

Security

Back in the old days ...

- Make it *look* encrypted
- `md5(static-key + pwd)`
- Unencrypted transport

Later...

- `hmac_sha256(salt + static-key, pwd)`
- Better but with the same issues
- Encrypted transport

Now: Public-Key Crypto

- Ship only public keys
- Provision multiple keys
- Think about key roll-over
- Sign *all* downloaded files
- Take care of our keys ...
  - Use a HSM
- TLS certificate pinning: good but dangerous

Murphy: What can go wrong?

- Self DoS
- Update Loops
- Something somewhere
  - The great wall(s)
  - Firewalls
    - Blocked ports
    - TLS interception
- Wrong Time (VM Snapshot restore) → TLS cert verification failed
- Outdated root CA
- Power outage
- Layer 8 issues
  - Impatience

Thank you