Migrate from PowerDNS to Knot with help of Catalog Zones

06.09.2023 · Klaus Darilion · Head of Operations

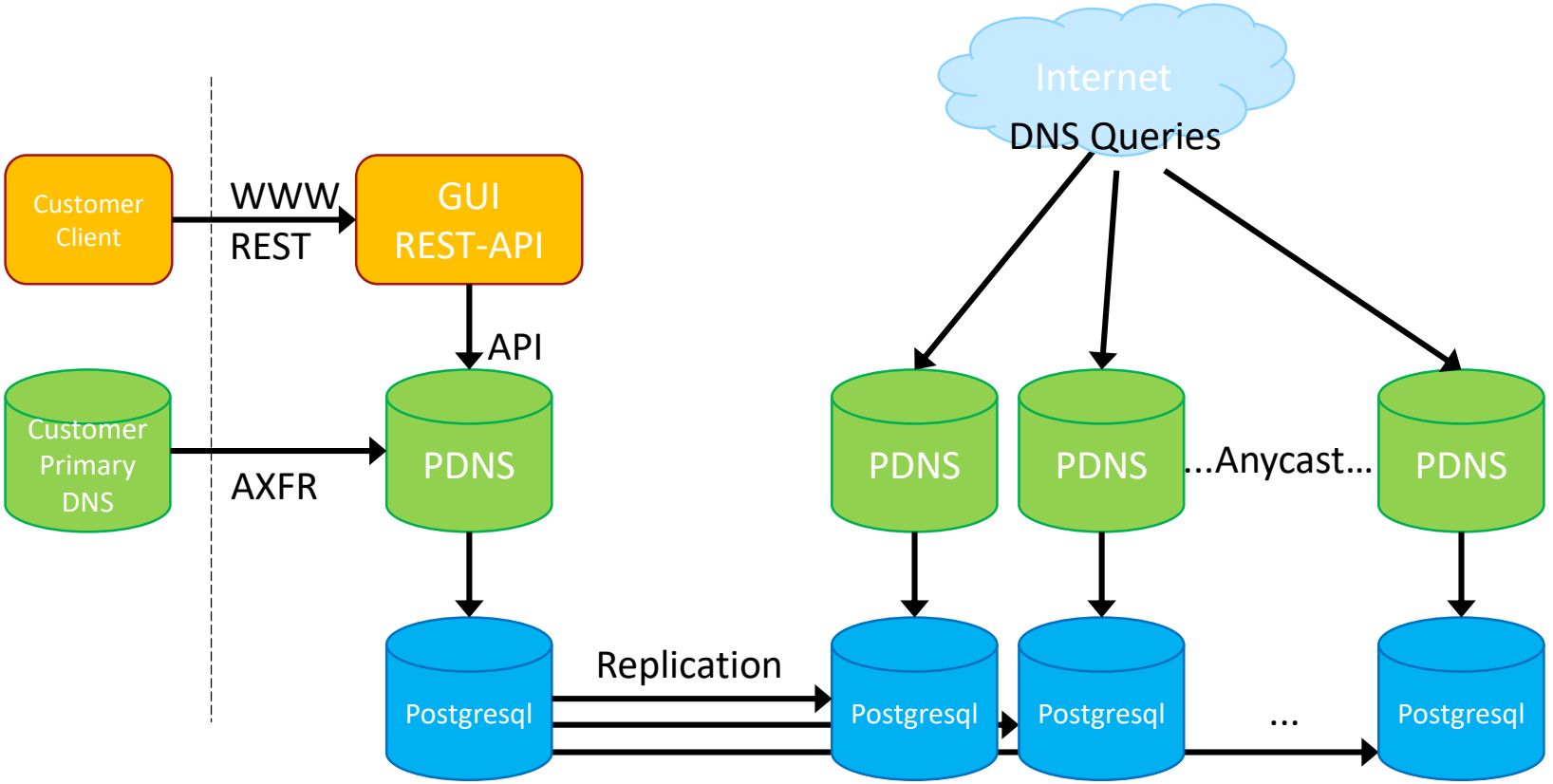
Klaus Darilion

- Head of Operations @ nic.at
  - .at TLD registry
- Linux Sysadmin
- RcodeZero from the beginning: 13 years

RcodeZero DNS

- Secondary DNS for TLDs (at, hu, si, pt, nl, eu, fi, ie …)
  - 30+ nodes (not topic today)
- Primary/Secondary DNS for 2nd level domains for Enterprises and Registrar/ISP
  - 50+ nodes, 2 Anycast Clouds
  - 4+ Mio Zones

Architecture



Architecture

- Replication fast and stable, PowerDNS flexible and stable,

but

- Random Subdomain Attacks: 24x7
- PDNS+DB too slow for Random Subdomain Attacks

→ We need a faster name server

→ Knot has reputation for fastest name server

Engineering Requirements

- Knot public facing
- Provisioning still needs PDNS (plenty of DB magic)

- How to get zones from PDNS to Knot?
  - Provisioning: add/delete zones
  - Zone data: AXFR

Zone Provisioning Options in Knot

- knotc
  - conf-begin
  - conf-set
  - conf-commit
  - → very very slow for 1mio zones

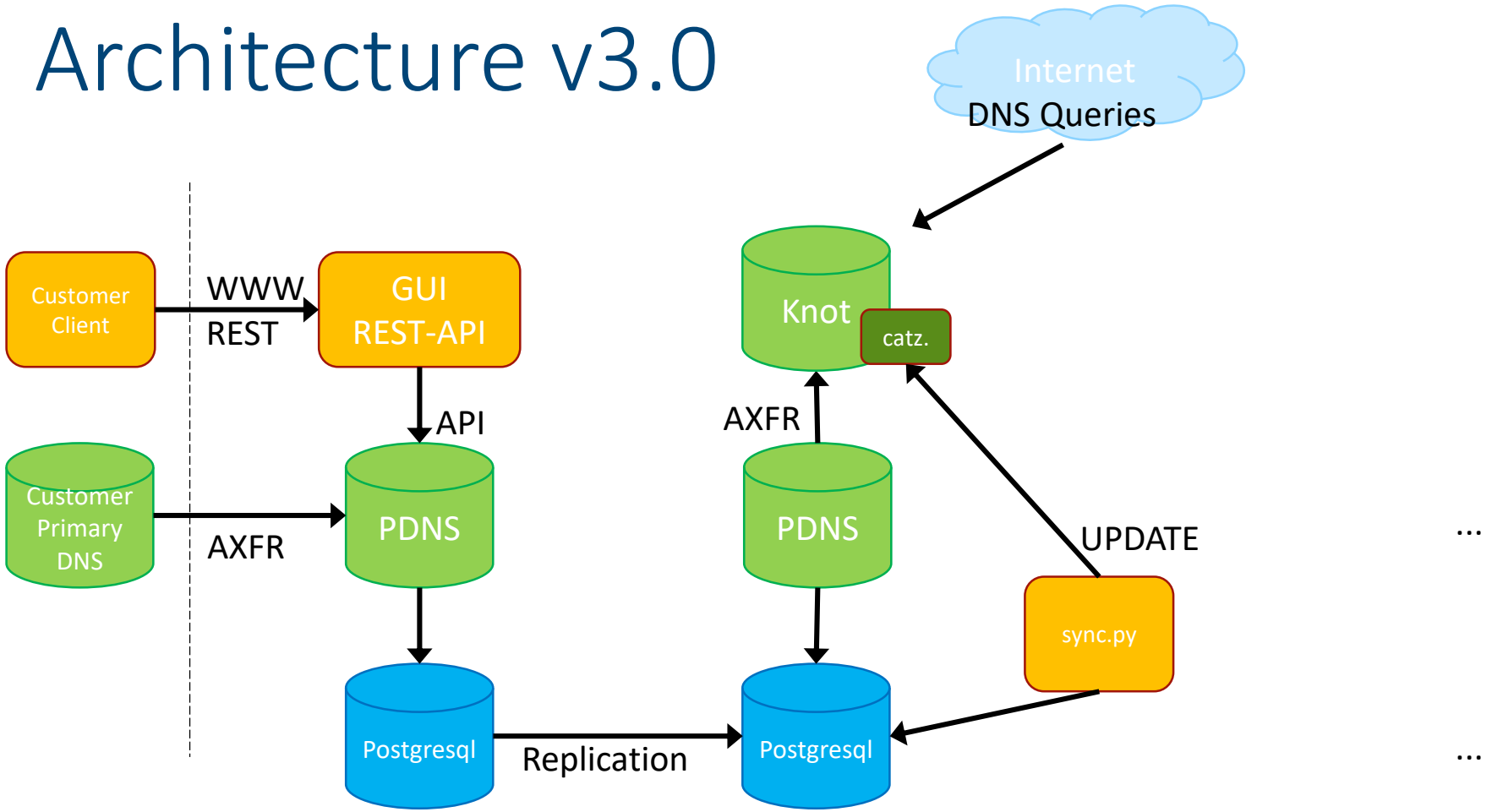
- Rather new „Catalog Zones“ support -> Let‘s try that

How to get the Catalog into Knot?

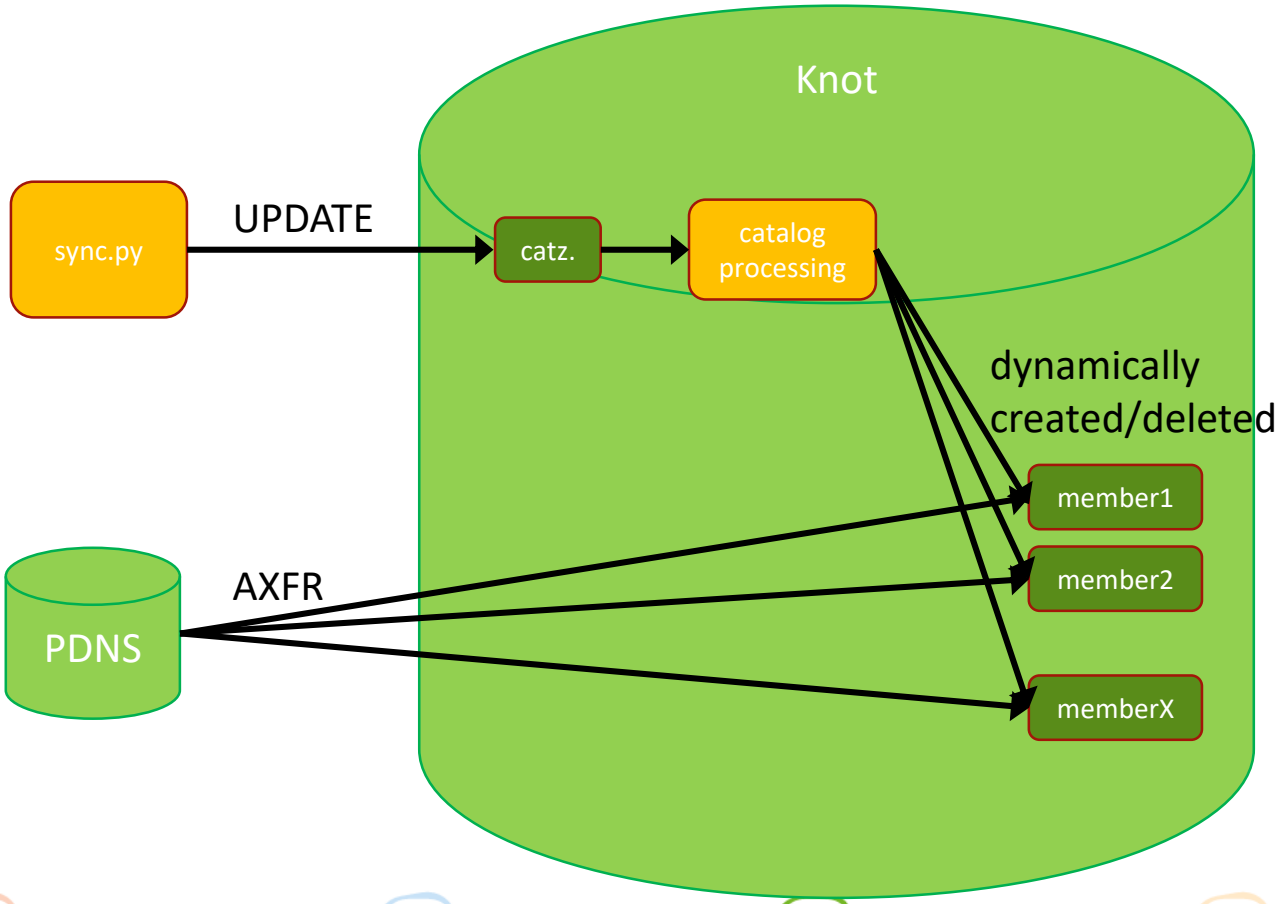
- PowerDNS 4.7 as catalog producer, Knot as consumer
  - No outgoing IXFR support in PDNS
  - AXFR of 1mio RRs every few seconds -> not nice
  - My insticts said: that may be dangerous

- Solution
  - Initially dump catalog zone from PDNS-DB
  - Knot loads catalog zone as primary zone
  - Changes in DB trigger DNS UPDATEs to catalog zone

Architecture v3.0



Knot Internals



Problem: Slow Catalog Updates in Knot

- UPDATE response: 1s -> OK
- Catalog processing: 30-200s (depending on hardware) -> not OK
- Workaround: Batching multiple events into one UPDATE
- Solution: Talking with Knot developers
  - after some refactoring: 1-5s

Problem: Zone Bootstrapping

- After initial catalog loading, the zones must be AXFRed from primary:
  - 2-8h for 1mio zones, depending on hardware
- How to know if bootstrapping is finished and server can be put into production?
  - No indication from Knot
  - Check from outside is necessary

Problem: Restarts

- Knot restart: 30-300 seconds (0.1 seconds for PDNS)
  - Config changes + restart require to temporarily disable the Anycast node (disable BGP)
  - Administrative overhead increases

Problem: ALIAS

- ALIAS (ANAME): (deprecated) substitute for CNAME on zone appex
  - Authoritative name server resolves the ALIAS target
- PowerDNS and our service supports it ☺
- Knot does not support ALIAS ☹

Workaround: ALIAS

- AXFR from PDNS to Knot triggers ALIAS resolving in PDNS
- Retransfer zones periodically to catch up ALIAS target changes
  - Some load balancers have 30s TTLs – retransfer every 30s?
- AXFR for 11000 zones with ALIAS RRs takes 150 seconds
  - Resolving takes time, AXFR-out not optimized in PDNS
- Knot unresponsive during massive AXFRs
- → ALIAS is still a problem

Biggest Problem: Zone Syncronicity

- Is Knot in sync with PDNS+Database?
  - Does Knot have the same zones provisioned as PDNS?
  - Do zones have the same serial?
- In DNS nobody knows if zones are in sync
  - Primary: sends NOTIFY, but does not care if secondary fetches zones, and does not track secondaries
  - Secondary: act on incoming NOTIFY or refresh TTL
- Syncronicity monitoring must be done out-of-protocol

Syncronicity Monitoring

- Between 2 nameservers (ie PDNS and Knot)
  - Get the list of zones
  - SOA queries for every zone against both nameservers
  - 1mio queries every 1-5 minutes? = Selfmade Random Subdomain Attack
- Alternative: Get zone list with serial and compare

Zone List with Serial

- PDNS + SQL Backend
  - SELECT content FROM records WHERE type='SOA' …
- Knot
  - knotc zone-status →blocks control sockets for 10 seconds -> Knot somehow sensible to concurrent knotc commands
  - kjournalprint → bypassing Knot, directly read the LMDB data files
    - sensible, stale read locks on LMDB cause rapid growing journal size -> fixes in Knot 3.2.9 and 3.3.0

Conclusion 1

- PDNS+SQL Backend: „Family Van“
  - Problems? Stop and start within (milli)seconds
  - Very flexible (database tricks)
  - Slow for Random Subdomain Attacks
- Knot: „Dragster“
  - Very fast in QPS
  - Knotd restart takes time (like for a Dragster)

Conclusion 2

- It runs in production since 10 months
  - Would not have been possible without extensive support and fixes/improvements from Knot and PDNS developers
  - Random Subdomain Attacks are not a problem anymore
  - Catalog zones ease the zone provisioning,
  - Checking zone syncronicity is complex and CPU intensive

Future

- Improve Zone Syncronicity Monitoring
- Alternatives:
  - PDNS+LMDB + Lightning Stream replication
    - Combines zone provisioning and zone data replication
    - Not as fast as Knot, but maybe worth due to much simpler syncronicity efforts
    - https://github.com/PowerDNS/lightningstream
  - Make PDNS+DB backend faster
    - Load zone completely from Backend if high qps
    - Implement „aggressive caching“ in PDNS cache

Klaus Darilion · Head of Operations

klaus.darilion@nic.at

Problem: non-compliant zones

- PowerDNS is not strict when reading from DB
  - Send to AXFR what is in the DB
  - Plenty of invalid RRs in our DB (received from customers)

- Knot strictly RFC compliant
  - CNAME/DNAME besides other RRs, …

- Solution (by Knot developers): semantic-checks: **soft**

Problem: corrupt AXFR

- PDNS Bug: „overfilled chunk“ in outbound AXFR
  - PDNS 4.6: AXFR from PDNS to Knot
    - Records were missing
  - Workaround available in PDNS 4.8
  - https://github.com/PowerDNS/pdns/issues/11804