Querying DNS for software updates

Mechiel Lukkien
mechiel@ueber.net
https://github.com/mjl-

Agenda

- Part 1:
  - Updates through DNS, manually
  - Automating DNS for Go with gopherwatch.org
- Part 2:
  - Automating upgrades with ysco
  - On-demand binaries with beta.gobuilds.org

Updates through DNS, manually

```
$ dig +short TXT _updates.xmox.nl
"v=UPDATES0;l=v0.0.15"
```

- Release involves updating DNS record

DNS is a good fit

- Low overhead
- DNS caching, redundant servers
- Can’t track exact installations/IP’s

Automate DNS records for Go ecosystem

- All Go software is in a transparency log (*tlog*) called *Go sumdb*
  - *modules* and semver versions and hashes of source code
  - Git tags like v0.1.2, also pseudo-versions for commits
- Gopherwatch.org was already monitoring the tlog and building database, just needed a DNS server

Example for github.com/mjl-/mox

```
$ dig +short TXT mox.mjl_2d._.github.com.v0.l.gopherwatch.org
"v=v0.0.15 t=6802a84d"
```

```
                                        .gopherwatch.org
                                    .l
                                .v0
                        .github.com
                    ._
            .mjl_2d
        mox
```

Example for Go toolchain

```
$ dig +short TXT toolchain.v0.l.gopherwatch.org

"v=go1.25.6 k=cur t=69693421; v=go1.24.12 k=prev
t=69693421; v=go1.26rc2 k=next t=69693421"
```

Gopherwatch DNS server

- Built with github.com/miekg/dns
- <1000 lines of extra Go code
- DNSSEC with online signing and compact denial of existence

How the Go sumdb gets filled

- Go toolchains request “latest” source code through Go module proxy
  - “go install github.com/mjl-/mox@latest”
  - Use external library and “go get”
  - “Latest” is resolved and new entries added to tlog

To do

- Add second DNS server to l.gopherwatch.org for redundancy
- Test/fuzz the DNS request handling. Tips for test suites?
- Responses with tlog proofs, and verifiable indexes.

Part 2

a) Automating upgrades with ysco

b) On-demand binaries with beta.gobuilds.org

Automating upgrades with ysco

- “./mygosvc” → “ysco run ./mygosvc”
- DNS query for updates every 24h
  - Also query for new Go toolchains (static linking)
- Upgrade (download, terminate & restart):
  - One-click, through web interface
  - Automatic: delayed, policy (patch/minor), schedule
- No modifications to applications needed!

Ysco - Managed automated updates

Check for updates

Pause updates

Scheduled updates

No updates scheduled.

Service: github.com/mjl-/mox / "v0.0.16-0.20251105170702-29f5e86b2f51" go1.25.6

Module

- ○ v0.0.15 tag
- ◉ v0.0.16-0.20251105170702-29f5e86b2f51

Go toolchains

- ◉ go1.25.6
- ○ go1.24.12
- ○ go1.26rc2

Update now

Version v0.0.16-0.2025110517 Go version go1.25.6 Update now

Ysco: github.com/mjl-/ysco / "v0.1.2" go1.25.6

Module

- ◉ v0.1.2 tag

Go toolchains

- ◉ go1.25.6
- ○ go1.24.12
- ○ go1.26rc2

Update now

Version v0.1.2 Go version go1.25.6 Update now

beta.gobuilds.org

- Service that compiles Go source on-demand
  - Source code(*) is verified through Go sumdb (*tlog*)
  - Automatically downloads new (verified) Go toolchains
- Cross-compiled for any OS/arch
- Reproducible builds (verified against 2nd server)
- Adds hashes of binaries to its own tlog
- Go toolchain only runs its own code to build, none of the project being compiled

https://beta.gobuilds.org

Summary

One-click or fully automated upgrades from Go source, without changes to applications

- Find updates for all Go software over DNS
- Use ysco to easily upgrade any Go service
- Using beta.gobuilds.org to compile on-demand

More

https://github.com/mjl-/gopherwatch

https://github.com/mjl-/ysco

https://github.com/mjl-/gobuild

https://www.ueber.net/who/mjl/blog/p/ysco-managed-automated-updates-for-go-services/