cert-manager-webhook-libdns: One Webhook, Many DNS Providers

2026-03-03

How cert-manager-webhook-libdns enables DNS-01 across many DNS providers via libdns, with practical operational improvements, compatibility automation, and a quick Kubernetes deployment path.

---

cert-manager-webhook-libdns: One Webhook, Many DNS Providers

Managing DNS-01 challenges with cert-manager often means choosing one provider-specific integration and staying tightly coupled to it. cert-manager-webhook-libdns takes a different approach: it uses the libdns ecosystem so one webhook can support many DNS providers through a shared interface.

If you run mixed infrastructure, migrate between DNS vendors, or just want a cleaner operational model, this project is built for that.

Why this project exists

In many Kubernetes setups, certificate automation becomes harder than it should be because DNS integrations are fragmented:

- Every provider has different APIs, credentials, and operational quirks.
- Switching providers often means switching tooling.
- Testing and maintenance can become provider-specific and brittle.

cert-manager-webhook-libdns solves this by exposing a single cert-manager webhook that delegates DNS record operations to pluggable libdns providers.

What it does

At a high level, the webhook:

- Receives DNS-01 challenge requests from cert-manager.
- Loads provider credentials from Kubernetes Secrets.
- Resolves and writes challenge TXT records through the selected libdns provider.
- Cleans up records after validation.

It supports important real-world behaviors, including:

- Handling multiple TXT values for wildcard + apex certificate requests.

- Safer fallback logic when record listing fails.
- Provider-specific TTL handling (for example, deSEC minimum TTL enforcement).
- Deterministic provider listing in CLI output.

Operational improvements in recent iterations

Several practical improvements were added to make this project easier to run and maintain:

- Semantic version output via `--version` (currently `v0.5.0`).
- No etcd/envtest dependency in the default `make test` path.
- Cleaner container workflow with `Containerfile` support.
- Release workflow hardening:
- tests run before image release
- `latest` is only published for stable tags
- release creation happens after successful image push
- concurrency guard to prevent overlapping tag releases

These are the kinds of changes that reduce surprises in CI/CD and day-to-day operations.

Provider compatibility is now automated

A major maintenance challenge with libdns-based projects is knowing which upstream provider modules are actually compatible with your pinned `libdns` version.

This repository now includes an automated compatibility process:

- Script: `scripts/check_libdns_provider_compat.sh`
- Workflow: `.github/workflows/libdns-provider-compat.yml`
- Report snapshot: `docs/01_provider-compat-latest.md`
- Historical snapshots: `docs/YYYY-MM-DD_provider-compat.md`

The check uses `gh api` to scan provider repositories, inspect their `go.mod`, and classify compatibility against the libdns version used in this project.

That gives maintainers a repeatable compatibility signal instead of manual guesswork.

Quick start

1. Build and publish the image

Use your preferred container runtime (Podman is preferred, Docker fallback is supported):

```
make container-build
make container-push IMAGE_TAG=v0.5.0 REGISTRY=ghcr.io/<your-org>
```

2. Deploy the Helm chart

```
helm install libdns-webhook ./deploy/libdns-webhook \
  --namespace cert-manager \
  --set image.repository=ghcr.io/<your-org>/cert-manager-webhook-libdns \
  --set image.tag=v0.5.0 \
  --set groupName=acme.yourdomain.com
```

3. Create provider credentials secret

Example for token-based providers:

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: dns-provider-credentials
  namespace: cert-manager
type: Opaque
stringData:
  api_token: "<YOUR_API_TOKEN>"
```

4. Configure ClusterIssuer

```yaml
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
  name: letsencrypt-prod
spec:
  acme:
    server: https://acme-v02.api.letsencrypt.org/directory
    email: your-email@example.com
    privateKeySecretRef:
      name: letsencrypt-prod-account
    solvers:
      - dns01:
          webhook:
            groupName: acme.yourdomain.com
            solverName: libdns
            config:
              provider: cloudflare
              secretRef:
                name: dns-provider-credentials
                namespace: cert-manager
```

Useful CLI checks

Before deployment, you can quickly inspect the binary:

```
./webhook --version
./webhook --list-providers
```

This is helpful for validating exactly what was compiled into your image.

Who should use this

This project is a strong fit if you:

- Run cert-manager and use DNS-01 challenges.

- Need flexibility across multiple DNS providers.
- Want one consistent webhook integration instead of many provider-specific implementations.
- Care about release discipline and compatibility visibility.

Final thoughts

cert-manager-webhook-libdns is not just a DNS integration layer. It is an operational strategy: keep cert-manager integration stable while letting providers vary underneath.

If your platform team wants fewer moving parts and more predictable certificate automation, this is a practical approach worth adopting.