KubeCon | CloudNativeCon

Europe 2022

WELCOME TO VALENCIA

KubeCon CloudNativeCon Europe 2022

CoreDNS: Intro & Deep Dive

John Belamaric, Google
Yong Tang, Ivanti

What is CoreDNS?

- Flexible DNS server written in Go
- Focus on service discovery
- Plugin based architecture, easily extended
- Support serving via DNS, DNS over TLS, DNS over HTTP/2, DNS over gRPC
- Support forwarding to upstream via DNS, DNS over TLS, DNS over gRPC
- Integration with Route53/Google Cloud DNS/Azure DNS
- Integrates with Prometheus, Open Tracing, OPA
- Default DNS server in Kubernetes
- Basis for node local cache feature in K8s

Latest Updates

- 1.8.5 – 1.9.2
  - 1.9.2 Released May, 2022
  - New plugins: **`geoip, header`**
    - **`geoip`** `reports where the query comes from`
    - **`header`** `allows fiddle with header bits`
  - Backwards incompatible changes
    - **kubernetes**: Removed wild card query functionality.
    - **route53**: Plaintext secret in Corefile deprecated.
  - Built with golang 1.17.8+ since 1.9.1
    - golang < 1.17.6 security issues

Security Audit

- Completed by Trail of Bits (March 2022)
- Sponsored by Linux Foundation
- 1 ***high*** severity issue (TOB-CDNS-8):
  - May lead to cache poisoning attacks.
- 1 ***medium*** issue (TOB-CDNS-12)
  - Mitigation possible without coredns update.
- 12 *low* or *informational* issues.
- All have been resolved now.
- Report available:
  - https://github.com/coredns/coredns#security-audits

CoreDNS Community

- 300+ Contributors (Big Thanks!)
- 26 Maintainers
- 33 Public Adopters
- 9200+ Stars
- LFX Program (Linux Foundation) & Google Summer of Code
  - 2017, 2018, 2019, 2020, 2021
  - 2022: ACME support for certificate management in tls plugin

Deep Dive

Extension Points for Developers

Three ways to customize CoreDNS

- Rebuilding with external plugins
- Using CoreDNS as a library
- Building your own plugin

CoreDNS

Rebuilding with External Plugins

You do not need to know Go to do this!

- “External”
  - Not built into the standard binaries and Docker images
  - Not supported by core team
- No dynamic loading of plugins
  - Plugins are built-in at compile time
  - Controlled by plugin.cfg
- Plugin **ordering** is fixed at compile time
- The ones we know about: https://coredns.io/explugins

External Plugins

Prerequisites: Docker and a shell

1. Clone CoreDNS
2. Modify plugin.cfg
3. Build CoreDNS

External Plugins

1. Clone CoreDNS

```
$ docker run --rm -u $(id -u):$(id -g) -v $PWD:/go golang:1.18 \
    /bin/bash -c \
   "git clone https://github.com/coredns/coredns.git && \
    cd coredns && \
    git checkout v1.9.2"
```

External Plugins

2. Modify plugin.cfg

```
$ cd coredns
$ vi plugin.cfg
```

```
...
dnstap:dnstap
acl:acl
firewall:github.com/coredns/policy/plugin/firewall
...
whoami:whoami
on:github.com/mholt/caddy/onevent
```

External Plugins

3. Build CoreDNS

```
$ docker run --rm -v $PWD:/coredns -w /coredns golang:1.18 make
```

CoreDNS as a Library

- Replace the CoreDNS `main.go`
- Allows you to:
  - Reduced the size and memory footprint of the binary
  - Limit the functionality and CLI flags
  - Do extra setup or initialization
- Used, for example, by Node Local DNS in K8s

Example: dnscached

- Source is in https://github.com/coredns/learning-coredns
- Simple caching DNS server
- Embeds only *bind*, *cache*, *errors*, *forward* and *log* plugins
- CLI args to generate a Corefile internally

Writing a Plugin

- Three categories of plugins
  - Best practice: stick to one of these in your plugin
- Backends
  - Source of data
  - *file*, *forward*, *hosts*, *clouddns*, *template*, *kubernetes*
- Mutators
  - Modify the inbound request, the outbound response, or both
  - *acl*, *cache*, *rewrite, nsid*
- Configurators
  - Modify the internal state or functioning of CoreDNS
  - *bind, log, health, ready*

Four functions

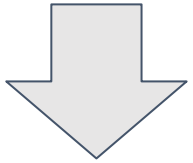
- Name - literally, just returns the name of the plugin
- ServeDNS - request handling
- init - register your plugin with Caddy
- setup - parse your config

Example: There can be only one!

- onlyone plugin from *Learning CoreDNS*
- Filters out all but one of specific record types

```
onlyone [ZONES...] {
    types TYPES
}
```

```
example.com.    18298   IN A  93.184.216.34
example.com.    18298   IN A  93.184.216.35
example.com.    18298   IN A  93.184.216.36
```



```
example.com.    18298   IN A  93.184.216.35
```

Functions: Name and init

onlyone.go

```
func (o *onlyone) Name() string { return "onlyone" }
```

setup.go

```
func init() {
   caddy.RegisterPlugin("onlyone", caddy.Plugin{
      ServerType: "dns",
      Action:     setup,
   })
}
```

Function: setup

setup.go

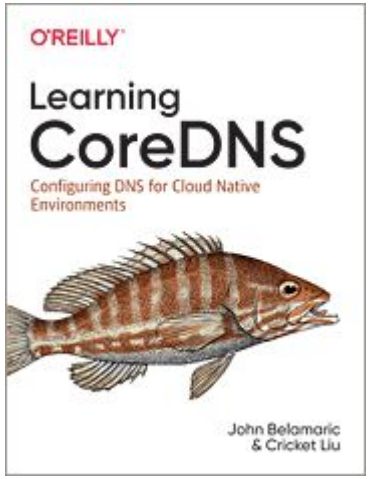
```
func setup(c *caddy.Controller) error {
    t, err := parse(c)
    if err != nil {
        return plugin.Error("onlyone", err)
    }

    dnsserver.GetConfig(c).AddPlugin(func(next plugin.Handler) plugin.Handler {
        t.Next = next
        return t
    })

    return nil
}
```

Function: ServeDNS

- Let’s look at it in GitHub
- It will be more readable there

Resources

- Plugin how-to: https://coredns.io/manual/toc/#writing-plugins
- GitHub: https://github.com/coredns/coredns
- Learning CoreDNS, John Belamaric & Cricket Liu, O'Reilly Media
  - https://github.com/coredns/learning-coredns
- Slack: **#coredns** on https://slack.cncf.io



Q & A

CoreDNS