CoreDNSであそんでみた

2021年6月24日

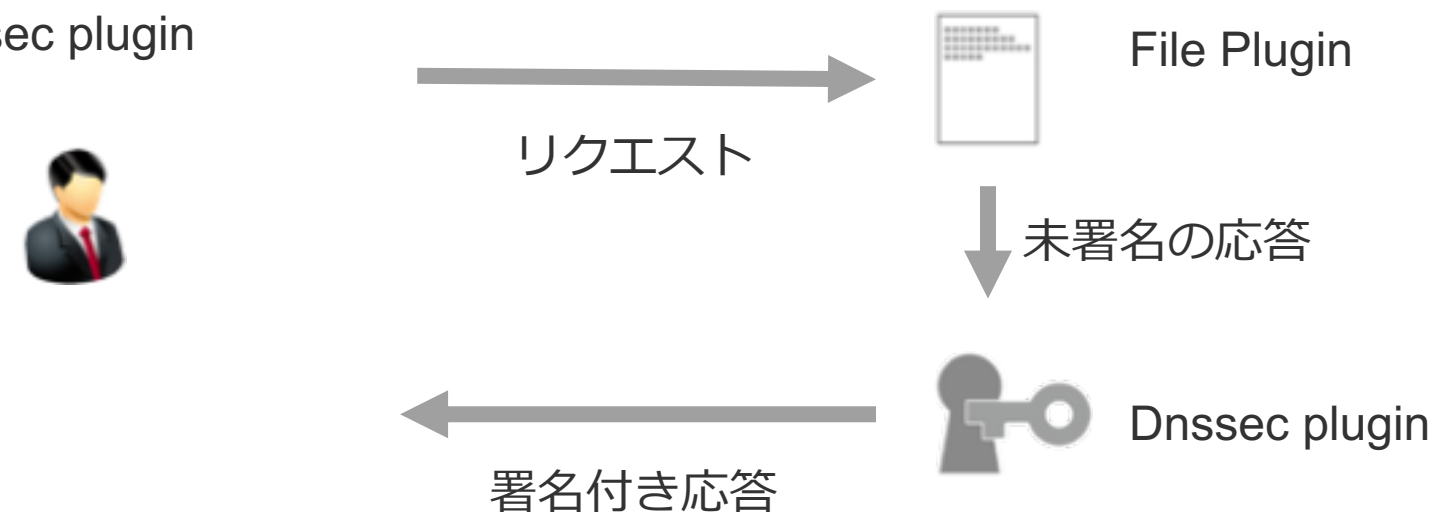
株式会社インターネットイニシアティブ
ネットワーク本部アプリケーションサービス部DNS技術課
其田 学

CoreDNSの紹介

CoreDNSは拡張性に優れた、DNSサーバソフトウェア

- CoreDNS自体は応答を受けつけ、Pluginに渡す部分のみ提供
- 応答データはPluginが作成
- PluginはChain機能があり、いろいろなPluginを組み合わせて応答すること可能
- CoreDNSはGoで書かれていて、自分でPluginを作ることも可能

例 file plugin + dnssec plugin



主なPlugin

file plugin

RFC1035形式のゾーンファイルを読み込み、その内容を応答する
RFC1035形式に加えて、$GENERATE, $INCLUDEに対応している

Corefile

```
example.jp. {
  file example.jp.zone
}
```

example.jp.zone

```
$ORIGIN example.jp
$TTL 3600
@ IN SOA ns1 root 1 3600 600 604800 900
@ IN NS localhost
$GENERATE 1-255 sv$ IN A 192.168.0.$
$INCLUDE hoeghoge.zone
```

file plugin

DNSSEC署名されたゾーンもサポート
ただし、サポートされている不在証明はNSECのみ、NSEC3は未対応

```
$ dig @localhost example.jp soa +dnssec
;; ANSWER SECTION:
example.jp.          3600    IN      SOA     g000.d-53.net. dns-managers.iij.ad.jp. 1 3600 600 86400 30
example.jp.          3600    IN      RRSIG   SOA 13 2 3600 20210723010411 (略
```

```
$ dig @localhost empty.example.jp soa +dnssec
;; AUTHORITY SECTION:
example.jp.          3600    IN      SOA     g000.d-53.net. dns-managers.iij.ad.jp. 1 3600 600 86400 30
example.jp.          3600    IN      RRSIG   SOA 13 2 3600 20210723010411 20210623010411 (略
example.jp.          30      IN      NSEC    example.jp. NS SOA RRSIG NSEC DNSKEY
example.jp.          30      IN      RRSIG   NSEC 13 2 30 20210723010411 20210623010411 (略
```

template plugin

動的にレスポンスデータを作成する

Corefile

```
example.jp {
  template IN AAAA {
    answer "{{ .Name }} 30 IN AAAA 2001:db8::53"
  }
}
```

```
$ dig @localhost a.example.jp. aaaa +short
2001:240::54
$ dig @localhost b.example.jp. aaaa +short
2001:240::54
```

- 例はAAAAを問い合わせると、必ず2001:db8::53を返す
- 逆引きとかの自動生成で使えるかも

dnssec plugin

レスポンスデータにRRSIG,NSECを付与する

Corefile

```
example.jp {
  template IN TXT {
     answer "{{ .Name }} 30 IN AAAA 2001:db8::53"
  }
  dnssec {
    key file Kexample.jp.zone.+013+33664
  }
}
```

- オンラインサイニングなので、templateみたいな動的な応答も扱える
- 鍵はZSK,KSKかCSK
- 不在応答はNSEC(black lie)

prometheus plugin

応答には使われないが、統計を作って、prometheus形式で出力する

Corefile

```
example.jp {
  prometheus :9153
}
```

```
$ curl http://localhost:9153/metrics
coredns_dns_requests_total{family="1",proto="udp",server="dns://:53",type="A",zone="example.jp."} 1000
coredns_dns_responses_total{rcode="NOERROR",server="dns://:53",zone=”example.jp."} 1000
…省略
```

- Exporter無しで、直接prometheusのmetricsが取れるのは便利

どんな時に利用が向いているか

- 通常のソフトウェア実装ではできないことがやりたい時
- Chain pluginが使えるので、全て実装する必要がないのがメリット
- 例
  - データソースを独自のものにしたい
  - 動的に応答を変えたい（プログラミングしたい）

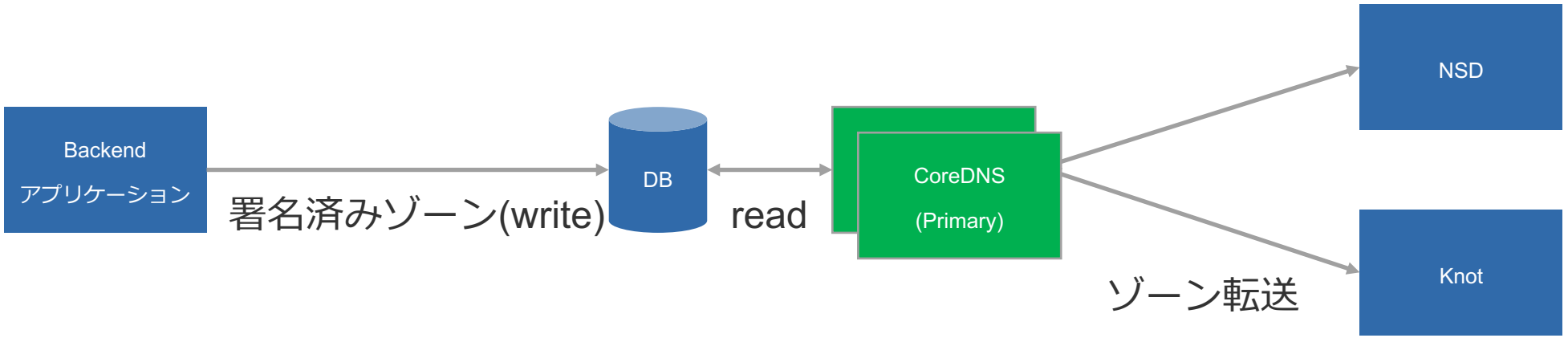
CoreDNS使ってみた

IIJ DNSプラットフォームサービスでのCoreDNSの利用例

IIJ DNSプラットフォームサービス(DPF)での利用例

Primary DNSとしての利用

現在の構成

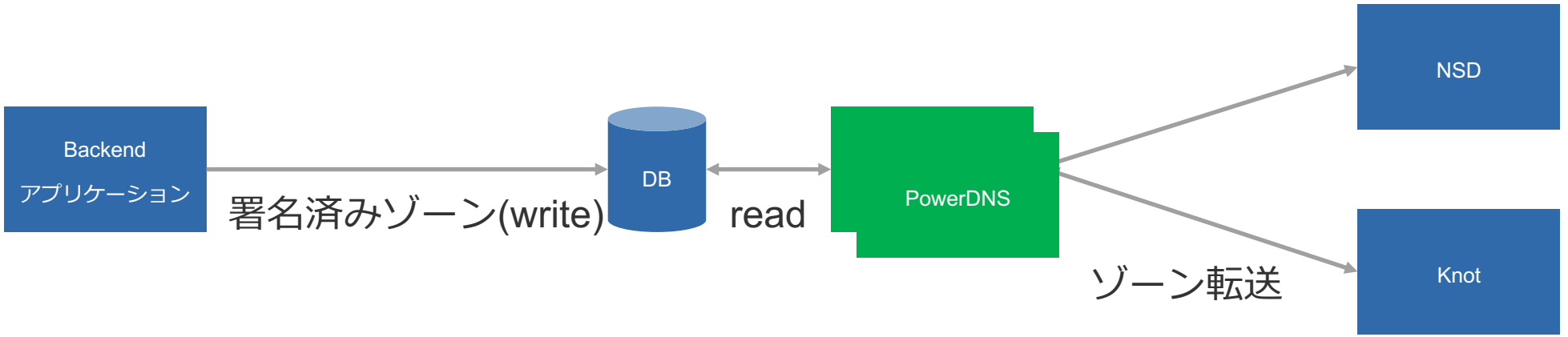


- ゾーン情報が通常のファイルベースだと、Primaryサーバ間の同期が課題
- DBはクラスタ構成が組めて、DBホスト間の同期は信頼できる
- PrimaryはDBから直接ゾーンデータを読み込むようにした
- 現在はPrimary DNSとしてCoreDNS+独自Pluginを採用中

IIJ DNSプラットフォームサービス(DPF)での利用例

Primary DNSとしての利用

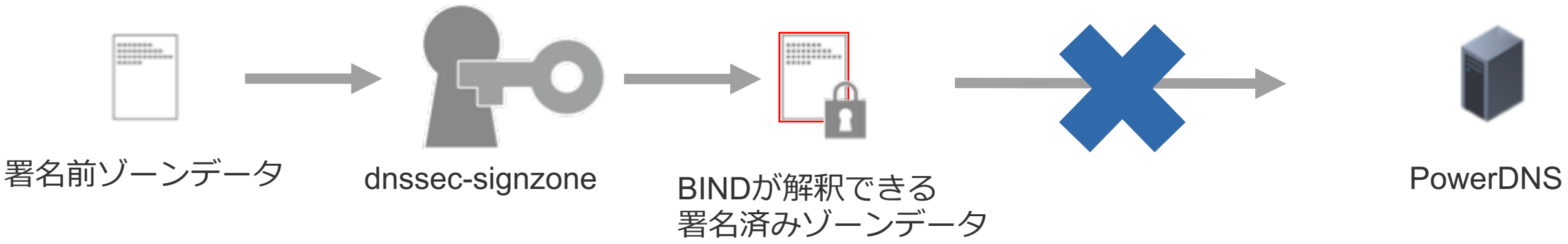
リリース当初(2019)の構成



当初は直接DBからゾーンデータを読み込み可能で、
唯一まともに動きそうなPowerDNSを採用した

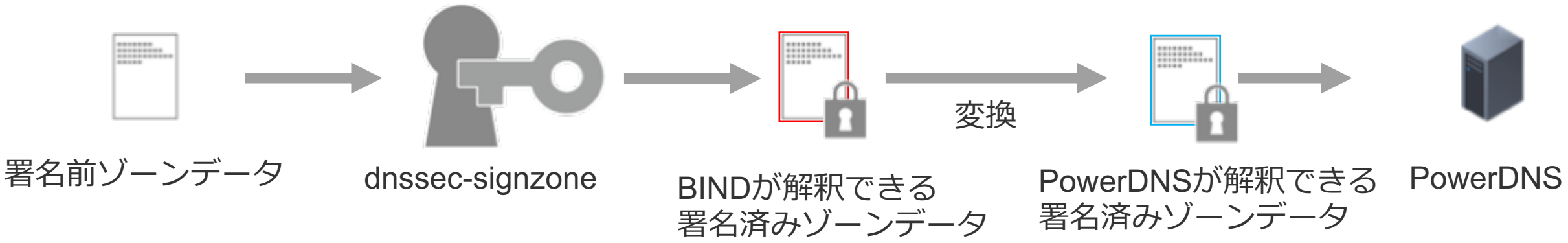
問題 1 Parserの差異

- 署名はdnssec-signzone(bind9)を利用していた。
- PowerDNSで解釈できないレコードはRFC3597形式に変換してからDB登録する必要があった



問題 1 Parserの差異

- 署名はdnssec-signzone(bind9)を利用していた。
- PowerDNSで解釈できないレコードはRFC3597形式に変換してからDB登録する必要があった



問題 2 署名済みゾーンデータの取り扱い

- PowerDNSは、PowerDNS自身が署名する機能は充実していた
- しかし署名済みデータを、インポートすると色々と不具合が生じた
  - 不在証明系のレコードはPowerDNSが作成、
  - 一部のレコードで、dnssec-sinzoneの署名時と違うものを出力
- dnssec-signzoneと同じ挙動にするパッチを書いて適用
- 開発元にIssueをあげるも、「そんな使い方想定してなかった」

代替のサーバを見つける必要が出てきた

そもそも、BIND9並に脆弱性も多いPowerDNSを選んだのは、
まともにDB使える実装がないから

作るしかない

DPFでのDNSサーバ実装

- 実は、DPFでは、この件以外でも、~~ヘンテコ~~独特な機能を持つDNSサーバをGo作っていた(miekg/dnsベース)
  - CNAME FlattenするDNSサーバ
  - k8s-apiと連携して、notifyのPodに転送するDNSサーバ
  - Notifyを受けてWebhookにするサーバ
  - etc…

- miekg/dnsとCoreDNSは作者が同じ=>新しいライブラリを覚えなくていい
- transfer pluginがあり、ゾーン転送に対応できそう

DBをデータソースとする独自Pluginを作成し、transfer plugin組み合わせて実装

問題 1 Parserの差異の解決

- 署名はdnssec-signzone(bind9)のまま、出力をbindの**raw形式**(binary)で出力
- raw形式でDBに保存し、pluginがraw形式を読み込み、応答に使用する
- raw形式なので、GoのDNS実装のParserの影響も受けない

問題 2 署名済みゾーンデータの取り扱い

- dnssec-sinezoneの結果をそのまま利用できるので、解決
- なお、ゾーン転送にしか利用しないので、+do bit付き応答は実装していない

運用してから

- 運用して数週間後に、メモリリークが見つかる
- prometheusのmetricsで、 goroutineの数が右肩上がりだった為、
  原因(goroutine Leek)に気づけた
  - これはtransfer pluginのバグで修正された
- また、起動中のプロセスのprofileを簡単に取れ、トラブルシュートしやすい
- この辺はCoreDNSというよりGo

まとめ

- CoreDNSはPluginをどう組み合わせて使うかが全て
- ただ、あまり利用されていないPluginを使うときは、十分に検証が必要
- metricsが取れたりprofileがとれたり、今時の運用で必要な機能は網羅
- 通常のDNSではできないようなことがしたいときには、選択肢の一つとして有り

日本のインターネットは1992年、IIJとともにはじまりました。以来、IIJグループはネットワーク社会の基盤をつくり、技術力でその発展を支えてきました。インターネットの未来を想い、新たなイノベーションに挑戦し続けていく。それは、つねに先駆者としてインターネットの可能性を切り拓いてきたIIJの、これからも変わることのない姿勢です。IIJの真ん中のIはイニシアティブ

IIJはいつもはじまりであり、未来です。