Knot Resolver6を調査した話

2025年06月27日

NTTコミュニケーションズ株式会社

小坂　良太

アジェンダ

- 調査背景

- 調査内容
  調査①　ソフトウェアの概要
  調査②　導入方法および具備する機能について
  調査③　運用を想定した調査

- 調査まとめ及び所感

■免責事項
本発表は第三者目線でKnot Resolverというソフトウェアを調査した結果の情報共有になります。
本資料中の内容について弊社(私含め)は一切の保証・責任を負いかねますのであらかじめご了承ください。

調査背景

フルサービスリゾルバのOSSとして様々な実装が存在

- BIND9/unbound/PowerDNS Recursor/Knot Resolverなど

これらのうちKnot Resolverは脆弱性が少なく高性能と言われているが国内での運用ノウハウが少ない印象

- Knot Resolver公式サイトによる導入実績
  - CETIN(チェコの電気通信会社)
  - Whalebone(チェコのサイバーセキュリティ会社)
  - DNS4EU(EUのISP向けにフルサービスリゾルバを提供するプロジェクト)
  - Cloudflare(現在はBigPineappleを利用 ※1)

DNSシステムを設計・開発する際に選択肢の1つとして扱えるかを把握するため今回調査を実施

(※1) Cloudflareの技術ブログより
https://blog.cloudflare.com/ja-jp/big-pineapple-intro/

調査①　ソフトウェアの概要

Knot Resolverについて

開発元はCZ.NIC

- チェコのinterest association of legal entities(直訳:法人の利益団体)でczドメインを運用
- CZ.NICでは権威DNSソフトウェアとしてKnot DNSも開発
- Knot ResolverはKnot DNSのライブラリ(knot-libs)を利用

以下の特徴を持つリゾルバソフトウェア

- オープンソース(GPLライセンス)
- 開発プロセスに透明性がありコミュニティのニーズベースで開発
- スケーラブル、耐障害性あり、フレキシブル
- コアアーキテクチャは小さく、効率的
- モジュールの追加によってリッチな機能を提供
  - モジュールの追加はオプショナル。使わなければ不具合や脆弱性のリスクもなし
- スレッド化なし、シェアードナッシングアーキテクチャ（共有可能なキャッシュを除く）

マルチプロセスで動作(かつプロセス間の連携なし。キャッシュは共有)

バージョンは5.xがstable、6.xがearly-access

- 以下マニュアルより抜粋
- 「6.0.x are “early access” versions, not generally recommended for production use.」

5.xと6.xの違いについて(1/4)

つなごう。驚きを。幸せを。

docomo business

Knot Resolver 5.x の起動方法とプロセスの見え方

```
# systemctl start kresd@1.service
# systemctl start kresd@2.service
# systemctl start kresd@doh.service

# ps -eo user:13,pid,args | grep kres[d]
knot-resolver   95284 /usr/sbin/kresd -c /usr/lib64/knot-resolver/distro-preconfig.lua -c /etc/knot-resolver/kresd.conf -n
knot-resolver   95294 /usr/sbin/kresd -c /usr/lib64/knot-resolver/distro-preconfig.lua -c /etc/knot-resolver/kresd.conf -n
knot-resolver   95687 /usr/sbin/kresd -c /usr/lib64/knot-resolver/distro-preconfig.lua -c /etc/knot-resolver/kresd.conf -n
```

- 起動したいプロセス数に合わせて複数起動
  - 識別のために@以降をユニークにする(何でも良い)
- 同じconfigで複数のプロセスが生成
  - 複数のプロセスがUDP 53番ポートをlisten (TCPは1プロセス)

Knot Resolver 6.x の起動方法とプロセスの見え方

```
# systemctl start knot-resolver.service

# ps -eo user:13,pid,args | grep -e kre[s] -e python | grep -v grep
knot-resolver  241732 /usr/bin/python3 -s /usr/bin/supervisord -c /run/knot-resolver/supervisord.conf
knot-resolver  241736 /usr/bin/python3 /usr/bin/knot-resolver --config=/etc/knot-resolver/config.yaml
knot-resolver  241741 /usr/sbin/kresd -c kresd0.conf -n
knot-resolver  241743 /usr/sbin/kresd -c kresd1.conf -n
knot-resolver  241745 /usr/sbin/kresd -c kresd2.conf -n
knot-resolver  241773 /usr/sbin/kres-cache-gc -c /var/cache/knot-resolver -d 1000 -u 80 -f 10 -l 100 -L 200 -t 0 -m 0 -w 0
```

- knot-resolver.serviceを起動するとconfigに記載したプロセス数に合わせてプロセスを起動
- config.yamlからkresd.confが自動生成

5.xと6.xの違いについて(2/4)

つなごう。驚きを。幸せを。
docomo business

Knot Resolver 5.x の起動方法とプロセスの見え方

```
# systemctl start kresd@1.service
# systemctl start kresd@2.service
# systemctl start kresd@doh.service

# ps -eo user:13,pid,args | grep kres[d]
knot-resolver   95284 /usr/sbin/kresd -c /usr/lib64/knot-resolver/distro-preconfig.lua -c /etc/knot-resolver/kresd.conf -n
knot-resolver   95294 /usr/sbin/kresd -c /usr/lib64/knot-resolver/distro-preconfig.lua -c /etc/knot-resolver/kresd.conf -n
knot-resolver   95687 /usr/sbin/kresd -c /usr/lib64/knot-resolver/distro-preconfig.lua -c /etc/knot-resolver/kresd.conf -n
```

Knot Resolver 6.x の起動方法とプロセスの見え方

```
# systemctl start knot-resolver.service

# ps -eo user:13,pid,args | grep -e kre[s] -e python | grep -v grep
knot-resolver  241732 /usr/bin/python3 -s /usr/bin/supervisord --configuration /run/knot-resolver/supervisord.conf
knot-resolver  241736 /usr/bin/python3 /usr/bin/knot-resolver --config=/etc/knot-resolver/config.yaml
knot-resolver  241741 /usr/sbin/kresd -c kresd0.conf -n
knot-resolver  241743 /usr/sbin/kresd -c kresd1.conf -n
knot-resolver  241745 /usr/sbin/kresd -c kresd2.conf -n
knot-resolver  241773 /usr/sbin/kres-cache-gc -c /var/cache/knot-resolver -d 1000 -u 80 -f 10 -l 100 -L 200 -t 0 -m 0 -w 0
```

・管理プロセスのようなものが３つ存在
  - supervisordはプロセス管理を実施
  - knot-resolverは起動スクリプトのようなもの
  - kres-cache-gcはおそらくガベージコレクタ

5.xと6.xの違いについて(3/4)

つなごう。驚きを。幸せを。

Knot Resolver 5.x の起動方法とプロセスの見え方

```
# systemctl start kresd@1.service
# systemctl start kresd@2.service
# systemctl start kresd@doh.service

# ps -eo user:13,pid,args | grep kres[d]
knot-resolver   95284 /usr/sbin/kresd -c /usr/lib64/knot-resolver/distro-preconfig.lua -c /etc/knot-resolver/kresd.conf -n
knot-resolver   95294 /usr/sbin/kresd -c /usr/lib64/knot-resolver/distro-preconfig.lua -c /etc/knot-resolver/kresd.conf -n
knot-resolver   95687 /usr/sbin/kresd -c /usr/lib64/knot-resolver/distro-preconfig.lua -c /etc/knot-resolver/kresd.conf -n
```

・knot-resolverの本体はいずれもkresdデーモン
  ・過去のリリースノートを見る感じだとおそらくほぼ同一
  ・不具合があれば同時期にパッチが出る印象
  ・安定性に違いはなさそう？

Knot Resolver 6.x の起動方法とプロセスの見え方

```
# systemctl start knot-resolver.service

# ps -eo user:13,pid,args | grep -e kre[s] -e python | grep -v grep
knot-resolver  241732 /usr/bin/python3 -s /usr/bin/supervisord --configuration /run/knot-resolver/supervisord.conf
knot-resolver  241736 /usr/bin/python3 /usr/sbin/knot-resolver --config=/etc/knot-resolver/config.yaml
knot-resolver  241741 /usr/sbin/kresd -c kresd0.conf -n
knot-resolver  241743 /usr/sbin/kresd -c kresd1.conf -n
knot-resolver  241745 /usr/sbin/kresd -c kresd2.conf -n
knot-resolver  241773 /usr/sbin/kres-cache-gc -c /var/cache/knot-resolver -d 1000 -u 80 -f 10 -l 100 -L 200 -t 0 -m 0 -w 0
```

5.xと6.xの違いについて(4/4)

つなごう。驚きを。幸せを。

docomo business

Knot Resolver 5.x の初期config

```
# cat /etc/knot-resolver/kresd.conf

-- Network interface configuration
net.listen('127.0.0.1', 53, { kind = 'dns' })
net.listen('127.0.0.1', 853, { kind = 'tls' })
--net.listen('127.0.0.1', 443, { kind = 'doh2' })
net.listen('::1', 53, { kind = 'dns', freebind = true })
net.listen('::1', 853, { kind = 'tls', freebind = true })
--net.listen('::1', 443, { kind = 'doh2' })

-- Load useful modules
modules = {
        'hints > iterate',  -- Allow loading /etc/hosts or custom root hints
        'stats',            -- Track internal statistics
        'predict',          -- Prefetch expiring/frequent records
}

-- Cache size
cache.size = 100 * MB
```

Knot Resolver 6.x の初期config

```
# cat  /etc/knot-resolver/config.yaml

rundir: /run/knot-resolver

workers: 2

cache:
  storage: /var/cache/knot-resolver

logging:
  level: info

network:
  listen:
    - interface: 127.0.0.1@53

management:
  unix-socket: /run/knot-resolver/kres-api.sock
```

- 5.xはluaフォーマット、6.xはyamlフォーマット
  - 6.xのconfigは起動時にluaフォーマットに自動で変換される(/var/run/knot-resolver/kresd0.conf)
  - luaとyamlは1対1対応していないためyamlフォーマットで記述できない設定あり(特にrpz周り)
  - 6.xでもluaフォーマットとして記述できるため必要に応じてluaで記述する(ただし非推奨。詳細はマニュアル参照)
- 長く使うことを想定し、今回は6.xを調査(6.0.14を調査。6.0.15以降でオプション名が色々と変更されるようです)

調査②　導入方法および具備する機能について

導入方法(インストール方法)

つなごう。驚きを。幸せを。
docomo business

RHEL系におけるインストール方法 公式ドキュメントより抜粋

```
( # dnf install 'dnf-command(copr)' )
( # dnf install epel-release )
# dnf copr enable @cznic/knot-resolver
# dnf install knot-resolver6

■以下は任意
# dnf install knot-resolver-module-dnstap knot-resolver-module-http
# dnf socat
  (socatはknot-resolverの管理APIに直接アクセスする時に便利なのでインストールしておく)

■インストール結果
# rpm -qa | grep -e knot -e socat
knot-libs-3.3.10-1.el9.x86_64
knot-resolver-6.0.14-cznic.1.el9.x86_64
knot-resolver-module-dnstap-6.0.14-cznic.1.el9.x86_64
knot-resolver-module-http-6.0.14-cznic.1.el9.x86_64
socat-1.7.4.1-5.el9.x86_64
```

(※)公式ドキュメントのリンク
https://www.knot-resolver.cz/documentation/v6.0.12/gettingstarted-install.html

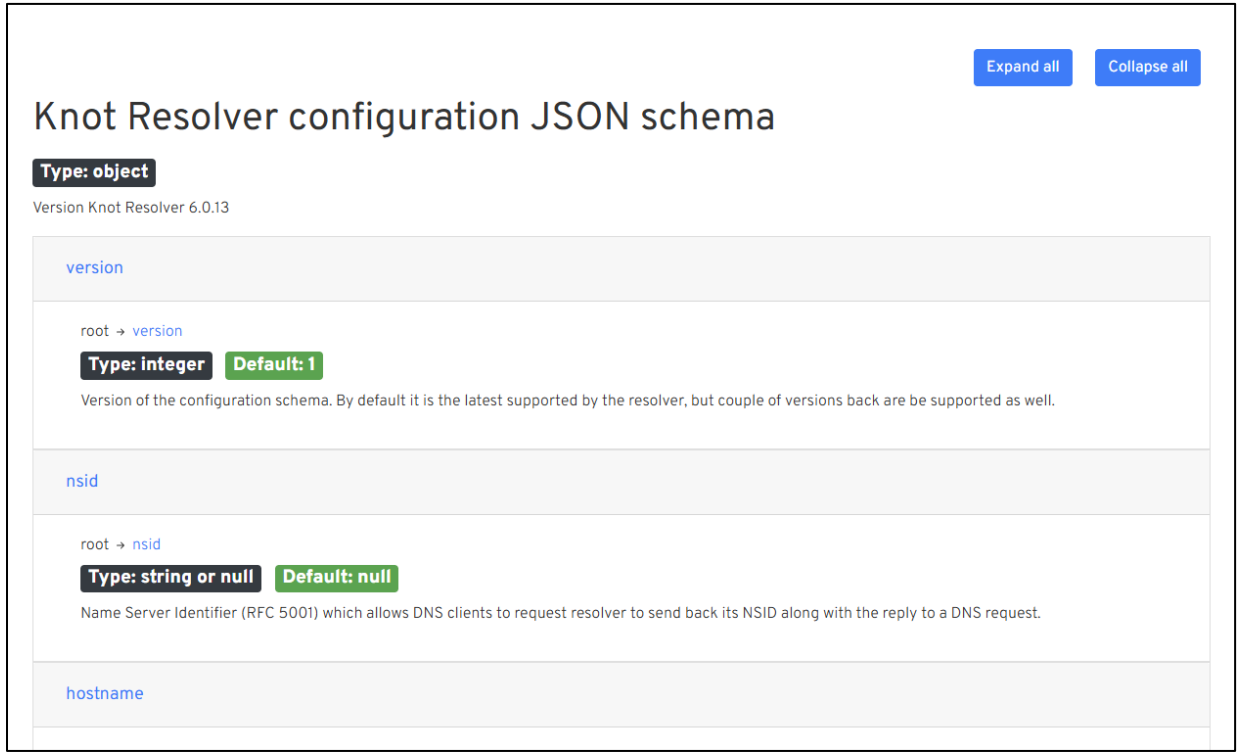
Knot Resolverの機能について　～概要～

つなごう。驚きを。幸せを。
docomo business

設定可能なオプションは約180件

- その中から主要な機能を抜粋して紹介

参考



・6.xは設定可能なオプションが一覧で掲載されている
・今回はこちらとマニュアルの両方を参考に調査
　・マニュアルにはRFCの対応表もあり

Knot Resolverの機能について　～基本設定～

つなごう。驚きを。幸せを。

docomo business

基本設定

```
network:
 listen:
    - interface: eth0@53
    - interface: lo@53

#  edns-buffer-size:
#    upstream: 1232B
#    downstream: 1232B

# out-interface-v4:
#    - interface: x.x.x.x@53
# do-ipv6: true
#

views:
   - subnets: [ 0.0.0.0/0, "::/0" ]
     answer: refused
   - subnets: [ 192.0.2.0/24 ]
     answer: allow
```

(※)明記が必須でない箇所はコメントアウト「#」で表記。

- プロトコルはudp53(tcp53)、doh、dot、xdpに対応
- IF名やIPでの指定が可能

- EDNS0のサイズはデフォルトで1232バイト
  (対クライアント、対権威DNSともに)
- 通常はデフォルト値のままで良いので設定不要

- 反復問い合わせの際に使うIPは設定可能
  - 送信元ポートの指定や制限は不可
  - 設定がなければOSのルーティングに従う
- IPv4を優先にするオプションはなし(IPv6の無効化は可能)

- ACL設定
- tag付けによるポリシー設定が可能だがキャッシュの分離は不可
  - BIND9のviewのような使い方はできない

Knot Resolverの機能について　～キャッシュ設定～

つなごう。驚きを。幸せを。
docomo business

キャッシュ設定

```
cache:
  storage: /var/cache/knot-resolver
  size-max: 100M
  ttl-min: 5s
  ttl-max: 1d
  ns-timeout: 1000ms
#   prefill:
#     - origin: "."
#       url: https://www.internic.net/domain/root.zone
#       refresh-interval: 12h
#       ca-file: /etc/pki/tls/certs/ca-bundle.crt
```

(※)上記はデフォルト値を記載(prefillはデフォルト設定なし)。
　　必要に応じてチューニングを実施

- 設定がない場合はデフォルトで動作
  - デフォルト値は左記参照
- 通常キャッシュはディスク上に保存
  - メモリをRAMディスク化して使うと性能向上
  - デフォルトではサイズが小さいのでチューニングが必要
- ttlやtimeoutの設定がデフォルト値で設定されているため
  こちらもチューニングが必要
- local root zone(RFC8806)はprefillモジュールで実装
  - デフォルト無効

RAMディスクの場合は
reboot時にデータは消失

Knot Resolverの機能について　～オプション機能～

つなごう。驚きを。幸せを。 docomo business

オプション機能

機能内容	パラメータ名	デフォルト値	備考
QNAME minimization	minimize	true	fallback機構あり
QNAME Case randomization	query_case_randomization	true	violators-workaroundsを有効にすると名前解決不可の時にfallbackする
RDビットなしクエリへの応答	refuse-no-rd	true	デフォルトでrefuse応答
DNSラウンドロビン	reorder_rrset	true	
プリフェッチ	prefetch	false	
DNS rebinding対策	rebinding-protection	false	Aレコード先がプライベートIPの場合にrefused応答を返す(デフォルトoff)
Minimal-response	なし	なし	機能は有効。チューニング不可
Aggressive Use of DNSSEC-Validated Cache(RFC 8198)	なし	なし	機能は有効。チューニング不可。
Type=ANYへの応答	なし	なし	NOTIMPを応答。チューニング不可。
DNS Cookie関連	なし	なし	RFC7873は非対応
EDNS Client Subnet(ESC)	なし	なし	RFC7871は非対応
Servfailのキャッシュ	なし	なし	servfail-ttl相当の機能なし。

Knot Resolverの機能について　～セキュリティ機能～

セキュリティ機能

```
#rate-limiting
#    rate-limit              xxx
#    instant-limit           50
#    slip                    2
#    capacity                524288
#    log-period              0s
#    dry-run                 false

#dnssec:
#  trust-anchors-files:
#    - file: /var/lib/knot-resolver/root.keys
#      read-only: false
#   trust-anchor-sentinel: true
#   trust-anchor-signal-query:true
```

(※)明記が必須ではないためコメントアウト「#」で表記。

・rate-limit機能あり(デフォルト無効)
 - 送信元IPベースのrate-limit
・dry-run機能もあるのでログonlyモードでのリリースが可能

・DNSSEC(バリデーション)機能あり (デフォルト有効)
・RFC5011、8145、8509に対応(デフォルト有効)
・基本的に設定変更不要(明記も不要)

その他のセキュリティ機能

・TCPアイドルタイムアウトは1秒(luaで変更可)
・TCPの最大接続数は設定不可(tcp-clients相当の機能なし)
・反復問い合わせの上限設定不可(recursive-clients相当の機能なし)

Knot Resolverの機能について　～ rpz機能～

つなごう。驚きを。幸せを。
docomo business

rpz機能

```
local-data:
   rpz:
    - file: /etc/knot-resolver/blocklist.rpz
       log: [name, ip]
       watchdog: auto

#lua:
#   script: |
#      policy.add(policy.rpz(policy.DENY_MSG('domain blocked.'), '/etc/knot-resolver/blocklist.rpz', true))
```

・rpzは通常local-dataで設定を行う(左記参照)
 - ただし一部の機能が動作しない

・luaで記述すると概ね正常に動作する
 - ただしluaの利用は非推奨

・local-dataに記述する方法では以下のような動作
 - ゾーンファイルにAレコードしかない場合、その他のレコード(AAAA)は乗っ取りしない
 - ワイルドカード(*)は動作しない
 - watchdogが動いていない(trueにすると起動不可)
 - ブロックしたクエリだけをログに出力することが可能

・lua方式で実装した場合は以下の動作
 - ゾーンファイルにAレコードしかない場合、その他のレコード(AAAA)はNODATA応答
 -ワイルドカード(*)は動作する
 - watchdogが設定できゾーンを書き換えると自動反映(watchdogの無効も可能)
 - ログ出力不可(DNSTAPで‘domain blocked.’の文字列でフィルタするといった対応は可能)

(※)luaでの記述はメーカー非推奨であることから
コメントアウト「#」で表記。

調査③　運用を想定した調査

統計情報の取得について

つなごう。驚きを。幸せを。
docomo business

以下の３つの方法で取得が可能

① management APIを用いる(新旧あり)
② graphite protocolで統計を送信
③ kresdデーモンのcontrol APIを用いる

(例)
```
# kresctl metrics
# curl http://127.0.0.1:8453/metrics/prometheus
```

(例)
```
# socat - UNIX-CONNECT:/run/knot-resolver/control/0
> cache.stats()
```

①②で取得可能なもの

- クエリ数(udp/tcp/doh/dot)
- レスポンス数(noerror/nxdomain/servfail、フラグ毎のカウント)

③で取得可能なもの

- キャッシュのread数、hit回数(＝キャッシュヒット率が取得可能)
- キャッシュのUsage
- 等々

統計は各プロセス毎の値が出力されるため合算処理が必要

そういえばシステムのログもプロセスの数だけ出力されます

クエリのログ出力について

つなごう。驚きを。幸せを。
docomo business

クエリのログ出力にはDNSTAPを用いる(設定例は以下)

```
logging:
  dnstap:
    unix-socket: /run/knot-resolver/dnstap.sock
    log-queries: true
    log-responses: true
```

} クライアント ～ Knot Resolver間のクエリとレスポンス
(Knot Resolver ～ 権威DNS間のクエリログは出力不可)

クエリログのサンプル(DNS-collectorを用いた簡易ファイル出力)

```
# tail -f /var/log/knot-resolver/query.log
2025-06-03T06:02:14.182627Z - CQ - 127.0.0.1 47918 IPv4 UDP 54b example.jp A 0.000000000
2025-06-03T06:02:14.192254Z - CR 0 127.0.0.1 47918 IPv4 UDP 58b example.jp A 0.000000000
```

- JSON形式で出力すればより詳細な情報が得られる
(DNSTAPクライアントの設定なので今回は説明省略)
- EDE(Extended DNS Errors)に対応しているため、
Servfailのログを出力させるとデバッグに役立つ

サポートについて

つなごう。驚きを。幸せを。

docomo business

プロフェッショナルサポート(有償)

- CZ.NICと契約

フリーサポート(無償)

- メーリングリスト
- CZ.NICのgitlubにissue報告
- Gitter

その他：セキュリティに関するバグ報告

- PGPを使って暗号化した上で開発元(開発者)に連絡
  - 連絡先等は公式サイト参照

脆弱性及び不具合情報

つなごう。驚きを。幸せを。

(本ページは当日の投影のみ)

【参考】過去に報告されたCVE番号とパッチリリース

つなごう。驚きを。幸せを。
docomo business

パッチリリース	CVE番号
1.5.2 (2018-01-22)	CVE-2018-1000002
2.3.0 (2018-04-23)	CVE-2018-1110
2.4.1 (2018-08-02)	CVE-2018-10920
4.1.0 (2019-07-10)	CVE-2019-10190、 CVE-2019-10191
4.3.0 (2019-12-04)	CVE-2019-19331
5.1.1 (2020-05-19)	CVE-2020-12667
5.3.2 (2021-05-05)	CVE-2021-40083
5.5.3 (2022-09-21)	CVE-2022-40188
5.7.0 (2023-08-22)	CVE-2023-46317
5.7.1 (2024-02-13)、 6.0.6 (2024-02-13)	CVE-2023-50868、 CVE-2023-50387(KeyTrap)

2018/1～2025/6の約7年で12件

- CVE番号が付与されていない脆弱性は今回除外

性能について

つなごう。驚きを。幸せを。

docomo business

今回は簡易的な性能試験のみ実施

試験環境

- CPU：　8コア16スレッド
- Mem:　64GB
- NIC:　10Gbps
- 負荷:　NAT配下で商用模擬トラヒックを印加

試験結果

- 20万qps印加時にロスが0.5%未満
- CPU使用率は60%程度

所感

- NAT配下でなければ30万qpsまでは処理できそう
- メニーコアCPUを採用すれば更に性能向上の見込みあり
  - XDPにも対応しているため、XDPを使ってみるのも面白そう

調査まとめ及び所感

調査まとめ及び所感

バージョン5.xと6.xにはそれぞれメリット・デメリットあり

- Configの書きやすさと長期利用を想定し今回は6.xを調査

基本機能や新しい機能はしっかり実装されている

- ただし6.xはrpz周りの実装が弱くluaでの設定が必要になるケースあり
- 統計情報は少し物足りない(必要に応じてDNSTAPを使う)

不具合および脆弱性についてはリリースノート参照

性能に関しては今回ほぼ未調査

- 20万qpsは処理可能であることを確認
- ポテンシャルを引き出すにはCPUコア数の多いサーバを採用すると良さそう

所感

- 6.xはearly-accessでマニュアルにも「本番環境には推奨しない」とあるためstableになるまでは様子を見ることをオススメ
- 機能・性能はしっかりしているためPoCとして使い始めるのはOK
  - 初期configにIP設定、ACL、キャッシュサイズの設定、＋αの設定をするだけ

つなごう。驚きを。幸せを。

ご清聴ありがとうございました