DNSアプリの機能比較 on Rocky8編

2022年6月23日

NTTコミュニケーションズ株式会社

小坂　良太

Agenda

- 自己紹介
- 発表の概要
- 調査を行った機能について

- フルサービスリゾルバの調査結果
  - bind9
  - unbound
  - knot-resolver

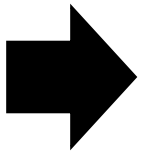
- 権威DNSの調査結果
  - bind9
  - NSD
  - knot DNS

- まとめ

自己紹介

・名前：小坂　良太
・所属：NTTコミュニケーションズ株式会社
・仕事：ISPとして提供しているフルサービスリゾルバ(キャッシュDNS)と
　　権威DNSの設計および開発　(回線サービス名：OCN)

・今回の発表の趣旨
- DNSTAPやDoHといった面白い機能をどんどん使っていきたい！
- bind9以外のソフトウェアの国内導入実績も多いのでbind9以外も使っていきたい！



各DNSアプリの対応状況を調査し比較してみました！

発表の概要

■調査を行ったDNSソフトウェア

フルサービスリゾルバ:	bind9	unbound	knot-resolver
権威DNS:	bind9	NSD	knotDNS
	ISC	NLnet Labs	CZ NIC

■調査を行った機能

共通：レポジトリからのインストール可否、dnstap、DoH、Prometheusとの連携、tcp-idle-timeout
フルサービスリゾルバ:　RFC8806対応、HTTPSレコードに対するA/AAAA付与
権威DNS:　HTTPSレコードの登録、CAAレコードの登録

■**免責事項**

今回ご紹介する手順や設定はあくまで動作させることのみを目的としたものになります。
実際にソフトウェアをご利用される場合はセキュリティを考慮した設定を行い、各自試験を行ってください。
不具合など発生した場合において弊社(私含め)は一切の責任を負いかねますのであらかじめご了承ください。

調査を行った機能について　共通機能(1/2)

■前提条件

Rocky8.6をminimalインストールし、#dnf –y update した状態で機能調査を実施(2023年5月頃)
updateの結果、**Rocky8.8・インストールされているrpm数は361の状態からスタート**
SELinux、filrewalldはoff、作業はrootで実施(一部を除く)

■共通機能1. レポジトリからのインストール可否

dnf install (パッケージ名)でインストールが行えるか確認
Rocky8公式レポジトリからインストールできない場合はEPELを用いてインストールが行えるか確認

■共通機能2. dnstap
■共通機能3. DoH

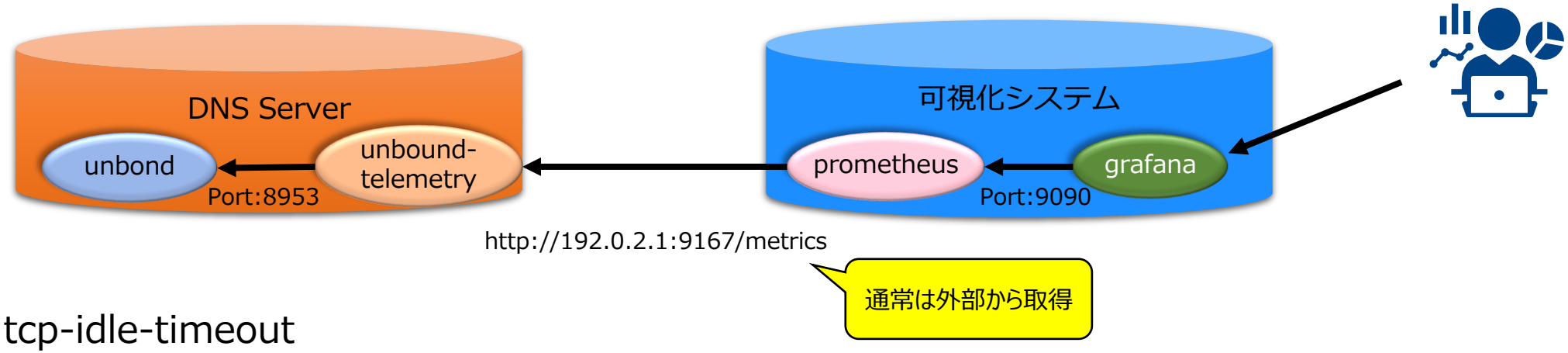
dnstapおよびDoHが利用できるか確認
DoHの確認にはbind9.18に付属するdigツールを用いて、dig +https @(DNSサーバ) example.comを実施

調査を行った機能について　共通機能(2/2)

■共通機能4. Prometheusとの連携

http://127.0.0.1:(ポート番号)/metricsにアクセスし、統計情報を取得可能か確認
アプリ単体で対応していない場合は追加でアプリをインストールし取得可能(提供可能)か確認

□ Prometheus/Grafanaを用いた可視化システムの例



■共通機能5. tcp-idle-timeout

TCP接続に対しidle-timeout機能を具備しているか確認
確認にはtelnet 127.0.0.1 53を用い、指定した秒数後に切断されることを確認

調査を行った機能について　フルサービスリゾルバの機能

■フルサービスリゾルバの機能1. RFC8806対応

Root DNSからRootゾーンを取得し、Root DNSに問い合わせせずに応答を返すか確認
(RFC8806に対して厳密に準拠しているかまでは今回確認しない)

■フルサービスリゾルバの機能2.
HTTPSレコードに対するA/AAAA付与

キャッシュにA/AAAAがある状態でHTTPSを名前解決し、
Additional SectionにA/AAAAが付与されるかを確認

```
$dig @192.0.2.1  www.google.co.jp https

; <<>> DiG 9.18.0 <<>> @ 192.0.2.1 www.google.co.jp https
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 27249
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 3

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;www.google.co.jp.              IN      HTTPS

;; ANSWER SECTION:
www.google.co.jp.       7114    IN      HTTPS   1 . alpn="h2,h3"

;; ADDITIONAL SECTION:
www.google.co.jp.       299     IN      A       142.251.222.35
www.google.co.jp.       292     IN      AAAA    2404:6800:4004:818::2003
```

調査を行った機能について　権威DNSの機能

■権威DNSの機能1. HTTPSレコードの登録
■権威DNSの機能2. CAAレコードの登録

HTTPSレコード及びCAAレコードが登録可能か確認
digを用いて名前解決可能か確認

```
# cat example.com
$TTL 3H
@       IN SOA  @ rname.invalid. (
                                1       ; serial
                                1D      ; refresh
                                1H      ; retry
                                1W      ; expire
                                3H )    ; minimum
        NS      @
        A       127.0.0.1
        AAAA    ::1

www     HTTPS   1 .
www     CAA     0 issue “example.com"
```

以下のようなレコードを登録し、
名前解決できるかどうかを確認

フルサービスリゾルバの調査結果

フルサービスリゾルバの調査結果　サマリ

		bind9	unbound	knot-resolver
共通機能1	レポジトリからのインストール可否	○9.16.23	○1.16.2	△5.6.0(EPEL)
共通機能2	dnstap	○	×要ビルド	○
共通機能3	DoH	×9.17から	×要ビルド	○
共通機能4	Prometheusとの連携	○	△(3rd Party製)	○
共通機能5	tcp-idle-timeout	○	○	未確認
フルサービスリゾルバの機能1	RFC8806対応	○	○	○
フルサービスリゾルバの機能2	HTTPSレコードに対するA/AAAA付与	×	×	×

フルサービスリゾルバの調査結果　bind9

		bind9
共通機能1	レポジトリからのインストール可否	○9.16.23

レポジトリからのインストール

```
# dnf install bind9.16

# rpm -qa | grep "^bind9.16" | sort
bind9.16-9.16.23-0.14.el8.x86_64
bind9.16-dnssec-utils-9.16.23-0.14.el8.x86_64
bind9.16-libs-9.16.23-0.14.el8.x86_64
bind9.16-license-9.16.23-0.14.el8.noarch
bind9.16-utils-9.16.23-0.14.el8.x86_64

# systemctl start named

# named -v
BIND 9.16.23-RH (Extended Support Version) <id:fde3b1f>
```

■ポイント
・#dnf install bind
　の場合は9.11系がインストールされる
・今回は新機能を試すため新しいバージョン(9.16)をインストールする

■補足
・レポジトリ版は機能がバックポートされることがあるため
　ISC版(OSS版)と機能が異なる
・バージョン番号はあくまで参考程度にすること

フルサービスリゾルバの調査結果　bind9

ソースコードからのインストール

```
# dnf install tar gcc openssl-devel libcap-devel perl
# dnf --enablerepo=devel install libuv-devel  libnghttp2-devel

# tar -Jxvf  bind-9.18.15.tar.xz
# cd bind-9.18.15

(configureオプションなし)
# ./configure
# make
# make install

(configureオプションあり)
# dnf install json-c-devel protobuf-c-devel fstrm-devel libidn2-devel
# ./configure --with-tuning=large --with-libidn2  --with-json-c  --enable-dnstap --enable-fixed-rrset ¥
     --enable-full-report
# make
# make install
```

■ポイント
・マニュアル参照
・ただし、デフォルトで有効な機能があるため
　マニュアルに記載のないものもインストールが必要
・Perlをインストールするとmakeなどもインストールされる

■補足
・マニュアルに記載のあるOptional Featuresのうち
　重要そうなものをピックアップ
・レポジトリ版のnamed –Vも参考にオプション指定

フルサービスリゾルバの調査結果　bind9

		bind9
共通機能2	dnstap	○

dnstap

```
(named.confの抜粋)
================================
options {
        dnstap { all; }; # Configure filter
        dnstap-output file "/var/named/data/dnstap.bin";
};
================================

# dnstap-read -y  /var/named/data/dnstap.bin
```

フルサービスリゾルバの調査結果　bind9

		bind9
共通機能3	DoH	×9.17から

DoH

(レポジトリ版は非対応のためビルド版で確認)

(named.confの抜粋)

```
==============================
tls local-tls {
   key-file "/var/named/server.key";
   cert-file "/var/named/server.crt";
};

options {
        listen-on port 443 tls local-tls http default {any;};
        listen-on port 53 { 127.0.0.1; };
}
==============================
# dig  +https @127.0.0.1  example.com a
```

フルサービスリゾルバの調査結果　bind9

		bind9
共通機能4	Prometheusとの連携	○

Prometheusとの連携

(named.confの抜粋)

```
================================
statistics-channels {
        inet 127.0.0.1    port 9000 allow { 127.0.0.1; };
};
================================
```

■ポイント
・Exporter(isc-stork-agent)が利用する
8080,9119,9547を避け、空いてるポートを設定

■補足
・ISC謹製のExpoterが利用可能(※1)
・namedが読み込んでいるnamed.confをパースし、
namedプロセスに接続を行う

```
# rpm -ivh isc-stork-agent-1.10.0.230404081327-1.x86_64.rpm
# systemctl start isc-stork-agent
# curl http://127.0.0.1:9119/metrics
```

(※1) 以下を参照
https://kb.isc.org/docs/exporting-statistics-to-prometheus
https://www.isc.org/download/#Stork

フルサービスリゾルバの調査結果　bind9

		bind9
共通機能5	tcp-idle-timeout	○
フルサービスリゾルバの機能1	RFC8806対応	○
フルサービスリゾルバの機能2	HTTPSレコードに対するA/AAAA付与	×

tcp-idle-timeout

```
(named.confの抜粋)

================================
options {
        tcp-initial-timeout 1;
        tcp-idle-timeout 1;
        #tcp-keepalive-timeout 1;
};

================================
```

RFC8806対応

```
(named.confの抜粋)

================================

zone "." { type mirror; };

================================
```

フルサービスリゾルバの調査結果　unbound

		unbound
共通機能1	レポジトリからのインストール可否	○1.16.2

レポジトリからのインストール

```
# dnf install unbound
# systemctl start unbound
#  /usr/sbin/unbound -V | head -1
Version 1.16.2
```

フルサービスリゾルバの調査結果　unbound

ソースコードからのインストール

```
# dnf install tar gcc make openssl-devel expat-devel

# tar zxvf unbound-1.17.1.tar.gz
# cd unbound-1.17.1

(configureオプションなし)
# ./configure
# make
# make install

(configureオプションあり)
# dnf install libevent-devel protobuf-c-devel fstrm
# dnf --enablerepo=devel install libnghttp2-devel
# ./configure --with-libevent --enable-dnstap  --enable-ipsecmod --enable-subnet ¥
  --with-pthreads --with-ssl --enable-sha2 --enable-ecdsa --enable-linux-ip-local-port-range  --with-libnghttp2
# make
# make install
```

■補足
・公式マニュアルにはUbuntuとmacOSの手順しかないため、
それを参考に依存関係を解消させる
・bisonとflexも必要らしいが、なくてもコンパイルは通る。
気になる人は # dnf install bison flex してください

■補足
・サンプルconfigをエラーなく動作させるために
いくつか指定
・dnstapとDoHを使う場合は赤文字の箇所が必要

フルサービスリゾルバの調査結果　unbound

		unbound
共通機能2	dnstap	×要ビルド

dnstap

(レポジトリ版は非対応のためビルド版で確認)

(unbound.confの抜粋)

```
==============================
dnstap:
        dnstap-enable: yes
        dnstap-socket-path: "/var/run/unbound/dnstap.sock"
        dnstap-send-identity: yes
        dnstap-send-version: yes
        dnstap-log-resolver-query-messages: yes
        dnstap-log-resolver-response-messages: yes
        dnstap-log-client-query-messages: yes
        dnstap-log-client-response-messages: yes
        dnstap-log-forwarder-query-messages: yes
        dnstap-log-forwarder-response-messages: yes
==============================
```

(参考) dnstapアプリの利用

```
# dnf install golang
# go install github.com/dnstap/golang-dnstap/dnstap@latest
# cp go/bin/dnstap /usr/local/bin/
# chmod 755 /usr/local/bin/dnstap
# sudo -u unbound /usr/local/bin/dnstap –u   ¥
                /var/run/unbound/dnstap.sock
```

フルサービスリゾルバの調査結果　unbound

		unbound
共通機能3	DoH	×要ビルド

DoH

(レポジトリ版は非対応のためビルド版で確認)

(unbound.confの抜粋)

```
==============================
server:
        interface: 192.0.2.1@53
        interface: 192.0.2.1@443
        tls-service-key: "/etc/unbound/unbound_server.key"
        tls-service-pem: "/etc/unbound/unbound_server.pem"
        access-control: 192.0.0.0/8 allow
==============================
```

フルサービスリゾルバの調査結果　unbound

		unbound
共通機能4	Prometheusとの連携	△(3rd Party製)

Prometheusとの連携

(unbound.confの抜粋)

```
==============================
server:
   statistics-interval: 0
   extended-statistics: yes
   statistics-cumulative: no
==============================
```

■ポイント
・デフォルトで8953ポートをlisten

(参考) unbound_exporterの利用

```
# dnf install golang
# go install github.com/letsencrypt/unbound_exporter@latest
# go/bin/unbound_exporter
# curl http://127.0.0.1:9167/metrics
```

■ポイント
・公式のExporterは未提供のため
　3rd patry製のアプリを利用する必要あり
・デフォルトでunboundの8953に接続を行う

フルサービスリゾルバの調査結果　unbound

		unbound
共通機能5	tcp-idle-timeout	○
フルサービスリゾルバの機能1	RFC8806対応	○
フルサービスリゾルバの機能2	HTTPSレコードに対するA/AAAA付与	×

tcp-idle-timeout

```
(unbound.confの抜粋)

==============================
server:
        tcp-idle-timeout: 1000
        edns-tcp-keepalive-timeout: 1000

==============================
```

RFC8806対応

```
(unbound.confの抜粋)
==============================
auth-zone:
        name: "."
        primary: 199.9.14.201        # b.root-servers.net
        primary: 192.33.4.12         # c.root-servers.net
        primary: 199.7.91.13         # d.root-servers.net
        primary: 192.5.5.241         # f.root-servers.net
        primary: 192.112.36.4        # g.root-servers.net
        primary: 193.0.14.129        # k.root-servers.net
        primary: 192.0.47.132        # xfr.cjr.dns.icann.org
        primary: 192.0.32.132        # xfr.lax.dns.icann.org
        primary: 2001:500:200::b     # b.root-servers.net
        primary: 2001:500:2::c       # c.root-servers.net
        primary: 2001:500:2d::d      # d.root-servers.net
        primary: 2001:500:2f::f      # f.root-servers.net
        primary: 2001:500:12::d0d    # g.root-servers.net
        primary: 2001:7fd::1         # k.root-servers.net
        primary: 2620:0:2830:202::132 # xfr.cjr.dns.icann.org
        primary: 2620:0:2d0:202::132  # xfr.lax.dns.icann.org
        fallback-enabled: yes
        for-downstream: no
        for-upstream: yes
==============================
```

■補足
・デフォルトで設定済

フルサービスリゾルバの調査結果　knot-resolver

		knot-resolver
共通機能1	レポジトリからのインストール可否	△5.6.0(EPEL)

レポジトリからのインストール

```
# dnf install epel-release
# dnf install knot-resolver
# dnf install knot-resolver-module-dnstap knot-resolver-module-http
# systemctl start kresd@1.service

#  /usr/sbin/kresd -V
Knot Resolver, version 5.6.0
```

■ポイント
・dnstapやprometheus連携を行うため
それらmoduleもインストール

フルサービスリゾルバの調査結果　knot-resolver

ソースコードからのインストール(1/2)

```
# dnf install gcc-c++ make gcc protobuf-c-devel fstrm-devel cmake git python38-pip
# dnf install autoconf automake libtool gnutls-devel luajit-devel
# dnf --enablerepo=powertools install userspace-rcu-devel lmdb-devel libedit-devel
# dnf --enablerepo=devel install libuv-devel

(libknotが必要なので事前にknotDNSをインストール)
# git clone https://gitlab.nic.cz/knot/knot-dns.git
# autoreconf -if
# ./configure
# make
# make install
# vi /etc/ld.so.conf
# cat /etc/ld.so.conf
include ld.so.conf.d/*.conf
/usr/local/lib                    ←末尾に/usr/local/libを追加
# ldconfig
```

フルサービスリゾルバの調査結果　knot-resolver

ソースコードからのインストール(2/2)

```
$ pip3  install meson ninja –user

$ git clone --recursive https://gitlab.nic.cz/knot/knot-resolver.git
$ cd knot-resolver/
$ export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig
$ meson setup build_dir --prefix=/tmp/kr --default-library=static
$ meson configure build_dir -Ddnstap=enabled
$ ninja -C build_dir
$ ninja install -C build_dir

$ su –
# cd /tmp/kr/
# sbin/kresd -c lib64/knot-resolver/distro-preconfig.lua -c etc/knot-resolver/kresd.conf -n
```

■補足
・マニュアル「14.2 Complilation」をそのまま実施
・ただし、デフォルトでdnstapが使えないため
　meson_options.txtを参考にオプションを指定

■補足
・EPEL版を参考にして引数を指定し起動

フルサービスリゾルバの調査結果　knot-resolver

		knot-resolver
共通機能2	dnstap	○
共通機能3	DoH	○
共通機能4	Prometheusとの連携	○

dnstap

```
(kresd.confの抜粋)

================================
modules = {
    dnstap = {
        socket_path = "/tmp/dnstap.sock",
        identity = "1",
        version = "My Custom Knot Resolver " .. package_version(),
        client = {
            log_queries = true,
            log_responses = true,
        },
    }
}
================================
```

DoH

```
(kresd.confの抜粋)

================================
net.listen('127.0.0.1', 443, { kind = 'doh2' })
================================
```

■補足
・デフォルトで設定済

Prometheusとの連携

```
(kresd.confの抜粋)

================================
net.listen('127.0.0.1', 9000, { kind = 'webmgmt' })
modules.load('http')
================================
# curl http://127.0.0.1:9000/metrics
```

■ポイント
・追加でexporterを入れる必要なし

フルサービスリゾルバの調査結果　knot-resolver

		knot-resolver
共通機能5	tcp-idle-timeout	未確認
フルサービスリゾルバの機能1	RFC8806対応	○
フルサービスリゾルバの機能2	HTTPSレコードに対するA/AAAA付与	×

tcp-idle-timeout

```
(kresd.confの抜粋)

================================
net.tcp_in_idle(1)
================================
```

■補足
・何らかのtimeoutは設定できるが詳細不明
・マニュアルにも記載なし

RFC8806対応

```
(kresd.confの抜粋)

================================
modules.load('prefill')
prefill.config({
    ['.'] = {
        url = 'https://www.internic.net/domain/root.zone',
        interval = 86400, -- seconds
        ca_file = '/etc/pki/tls/certs/ca-bundle.crt', -- optional
    }
})
modules = { 'serve_stale < cache' }
================================
```

フルサービスリゾルバの調査結果　所感

		bind9	unbound	knot-resolver
共通機能1	レポジトリからのインストール可否	○9.16.23	○1.16.2	△5.6.0(EPEL)
共通機能2	dnstap	○	×要ビルド	○
共通機能3	DoH	×9.17から	×要ビルド	○
共通機能4	Prometheusとの連携	○	△(3rd Party製)	○
共通機能5	tcp-idle-timeout	○	○	未確認
フルサービスリゾルバの機能1	RFC8806対応	○	○	○
フルサービスリゾルバの機能2	HTTPSレコードに対するA/AAAA付与	×	×	×

- コンパイルせずにdnstapとDoHを使う場合の選択肢はknot-resolverになるが、何か設定をしようとした時にhoge-option yes/no/auto; のように設定するものではなく、それ故にマニュアルも読みづらいと感じた
- 一方でbind9は9.18のマニュアル「8.2.13 options Block Grammar」を確認すると非常にオプションが多く、総ページ数も584ページと膨大なので今後チューニングしていくのは負担が大きい
- 脆弱性の少なさ、クセのなさからunboundが今の所はオススメ(ある程度運用ができればビルドに挑戦？)

権威DNSの調査結果

権威DNSの調査結果　サマリ

		bind9	NSD	knotDNS
共通機能1	レポジトリからのインストール可否	○9.16.23	△4.3.8(EPEL)	△3.2.6(EPEL)
共通機能2	dnstap	○	×要ビルド	×要ビルド
共通機能3	DoH	×9.17から	×(DoTのみ対応)	×
共通機能4	Prometheusとの連携	○	△(3rd Party製)	△(3rd Party製)
共通機能5	tcp-idle-timeout	○	○	○
権威DNSの機能1	HTTPSレコードの登録	○	○	○
権威DNSの機能2	CAAレコードの登録	○	○	○

権威DNSの調査結果　NSD

		NSD
共通機能1	レポジトリからのインストール可否	△4.3.8(EPEL)

レポジトリからのインストール

```
# dnf install epel-release
# dnf install nsd
# systemctl start nsd

# nsd –v 2>&1 | head -1
NSD version 4.3.8
```

権威DNSの調査結果　NSD

ソースコードからのインストール

```
# dnf install tar gcc make libevent-devel openssl-devel

# tar zxvf nsd-4.6.1.tar.gz
# cd nsd-4.6.1

(configureオプションなし)
# ./configure
# make
# make install

(configureオプションあり)
# dnf install  protobuf-c-devel fstrm-devel
# ./configure --enable-dnstap
# make
# make install
```

■補足
・公式マニュアルにはUbuntuとmacOSの手順しかないため、
それを参考に依存関係を解消させる
・bisonとflexも必要らしいが、なくてもコンパイルは通る。
気になる人は # dnf install bison flex してください
(unboundと同様)

権威DNSの調査結果　NSD

		NSD
共通機能2	dnstap	×要ビルド

dnstap

(レポジトリ版は非対応のためビルド版で確認)

(nsd.confの抜粋)

```
==============================
dnstap:
        dnstap-enable: yes
        dnstap-socket-path: "/tmp/dnstap.sock"
        dnstap-send-identity: no
        dnstap-send-version: no
        dnstap-identity: ""
        dnstap-version: ""
        dnstap-log-auth-query-messages: yes
        dnstap-log-auth-response-messages: yes
==============================
```

■補足
・レポジトリ版のnsd.confにdnstapの設定が
コメントアウトで入っているためそれを参考にすると良い

権威DNSの調査結果　NSD

		NSD
共通機能3	DoH	×(DoTのみ対応)
共通機能4	Prometheusとの連携	△(3rd Party製)
共通機能5	tcp-idle-timeout	○

Prometheusとの連携

(https://github.com/NLnetLabs/nsd から以下を取得し実行)
./nsd-control-setup.sh
(↑/etc/nsd配下に秘密鍵や証明書が作成される)

(nsd.confの抜粋)
==========
remote-control:
　　control-enable: yes
　　control-interface: 127.0.0.1

　　server-key-file: /etc/nsd/nsd_server.key
　　server-cert-file: /etc/nsd/nsd_server.pem
　　control-key-file: /etc/nsd/nsd_control.key
　　control-cert-file: /etc/nsd/nsd_control.pem
==========

(参考) nsd_exporterの利用

```
# dnf install golang
# go install github.com/optix2000/nsd_exporter@latest
# go/bin/nsd_exporter -ca /etc/nsd/nsd_server.pem ¥
  -key /etc/nsd/nsd_control.key ¥
    -cert /etc/nsd/nsd_control.pem -nsd-address 127.0.0.1:8952

# curl http://127.0.0.1:8080/metrics
```

tcp-idle-timeout

(nsd.confの抜粋)
==========
server:
　　tcp-timeout: 1
==========

権威DNSの調査結果　knotDNS

		knotDNS
共通機能1	レポジトリからのインストール可否	△3.2.6(EPEL)

レポジトリからのインストール

```
# dnf install epel-release
# dnf install knot
# systemctl start knot.service

# knotd -V
knotd (Knot DNS), version 3.2.6
```

権威DNSの調査結果　knotDNS

ソースコードからのインストール

```
# dnf install tar libtool autoconf automake pkgconfig  gnutls-devel make gcc
# dnf --enablerepo=devel install userspace-rcu-devel libedit-devel lmdb-devel

# tar xvf knot-3.2.6.tar.xz
# cd knot-3.2.6

(configureオプションなし)
# ./configure
# make
# make install

(configureオプションあり)
# dnf --enablerepo=devel  install libcap-ng-devel systemd-devel libidn2-devel protobuf-c-devel ¥
                fstrm-devel libmaxminddb-devel libnghttp2-devel libmnl-devel
# ./configure --with-libnghttp2 --enable-dnstap  --with-module-dnstap=yes
# make
# make install
```

■補足
・dnstapとprometheus連携のためのオプション追加

権威DNSの調査結果　knotDNS

		knotDNS
共通機能2	dnstap	×要ビルド

dnstap

(レポジトリ版は非対応のためビルド版で確認)

(knot.confの抜粋)

```
==============================
mod-dnstap:
  - id: capture_all
    sink: unix://tmp/caputure.tap

template:
  - id: default
    global-module: mod-dnstap/capture_all
==============================
```

権威DNSの調査結果　knotDNS

		knotDNS
共通機能3	DoH	×
共通機能4	Prometheusとの連携	△(3rd Party製)
共通機能5	tcp-idle-timeout	○

Prometheusとの連携

(knot.confの抜粋)

```
==============================
mod-stats:
  - id: custom
    edns-presence: on
    query-type: on

template:
  - id: default
    module: mod-stats/custom
==============================
```

(参考) knot_exporterの利用

https://github.com/ghedo/knot_exporter をダウンロード

■補足
・中身は単なるpythonスクリプト

```
# dnf install python3
# pip3 install prometheus_client
# chmod 755 ./knot_exporter
# ./knot_exporter --knot-library-path /usr/lib64/libknot.so.13
# curl http://127.0.0.1:9433/metrics
```

tcp-idle-timeout

(knot.confの抜粋)

```
==============================
server:
    tcp-idle-timeout: 1
==============================
```

権威DNSの調査結果　所感

		bind9	NSD	knotDNS
共通機能1	レポジトリからのインストール可否	○9.16.23	△4.3.8(EPEL)	△3.2.6(EPEL)
共通機能2	dnstap	○	×要ビルド	×要ビルド
共通機能3	DoH	×9.17から	×(DoTのみ対応)	×
共通機能4	Prometheusとの連携	○	△(3rd Party製)	△(3rd Party製)
共通機能5	tcp-idle-timeout	○	○	○
権威DNSの機能1	HTTPSレコードの登録	○	○	○
権威DNSの機能2	CAAレコードの登録	○	○	○

・dnstapやDoHの要件を気にしなければNSDやknotDNSも選択肢となる
  - フルサービスリゾルバは反復問い合わせにDoHを使わないため権威DNSのDoH化は優先度低
  - 通信が暗号化されないためトラヒックの可視化にポートミラーやDSCは引き続き使えるため
    dnstapの導入も優先度は若干低
・EPELを使わないのであればbind9。脱bindを目指すなら、何かあった時にコンパイルする可能性を考慮し
  依存関係が少ないNSDの方が今の所はオススメ

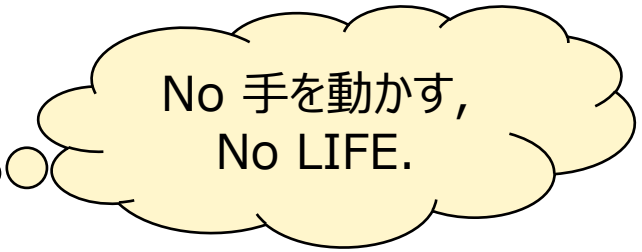
全体を通して

- 前述の所感には「普段使ってみての所感」を含んでいないため実際に利用/運用するとまた変わってくるはず

- 「単に機能を使ってみる」ぐらいであればそこまで時間はかからない
  - 依存関係も似たり寄ったりなのでビルド用VMを1つ準備しておくと良いように感じた
    - DoHの動作確認用にbind9.18のdigは一家に一台必要
  - knot-resolverだけビルドのハードルが高いが、おそらく「モジュールを交換/追加」することを想定していてknot-resolver本体はあまりアップデートしない思想かも？

- Rocky8公式レポジトリだけだと意外と選択肢がないことが分かった
  - Rocky9は未調査だが見た所あまり変化なし
  - Ubuntuはどうなんでしょう…？

- 皆さんも是非ビルドしたり、使ってみたりしてください



■免責事項(再掲)

今回ご紹介する手順や設定はあくまで動作させることのみを目的としたものになります。
実際にソフトウェアをご利用される場合はセキュリティを考慮した設定を行い、各自試験を行ってください。
不具合など発生した場合において弊社(私含め)は一切の責任を負いかねますのであらかじめご了承ください。

フルサービスリゾルバと権威DNSのサマリ

		bind9	unbound	knot-resolver
共通機能1	レポジトリからのインストール可否	○9.16.23	○1.16.2	△5.6.0(EPEL)
共通機能2	dnstap	○	×要ビルド	○
共通機能3	DoH	×9.17から	×要ビルド	○
共通機能4	Prometheusとの連携	○	△(3rd Party製)	○
共通機能5	tcp-idle-timeout	○	○	未確認
フルサービスリゾルバの機能1	RFC8806対応	○	○	○
フルサービスリゾルバの機能2	HTTPSレコードに対するA/AAAA付与	×	×	×

		bind9	NSD	knotDNS
共通機能1	レポジトリからのインストール可否	○9.16.23	△4.3.8(EPEL)	△3.2.6(EPEL)
共通機能2	dnstap	○	×要ビルド	×要ビルド
共通機能3	DoH	×9.17から	×(DoTのみ対応)	×
共通機能4	Prometheusとの連携	○	△(3rd Party製)	△(3rd Party製)
共通機能5	tcp-idle-timeout	○	○	○
権威DNSの機能1	HTTPSレコードの登録	○	○	○
権威DNSの機能2	CAAレコードの登録	○	○	○

フルサービスリゾルバと権威DNSのサマリ (補足なし)

		bind9	unbound	knot-resolver
共通機能1	レポジトリからのインストール可否	○	○	△
共通機能2	dnstap	○	×	○
共通機能3	DoH	×	×	○
共通機能4	Prometheusとの連携	○	△	○
共通機能5	tcp-idle-timeout	○	○	ー
フルサービスリゾルバの機能1	RFC8806対応	○	○	○
フルサービスリゾルバの機能2	HTTPSレコードに対するA/AAAA付与	×	×	×

		bind9	NSD	knotDNS
共通機能1	レポジトリからのインストール可否	○	△	△
共通機能2	dnstap	○	×	×
共通機能3	DoH	×	×	×
共通機能4	Prometheusとの連携	○	△	△
共通機能5	tcp-idle-timeout	○	○	○
権威DNSの機能1	HTTPSレコードの登録	○	○	○
権威DNSの機能2	CAAレコードの登録	○	○	○

フルサービスリゾルバと権威DNSのサマリ (ビルド版)

		bind9	unbound	knot-resolver
共通機能1	レポジトリからのインストール可否			
共通機能2	dnstap	○	○	○
共通機能3	DoH	○	○	○
共通機能4	Prometheusとの連携	○	△	○
共通機能5	tcp-idle-timeout	○	○	ー
フルサービスリゾルバの機能1	RFC8806対応	○	○	○
フルサービスリゾルバの機能2	HTTPSレコードに対するA/AAAA付与	×	×	×

		bind9	NSD	knotDNS
共通機能1	レポジトリからのインストール可否			
共通機能2	dnstap	○	○	○
共通機能3	DoH	○	×	×
共通機能4	Prometheusとの連携	○	△	△
共通機能5	tcp-idle-timeout	○	○	○
権威DNSの機能1	HTTPSレコードの登録	○	○	○
権威DNSの機能2	CAAレコードの登録	○	○	○