キャッシュサーバの設定

IIJ 山口崇徳

DNS Summer Days 2014

自己紹介

- IIJというところでDNSの運用やってます
  - お客様用参照サーバ
  - お客様のゾーンを預かる権威サーバ
  - 某ccTLDのセカンダリ
- 最初のDNSのお仕事は BIND4 → BIND8 の移行
  - 前世紀末
- その他、メールやったり web いじったり
  - L7 のなんでも屋さん

はじめに

- BIND、Unbound が対象
  - ほかはさわったことないので…
  - 設定のコピペはできなくても、考え方自体は他でも通用するはず
- 設定ファイルの文法レベルの話、BIND で権威サーバと共通する設定は触れません
  - 時間が足りないっす…
  - マニュアル読んでください
- DNSSEC は触れません

基本設定

RTFM

- BIND 9 Administrator Reference Manual
  - ソースアーカイブの bind-9.x.x/doc/arm/Bv9ARM.html
  - パッケージでインストールすると /usr/{,local/,pkg/}share/doc/ の下あたりにあることが多い
    - debian は bind9-doc パッケージを別途インストール
- Unbound は web で
  - http://unbound.net/documentation/
    - http://unbound.jp/unbound/ に和訳あり
  - man も読むべし
    - unbound(8), unbound.conf(8) など
- ぐぐる前にまずこちらを参照しよう
  - 残念ながらぐぐって見つかる情報には嘘が多いです

最低限の設定

- BIND
  - 再帰検索をおこなうことを明示

```
options {
    recursion yes;
};
```

    - デフォルト yes なのでなくても動くが、権威サーバでないことをはっきりさせるためにも書いておくべき
- Unbound
  - 空っぽでも動く
- 両者とも、アクセス制限を追加すべし

なぜアクセス制限?

- アクセス制限されていないキャッシュDNSサーバ = オープンリゾルバ
- DNS amp 攻撃の踏み台にされやすい
  - アドレス詐称クエリによりサイズを増幅させて帯域飽和させる攻撃
  - アクセス制限されていれば影響が限定的になる
- キャッシュポイズニング攻撃を受けやすい
  - アクセス制限があってもポイズニング攻撃自体は可能
  - が、アクセス制限されていればポイズニングに成功したかどうか攻撃者は判別が困難

BIND: allow-なんとか(1)

- allow-query
  - サーバへのクエリそのものの許可
- allow-query-cache
  - キャッシュ情報へのアクセス許可
- allow-recursion
  - 再帰クエリの許可
- キャッシュサーバを利用する上では、以上3つすべてが許可されている必要がある
  - 権威サーバでは allow-query だけ許可されていればよい

allow-なんとか(2)

- allow-query が設定されていない場合
  - すべてのホストからのアクセスが許可される
- allow-query-cache が設定されていない場合
  - allow-recursion の設定が使われる
  - allow-recursion も未設定なら、allow-query が使われる
- allow-recursion が設定されていない場合
  - allow-query-cache の設定が使われる
  - allow-query-cache も未設定なら、allow-query が使われる
  - allow-query{,-cache} どちらも未設定なら、localhost, localnets が許可される

allow-なんとか(3)

- allow-hoge はひとつ設定すると、他の allow-fuga の設定にも影響を与える
  - キャッシュサーバとして利用するには、3つすべてが許可されている必要があるが、ひとつ設定すれば他は設定しなくてもしたことになる
  - しかも条件がビミョーに違う
  - ひじょーにわかりにくい
- 横着せずに、3つとも明示的に記述すると間違いがない

allow-なんとか(4)

- 以上、BIND 9.4 以降の話
- 9.3 まではデフォルトが違っていた
  - allow-query-cache は存在しなかった
  - allow-recursion, allow-query どちらも「ぜんぶ許可」
    - デフォルトでオープンリゾルバ
- 公式にはすでにディスコンになっているバージョン
  - が、RHEL4/5 の標準 RPM がそれぞれ 9.2、9.3 なので使っている人はいまだに多いかと…
  - バージョンアップするときには要注意

unboundのアクセス制限

```
access-control: 0.0.0.0/0 refuse
access-control: 127.0.0.0/8 allow
```

- マッチしたクライアントに対する挙動
  - allow: アクセス許可(ただし非再帰クエリは拒否)
  - allow_snoop: アクセス許可(非再帰も許可)
  - deny: クエリを捨てる(応答を返さない)
  - refuse: クエリを拒否する(拒否応答を返す)
  - {deny,refuse}_non_local
- 0.0.0.0/0 = ipv4 全体
  - ::0/0 = ipv6 全体

BINDのログ

- bind のロギングはそれだけで30分ぐらい喋れそうなぐらい細かい設定ができる
  - そんな時間はないのでばっさり割愛

- デフォルトではかなり冗長なログを吐くので、重要度の低いものは黙らせておくといいかも

```
logging {
    category lame-servers { null; };
    category edns-disabled { null; };
};
```

Unboundのログ

- 設定するところほとんどないし…
  - 冗長度(verbosity: 1)
  - syslog 経由かファイルに出力か(use-syslog: yes)
  - query log を取るかどうか(log-queries: no)
  - …ぐらいしか
- デフォルトのままでいいんじゃないかな

クエリログ

- 規模にもよるが、ふだんは出力しない方がよいかと
  - あっというまにサイズが膨れあがるので
  - 調査の必要があるときだけ一時的に出力させる
  - BIND: `querylog no;`
  - Unbound: `log-queries: no`
- 一時的なクエリログ出力の切り替え
  - BIND: `rndc querylog [on|off]`
    - 9.8 以前は on/off 指定できない(toggle)
  - Unbound: `unbound-control set_option log-queries: [yes|no]`
  - どちらも制御コマンドを使えるように事前準備が必要

ラウンドロビン

- BIND
  - RRset ごとに細くラウンドロビンの挙動を設定できる
    - が、通常は全部まとめて同じ設定でよいかと
  - 9.9 のデフォルトはランダム順
    - rrset-oder { order random; };
    - 9.8 までは循環(cyclic)
    - 固定順(fixed)にするには configure --enable-fixed-rrset
    - 強いこだわりがなければ random も cyclic もどちらでもよいかと
- unbound
  - デフォルトはラウンドロビンしない(固定順)
  - rrset-roundrobin: yes でランダムに (1.4.17以降)

パフォーマンスチューニング

- チャンネルはそのまま! この後すぐ!!
  - wktk しながら東さんのセッションを待とう

セキュリティ

キャッシュポイズニング

- キャッシュサーバが権威サーバに出したクエリへの応答として、第三者が偽造応答を割り込ませれば、嘘の情報をキャッシュさせられるよ
- 偽造応答に必要な情報
  - IP アドレス: 1/(権威サーバの数) の確率で的中
  - ポート番号: 16ビット(65536とおり)からひとつ
  - クエリ ID: DNS クエリの識別子; 16ビットランダム値
- ポート番号固定→推測の的中確率は 1/6.5万
- ポート番号ランダム→的中確率 1/43億

やっちゃいけない設定

```
query-source port 53;
```

- 問い合わせ元のソースポート固定禁止
  - 53 がダメなのではなく、固定するのがダメ
  - こうしないとファイアウォールを越えられない場合、bindではなくFWの方を直すべし
- unbound ではふつーに設定していればランダム
  - トリッキーな方法でポート固定もできるがヒミツ

NATとポートランダム化(1)

- NAT 箱の裏側で動くキャッシュサーバに毒入れを試みるパケットが届くの?
  - 届きます
  - ポート番号の推測が的中した場合、それは NAT テーブルに載っている
    - 応答パケットの偽造なので
    - NAT 箱が dst port をちゃんと書き換えた上でキャッシュサーバにパケットを届けてくれる
  - 的中しなければ NAT テーブルにないので弾かれる

NATとポートランダム化(2)

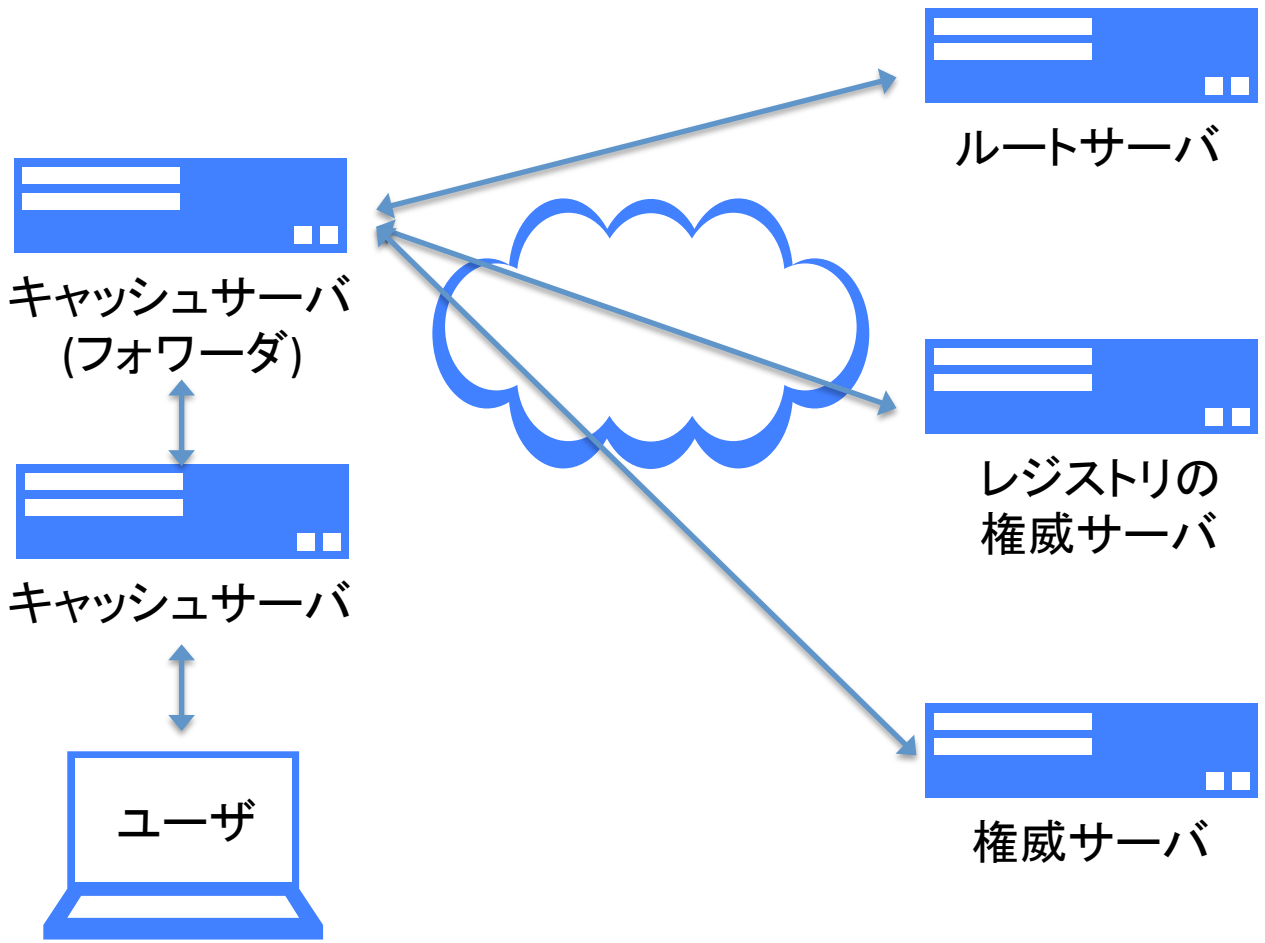
- NAT 箱がソースポートをどう書き換えるか確認
  - https://www.dns-oarc.net/oarc/services/dnsentropy
  - 結果が GREAT となっていればよい、わけではない
    - 実際に観測されたポート番号(Values Seen) の値を確認すべし
  - もし連番(ないしはそれに近い連続した値)になっていたら、GREAT でもダメ
    - 推測しやすい
- NAT 箱の入れ換え、ネットワーク構成の変更などを検討すべし
  - キャッシュサーバの設定では対処不可

他DNSサーバとの連携

フォワーダ(1)

- 自前でできない/やりたくない名前解決をよそに回送(forward)するというアプローチ
  - よそのキャッシュサーバ = フォワーダ
- どんなときに使うの?
  - グローバルへの疎通のないキャッシュサーバが疎通のあるキャッシュサーバに問い合わせをフォワード
  - 再帰検索は遅いので、キャッシュが溜まってる別サーバにフォワード
  - 家庭用ルータの DNS 機能はたいていフォワードしてるだけ

フォワーダ(2)



フォワーダの設定

- すべての名前解決を別のキャッシュサーバにフォワードする
  - 特定ゾーンの名前解決だけをフォワードするなら、下の例の "." をそのゾーンに変えればよい

BIND

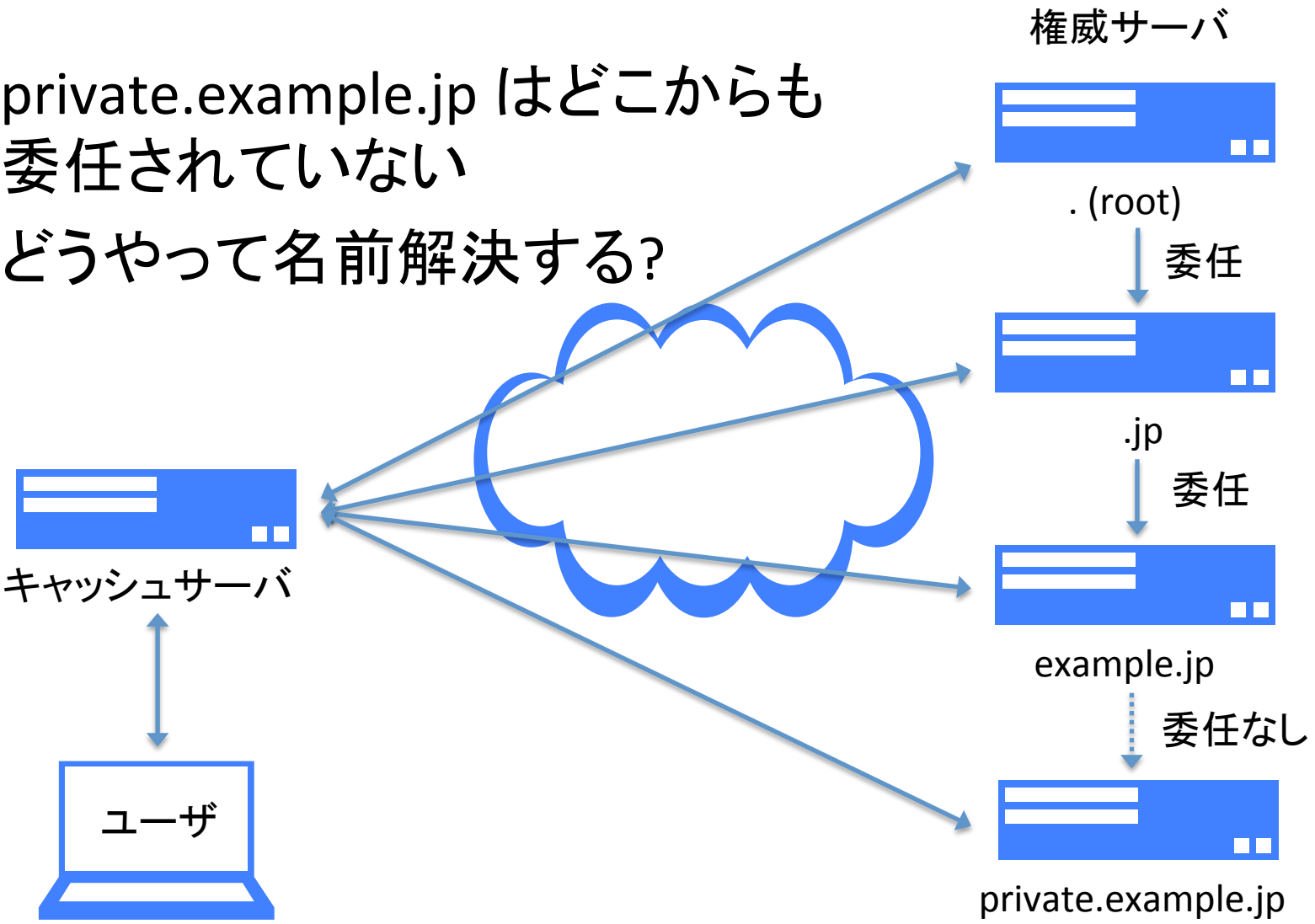
```
zone "." {
    type forward;
    forwarders { 10.0.0.1; };
    forward only;
};
```

Unbound

```
forward-zone:
    name: "."
    forward-addr: 10.0.0.1
```

どこからも委任されないゾーン

- private.example.jp はどこからも委任されていない
- どうやって名前解決する?



内部専用ゾーンの名前解決

- 権威サーバはルートサーバから NS レコードを辿って探すのが基本
- 先の例では、private.example.jp を example.jp からふつーに委任しちゃうという方法もできなくはない
  - が、内部用の委任が外部から見えちゃうのはよろしくない
  - 正引きならともかく、10.0.0.0/8 のようなプライベートアドレスの逆引きゾーン(10.in-addr.arpa) を in-addr.arpa から自分のところに委任してもらうことはそもそもできない

内部専用ゾーン

- localhost とその逆引き
- 組織内だけで使われる名前
  - private.example.jp
- プライベートアドレスの逆引き
  - 168.192.in-addr.arpa など
- リンクローカルアドレスその他予約アドレスの逆引き
  - RFC1918, RFC4193, RFC5737, RFC6303, RFC6598

プライベートアドレスの逆引き(1)

- ローカルで名前解決できるようにする
  - たとえ逆引きなんか引けなくてもいいや、という場合であっても、空ゾーン(SOA と NS だけのゾーン)でいいので用意しておく
  - ないとどうなるの?
    - プライベートアドレスの逆引きを調べるのに、インターネットにクエリが出ていってしまう
    - 障害でインターネットとの疎通がなくなると、インターネットとは無関係なイントラ内のホストに対するアクセスも困難になる(かも)

プライベートアドレスの逆引き(2)

- BIND、Unbound とも予約アドレス空間の逆引き空ゾーンが組み込まれている
  - BIND は 9.5 以降
- 空でない逆引きゾーンが必要なら自前で用意する
  - BIND 9.4 以前は空ゾーンも自前で用意
- 組織内用正引きゾーン、localhost ゾーンも自前で
  - Unbound は localhost も組み込み
- どうやって?
  - キャッシュサーバ自身でゾーンを持つ
  - 別の権威サーバにゾーンを置いてそちらに問い合わせる

empty-zones-enable

- BIND 9.5 以降で組み込みの逆引き空ゾーンを使うための設定
  - とくに理由がなければ yes にしておきましょう
- 有効だと起動時に大量のログが出力される
  - エラーではない
  - このログ出力をやめさせるためにわざわざ no にするという間違った設定例が蔓延しています
  - 真似しないでください
  - あえて no にするのであれば、9.4 以前のように自前で空ゾーンを用意しましょう

自前ゾーン: BIND

- もともと権威と共用なので、自前でゾーンを持たせるのはカンタン

```
zone "localhost" {
    type master; file "localhost.zone";
};
zone "127.in-addr.arpa" {
    type master; file "localhost-rev.zone";
};
zone "1.0.0.(略).0.0.ip6.arpa" {
    type master; file "localhost-rev6.zone";
};
```

自前ゾーン: Unbound

- キャッシュ専用なので、権威機能は簡易なものだけ
- localhost ゾーンの例(man より)

```
local-zone: "localhost." static
local-data: "localhost. 10800 IN NS localhost."
local-data: "localhost. 10800 IN SOA localhost. nobody.invalid.
1 3600 1200 604800 10800"
local-data: "localhost. 10800 IN A 127.0.0.1"
local-data: "localhost. 10800 IN AAAA ::1"
```

  - 自前ゾーンはこんなふうに unbound.conf にいちいち書く必要あり
  - localhost ゾーンは実際には組み込みなので不要
- 実際のところ local-data だけでけっこうなんとかなる

ローカルゾーン専用権威サーバ

- BIND でも Unbound でも、内部専用ゾーンがある程度大きくなるなら、別途専用の権威サーバを構築し、そちらにクエリを投げるようにすべき
- サーバ増設が必要?
  - この権威サーバは世界中からアクセスされるわけではない
    - 内部のキャッシュサーバからのみアクセスできればよい
  - キャッシュサーバと同一ホスト内の別プロセスでも十分
    - リソースが競合しないように注意
    - ポート 53 ではなく 54 で listen させるとか 127.0.0.2 を使うとか
    - pid ファイルその他の置き場所を変える

stub ゾーン

- 特定ゾーンの名前解決をDNS の委任ツリーに含まれない権威サーバに問い合わせる
  - 設定・動作はフォワーダによく似ているが、問い合わせ先が権威サーバなので、再帰要求しない
  - ぶっちゃけ、forwarder として設定してもちゃんと動いてしまう

BIND

```
zone "private.example.jp" {
    type stub;
    masters { 10.0.0.1; };
};
```

Unbound

```
stub-zone:
    name: "private.example.jp"
    stub-addr: 10.0.0.1
```

Unbound ハマリどころ

- stub-zone を設定しても、組み込みの逆引きゾーンの方が優先される
  - 組み込みゾーンを使わないようにさせる

```
local-zone: "168.192.in-addr.arpa" transparent
```

- 権威サーバからプライベートアドレスな応答が返ってきても削られる
  - プライベートアドレスを応答に含んでもよいゾーンを明示する

```
private-domain: "private.example.jp"
```

- 権威サーバを同じホストに同居させると名前解決できない
  - localhost への問い合わせを許す

```
do-not-query-localhost: no
```

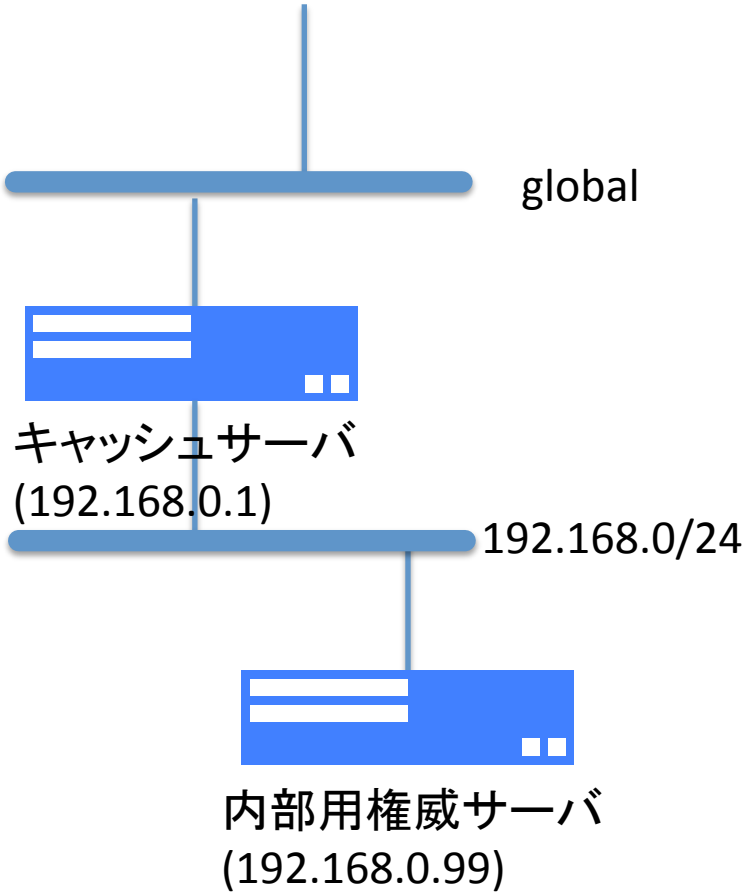
内部専用ドメイン

- キャッシュサーバの設定とは関係ないんですが…
- 内部用のドメインとして、勝手 TLD は使わないようにしましょう
  - gTLD が現在爆発的に増加中
  - 内部ドメインとして .mydomain を利用→ .mydomain が新規に gTLD として登録される → うぎゃー
  - 名前衝突問題 (name collision)
    - https://www.nic.ad.jp/ja/dom/new-gtld/name-collision/
- 内部用に新規ドメインを取得するか、既存ドメインのサブドメインを使う
  - お金かかるけど…

まとめ

実際に設定してみる

- こんな構成
  - サーバアドレス 192.168.0.1
  - 内部用正引きドメイン private.example.jp
  - ローカルネットワーク 192.168.0.0/24, 2001 : db8::/48
  - アクセス許可もこの範囲からのみ
  - 正引き、逆引きとも別途権威サーバ 192.168.0.99 にゾーンがある



named.conf (1)

```
options {
     // ワーキングディレクトリ
     directory "/var/named";
     // 各種出力ファイルの設定
     // directory で指定したところからの相対パス
     pid-file "log/named.pid";
     statistics-file "log/named.stats";
     memstatistics-file "log/named.memstats";
     dump-file "log/named.dumpdb";
     recursing-file "log/named.recursing";

     // listen アドレス
     listen-on { 127.0.0.1; 192.168.0.1; };
     listen-on-v6 { any; };

     recursion yes;            // 再帰検索する
     empty-zones-enable yes;   // 組み込みの逆引き空ゾーンの有効化
     querylog no;              // クエリログ取得しない
     minimal-responses yes;    // 可能なら authority/additional section を削る
```

named.conf (2)

```
        // アクセス制限
        allow-query { localhost; private; };
        allow-recursion { localhost; private; };
        allow-query-cache { localhost; private; }; // 9.4 以降

        // パフォーマンスチューニング
        recursive-clients 1000;
        max-cache-size 1G;
};

// アクセス許可ネットワークの定義
acl private {
        192.168.0.0/24;
        2001:db8::/48;
};
```

named.conf (3)

```
// ログ設定
// query log はファイルへ出力する
// (querylog no; と設定してるので rndc querylog で有効にしたときのみ出力)
// それ以外はデフォルトのまま(syslog)
logging {
     channel log_query {
          file "log/query.log" versions 5 size 10m; // 10MB でローテーション、5世代
          print-time yes;
     };
     category queries { log_query; };
     // 冗長なログは捨てる
     category lame-servers { null; };
     category edns-disabled { null; };
};

// rndc の設定
include "etc/rndc.key";
controls {
     inet 127.0.0.1 port 953 allow { 127.0.0.1; } keys { "rndc-key"; };
};
```

named.conf (4)

```
// ゾーン定義

// ルートヒントはちゃんと管理できないなら設定しなくてもよい
zone "." {
     type hint;
     file "etc/root.hint";
};
// localhost とその逆引きは自前で持つ
zone "localhost" {
     type master;
     file "etc/localhost.zone";
};
zone "127.in-addr.arpa" {
     type master;
     file "etc/localhost-rev.zone";
};
zone "1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.ip6.arpa" {
     type master;
     file "etc/localhost-rev6.zone";
};
```

named.conf (5)

```
// 内部専用ゾーンは専用の権威サーバへ
zone "private.example.jp" {
     type stub;
     masters { 192.168.0.99; };
};
zone "0.168.192.in-addr.arpa" {
     type stub;
     masters { 192.168.0.99; };
};
zone "0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa" {  // 2001:db8::/48
     type stub;
     masters { 192.168.0.99; };
};

// プライベートアドレスの逆引きは BIND のデフォルトに任せるので設定しない
// (9.4 以前では別途設定が必要)

// EOF
```

unbound.conf (1)

```
server:
     verbosity: 1
     interface: 127.0.0.1
     interface: 192.168.0.1
     interface: ::0
     chroot: ""
     username: "unbound"
     directory: "/usr/local/etc/unbound"
     pidfile: "/var/run/unbound.pid"
     # ルートヒント; ちゃんと管理できないならコメントアウト
     root-hints: "/usr/local/etc/unbound/root.hint"

     # アクセス制限
     access-control: 0.0.0.0/0 refuse
     access-control: ::0/0 refuse
     access-control: 127.0.0.0/8 allow
     access-control: 192.168.0.0/24 allow
     access-control: ::1 allow
     access-control: 2001:db8::/48 allow
```

unbound.conf (2)

```
# ラウンドロビンさせる
rrset-roundrobin: yes
# 可能なら authority/additional section を削る
minimal-responses: yes

# 応答にプライベートアドレスを含んでよいゾーン(内部ゾーン)
private-domain: "private.exmaple.jp"
# 0.168.192.in-addr.arpa は組み込み逆引きゾーンを使わない
local-zone: "0.168.192.in-addr.arpa" transparent
# localhost への query を許容するかどうか(今回の例では yes のままで問題ない)
# do-not-query-localhost: no

# パフォーマンスチューニング
# unbound のデフォルトはかなり小規模向けなので注意(以下の例もかなり控え目)
num-queries-per-thread: 1024
outgoing-num-tcp: 100
incoming-num-tcp: 100
num-threads: 2
msg-cache-size: 512m
rrset-cache-size: 512m
```

unbound.conf (3)

```
# 内部専用ゾーン
stub-zone:
     name: "private.example.jp"
     stub-addr: 192.168.0.99
stub-zone:
     name: "0.168.192.in-addr.arpa"
     stub-addr: 192.168.0.99
stub-zone:
     name: "0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa"
     stub-addr: 192.168.0.99
# localhost とその逆引き、プライベートアドレスの逆引きは
# デフォルトの組み込みゾーンをそのまま使うので定義しない

# unbound-control を使えるようにする
remote-control:
     control-enable: yes

# EOF
```