KNOT RESOLVER

A flexible DNSSEC-validating Resolver

Ondřej Surý • ondrej.sury@nic.cz • 1.4.2016

CZ DOMAIN REGISTRY

What is Knot DNS Resolver?

- **Platform for building recursive DNS service**
- Open-source DNS Resolver (GPLv3+)
- Full DNSSEC support:
  - RFC 6650 – ECDSA support
  - RFC 5011 – Automated Trust Anchor Management
  - RFC 7646 – Negative Trust Anchors

What is Knot DNS Resolver?

- Written in C and LuaJIT
- Built on top of Knot DNS Libraries
- Scriptable daemon with dynamic configuration in Lua
- Simple **core** extensible with modules in C, Lua & Go
- “Happy Eyeballs” IPv6 (20ms headstart)
- No internal threading, scales by self-replication

Who is it for? Everybody!

- Large recursive DNS farms
- Small recursors in private networks
- Personal resolvers
- Geeks, tinkerers, you :)

Large recursive DNS farms

- Scales, the really fast scriptable engine allows you to change resolution
- Flexible shared cache backends
  - Local: lmdb
  - Networked: memcached, redis
- New instances just pick the data from the shared cache
- Great statistics, metrics, and plotting with Graphite backend
  - and f.e. InfluxDB, Grafana
- RF7646 Negative Trust Anchors
- Cluster-aware – etcd module for shared self-configuration
- Views and ACL support
- Prefetching (keeping the cache hot)

Small recursors in private networks

- QNAME minimisation for DNS privacy
- DNSSEC and RFC5011 key management
- Low memory consumption (cache can be paged out)
- Query policy based resolution
  - Match: pattern, suffix, RPZ
  - Action: PASS, DENY, DROP, FORWARD, TC
- DNS64 support to complement NAT64

Personal resolvers

- Simple config-less operation
  - Just give it a writeable file for DNSSEC root trust anchor and you are good to go
- Persistent caching (survives reloads/reboots)
- Tinyweb module for monitoring your queries
  - Live Demo: https://kitsune.labs.nic.cz/
- Future work:
  - DNS over TLS
  - Better handling of the hotel almost-WiFi
    - Captive portals
    - Unreliable transports

Geek, Tinkers, …

- kresd is scriptable without binding go port 53
- scripts/kresd-host.lua
  - dig/host like utility

```
$ ./scripts/kresd-host.lua -c IN -t AAAA www.fosdem.org
www.fosdem.org has IPv6 address 2001:67c:1808::5
```

- scripts/kresd-query.lua
  - Prints DNS response QNAME

```
kresd-query.lua -t SOA cz "print(pkt:qname())"
cz
```

  - Prints RCODE from the DNS response

```
kresd-query.lua -t SOA nan. "print(pkt:rcode())"
3 # ← NXDOMAIN
```

  - API specification in the documentation

Current status

- Brace yourself! 1.0.0 is coming!
- Comes with extensive documentation
  - http://knot-resolver.rtfd.org
- Give it a try!
  - Shiny new website: https://www.knot-resolver.cz/
  - Deb and RPM packages (see the website)
  - Sources: https://gitlab.labs.nic.cz/knot/resolver
  - Docker # docker run cznic/knot-resolver
- Throw a normal and a weird DNS stuff on it
- Report back any oddities or success stories

Policies – Filtering

- ```
  -- Load default policies
  modules = { 'policy' }
  ```

- ```
  -- Whitelist 'www[0-9].phishing'
  policy:add(policy.pattern(policy.PASS, '\4www[0-9]\8phishing'))
  -- Block all names below rhybar.cz
  policy:add(policy.suffix(policy.DENY, {'\8phishing'}))
  ```

- ```
  -- Custom rule
  policy:add(function (req, query)
          if query:qname():find('%d.%d.%d.192\7in-addr\4arpa') then
                  return policy.DENY
          end
  end)
  ```

- ```
  -- Disallow ANY queries
  policy:add(function (req, query)
          if query.type == kres.type.ANY then
                  return policy.DROP
          end
  end)
  ```

Policies – Forwarding

- -- Load default policies
  modules = { 'policy' }
- -- Forward all queries below 'company.se' to given resolver
  policy:add(policy.suffix(policy.FORWARD('192.168.1.1'), {'\7company\2se'}))
- -- Forward all queries matching pattern
  policy:add(policy.pattern(policy.FORWARD('2001:DB8::1'), '\4bad[0-9]\2cz'))
- -- Forward all queries (complete stub mode)
  policy:add(policy.all(policy.FORWARD('2001:DB8::1')))

Views and ACLs

- -- Load modules
  modules = { 'policy', 'view' }
- -- Whitelist queries identified by TSIG key
  view:tsig('\5mykey', function (req, qry) return policy.PASS end)
- -- Block local clients (ACL like)
  view:addr('127.0.0.1', function (req, qry) return policy.DENY end))
- -- Drop queries with suffix match for remote client
  view:addr('10.0.0.0/8', policy.suffix(policy.DROP, {'\3xxx'}))
- -- RPZ for subset of clients
  view:addr('192.168.1.0/24', policy.rpz(policy.PASS, 'whitelist.rpz'))
- -- Forward all queries from given subnet to proxy
  view:addr('10.0.0.0/8', policy.all(policy.FORWARD('2001:DB8::1')))

Networked cache

- Memcached

```
modules = { 'kmemcached' }
cache.storage = 'memcached://--SOCKET="/var/sock/memcached"'
cache.storage = 'memcached://--SERVER=<ipaddr> --SERVER=<name>'
```

- Redis

```
modules = { 'redis' }
-- redis over TCP
cache.storage = 'redis://127.0.0.1:6398'
-- redis over UNIX socket
cache.storage = 'redis:///tmp/redis.sock'
```

More examples

- DNS64

```
modules = { dns64 = 'fe80::21b:77ff:0:0' }
```

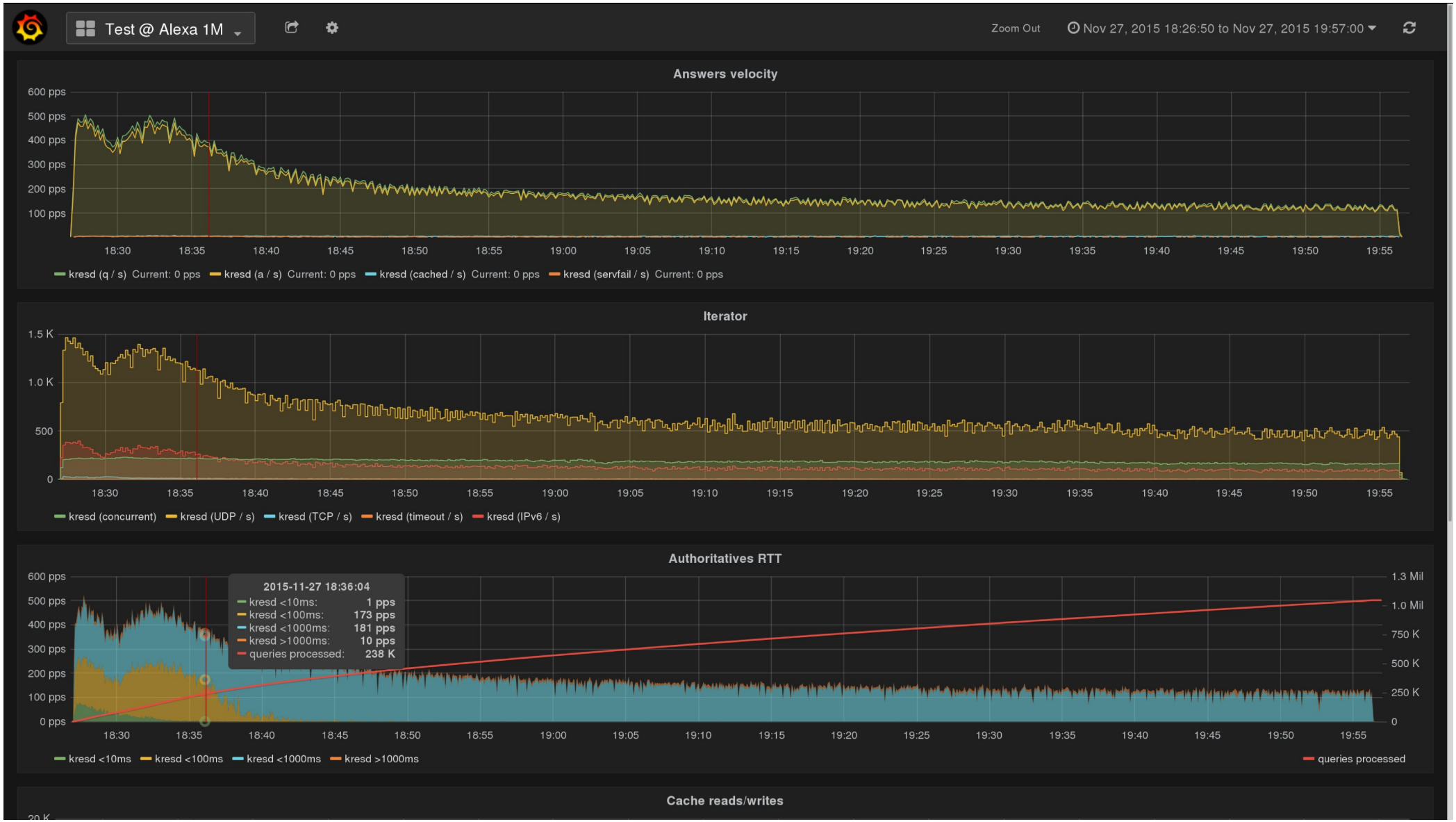
- Graphite

```
modules = {
        graphite = {
                prefix = hostname(), -- optional metric prefix
                host = '127.0.0.1',  -- graphite server address
                port = 2003,         -- graphite server port
                interval = 5 * sec,  -- publish interval
        }
}
```

- Tinyweb

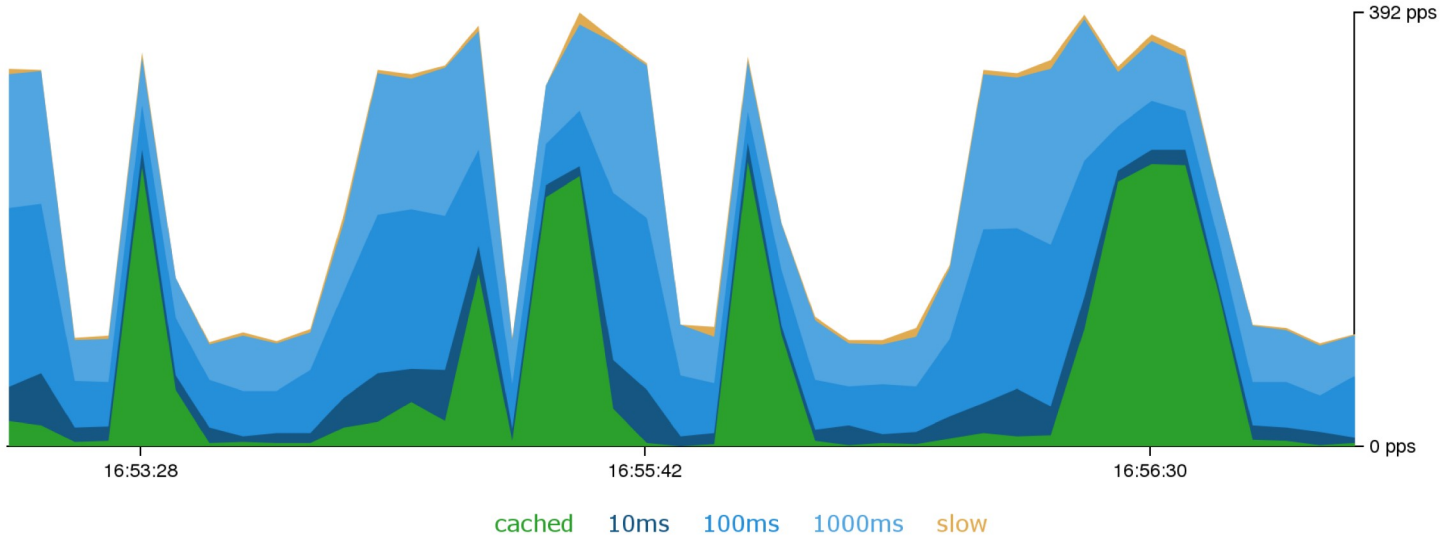
```
modules = {
  tinyweb = {
    addr = 'localhost:8080', -- Custom address
    geoip = '/usr/local/var/GeoIP' -- Different path to GeoIP DB
  }
}
```

Grafana – vizualizace statistik

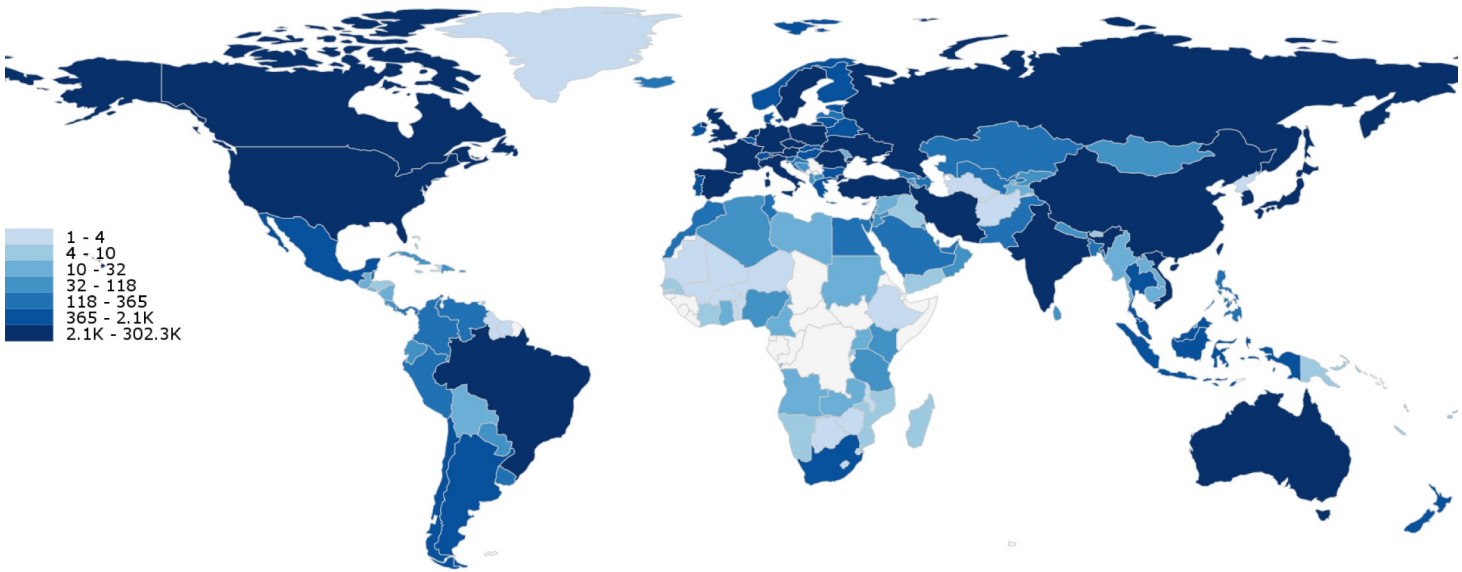


Tinyweb output

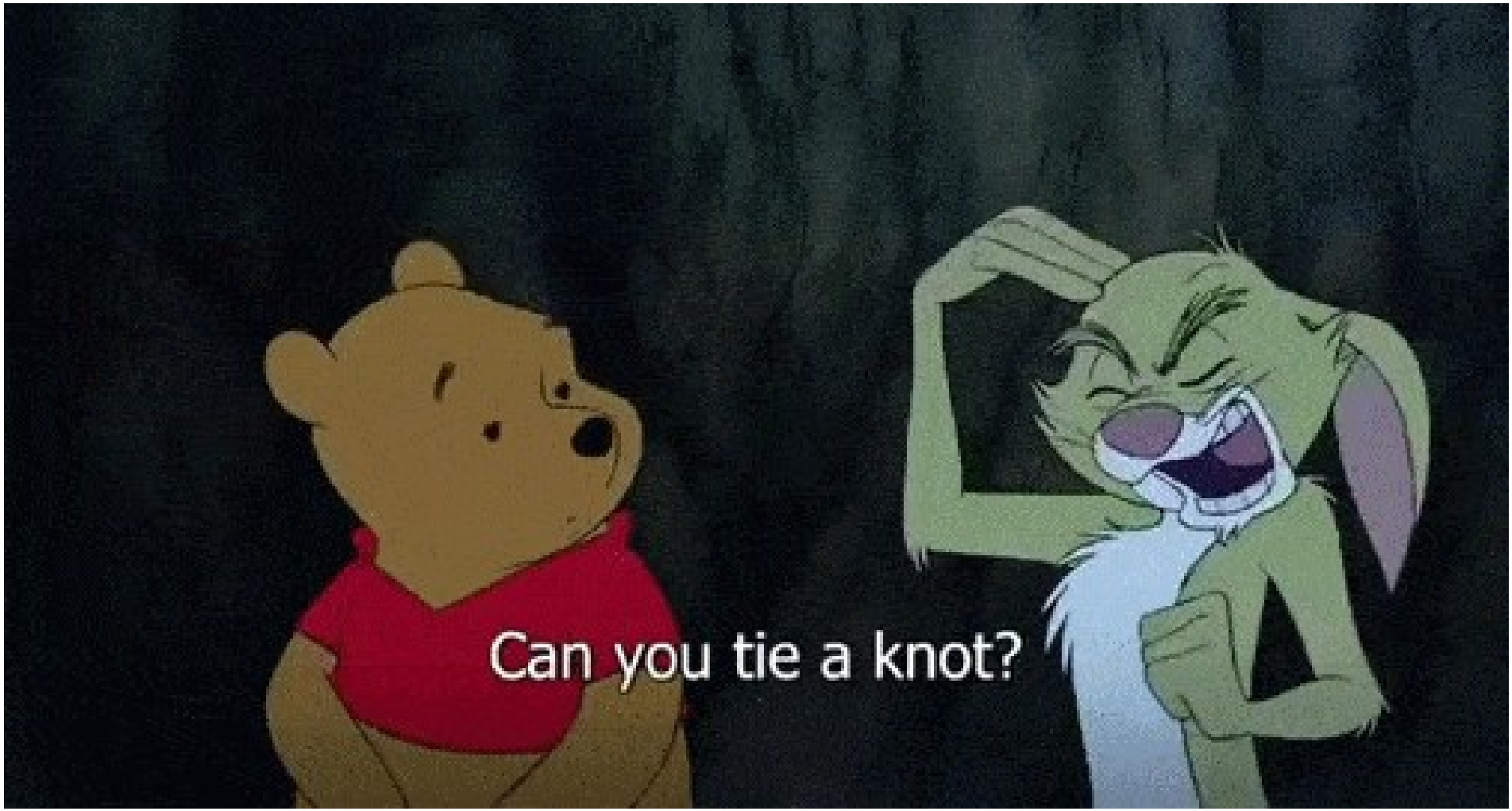
kresd @ kitsune



Queried servers



Thank you and you can Knot!



https://www.youtube.com/watch?v=aMxcAaR0oHU