The Kea Config Backend
Scalable DHCP configuration management with MySQL

Alan Clegg
August 14, 2019
Some notes on Kea

- Modern DHCPv4 and DHCPv6 server (1.6 in Aug 2019)
- Performance (1000s leases/sec)
- High Availability, NETCONF, RADIUS
- Databases (CSV, MySQL, PostgreSQL, Cassandra)
- Hooks (ISC and 3rd party libraries)
- REST API (120+ commands and counting)
Some notes on Kea

- Modern DHCPv4 and DHCPv6 server (1.6 in Aug 2019)
- Performance (1000s leases/sec)
- High Availability, NETCONF, RADIUS
- Databases (CSV, MySQL, PostgreSQL, Cassandra)
- Hooks (ISC and 3rd party libraries)
- REST API (120+ commands and counting)
The Backend Concept

- Leases (addresses, prefixes)
- Host reservations (per host details)
- Options
  - Pools
  - Subnets
  - Shared networks
  - Option definitions
  - Global parameters
Config Backend

Ability to retrieve configuration from a database

- Database stores DHCP configuration elements
- MySQL server can be colocated with DHCP or remote
- Multiple Kea servers can share one remote MySQL DB
- Database can be modified when DHCP servers are off-line

© 2019 - Internet Systems Consortium
Capabilities

- Changes made to the configuration database are applied to all servers without needing to restart.

- Pull model with configurable delay:
  - `config-fetch-wait-time`
Local or remote

- MySQL can be colocated with DHCP or remote
- Multiple DHCP servers can share one MySQL instance
- Configuration database can co-exist with other databases
Offline Configuration

- Dedicated Kea node for configuration management
- Manage configuration even when servers are offline
- Allows firewalling of configuration management
Use Cases

• Sharing configuration between HA partners
• Frequently changing configuration
• Automated configuration management
• Large configuration / Large deployments
• Scaling up or down
• Database Monitoring, Reporting, Backup, Redundancy
Config Backend

- Currently MySQL only
  - PostgreSQL considered in a future release
- DHCPv4 and DHCPv6 servers only for now
  - DDNS, Control Agent, etc. are not (yet) supported
- Some parameters must be configured via the JSON configuration file
  - For example: `server-tag`
Minimalist Configuration

- Using the configuration backend, "hard-coded" configuration can be significantly reduced:

  "server-tag"

  "interfaces-config"

  "control-socket"

  "config-control"

  "hooks-libraries"

  "lease-database"

  "expired-leases-processing"

- Who am I?
- Where am I?
- How do I talk?
- Where is my config?
- Where do I store data?
- When do I get rid of it?
Minimalist Configuration

• "Everything else" obtained from the Config Backend

• Global Parameters

• Option Definitions

• Subnets & Shared Networks

• Pools

• Options
Communicating with Kea

- **kea-ctrl-agent**
  - Listens on a configurable TCP port
    - `http` - directly
    - `https` - reverse proxy (nginx)
  - Listens on a UNIX socket (per service)
  - Consumes and Emits JSON
Enabling "Remote" Control

A sample kea-ctrl-agent.conf configuration file:

```json
{
    "Control-agent": {
        "http-host": "127.0.0.1",
        "http-port": 8080,

        "control-sockets": {
            "dhcp4": {
                "socket-type": "unix",
                "socket-name": "/tmp/kea-dhcp4.sock"
            },
            "dhcp6": {
                "socket-type": "unix",
                "socket-name": "/tmp/kea-dhcp6.sock"
            },
            "d2": {
                "socket-type": "unix",
                "socket-name": "/tmp/kea-dhcp-ddns.sock"
            }
        }
    }
}
```

Multiplexed TCP Communication here requires the service to be specified.

UNIX Socket
Communicating with Kea

- **kea-shell**
  - Included with the Kea distribution
  - Built when configured with `--with-shell`

- **socat**

- **curl**
  - Available with most UNIX/Linux distributions
Communicating with Kea

• kea-shell

```
kea-shell --host 127.0.0.1 --port 8080 --service dhcp4 config-get
```

• Postprocess

  • | jq .

  • | python -m json.tool
Communicating with Kea

• socat

```bash
echo '{ "command": "config-get" }' | \ socat /tmp/kea-dhcp4.sock -,ignoreeof
```

• curl

```bash
curl -X POST -H "Content-Type: application/json" \ -d '{ "command": "config-get", "service": [ "dhcp4" ] }' \ http://127.0.0.1:8080/
```
Enabling Config Backend

A sample `kea-dhcp6.conf` configuration file:

```
"Dhcp6": {
  "server-tag": "headquarters",
  "config-control": {
    "config-databases": [{
      "type": "mysql",
      "name": "kea",
      "user": "kea",
      "password": "password",
      "host": "192.0.2.1",
      "port": 3306
    }],
    "config-fetch-wait-time": 20
  },
  "hooks-libraries": [{
    "library": "/lib/kea/hooks/libdhcp_mysql_cb.so"
  }, {
    "library": "/lib/kea/hooks/libdhcp_cb_cmds.so"
  }],
  ...
}
```

- **Server Tag**: where to look for config
- **DB Credentials**: exposer JSON-based REST API
- **Refresh Interval**
- **CB Hook**
- **CB Command Hook**
- **Other DHCP6 Server Parameters**
Command List

```json
{
    "command": "list-commands",
    "service": [ "dhcp6" ]
}

{
    "arguments": [
        "build-report",
        "config-get",
        "config-set",
        "config-test",
        "remote-global-parameter4-del",
        "remote-global-parameter4-get",
        "remote-global-parameter4-get-all",
        ...
        "remote-subnet6-list",
        "remote-subnet6-set",
        "shutdown",
        "statistic-{get,remove,reset}"",
        "statistic-{get,remove,reset}-all",
        "version-get"
    ],
    "result": 0
}
```
Server Tags

- Server tags allow the creation of "groups" that are then be applied to one or more servers.

- Each server has exactly one server tag associated with it.

- Each object can have one or more server tags associated with it.

- There is a global "all" tag that will be associated with all servers.
Server Tags

Given Objects with IDs and server tags as shown:

- id: 100, server-tags: ["all"]
- id: 101, server-tags: ["headquarters"]
- id: 102, server-tags: ["headquarters", "remote"]
- id: 103, server-tags: ["backup"]
- id: 104, server-tags: []

© 2019 - Internet Systems Consortium
Create Server Tags

```bash
#!/bin/bash

echo ""servers": [ {
    "server-tag": "headquarters",
    "description": "The machine at HQ (v4)" } ]' | \ 
kea-shell --host 127.0.0.1 --port 8080 --service dhcp4 \ 
    remote-server4-set | jq ""

echo ""servers": [ {
    "server-tag": "headquarters",
    "description": "The machine at HQ (v6)" } ]' | \ 
kea-shell --host 127.0.0.1 --port 8080 --service dhcp6 \ 
    remote-server6-set | jq ""

echo ""servers": [ {
    "server-tag": "remote",
    "description": "The machine in the field (v4)" } ]' | \ 
kea-shell --host 127.0.0.1 --port 8080 --service dhcp4 \ 
    remote-server4-set | jq ""

[...]
```
Create Server Tags (output)

```json
[
  {
    "arguments": {
      "servers": [
        {
          "description": "The machine at HQ (v4)",
          "server-tag": "headquarters"
        }
      ]
    },
    "result": 0,
    "text": "DHCPv4 server successfully set."
  }
]
[...]
Get all DHCPv4 Server Tags

```json
{
    "command": 
        "remote-server4-get-all",
    "service": [ "dhcp4" ]
}

{
    "arguments": {
        "count": 2,
        "servers": [
            {
                "description": "The machine at HQ (v4)",
                "server-tag": "headquarters"
            },
            {
                "description": "The machine in the field (v4)",
                "server-tag": "remote"
            }
        ]
    },
    "result": 0,
    "text": "2 DHCPv4 server(s) found."
}
```
Create IPv6 Subnet

```
{
  "subnets": [
    {
      "id": 100,
      "subnet": "2001:db8:1::/48",
      "shared-network-name": "",
      "pools": [
        {
          "pool": "2001:db8:1::/64"
        }
      ]
    }
  ],
  "server-tags": [
    "remote"
  ],
  "command": "remote-subnet6-set"
}
```

```
{
  "arguments": {
    "subnets": [
      {
        "id": 100,
        "subnet": "2001:db8:1::/48"
      }
    ],
    "result": 0,
    "text": "IPv6 subnet successfully set."
  }
}
```
Get a specific Net by ID

```json
{
    "command":
        "remote-subnet6-get-by-id",
    "subnets": [ { "id": 100 } ]
}
```

```json
{
    "arguments": {
        "count": 1,
        "subnets": [ { "id": 100,  
                        "metadata": {      
                            "server-tags": [   
                                "remote"    
                            ]
                        },
                        "option-data": [],
                        "pd-pools": [],
                        "pools": [ {
                            "option-data": [],
                            "pool": "2001:db8:1::/64"
                        } ]
        } ]
    }
    "result": 0,
    "text": "IPv6 subnet 100 found."
}
```
List All IPv6 Subnets (tag)

```json
{
    "server-tags": [ "all" ],
    "command": "remote-subnet6-list"
}
```

```json
{
    "arguments": {
        "count": 4,
        "subnets": [
            {
                "id": 101,
                "metadata": {
                    "server-tags": [ "all" ]
                },
                "shared-network-name": null,
                "subnet": "2001:db8:2::/48"
            }
            [
                "result": 0,
                "text": "4 IPv6 subnet(s) found."
            ]
```
Create an Option Definition

```json
{
  "command": "remote-option-def4-set",
  "arguments": {
    "option-defs": [
      {
        "name": "foo",
        "code": 222,
        "type": "uint32",
        "array": false,
        "record-types": "",
        "space": "dhcp4",
        "encapsulate": ""
      }
    ],
    "server-tags": [
      "headquarters"
    ]
  }
}
```

```json
[
  {
    "arguments": {
      "option-defs": [
        {
          "code": 222,
          "space": "dhcp4"
        }
      ],
      "result": 0,
      "text": "DHCPv4 option definition successfully set."
    }
  }
]
```
Get an Option Definition

```
{
  "command": "remote-option-def4-get",
  "arguments": {
    "option-defs": [
      {
        "array": false,
        "code": 222,
        "space": "dhcp4"
      }
    ],
    "server-tags": ["headquarters"]
  }
},
"service": ["dhcp4"]
}
```

```
{
  "arguments": {
    "count": 1,
    "option-defs": [
      {
        "array": false,
        "code": 222,
        "encapsulate": "",
        "metadata": {
          "server-tags": ["headquarters"]
        },
        "name": "foo",
        "record-types": "",
        "space": "dhcp4",
        "type": "uint32"
      }
    ],
    "server-tags": ["headquarters"]
  },
  "result": 0,
  "text": "DHCPv4 option definition 222 in 'dhcp4' found."
}
```
Moving to Config Backend

• Moving from a non-database configuration to the Config Backend requires understanding of the existing infrastructure

• Common elements must be pulled out of existing configurations and grouped

• This will be a topic for a future webinar!
Feedback requested

https://gitlab.isc.org/isc-projects/kea/-/boards?label_name[]=config-backend
Questions?

Comments?