Intro to Kea DHCP

For IPv4 and IPv6

Tomek Mrugalski

Mar 2020, APNIC Webinar

About presenter

- MSc (2003), PhD (2010), both about DHCPv6
- 7 years at Intel
- IETF (since 2009)
  - DHC WG co-chair at IETF
  - 11 RFCs published
- ISC (since 2011)
  - As an engineer started the Kea project
  - Currently Director of DHCP engineering
  - Managing ISC DHCP, Kea and Stork projects
- Several RIPE, UKNOF, PLNOG presentations
- Open source enthusiast

Tomek Mrugalski

A bit of history :: ISC DHCP

- In development since 1995
- A very different era after 25 years
- Limited documentation, confusing code, hard to maintain and extend
- Large parts of the code really should be rewritten
- Decided to ***rewrite everything***

ISC DHCP Future

- There’s no EOL date defined
- Currently in maintenance mode
- 4.4.2 released in Jan 2020
- Futures releases **not planned**, but will release when needed (serious security problems, critical bugfixes for customers)
- No new development
- On GitHub now: https://github.com/isc-projects/dhcp/
- Will be handed over to the community on github

If you never heard about Kea…

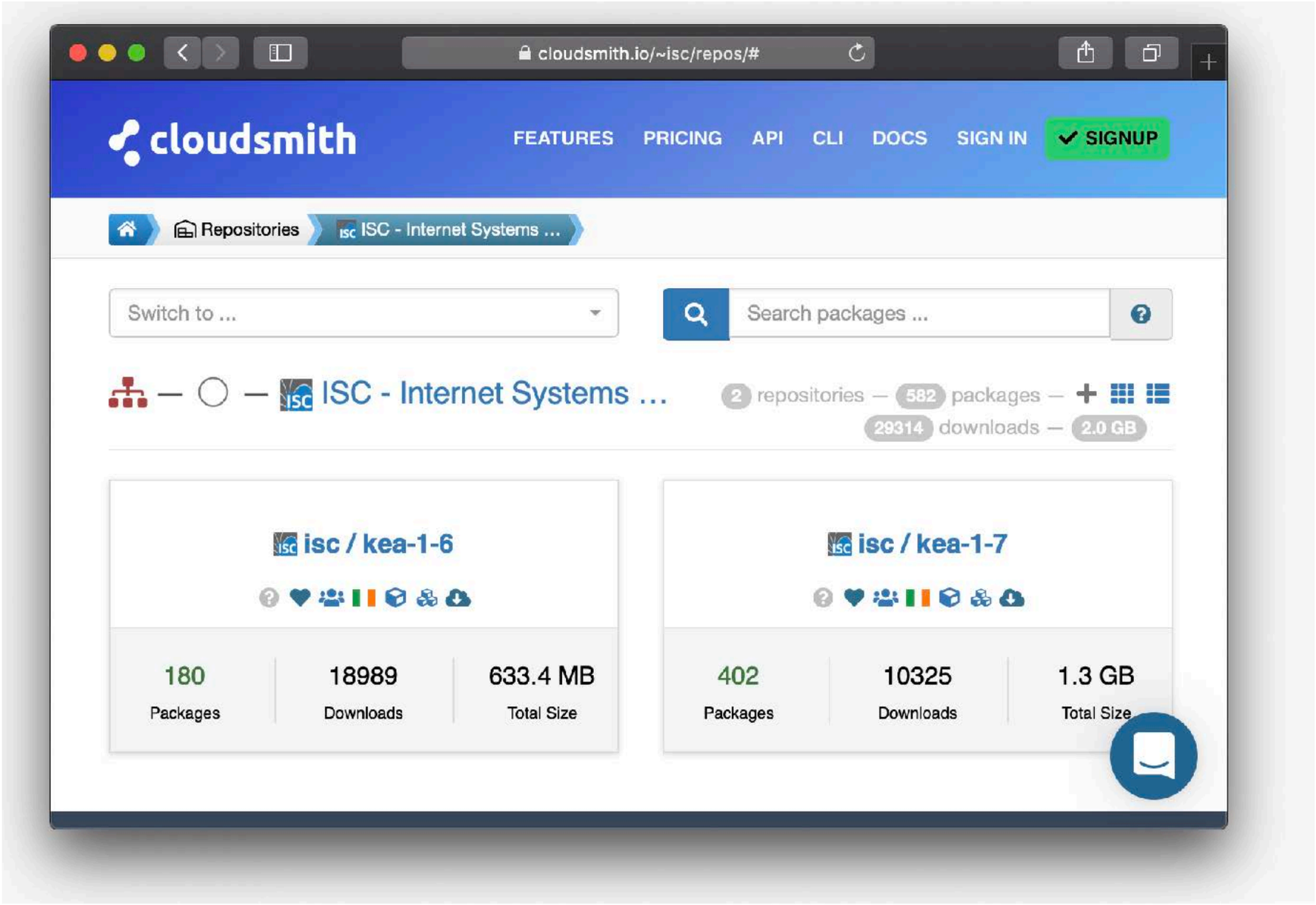
- Modern DHCPv4 and DHCPv6 server (1.6 in Aug 2019)
- Features: **High Availability**, DDNS, NETCONF, RADIUS
- **Database** support (CSV, MySQL, PostgreSQL, Cassandra)
- **Flexible** (leases, host reservations, configuration)
- **Hooks** (optional libraries, including many from ISC)
- **REST API** (140+ commands and counting)
- Linux, BSDs, MacOS, …
- **Open source** (MPL2), with some **paid add-ons** (hooks)
- Commercial support available

Kea vs ISC DHCP Differences

	ISC DHCP	Kea
Performance	OK (with ramdisk tricks)	Much better (many 1000s leases/sec)
Management	OMAPI (custom C interface)	**JSON over REST API/http, JSON over Unix socket**
HA	DHCPv4 failover	**HA for DHCPv4 and DHCPv6, multiple options for DB clustering**
Extensibility	Shell scripts (out only), configuration language	JSON everywhere, Hooks (C++), stable API
Configuration	Custom complex syntax (almost programming language)	JSON with optional DB storage for some elements (more to come)
Leases information	Custom	**CSV, MySQL, PgSQL, Cassandra**
Hosts information	Custom config	**JSON, MySQL, PgSQL**
Multi-core support	No	**Coming soon!**

Kea installation

- Compile from sources
- Native packages: Ubuntu, Fedora, CentOS, Debian, …



A basic Kea config

```
{
"Dhcp4": {
  "interfaces-config": {
      "interfaces": [ "eth0" ]
  },

  "lease-database": {
      "type": "memfile",
      "lfc-interval": 3600
  },

  "valid-lifetime": 4000,

  "subnet4": [ {
       "pools": [ { "pool":  "192.0.2.1 - 192.0.2.200" } ],
       "subnet": "192.0.2.0/24",
       "interface": "eth0"
  }],

  "loggers": [ {
        "name": "kea-dhcp4",
        "output_options": [ {
            "output": "/var/log/kea4.log" } ],
        "severity": "INFO"
  } ],
  ...
} }
```

Network interfaces

DB credentials

Global parameters
(e.g. lease lifetime)

Topology of your network
(subnets, shared networks)

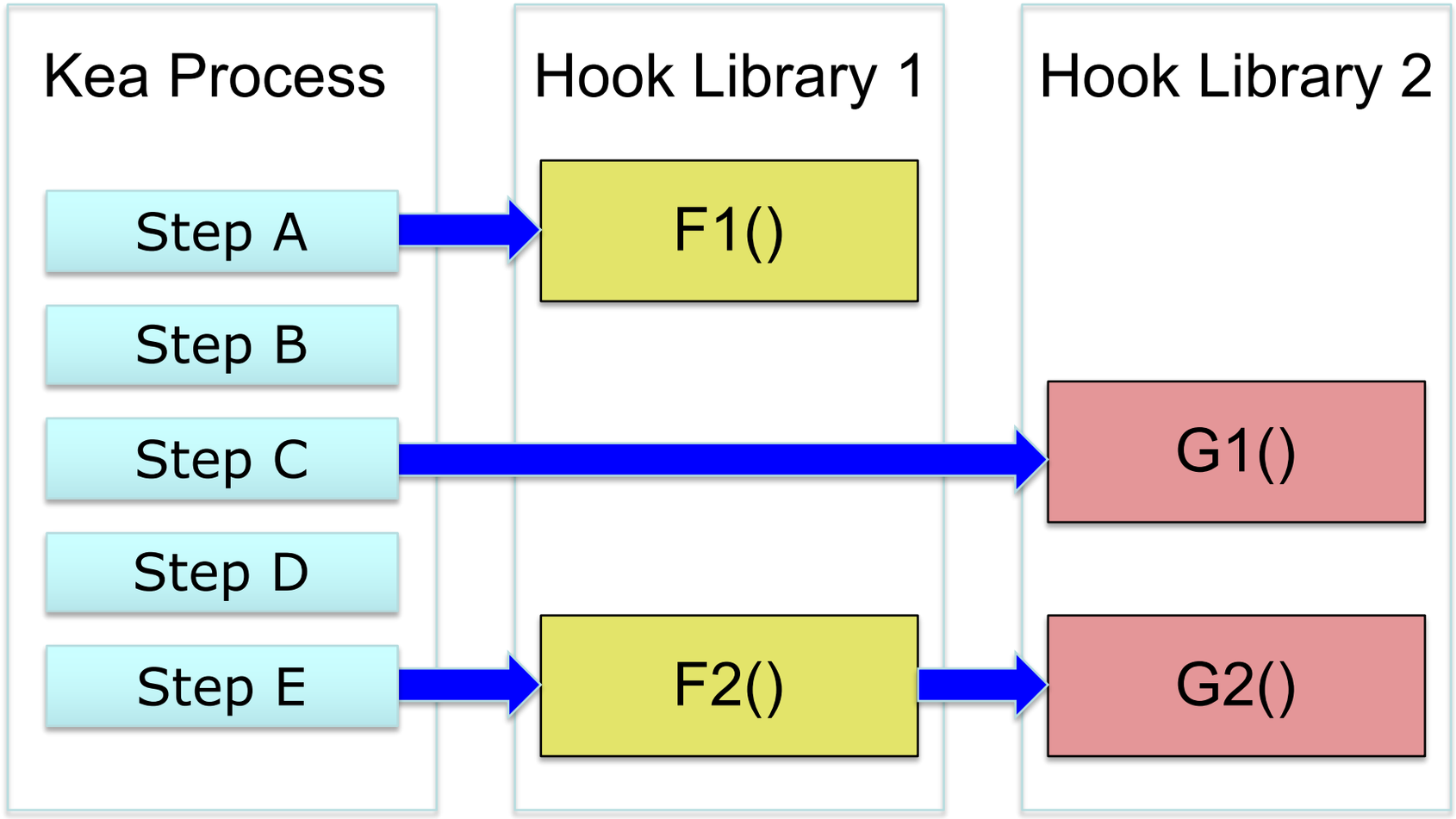
Logging details

Other usual Kea parameters (see Kea ARM)

Hooks

Hooks overview

- Kea has lots of features, but rarely all of them are used
- Kea adopted the approach of reasonably small core with extensions
- Hook points
- Multiple libraries
- Bi-directional data flow (hooks can get data, influence Kea)



Hooks (1 of 2)

- 1.1: **User Check** – example access control
- 1.2: **Forensic Logging** – audit trail for legal purposes
- 1.2: **Flexible Identifier** – identify hosts by expression
- 1.2: **Host Commands** – query, add and delete host reservations using REST interface
- 1.3: **Subnet management** (add, get, update, delete subnets and shared networks via REST API)
- 1.3: **Lease commands** (add, get, update, delete, wipe all, get all leases via REST API)

Open source

Premium ($)

Hooks (2 of 2)

- 1.4: **HA** – high availability solution (heartbeat, failure detection, lease updates, recovering DB from partner)
- 1.4: **Radius** – access control and host reservation using FreeRadius, accounting
- 1.4: **DB Statistics** – Multiple servers sharing DB
- 1.4: **Host Caching** – cache host responses locally from slower backends for extra performance (includes negative caching)
- 1.5:**Class commands** – extra API for classification management
- 1.6: **Config Backend for MySQL** – Config Backend
- 1.6: **Config Backend commands** – Config Backend management
- 1.7.1: **Flex Options** – Option values defined by expression
- 1.7.3: **BOOTP** – support for legacy devices

Open source

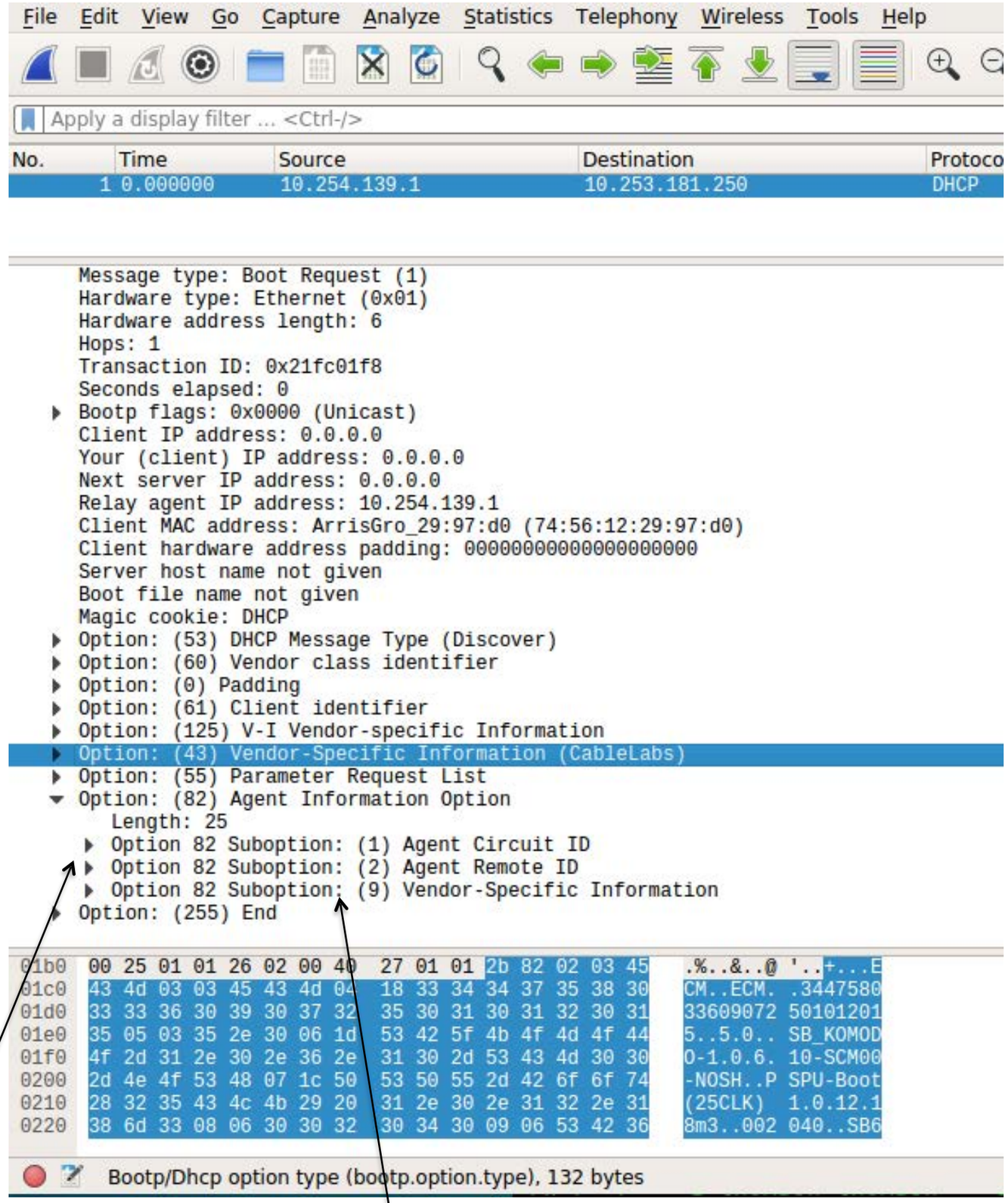
Premium ($)

12

Flex-id (1.2)

- Flexible Identifier

  How to identify hosts:
- **Open source**
- MAC, duid, circuit-id, client-id
- **Premium**
- Almost anything could be used (40+ different expressions)
- Options (client, relay, vendor)
- Fixed fields
- Concat, substring
- Meta-data (interface name, src/dst IP, …)



concat(relay4[1].hex, relay4[2].hex)

Databases

Database backends

- SQL data can be modified any time
- All changes applied instantly (no restart)
- Adapt your provisioning systems to write directly to the database or
- Use the api (some of these require premium hooks libraries)

PostgreSQL

The backend concept

DHCPv4, DHCPv6 server

MySQL

Changing: Often → Rarely

- Leases (addresses, prefixes)

Lease backend

- Host reservations (per host details)

Hosts backend

- Options
- Pools
- Subnets
- Shared networks
- Option definitions
- Global parameters

Configuration backend

The lease backend

- Run-time state (the fastest changing data)
- Most susceptible to performance gain/loss
- **Memfile** (custom developed, C++, in-memory/disk) is faster by an order of magnitude (11000 lps)
- **MySQL**, **PostgreSQL** comparable (500-2000 lps)
- Cassandra by far slowest* (~200 lps)

```
"Dhcp4": {
    "lease-database": {
        "type": "memfile",
        "name": "/tmp/kea-leases4.csv",
        "lfc-interval": 1800,
        "max-row-errors": 100
    },
    . . .
}
```

The hosts backend

- Per device information
- Config file, MySQL, PostgreSQL, Cassandra
- Use config file, if you have few, rarely changing hosts
- Use DB if having many, frequently changing hosts (customers)
- Many configuration knobs (reservation-mode, host-reservation-identifiers, flex-id)
- Host Cmds available ($)

DB init

```
$ kea-admin db-init mysql ...
```

Config example

```
"Dhcp6": {
    "host-reservation-identifiers": [
       "circuit-id", "hw-address",
       "duid", "client-id", "flex-id" ],

    "hosts-database": {
        "type": "mysql",
        "name": "kea",
        "user": "kea",
        "password": "secret123",
        "host": "localhost",
        "port": 3306
    },
    . . .
}
```

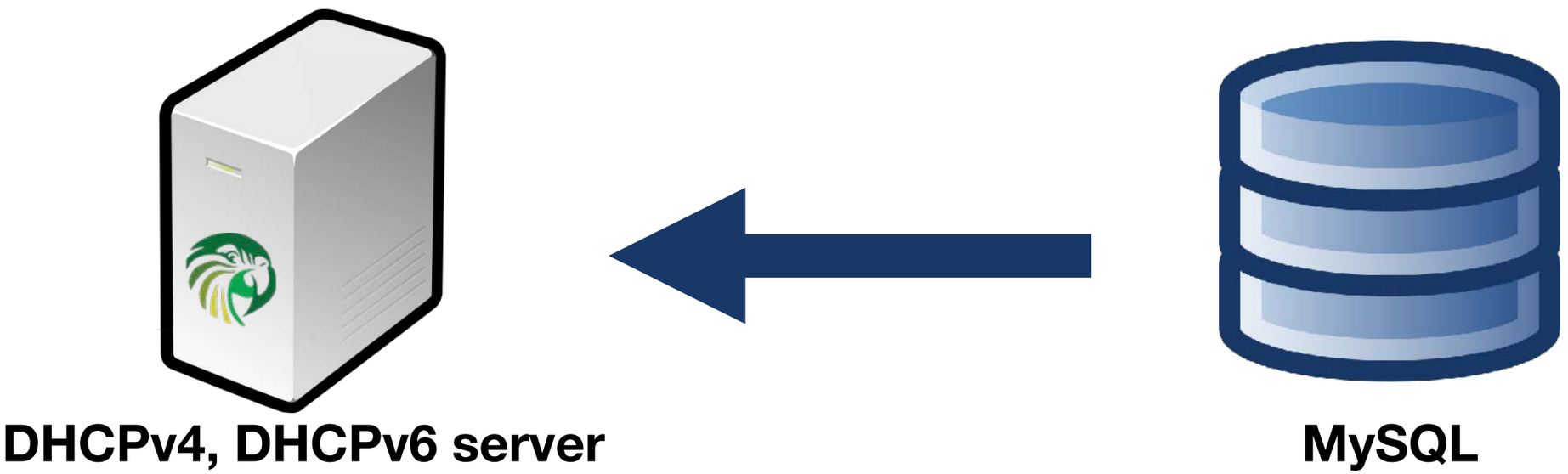
Reservation examples

```
{
    "hw-address": "00:11:22:33:44:55",
    "ip-address": "192.0.2.204",
    "hostname": "printer-floor1",
    "option-data": [
        {
            "name": "vivso-suboptions",
            "data": "4491"
        },
        {
            "name": "tftp-servers",
            "space": "vendor-4491",
            "data": "10.1.1.202,10.1.1.203"
        }
    ],
    "client-classes": [ "cgn-class1", "silver" ]
}
```

```
{
    "client-id": "01:0a:0b:0c:0d:0e:0f",
    "ip-address": "192.0.2.205",
}
```

```
{
    "flex-id": "s0mEVaLue",
    "ip-address": "192.0.2.206"
}
```

Config Backend



- Colocate or remote
- Multiple Kea servers can share one MySQL DB
- Works when the DHCP servers are on-line or off-line
- Sharing configuration between HA partners
- Frequently changing configuration (options, pools, subnets, shared networks)
- Automated configuration deployment
- Large configuration (100+ subnets)
- Large scale deployments (many DHCP servers)
- Scaling up or down (add new or delete not needed VMs)

Server tags

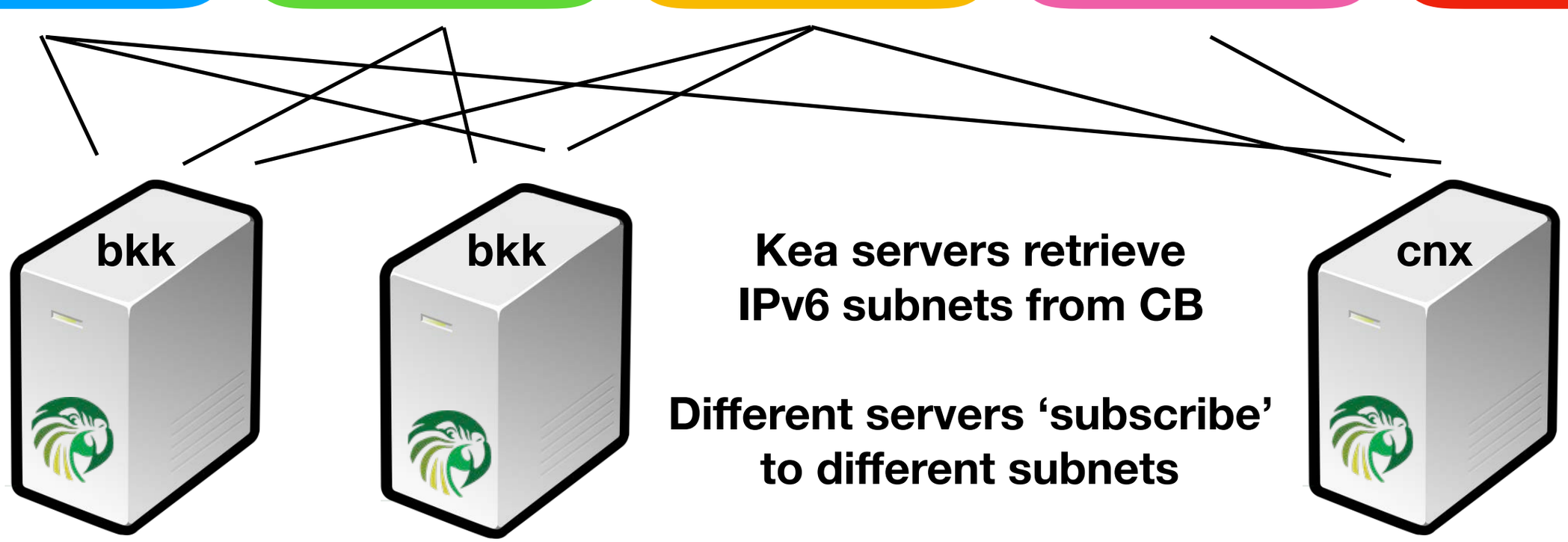
id: 100,
server-tags:
[“all”]

id: 101,
server-tags:
[“bkk”]

id: 102,
server-tags:
[“bkk, “cnx”]

id: 103,
server-tags:
[“cnx”]

id: 104,
server-tags: [ ]



Kea servers retrieve IPv6 subnets from CB

Different servers ‘subscribe’ to different subnets

Enabling CB

A sample /etc/kea/kea-dhcp6.conf configuration file:

```
"Dhcp6": {
    "config-control": {
        "config-databases": [{
            "type": "mysql",
            "name": "kea",
            "user": "kea",
            "password": "secret1",
            "host": "192.0.2.1",
            "port": 3302
        }],
        "config-fetch-wait-time": 20
    },
    "hooks-libraries": [{
        "library": "/opt/kea/hooks/libdhcp_mysql_cb.so"
    }, {
        "library": "/opt/kea/hooks/libdhcp_cb_cmds.so"
    }],
    ...
}
```

DB credentials

Refresh interval

CB hook
(tells Kea to look at the DB for config)

CB commands hook
(tells Kea to expose JSON-based REST API), optional ($)

Other usual Kea parameters (see Kea ARM)

HA

HA: High Availability

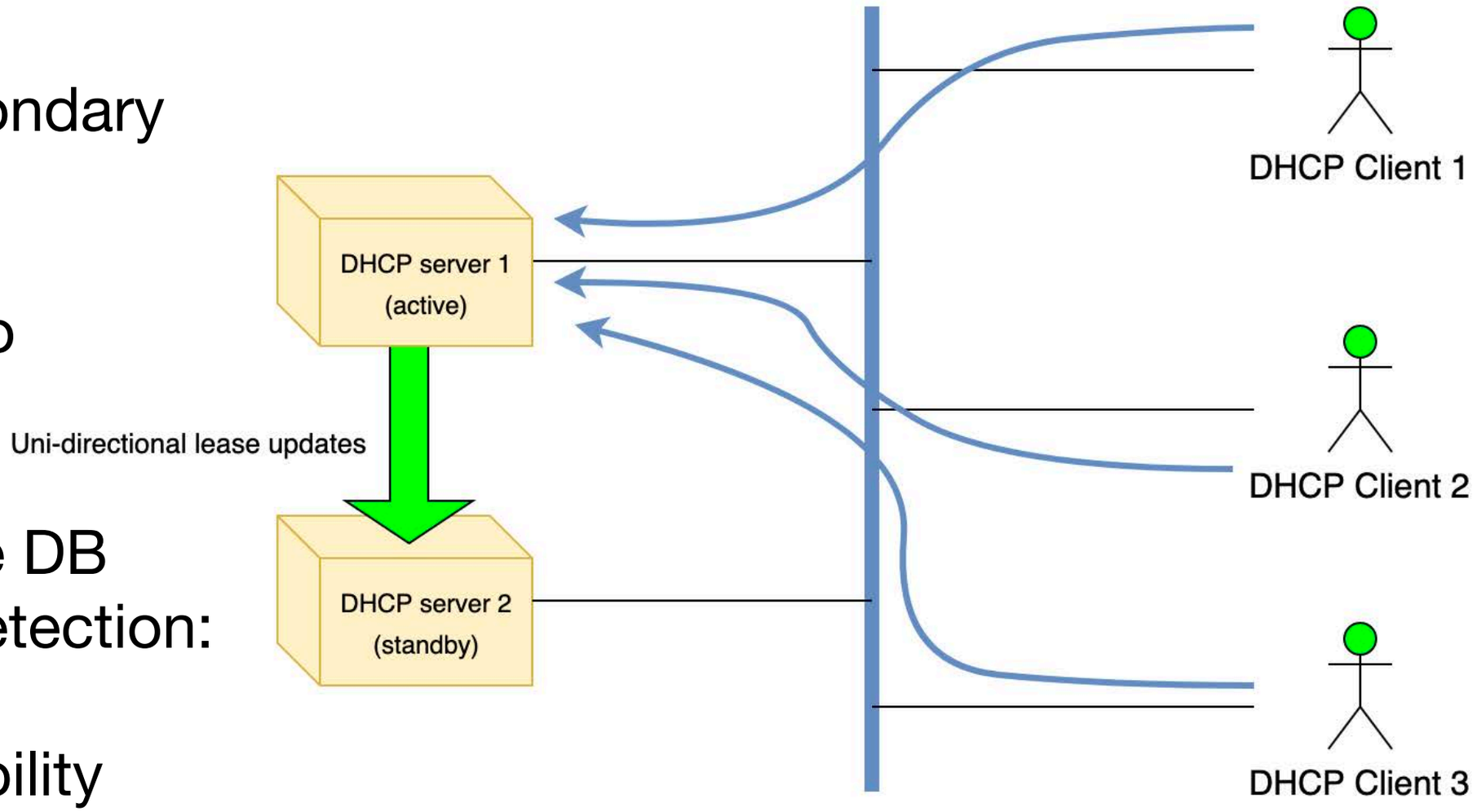
- **more than one server => Resilience to software/hardware failures**
- **Choice 1: Mulitple servers sharing the same DB**
  - DB redundancy needed, otherwise just swapping one single point of failure for another
- **Choice 2: HA**
  - Works for every backend (including DB)

HA vs Failover

- **DHCP Failover in ISC DHCP**
  - IPv4 only
  - The standardization work on the DHCPv4 failover draft was never completed in IETF; as such, it is not a standard.
  - When ISC DHCP was in active development, there was no IETF draft for DHCPv6 failover. There is now (see RFC 8156), but it was never implemented.
  - The failover is complex.
    - MCLT, loose coherency between client, server and partner
    - Failover is asynchronous
    - 2 servers (a pair)
    - Pool rebalancing
    - ping before use
- **HA in Kea**
  - IPv4 and IPv6
  - HA is synchronous
  - HA has fewer config knobs
  - easy to understand (you can even interact on your own)
  - 2 servers (a pair) + optional backup servers (unlimited*)

HA: Hot-standby

- Primary and secondary
- Active handles
  100% traffic
- Sends updates to
  secondary
- Secondary has
  100% up-to-date DB
- Primary failure detection:
  - HA heartbeat
  - connection stability
  - clients information (elapsed field)
  - administrative action
- Additional backup servers possible
  - Numbers not limited, but each decrease performance



HA: Hot-standby example

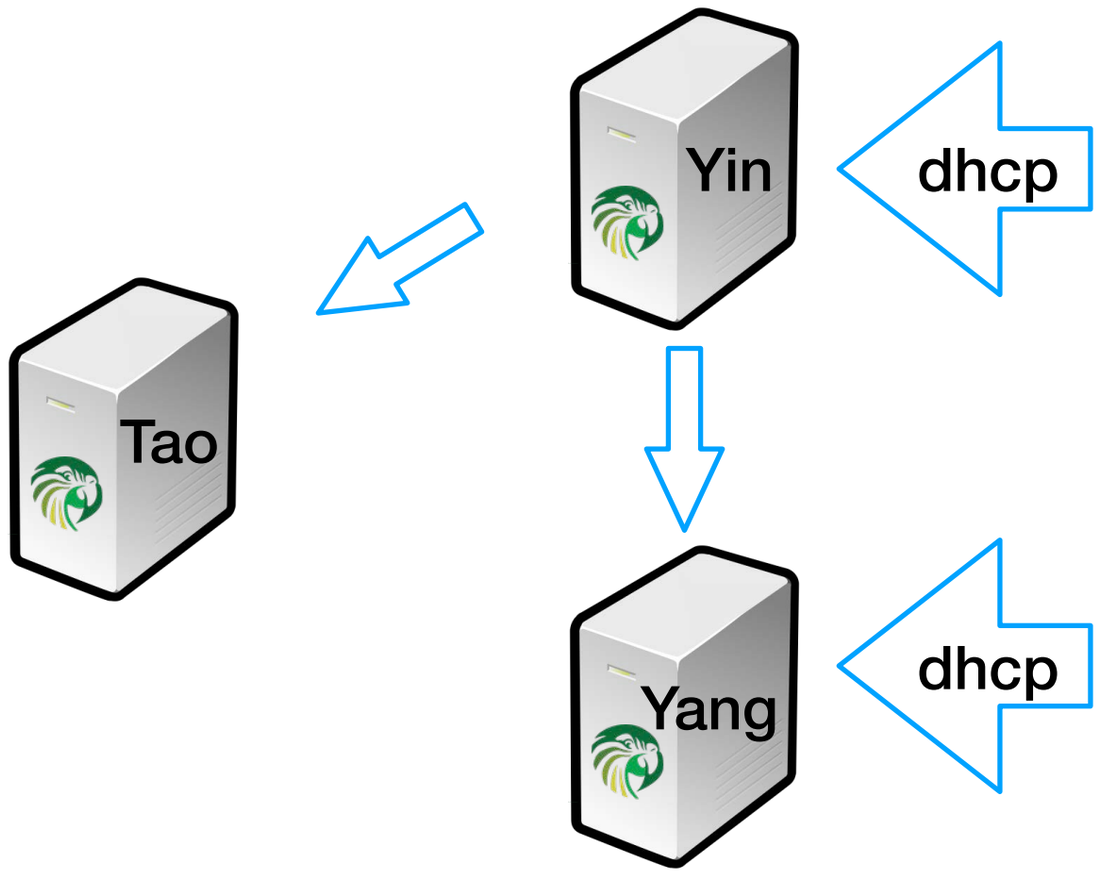
```
"Dhcp4": {
    "hooks-libraries": [{
        "library": "/usr/lib/kea/hooks/libdhcp_lease_cmds.so",
    }, {
        "library": "/usr/lib/kea/hooks/libdhcp_ha.so",
        "parameters": {
            "high-availability": [{
                "this-server-name": "yin",
                "mode": "hot-standby",

                "heartbeat-delay": 10000,
                "max-response-delay": 10000,
                "max-ack-delay": 5000,
                "max-unacked-clients": 5,

                "peers": [{
                    "name": “yin",
                    "url": "http://192.168.56.33:8000/",
                    "role": "primary",
                    "auto-failover": true
                }, {
                    "name": "yang",
                    "url": "http://192.168.56.66:8000/",
                    "role": "standby",
                    "auto-failover": true
                }, {
                    "name": “tao",
                    "url": "http://192.168.56.99:8000/",
                    "role": "backup",
                    "auto-failover": false
                }]
            }]
        }
    }],
```

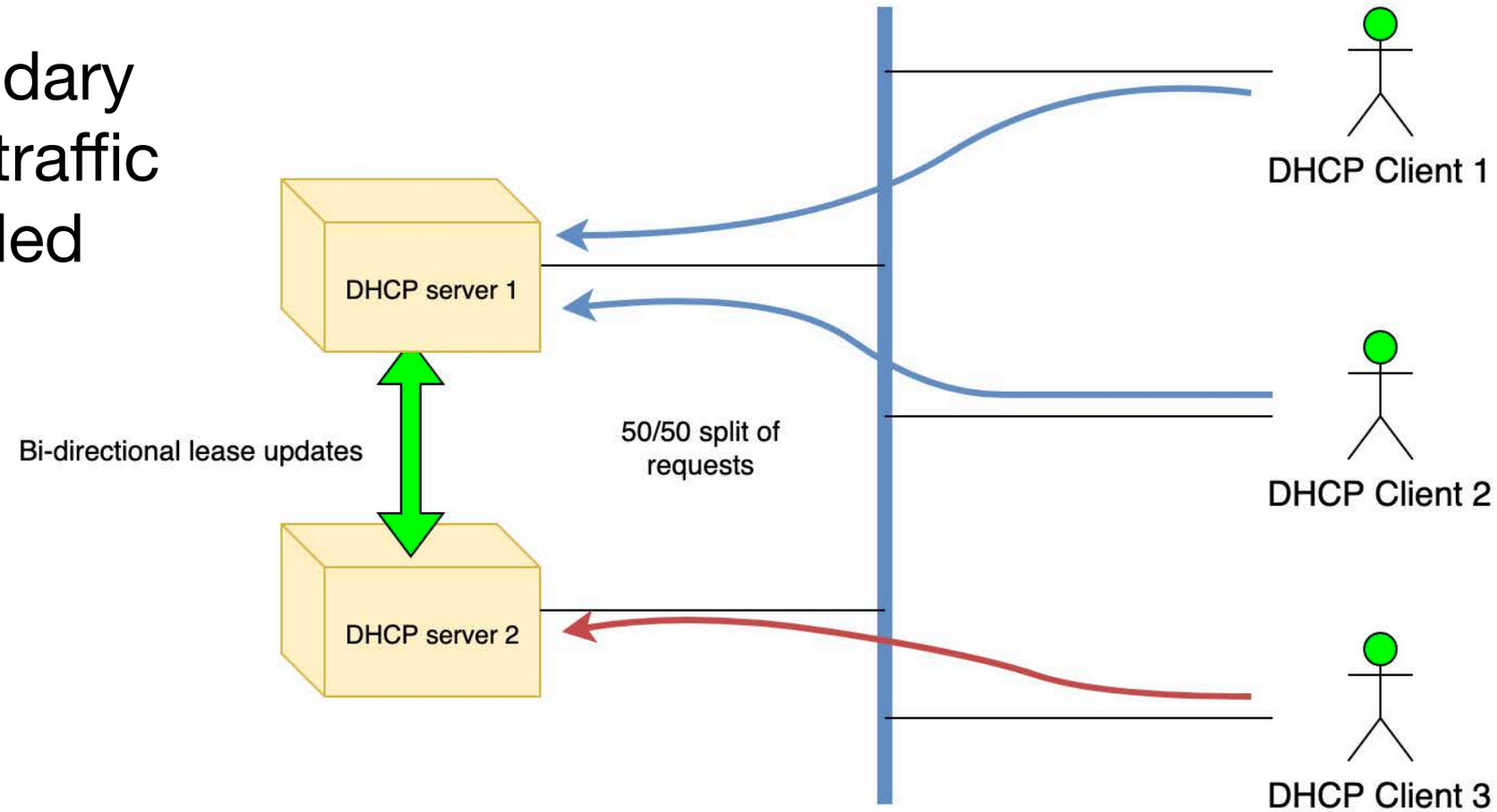
```
    "subnet4": [{
        "subnet": "192.0.3.0/24",
        "pools": [{
            "pool": "192.0.3.100 - 192.0.3.250",
            "client-class": “HA_yin"
        }],

        "option-data": [{
            "name": "routers",
            "data": "192.0.3.1"
        }],
        . . .
    }]
}
```



HA: Load Balancing

- Primary and secondary
- Each handle 50% traffic
- Classification needed



- Additional backup servers possible
  - Numbers not limited, but each decrease performance

HA: Load Balancing example

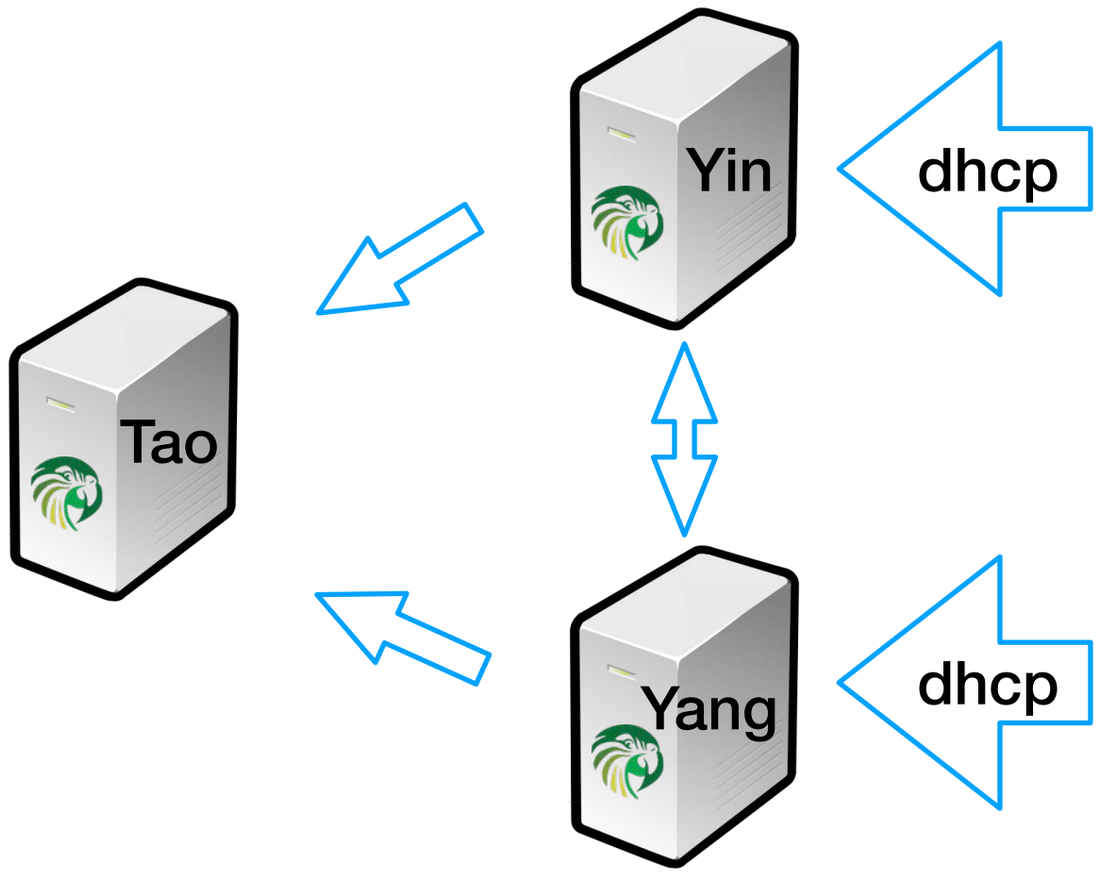
```
"Dhcp4": {
    "hooks-libraries": [{
        "library": "/usr/lib/kea/hooks/libdhcp_lease_cmds.so",
    }, {
        "library": "/usr/lib/kea/hooks/libdhcp_ha.so",
        "parameters": {
            "high-availability": [{
                "this-server-name": "yin",
                "mode": “load-balancing”,

                "heartbeat-delay": 10000,
                "max-response-delay": 10000,
                "max-ack-delay": 5000,
                "max-unacked-clients": 5,

                "peers": [{
                    "name": “yin",
                    "url": "http://192.168.56.33:8000/",
                    "role": "primary",
                    "auto-failover": true
                }, {
                    "name": "yang",
                    "url": "http://192.168.56.66:8000/",
                    "role": "standby",
                    "auto-failover": true
                }, {
                    "name": “tao",
                    "url": "http://192.168.56.99:8000/",
                    "role": "backup",
                    "auto-failover": false
                }]
            }]
        }
    }],
```

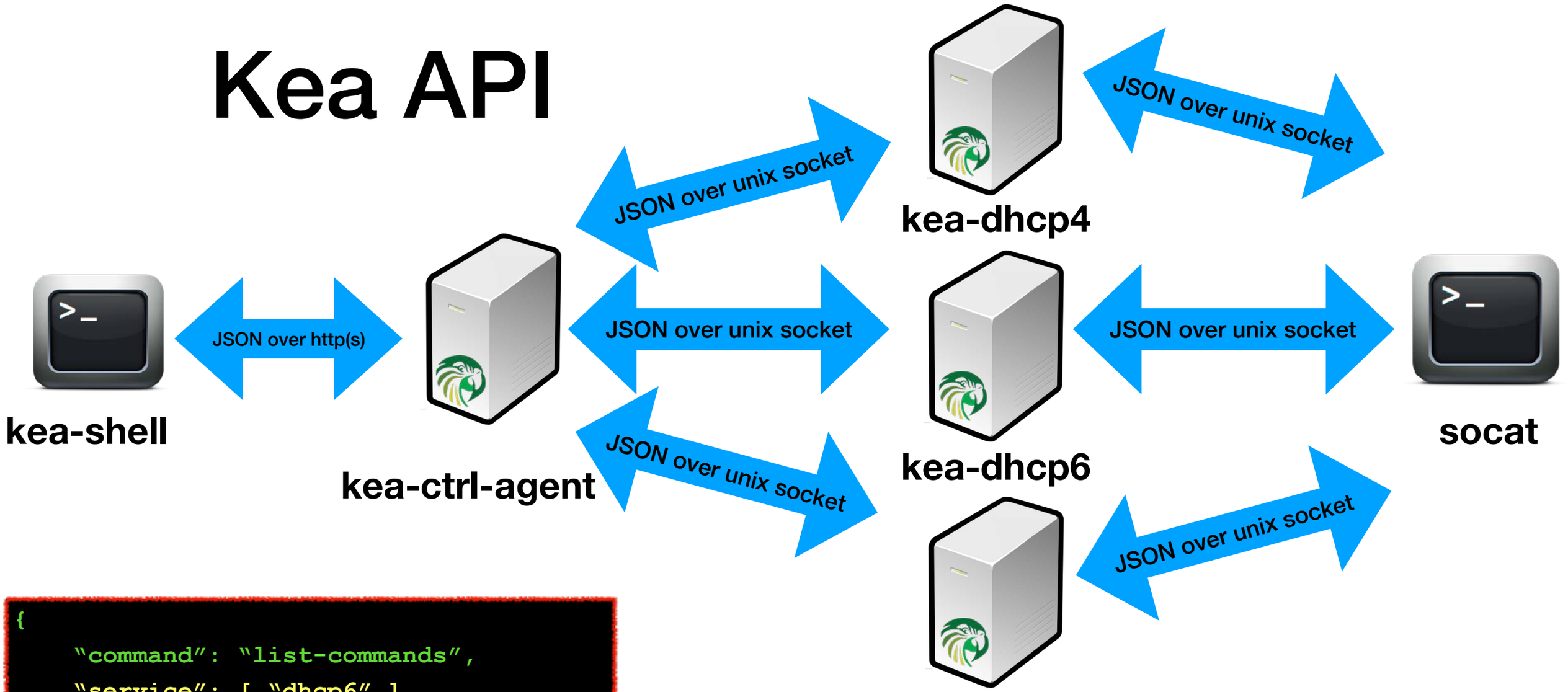
```
    "subnet4": [{
        "subnet": "192.0.3.0/24",
        "pools": [{
            "pool": "192.0.3.100 - 192.0.3.150",
            "client-class": "HA_yin"
         }, {
            "pool": "192.0.3.200 - 192.0.3.250",
            "client-class": "HA_yang"
         }],

        "option-data": [{
            "name": "routers",
            "data": "192.0.3.1"
        }],
        . . .
    }]
}
```



API

Kea API




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

Command

```
{
  "arguments": [
    "build-report",
    "config-get",
    . . .
  ],
  "result": 0
}
```

Response

- JSON in, JSON out
- 140+ commands supported
- New commands every release
- Some provided by hooks (optional libs)

API :: Basics

1. Send **`list-commands`** command:

```
# kea-shell --host ::1  --port 8080 --service dhcp6 list-commands
^D
```

2. Get list of currently supported commands in return:

```
{
    “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”,
    "server-tag-get",
    "shutdown",
    “statistic-{get,remove,reset}“,
    “statistic-{get,remove,reset}-all”,
    "version-get"
  ],
  "result": 0
}
```

API :: List IPv6 subnets

- Send **`list-commands`** command:

```
# echo ' "server-tags": [ "all" ] ' | \
  kea-shell --host ::1 --port 8000 --service dhcp6 remote-subnet6-list
```

- Get list of currently supported commands in return:


```
[
  {
    "arguments": {
      "count": 0,
      "subnets": [

      ]
    },
    "result": 3,
    "text": "0 IPv6 subnet(s) found."
  }
]
```

API :: Add new IPv6 subnet

- Send **`remote-subnet6-set`** command:

```
echo ' "subnets": [ { "id": 100, "subnet": "2001:db8:1::/48", "shared-network-name": "",
"pools": [ { "pool": "2001:db8:1::/64" } ] } ], "server-tags": [ "all" ] ' |
kea-shell --host ::1 --port 8000 --service dhcp6 remote-subnet6-set
```

- Get list of currently supported commands in return:

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

```
{
  "arguments": {
    "subnets": [
      {
        "id": 100,
        "subnet": "2001:db8:1::/64"
      }
    ]
  },
  "result": 0,
  "text": "IPv6 subnet successfully set."
}
```

API :: Get IPv6 subnet details

- Send **`remote-commands`** command:

```
echo ' "subnets": [ { "id": 100 } ] ' | \
kea-shell --host ::1 --port 8000 --service dhcp6 remote-subnet6-get-by-id
```

- Get list of currently supported commands in return:


```
{
  "arguments": {
    "count": 1,
    "subnets": [
      {
        "id": 100,
        "metadata": {
          "server-tags": [ "all" ]
        },
        "option-data": [ ],
        "pd-pools": [ ],
        "pools": [ {
            "option-data": [ ],
            "pool": "2001:db8:1::/64"
          } ],
        "relay": {
          "ip-addresses": [ ]
        },
        "shared-network-name": null,
        "subnet": "2001:db8:1::/48"
      }
    ]
  },
  "result": 0,
  "text": "IPv6 subnet 100 found."
}
```

Kea-dhcp6 logs

- Send **`remote-subnet6-set`** command:

```
echo ' "subnets": [ { "id": 100 } ] ' | \
kea-shell --host ::1 --port 8000 --service dhcp6 remote-subnet6-get-by-id
```

- Kea's reaction:

```
14:43:07.083 INFO DCTL_CONFIG_FETCH Fetching configuration data from config backends.
14:43:27.140 INFO DCTL_CONFIG_FETCH Fetching configuration data from config backends.
14:43:47.200 INFO DCTL_CONFIG_FETCH Fetching configuration data from config backends.
14:43:49.222 INFO COMMAND_RECEIVED Received command 'remote-subnet6-set'
14:43:49.228 INFO DHCPSRV_CFGMGR_NEW_SUBNET6 a new subnet has been added to
configuration: 2001:db8:1::/48 with params  preferred-lifetime=0, valid-lifetime=0,
rapid-commit is disabled
14:43:58.784 INFO COMMAND_RECEIVED Received command 'remote-subnet6-list'
14:44:07.820 INFO DCTL_CONFIG_FETCH Fetching configuration data from config backends.
14:44:07.832 INFO DHCPSRV_CFGMGR_CONFIG6_MERGED Configuration backend data has been
merged.
14:44:27.889 INFO DCTL_CONFIG_FETCH Fetching configuration data from config backends.
14:44:47.934 INFO DCTL_CONFIG_FETCH Fetching configuration data from config backends.
14:45:07.985 INFO DCTL_CONFIG_FETCH Fetching configuration data from config backends.
```

Migration Assistant

ISC DHCP to Kea Migration

- Modified ISC DHCP to produce Kea config
- Made public with ISC DHCP 4.4.2
- The configuation syntax is very different
- Some concepts work differently
  (client classes, identifying hosts, …)

```
wget -nd https://ftp.isc.org/isc/dhcp/4.4.2/dhcp-4.4.2.tar.gz
tar zxvf dhcp-4.4.2.tar.gz
cd dhcp-4.4.2
./configure
make
cd keama
make
```

Translating Configuration

```
NAME
       keama - Kea Migration Assistant

SYNOPSIS
       keama  [ -4 | -6] [ -N ] [ -r {perform|fatal|pass} ] [ -l hook-library-
       path ] [ -i input-file ] [ -o output-file ]

DESCRIPTION
       The Kea Migration Assistant converts an  ISC  DHCP  configuration  file
       into the corresponding Kea configuration file.

COMMAND LINE
       Protocol selection options:

       -4     The input configuration is for DHCPv4.  Incompatible with the -6
              option.

       -6     The input configuration is for DHCPv6.  Incompatible with the -4
              option.

       -N     Instead  of  using  global  host  reservations,  put them in the
              matching subnet.
```

A Small Network

```
option domain-name "boat";
option domain-name-servers 44.127.8.1;
default-lease-time 600;
max-lease-time 7200;
authoritative;

subnet 44.127.8.0 netmask 255.255.255.0 {
   range 44.127.8.128 44.127.8.249;
   option routers 44.127.8.1;
}

host roku {
    hardware ethernet 1c:1e:e3:9b:48:83;
    option host-name "roku-tv";
    fixed-address 44.127.8.2;
}
```

keama -4 -i home.conf -o home.kea

A Small Network

- Much more verbose
  - Input: 16 lines
  - Output: 70 lines
- And… some issues.

A Small Network

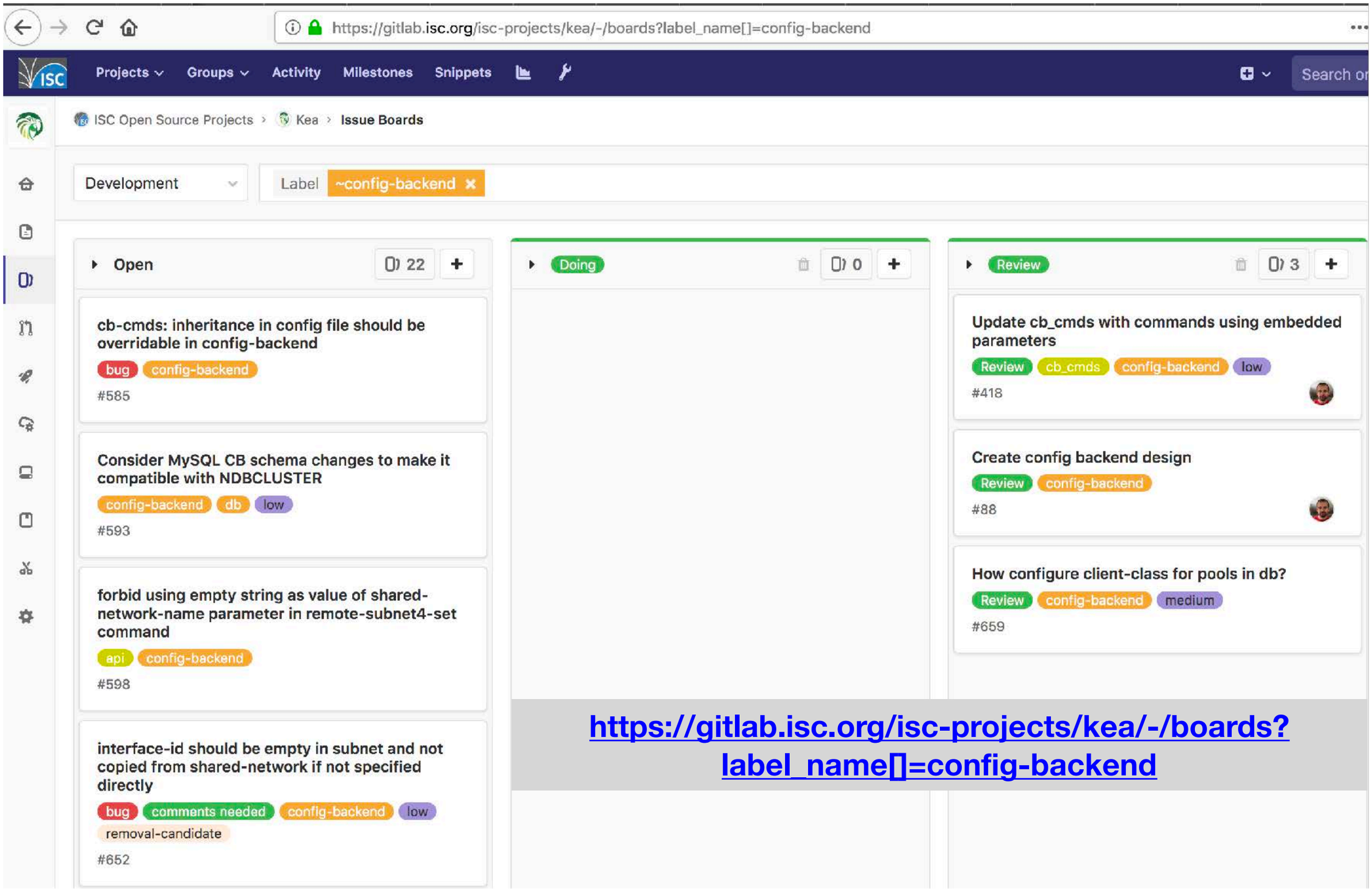
```
{
  /// This configuration declares some subnets but has no interfaces-config
  /// Reference Kea #245
  "Dhcp4": {
    "option-data": [
      {
        "space": "dhcp4",
        "name": "domain-name",
        "code": 15,
        "data": "boat"
      },
      {
        "space": "dhcp4",
        "name": "domain-name-servers",
        "code": 6,
        "data": "44.127.8.1"
      }
    ],
    "valid-lifetime": 600,
```

https://gitlab.isc.org/isc-projects/kea/issues/245

Migrating to Kea

- Different configuration syntax
- Migration Assistant available
- Configuration only, not leases
- ISC webinar
  https://www.isc.org/presentations/
- NANOG’76 talk
  https://pc.nanog.org/static/published/meetings//NANOG76/daily/day_2.html#talk_1998

Feedback requested



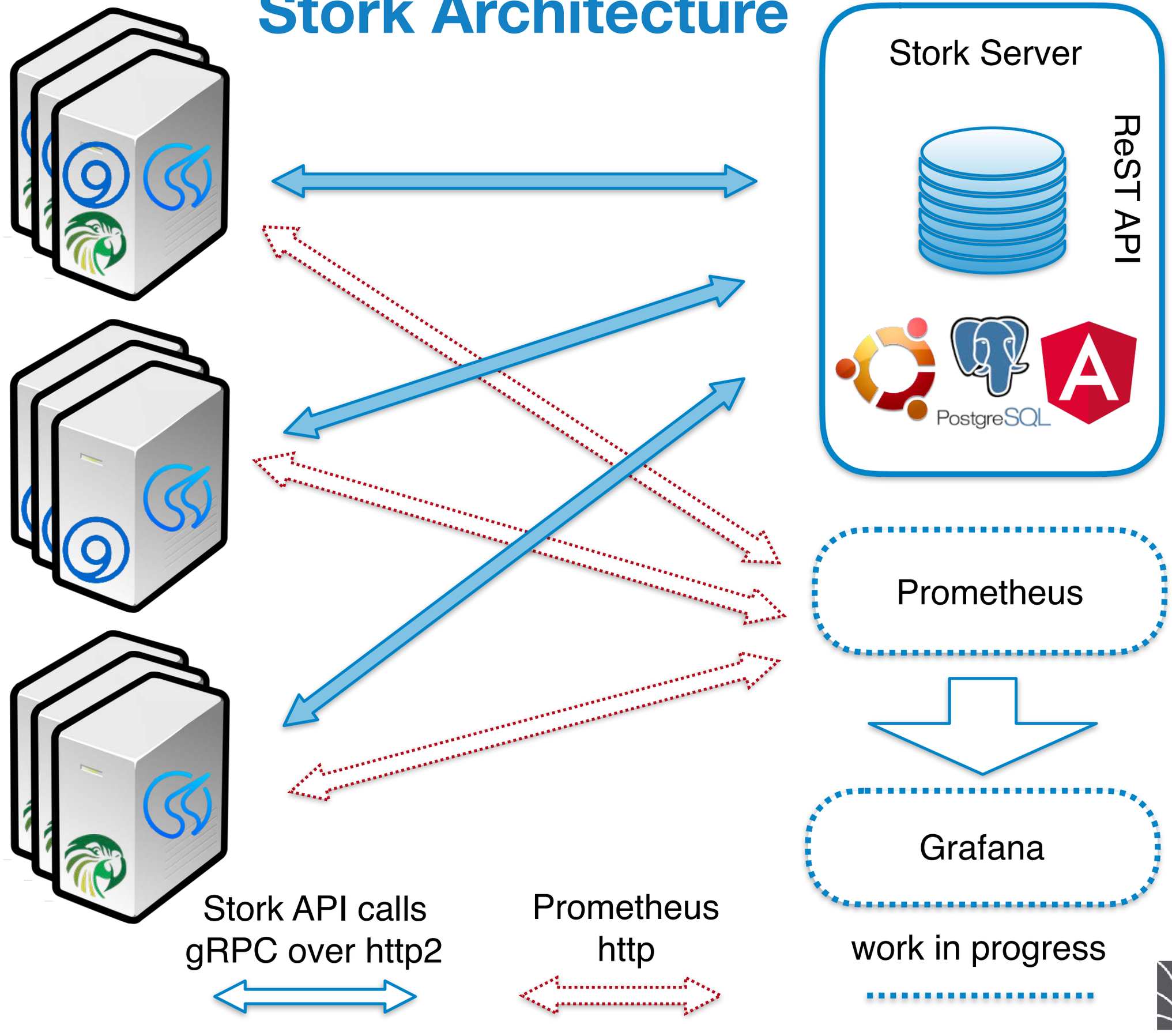
https://gitlab.isc.org/isc-projects/kea/-/boards?label_name[]=config-backend

ISC Stork

What is Stork?

- A **dashboard** for now
- Evolution towards **IPAM**
- **Kea** & **BIND 9** (leaning towards Kea for now)
- Open source
- Central server + agents
- Rapidly **getting new features** monthly
  - 0.5 available
  - 0.6 coming up early April
- Linux for now, FreeBSD coming soon

Stork Architecture



Stork :: HA status

Machine: agent-kea-ha1

DHCPv4 ✓ | DHCPv6 ! | DDNS ! | CA ✓

Overview

Version	1.7.4
Version Ext	1.7.4 tarball linked with: log4cplus 1.1.2 OpenSSL 1.1.1 11 Sep 2018 database: MySQL backend 9.1, library 5.7.29 PostgreSQL backend 6.0, library 100010 Memfile backend 2.1
Hooks	/usr/lib/x86_64-linux-gnu/kea/hooks/libdhcp_lease_cmds.so /usr/lib/x86_64-linux-gnu/kea/hooks/libdhcp_ha.so
Uptime	6 minutes 19 seconds
Last Reloaded At	2020-03-10 12:19:50

Subnet ID	Subnet	Pools
1	192.0.3.0/24	192.0.3.1-192.0.3.200
1	192.0.3.0/24	192.0.3.1-192.0.3.200

1 of 1 pages | 1 | 10 | Total: 1 subnet

High Availability

Local server		Remote server (8 seconds ago)	
State:	*hot-standby*	State:	*hot-standby*
Role:	*primary*	Role:	*standby*
Scopes served:	*server1*	Scopes served:	*(none)*

Note

The local server responds to the entire DHCP traffic.

Stork :: Pool Utilization

Stork DHCP Services Configuration Profile

DHCP Subnets

Filter subnets: subnet or any other field Protocol: any

Subnet ID	Subnet	Addresses			Pools
		Total	Assigned	Used %	
11	192.0.5.0/24	50	0	0%	192.0.5.1-192.0.5.50
12	192.0.6.0/24	110	0	0%	192.0.6.1-192.0.6.40 192.0.6.61-192.0.6.90 192.0.6.111-192.0.6.150
13	192.0.7.0/24	50	45	90% ⚠	192.0.7.1-192.0.7.50
14	192.0.8.0/24	50	30	60%	192.0.8.1-192.0.8.50
15	192.0.9.0/24	50	0	0%	192.0.9.1-192.0.9.50
21	192.1.15.0/24	50	37	74%	192.1.15.1-192.1.15.50
22	192.1.16.0/24	150	0	0%	192.1.16.1-192.1.16.50 192.1.16.51-192.1.16.100 192.1.16.101-192.1.16.150
23	192.1.17.0/24	245	245	100% ❗	192.1.17.1-192.1.17.20 192.1.17.21-192.1.17.40 192.1.17.41-192.1.17.60 192.1.17.66-192.1.17.80 192.1.17.101-192.1.17.120 192.1.17.121-192.1.17.140 192.1.17.141-192.1.17.160 192.1.17.161-192 192.1.17.201-192.1.17.220 192.1.17.221-192.1.17.240 192.1.17.241-192.1.17.243 192.1.17.244-192
1	192.0.2.0/24	200	200	100% ❗	192.0.2.1-192.0.2.50 192.0.2.51-192.0.2.100 192.0.2.101-192.0.2.150 192.0.2.151-192.0.2.200
1	4001:db8:1::/64	9.0 P	0	0%	4001:db8:1:0:abcd::-4001:db8:1:0:ffff:ffff:ffff:ffff

1 of 2 pages 1 2 10 Total: 18 subnets

Stork DHCP Services Configuration Profile

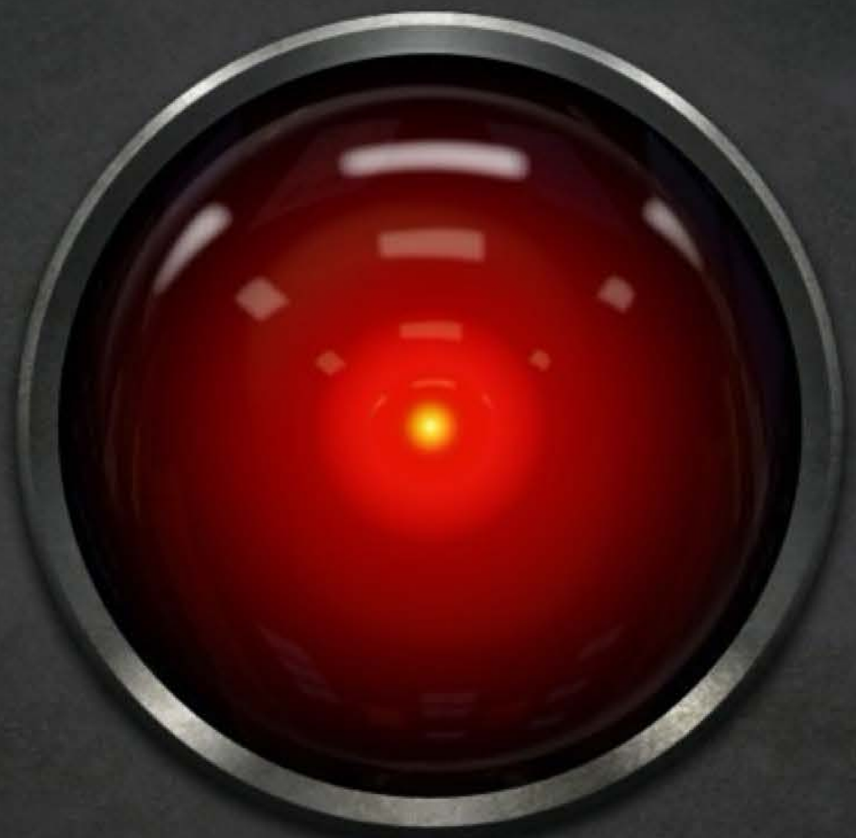
DHCP Shared Networks

Filter networks: network or any other field Protocol: any

Name	Addresses			Subnets
	Total	Assigned	Used %	
frog	310	45	14 %	192.0.5.0/24 192.0.6.0/24 192.0.7.0/24 192.0.8.0/24 192.0.9.0/24
mouse	445	301	67 %	192.1.15.0/24 192.1.16.0/24 192.1.17.0/24
elephant	10.1 P	0	0 %	4001:db8:1::/64 3000::/64 3001::/72 3002::/80 3003::/88 3004::/96 3005::/104

1 of 1 pages 1 10 Total: 3 shared networks

Stork :: Grafana Integration



kea.isc.org