Kea Webinar

Database and High-Availability options

Carsten Strotmann

28th October 2020

https://www.isc.org
Welcome

• Welcome to part four of our webinar series "the KEA DHCP Server"
About this Webinar

• Database backend support for Kea
• Database setup
• Maintaining the Kea-Database with kea-admin
• Kea Database Configuration
• Host/Reservation in a SQL Database
• Kea Configuration Database Backend
• Recovering from Database failures
• High Availability
• Kea HA Maintenance
• "so many options, which should I implement?"
Database backend support for Kea
Why a database backend?

• The Kea DHCP server can store in a database:
  • Lease information
  • Host addresses and prefixes
  • Host options
  • Host names
  • Host classification
  • Configuration (MySQL only)
Benefits of using a database backend

• Faster turn-around for configuration changes in large deployments (many DHCP servers)
• Easy access to DHCP information from scripts
• Option to build a custom management interface for the DHCP service
• High-Availability through database redundancy
• Easier to integrate into existing backup systems
Drawbacks of using a database backend

• when issuing a lease, Kea DHCP must wait for the storage backend to acknowledge the successful storage of lease information

• depending on the database setup and implementation, this is often slower than the Kea in-memory (lease-file) storage

• Some databases cannot store lease information that reaches beyond the year 2038
PostgreSQL

- powerful, open source object-relational database system
  - flexible and extensible
- performance is in par with commercial database offerings on Linux/Unix
- PostgreSQL License (permissive open source license)
- PostgreSQL: https://www.postgresql.org
MySQL/MariaDB

- MySQL is the most popular (network based) open source SQL database system
  - MariaDB is a compatible fork of MySQL by the original MySQL development team
  - MySQL/MariaDB is available in most Linux/Unix systems
  - MySQL and MariaDB are GPLv2 licensed
- MariaDB:
  https://mariadb.com
- MySQL:
  https://www.mysql.com
Cassandra

• Apache Cassandra is a NoSQL database for large amounts of data
  • it provides linear scalability and fault-tolerance on commodity hardware
• Support for the Cassandra Database in Kea is experimental
• Cassandra is available under the Apache License 2.0
• Cassandra Database: https://cassandra.apache.org
Database setup
Preparing the database

• steps required before Kea can connect to a database system:
  • create the database
  • create a user for Kea in the database system
  • set the access permissions on the Kea database
PostgreSQL (1/2)

• Most PostgreSQL installations come with a dedicated operating system user account for the PostgreSQL database (in our examples, the user postgres).

• All database configuration steps should be done as this database user.
Creating the database for storing Kea lease information
and giving the user kea access permissions on this database

(kea-server)# su - postgres
(kea-server)$ psql -U postgres
Password for user postgres:
psql (12.4)
Type "help" for help.
postgres=# CREATE USER kea WITH PASSWORD 'secure-password';
CREATE ROLE
postgres=# CREATE DATABASE kea_lease_db;
CREATE DATABASE
# GRANT ALL PRIVILEGES ON DATABASE kea_lease_db TO kea;
postgres=# \q
Preparing a MariaDB/MySQL database (1/2)

• To prepare the MySQL/MariaDB database for Kea, first the database are created (in this example, one database for leases)

```sql
mysql -u root -p
Enter password:
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 8
Server version: 10.4.14-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> CREATE DATABASE kea_lease_db;
Query OK, 1 row affected (0.000 sec)
```
Preparing a MariaDB/MySQL database (2/2)

• In the next step, the database user kea is created and given access to the lease-database:

```sql
MariaDB [(none)]> CREATE USER 'kea'@'localhost' IDENTIFIED BY 'secure-password';
Query OK, 0 rows affected (0.006 sec)

MariaDB [(none)]> GRANT ALL ON kea_lease_db.* TO 'kea'@'localhost';
Query OK, 0 rows affected (0.005 sec)

MariaDB [(none)]> quit
Bye
```
MySQL performance tuning

• If MySQL is used with the InnoDB database backend (the default), changing the MySQL internal value `innodb_flush_log_at_trx_commit` from default value 1 to 2 can result with huge gain in Kea performance
  • It can be set per session for testing:
    ```
    mysql> SET GLOBAL innodb_flush_log_at_trx_commit=2;
    mysql> SHOW SESSION VARIABLES LIKE 'innodb_flush_log%';
    ```
  • or permanently in `/etc/mysql/my.cnf`
    ```
    [mysqld]
    innodb_flush_log_at_trx_commit=2
    ```
  • Changing this value can create problems during data recovery after a database crash - please check the MySQL documentation
Maintaining the Kea-Database with kea-admin
Initializing the Kea database (1/3)

• The command `db-init` of the `kea-admin` tool is used to initialize the database

• while it is possible to initialize the SQL databases with the SQL-scripts provided with Kea, it is recommended to use the `kea-admin` tool, as it provides extra security checks in the process
Initializing the Kea database (2/3)

- Example: initializing a PostgreSQL database for lease database

```bash
# kea-admin db-init pgsql -u kea -h 127.0.0.1 -p secure-password -n kea_lease_db
Checking if there is a database initialized already. Please ignore errors.
Initializing database using script /opt/kea/share/kea/scripts/pgsql/dhcpdb_create.pgsql
psql:/opt/kea/share/kea/scripts/pgsql/dhcpdb_create.pgsql:142: NOTICE:  function lease4dumpheader() does not exist, skipping
psql:/opt/kea/share/kea/scripts/pgsql/dhcpdb_create.pgsql:150: NOTICE:  function lease4dumpdata() does not exist, skipping
psql:/opt/kea/share/kea/scripts/pgsql/dhcpdb_create.pgsql:180: NOTICE:  function lease6dumpheader() does not exist, skipping
psql:/opt/kea/share/kea/scripts/pgsql/dhcpdb_create.pgsql:188: NOTICE:  function lease6dumpdata() does not exist, skipping
psql:/opt/kea/share/kea/scripts/pgsql/dhcpdb_create.pgsql:892: WARNING:  there is already a transaction in progress
Database version reported after initialization: 6.1
```
Initializing the Kea database (3/3)

• Example: initializing a MySQL database for leases

```
# kea-admin db-init mysql -u kea -p secure-password -n kea_lease_db
Checking if there is a database initialized already. Please ignore errors.
Verifying create permissions for kea
MySQL Version is: 10.4.14-MariaDB
Initializing database using script /opt/kea/share/kea/scripts/mysql/dhcpdb_create.mysql
mysql returned status code 0
Database version reported after initialization: 9.3
```
Upgrade of database schema (1/3)

- Sometimes a new Kea version may require a new database schema
- The existing database will need to be upgraded
- After upgrade, it may be impossible to subsequently downgrade to an earlier version
- Before upgrading, please make sure that the database is backed up
- The `kea-admin db-upgrade` command can be used to upgrade an existing database
Upgrade of database schema (2/3)

- To check the current version of the database, use the following command (<db-product> can be mysql or pgsql):

```
$ kea-admin db-version <db-product> -u <db-user> -p <db-password> -n <db-name>
```

- If the version does not match the minimum required for the new version of Kea (as described in the release notes), the database needs to be upgraded.

- see also Databases and Database Version Numbers
Upgrade of database schema (3/3)

• The `kea-admin` command is used to upgrade the database schema of the database (`<db-product>` can be `mysql` or `pgsql`):

```
$ kea-admin db-upgrade <db-product> -u database-user -p database-password -n database-name
```
Kea Database Configuration
Configuration Example: Lease Database in PostgreSQL

- Example of a lease database configuration in Kea (file kea-dhcp4.conf or kea-dhcp6.conf)
- For MySQL/MariaDB, just change the type to mysql

```json
"lease-database": {
  "type": "postgresql",
  "name": "kea_lease_db",
  "user": "kea",
  "password": "secure-password",
  "host": "localhost"
},
```
Test the configuration

# kea-dhcp4 -t /opt/kea/etc/kea/kea-dhcp4.conf

HOSTS_BACKENDS_REGISTERED the following host backend types are available: mysql

DHCPSRV_CFGMGR_SOCKET_TYPE_DEFAULT "dhcp-socket-type" not specified, using default socket type raw

DHCPSRV_CFGMGR_NEW_SUBNET4 a new subnet has been added to configuration: 192.0.2.0/24 with params: t1=900, t2=1800, valid-lifetime=3600
Host/Reservation in a SQL Database
Host/Reservation in a SQL Database - Why?

• Larger deployments might want to change the DHCP reservations dynamically and programatically via the API
  • The Host Commands hook (part of the Premium hooks package) adds a number of new commands to Kea used to query and manipulate host reservations
  • see Webinar 3 of this series for a discussion of the Host-Commands API
• the Host Commands hook requires a database for storing the host reservations
• If reservations are specified in both file and database, file reservations take precedence over the ones in the database
Creating a PostgreSQL database to store host reservations

• Creating the database for storing Kea host information (reservations) and giving the user kea access permissions on this database

(kea-server)# su - postgres
(kea-server)$ psql -U postgres
Password for user postgres:
psql (12.4)
Type "help" for help.

postgres=# CREATE DATABASE kea_host_db;
CREATE DATABASE
postgres=# GRANT ALL PRIVILEGES ON DATABASE kea_host_db TO kea;
GRANT
postgres=# \q
Initializing the Host reservation database

• The command db-init of the kea-admin tool is used to initialize the database.
  • While it is possible to initialize the SQL databases with the SQL-scripts provided with Kea, it is recommended to use the kea-admin tool, as it provides extra security checks in the process.
  • Example: initializing a PostgreSQL database for use as a host reservation database.

```bash
# kea-admin db-init psql -u kea -h 127.0.0.1 -p secure-password -n kea_host_db
Checking if there is a database initialized already. Please ignore errors.
Initializing database using script /opt/kea/share/kea/scripts/pgsql/dhcpdb_create.psql
psql:/opt/kea/share/kea/scripts/pgsql/dhcpdb_create.psql:142: NOTICE:  function lease4dumpheader() does not exist, skipping
psql:/opt/kea/share/kea/scripts/pgsql/dhcpdb_create.psql:150: NOTICE:  function lease4dumpdata() does not exist, skipping
psql:/opt/kea/share/kea/scripts/pgsql/dhcpdb_create.psql:180: NOTICE:  function lease6dumpheader() does not exist, skipping
psql:/opt/kea/share/kea/scripts/pgsql/dhcpdb_create.psql:188: NOTICE:  function lease6dumpdata() does not exist, skipping
psql:/opt/kea/share/kea/scripts/pgsql/dhcpdb_create.psql:892: WARNING:  there is already a transaction in progress
Database version reported after initialization: 6.1
```
Configuration Example: Host Database in PostgreSQL

• Host database for PostgreSQL configuration in Kea (file `kea-dhcp4.conf` or `kea-dhcp6.conf`)

• for MySQL/MariaDB, just change the type to `mysql`

```json
"hosts-database": {
  "type": "postgresql",
  "name": "kea_host_db",
  "user": "kea",
  "password": "secure-password",
  "host": "localhost"
},
```
Using Read-Only Databases with Host Reservations

- the host reservation information might be stored in a database that contains other (sensible) inventory information for the network
- in some cases, for policy and security reasons, the Kea DHCP server should not be able to write into this database
- read-only access for retrieving reservations for clients and/or assigning specific addresses and options, can be configured explicitly in Kea with the read-only mode

```
"Dhcp4": {
  "hosts-database": {
    "readonly": true,
    ...
  },
  ...
}
```
Kea Configuration Database Backend
Storing Kea configuration in a database

- The Kea DHCPv4 and DHCPv6 servers support loading their configuration from an MySQL database.
- The configuration back end supports:
  - subnet and shared-network configurations
  - global DHCPv4/DHCPv6 parameter
  - option definitions
  - global, network and pool options
Benefits of the Kea database configuration backend

- The local Kea configuration on each server can be simple and static
- Each DHCP server can share an almost identical preconfigured configuration
- Enables offline configuration
- Sharing configuration between HA cluster members -> keeping the config in sync
- Helps with automatic configuration management
Kea configuration backend design (1/4)

• Administrators apply changes to the configuration into the database via the Kea Configuration Backend Commands hook
  • it is also possible to directly access the configuration on the SQL database level
  • changing the configuration on the database level requires a good understanding of the configuration database schema
• the Kea configuration command hook provides essential business logic that ensures logical integrity of the data
Kea configuration backend design (2/4)

- Kea DHCP server will pull/poll the configuration from the database
  - the poll interval is configured with the config-fetch-wait-time parameter
  - if a change is detected, the Kea DHCP server will fetch the delta to its current configuration and will reconfigure the service
Kea configuration backend design (3/4)
Kea configuration backend design (4/4)
Basic configuration for a Kea DHCP using the Config Backend

- each DHCP server that uses the Kea configuration backend can run on a simple and static configuration
- the server-tag is selecting the individual configuration for this DHCP server
- config-fetch-wait-time parameter defines the poll interval for new configuration (default 30) in seconds

```
"Dhcp6": {
    "server-tag": "office-1",
    "config-control": {
        "config-databases": [{
            "type": "mysql",
            "name": "kea_config_db",
            "user": "kea",
            "password": "secure-password",
            "host": "2001:db8:568::568"
        }],
        "config-fetch-wait-time": 120
    }
}
```
Hooks required for the config backend

• The hook `libdhcp_mysql_cb.so` is the implementation of the Configuration Backend for MySQL.
• It must be always present when the server uses MySQL as the configuration repository.
• This hook is part of the base open source Kea distribution

```
"hooks-libraries": [{
    "library": "/usr/lib/kea/hooks/libdhcp_mysql_cb.so"
}, {
    "library": "/usr/lib/kea/hooks/libdhcp_cb_cmds.so"
}]
```
Hooks required for the config backend

• The hook `libdhcp_cb_cmds.so` is optional.
• It should be loaded when the Kea server instance is to be used for managing the configuration in the database.
• This hooks library is only available to ISC customers with a support contract.

```
"hooks-libraries": [{
    "library": "/usr/lib/kea/hooks/libdhcp_mysql_cb.so"
}, {
    "library": "/usr/lib/kea/hooks/libdhcp_cb_cmds.so"
}],
```
Objects in the configuration database

- Kea can read the configuration of different DHCP objects from the database
  - global parameter
  - option definitions
  - options
  - shared subnet
  - subnet
  - pools
The role of server tags (1/2)

- DHCP server select their configuration objects by the use of server-tags
- each DHCP server has one tag
- multiple servers can share a tag
- a server without an explicit server-tag configuration uses the special all tag
The role of server tags (2/2)

- Configuration objects in the database have tags assigned
- Objects can reference multiple server tags
- Objects can reference no server (empty tag """)
- Objects can reference the special tag "all" to reference all DHCP servers
Use of Server-Tags in the configuration backend (1/4)
Use of Server-Tags in the configuration backend (2/4)
Use of Server-Tags in the configuration backend (3/4)
Use of Server-Tags in the configuration backend (4/4)

Kea Configuration Database

- Option definition
  - Office 1
  - Office 3

- Pool
  - Office 2

- Option "domain-name"
  - Office 1
  - Office 2

- Option "zimhutssio-server"
  - Office 2
  - Office 3

Kea DHCPv4

- Office 1
- Office 2
- Office 3
Configuration Backend Resources

• Kea Reference Manual: Kea Configuration Backend
  https://kea.readthedocs.io/en/kea-1.8.0/arm/config.html#kea-configuration-backend

• Kea Configuration Backend Design
  https://gitlab.isc.org/isc-projects/kea/-/wikis/designs/configuration-in-db-design

• Video: Alan Clegg: Using the Kea Configuration Backend
  https://www.youtube.com/watch?v=gnVEO4ThE10
Recovering from Database failures
Recovering from Database failures (1/3)

• When operating Kea with a database backend, the database or the connection to the database might fail
  • During server start-up, the inability to connect to the database is considered fatal and the server will exit
• During dynamic reconfiguration, the databases are disconnected and then reconnected using the new configuration
  • If connectivity to the database(s) cannot be established, the server will log a fatal error but remain up.
  • It will be able to process commands but will not serve clients
    • This allows the configuration to be fixed via a remote command, if required
Recovering from Database failures (2/3)

• During normal operations, if connectivity to database is lost and automatic recovery is enabled …
  • … the server will disconnect from all databases
  • … and then attempt to reconnect them
• During the recovery process, the server will cease serving clients, but continue to respond to commands
  • Once connectivity to all databases is restored, the server will return to normal operations
  • If connectivity cannot be restored after max-reconnect-tries, the server will issue a fatal error and exit
Recovering from Database failures (3/3)

- The parameter `reconnect-wait-time` configures the number of milliseconds the server will wait between attempts to reconnect to the database after connectivity has been lost.
- The default value for MySQL and PostgreSQL is 0, which disables automatic recovery and causes the server to exit immediately upon detecting the loss of connectivity to a database.
High Availability
DHCP High-Availability

• DHCP is a critical resource in most networks
• If the DHCP service is down, machines and computers cannot join the network
• DHCP administrators like to make the DHCP service redundant and high-available
Kea High-Availability options

• Kea DHCP supports different high availability (HA) options
  • some require only configuration changes
  • other require the (free) HA hook

• Kea does not support the standardized DHCPv6 fail-over protocol (RFC 8156 "DHCPv6 Failover Protocol")
  • it supports a HA implementation that aligns with the Kea software design and covers most use-cases
**DHCPv6 Split Pool / Shared Pool**

- The DHCPv6 split pool or shared pool HA solution are independent from the DHCPv6 server implementation.
- These HA solutions do not require any synchronization between the DHCP server.
- These solutions make use of the vast address space available in one IPv6 /64 subnet.
- These solutions are not good solutions for DHCPv4, because the address space in IPv4 is too small.
DHCPv6 Split Pool

• Split-Pool: because one IPv6 /64 is so large, it usually can be split in two parts that are served by two independent DHCPv6 servers
• the pools are not overlapping, it is impossible that the two DHCPv6 servers will return the same lease address to different clients
• if one DHCPv6 server stops responding, the clients will receive a new lease from the remaining DHCPv6 server (after lease expiry)
DHCP Split Pool

Network 2001:db8:100::/64
Pool 2001:db8:100:0:1::/64
Network 2001:db8:100::/64
Pool 2001:db8:100:0:2::/64

All content © 2020 Internet Systems Consortium, Inc.
DHCPv6 Shared Pool

• Shared-Pool: two DHCPv6 server are authoritative for the same addresses from a pool
  • because of the size of one IPv6 /64 subnet, the chance that both servers give out the same address to different clients is statistically very low
  • and if they did, IPv6 duplicate address detection (DAD) will cover this rare edge case
  • if one DHCPv6 server stops responding, the clients will receive a new lease from the remaining DHCPv6 server (after lease expiry)
DHCPv6 Shared Pool
IPv4/IPv6 pure static DHCP

• The pure static solution also works with any DHCP server, in IPv4 and IPv6 networks
  • The idea is to not use dynamic allocation of addresses, but only static reservations
  • Two or more DHCP server are equipped with the same reservation configuration
  • Each server will always return the same IP address lease to the same client
• This solution requires an out-of-band synchronization of the reservation
  • This could be done on the database level with a shared host reservation database
IPv4/IPv6 pure static DHCP

- Both DHCP server contain no pools but only reservations.
- Network 192.0.2.0/24
  - HW: 01:0F:13:04:05:06 -> 192.0.2.10
  - HW: 40:FFFF:FF:00:11 -> 192.0.2.11
  - HW: 80:4A:11:21:34:13 -> 192.0.2.100
- Network 192.0.2.0/24
  - HW: 01:0F:13:04:05:06 -> 192.0.2.10
  - HW: 40:FFFF:FF:00:11 -> 192.0.2.11
  - HW: 80:4A:11:21:34:13 -> 192.0.2.100
IPv4/IPv6 shared database

- The shared database solution moves the redundancy to the database level
  - this solution allows high availability with more than two DHCP server nodes
  - two or more DHCP server are connected to the same (logical) database containing the lease information
- the database itself should be made high available
- all DHCP servers read and write lease information from/to the same database
- database locking can lead to performance degradation(!) on high rate of leases/renewals
IPv4/IPv6 shared database
High Availability Hook

• Using the Kea DHCP High-Availability extension (HA hook) is the most feature rich high availability solution
• The HA hook offers different operation modes
  • load-balancing: all DHCP server are active and return leases
  • hot-standby: all DHCP server are in sync but only one is active and returns leases
  • passive-backup: one DHCP server is active and send lease database updates to a number of backup servers.
Kea HA Mode: load-balancing

- When operating in **load-balancing** mode, two Kea DHCP server are active and respond to lease requests
  - The lease information is synced between the Kea DHCP HA servers
  - the pools are split 50/50 between the two DHCP servers
  - Every DHCP server can take over the full service if needed
  - Via the HA protocol a DHCP HA node will detect if one partner node is down and takes over the service
    - once the partner is online again, the lease database is synced
Kea HA Mode: load-balancing
Kea HA Mode: hot-standby

• A Kea DHCP cluster configured for the hot-standby mode will have the primary node serving DHCP clients and another node (secondary) only receiving the lease-database updates, but not serving clients

• If the secondary server detects the failure of the primary, it starts responding to all DHCP queries
Kea HA Mode: hot-standby
Kea HA Mode: Backup Servers

• Kea DHCP supports any number of backup servers
  • Backup server receive lease database updates but are not an active part of an HA setup
  • Backup server can be deployed in addition to the other Kea HA modes
Kea HA Mode: Backup Servers
Kea HA Mode: passive-backup

- In the passive-backup configuration, only one Kea server is active and is serving leases to the clients
  - Any number of passive (not answering to clients) backup servers receive lease database backups
  - Since Kea 1.7.8, the active server does not need to wait for a lease update confirmation from the backup servers before giving the lease to a client
    - this reduces the latency compared to the other HA modes
- In case of an failure of the active server, a backup server needs to be manually promoted to be active
  - this could be automated outside of Kea with API calls from a monitoring system
Kea HA Mode: passive-backup
Example Configuration: Kea DHCP Failover Cluster
Kea HA Configurations

• The Kea HA configuration parts are symmetric, all HA peers can share an almost identical configuration file.
  • The only difference in the HA configuration is the `this-server-name` parameter.
  • The HA mode is selected with the `mode` parameter.
Example Load-Balancing Configuration

```json
"Dhcp4": {
    "hooks-libraries": [{
        "library": "/usr/lib/kea/hooks/libdhcp_lease_cmds.so", "parameters": {}}, {
        "library": "/usr/lib/kea/hooks/libdhcp_ha.so", "parameters": {
            "high-availability": {
                "this-server-name": "server1",
                "mode": "load-balancing",
                "heartbeat-delay": 10000, "max-response-delay": 40000, "max-ack-delay": 5000,
                "max-unacked-clients": 5,
                "peers": [{
                    "name": "server1",
                    "url": "http://192.0.2.33:8000/",
                    "role": "primary", "auto-failover": true
                }, {
                    "name": "server2",
                    "url": "http://192.0.2.66:8000/",
                    "role": "secondary", "auto-failover": true
                }, {
                    "name": "server3",
                    "url": "http://192.0.2.99:8000/",
                    "role": "backup",
                    "basic-auth-user": "foo", "basic-auth-password": "bar",
                    "auto-failover": false
                }]
            }
        }
    }
}]}},
```
"Dhcp4": {
  "hooks-libraries": [{
    "library": "/usr/lib/kea/hooks/libdhcp_lease_cmds.so",  "parameters": { 
  },
  "library": "/usr/lib/kea/hooks/libdhcp_ha.so", "parameters": {
    "high-availability": [{
      "this-server-name": "server1",
      "mode": "hot-standby",
      "heartbeat-delay": 10000,  "max-response-delay": 40000,
      "max-ack-delay": 5000,   "max-unacked-clients": 5,
      "peers": [{
        "name": "server1",
        "url": "http://192.0.2.33:8000/",
        "role": "primary", "auto-failover": true
      },
      "name": "server2",
      "url": "http://192.0.2.66:8000/",
      "role": "standby", "auto-failover": true
    }, {
      "name": "server3",
      "url": "http://192.0.2.99:8000/",
      "basic-auth-user": "foo",  "basic-auth-password": "bar",
      "role": "backup", "auto-failover": false
    }
    ]
  }
},
  [...]
}]},
Kea HA Maintenance
Sending control commands to the Kea HA Module

• As many other parts of the Kea system, the HA module can be controlled over the network with the REST-API
• It receives commands in JSON format via the Kea Control Agent (CA)
• The following slides give examples of useful API commands
• More commands and details can be found in the Kea Reference Manual
  https://kea.readthedocs.io/en/latest/arm/hooks.html#control-commands-for-high-availability
Database synchronization

• the `ha-sync` command triggers the server to sync the lease database with the selected peer

```json
{"command": "ha-sync",
 "service": [ "dhcp4 "],
 "arguments": {
   "server-name": "server2",
   "max-period": 60
 }
}
```
Retrieving the HA status

• The command `ha-heartbeat` can be used to check the current state of a Kea DHCP server HA node

```json
{  "service": [ "dhcp4" ],  "command": "ha-heartbeat" }
```

• The returned JSON structure describes the current DHCP server state

```json
{  "result": 0,  "text": "HA peer status returned.",  "arguments": {  "state": "partner-down",  "date-time": "Thu, 07 Nov 2019 08:49:37 GMT"  } }
```
Fetching the HA configuration

• With the status-get command, the administrator can request the current HA configuration from a Kea DHCP server node

```json
{
    "result": 0,
    "text": "",
    "arguments": {
        "pid": 1234,
        "uptime": 3024,
        "reload": 1111,
        "high-availability": {
            "ha-mode": "load-balancing",
            "ha-servers": {
                "local": {
                    "role": "primary",
                    "scopes": ["server1" ],
                    "state": "load-balancing" },
                "remote": {
                    "age": 10,
                    "in-touch": true,
                    "role": "secondary",
                    "last-scopes": ["server2" ],
                    "analyzed-packets": 8 }
            }},
        "multi-threading-enabled": true,
        "thread-pool-size": 4,
        "packet-queue-size": 64
    }
}```
Controlling Maintenance Mode

• Before removing a Kea DHCP server from a HA setup, the server should be set into maintenance mode.

• The commands `ha-maintenance-start` and `ha-maintenance-cancel` commands can be used to bring a server in or out of maintenance mode.
Decision tree for production systems
"so many options, which should I implement?"

- Kea offers many different high-availability options
- for an user new to Kea or DHCP administration, this can be a hard choice
- the next slides give some general recommendations and guidance on how to select an high-availability option for a Kea deployment
Load-balancing vs. hot-standby

• As the name implies, in the load-balancing mode the load is distributed across both active DHCP servers
  • with complex client classing rules, this can be faster than a single active server
  • the load-balancing mode requires a 50/50 split of the pools across both HA server nodes
• The hot-standby mode is simpler
  • only one active server, one active log file for trouble shooting
  • no split pools required
HA Module vs. shared database

• A shared database setup offers redundancy for more than two active DHCP servers
• In a shared database setup, two clients might be offered the same IP address
  • one will succeed, the other will get a DHCPNAK from the server and has to start the DHCP process again.
• The HA module works with the memfile lease database, which offers better performance most of the time compared to an SQL database
HA Module vs. split/shared Pool

• Split- or shared pools only work well with DHCPv6
  • these are good options for IPv6-only networks
• Split- or shared pools are simple and easy to maintain
• The HA module is more universal
  • it works for DHCPv4 and DHCPv6 and across all supported lease file storage back ends (memfile, SQL-Database, Cassandra)
Next Webinars

• 18th November - Kea DHCP - Monitoring, Logging, and Stork
• 2nd December - Kea DHCP - Migrating to Kea from ISC DHCP
Resources

• Kea Performance Optimization

• MariaDB 10.x and Kea
  https://kb.isc.org/docs/en/maria-10x-and-kea

• Using the Kea Configuration Backend
Questions and Answers