Kea DHCP

Database Support

Carsten Strotmann and the ISC Kea Team

CREATED: 2025-11-13 THU 10:59

In this Chapter

- Database backends for Kea
- Database setup and maintenance
- Host/Reservation Data in the database

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

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 then 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

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

PostgreSQL (2/2)

- 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)

```
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:

```
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 (1/2)

- 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 lease database

```
# 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 lease4dumphe
psql:/opt/kea/share/kea/scripts/pgsql/dhcpdb_create.pgsql:150: NOTICE:  function lease4dumpda
psql:/opt/kea/share/kea/scripts/pgsql/dhcpdb_create.pgsql:180: NOTICE:  function lease6dumphe
psql:/opt/kea/share/kea/scripts/pgsql/dhcpdb_create.pgsql:188: NOTICE:  function lease6dumpda
psql:/opt/kea/share/kea/scripts/pgsql/dhcpdb_create.pgsql:892: WARNING:  there is already a t
Database version reported after initialization: 6.1
```

Initializing the Kea database (2/2)

- Example: initializing a MySQL/MariaDB 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 https://kea.readthedocs.io/en/latest/arm/admin.html#kea-database-version

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`**

```
 "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

2020-10-22 11:43:23.772 INFO  [kea-dhcp4.hosts/61595.139911418369920]
           HOSTS_BACKENDS_REGISTERED the following host backend types are available: mysql
2020-10-22 11:43:23.773 INFO  [kea-dhcp4.dhcpsrv/61595.139911418369920]
           DHCPSRV_CFGMGR_SOCKET_TYPE_DEFAULT "dhcp-socket-type" not specified , using defaul
2020-10-22 11:43:23.774 INFO  [kea-dhcp4.dhcpsrv/61595.139911418369920]
           DHCPSRV_CFGMGR_NEW_SUBNET4 a new subnet has been added to configuration: 192.0.2.0
```

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 adds a number of new commands to Kea used to query and manipulate host reservations
    - see Chapter 3 of this training 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 initialized 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

```
# kea-admin db-init pgsql -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.pgsql
psql:/opt/kea/share/kea/scripts/pgsql/dhcpdb_create.pgsql:142: NOTICE:  function lease4dumphe
psql:/opt/kea/share/kea/scripts/pgsql/dhcpdb_create.pgsql:150: NOTICE:  function lease4dumpda
psql:/opt/kea/share/kea/scripts/pgsql/dhcpdb_create.pgsql:180: NOTICE:  function lease6dumphe
psql:/opt/kea/share/kea/scripts/pgsql/dhcpdb_create.pgsql:188: NOTICE:  function lease6dumpda
```

```
psql:/opt/kea/share/kea/scripts/pgsql/dhcpdb_create.pgsql:892: WARNING:  there is already a t
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`

```
 "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 (sensitive) 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 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

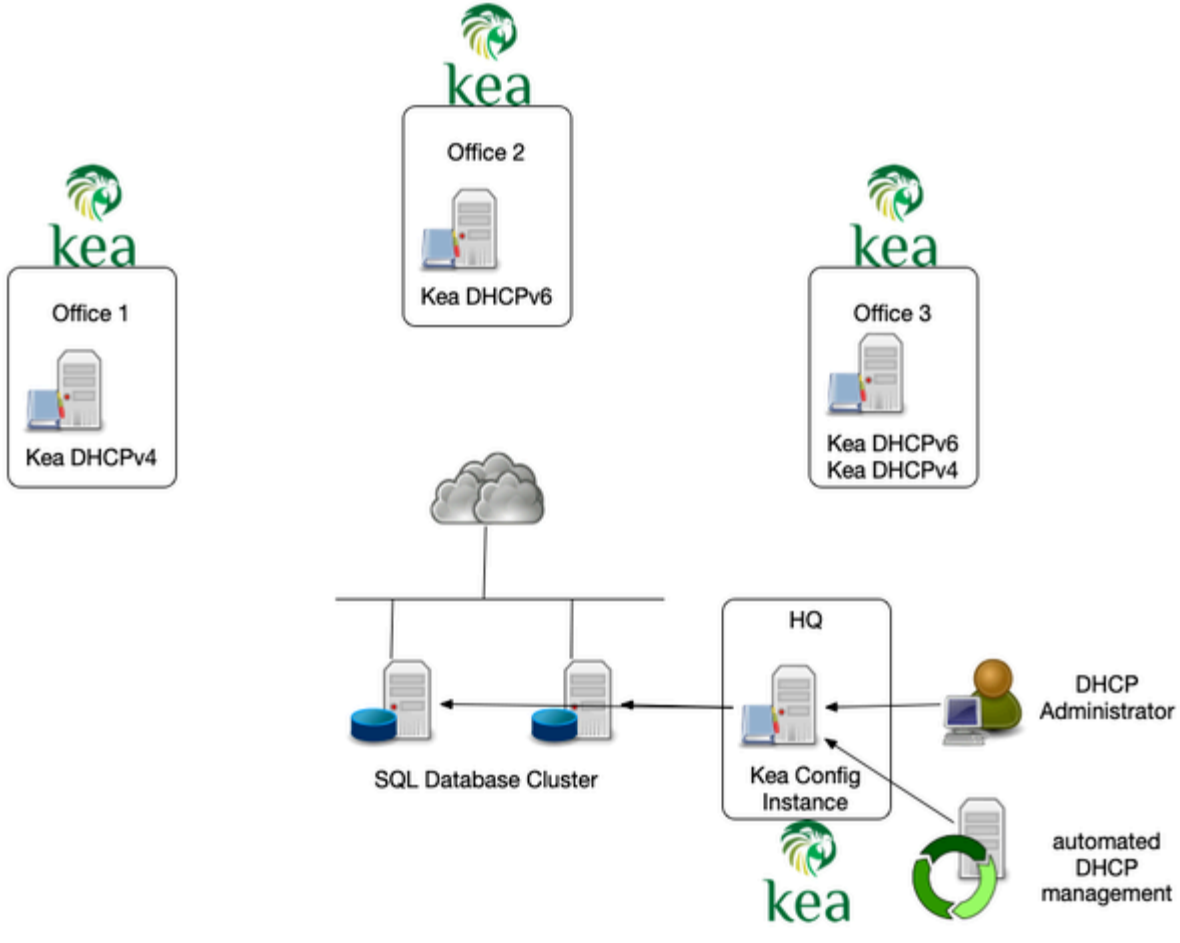
Kea configuration backend design (1/4)

- 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

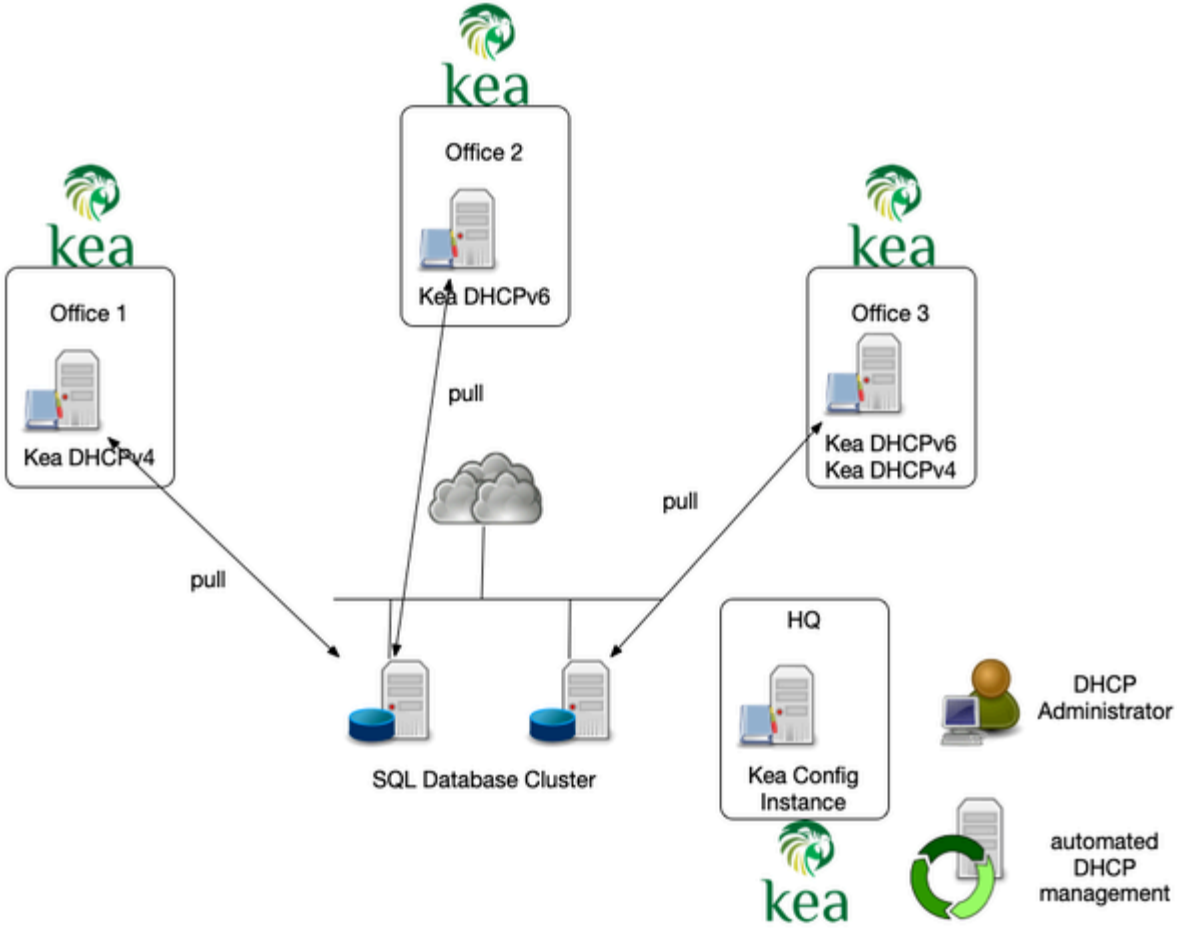
Kea configuration backend design (2/4)

- 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 intervall 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

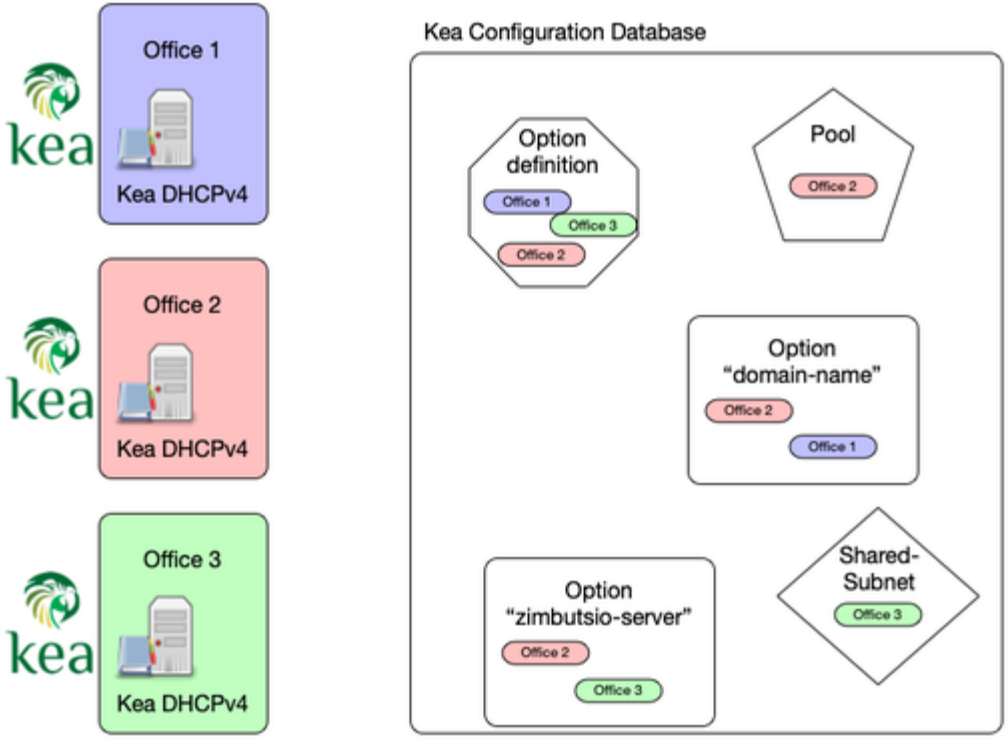
The role of server tags (1/2)

- 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

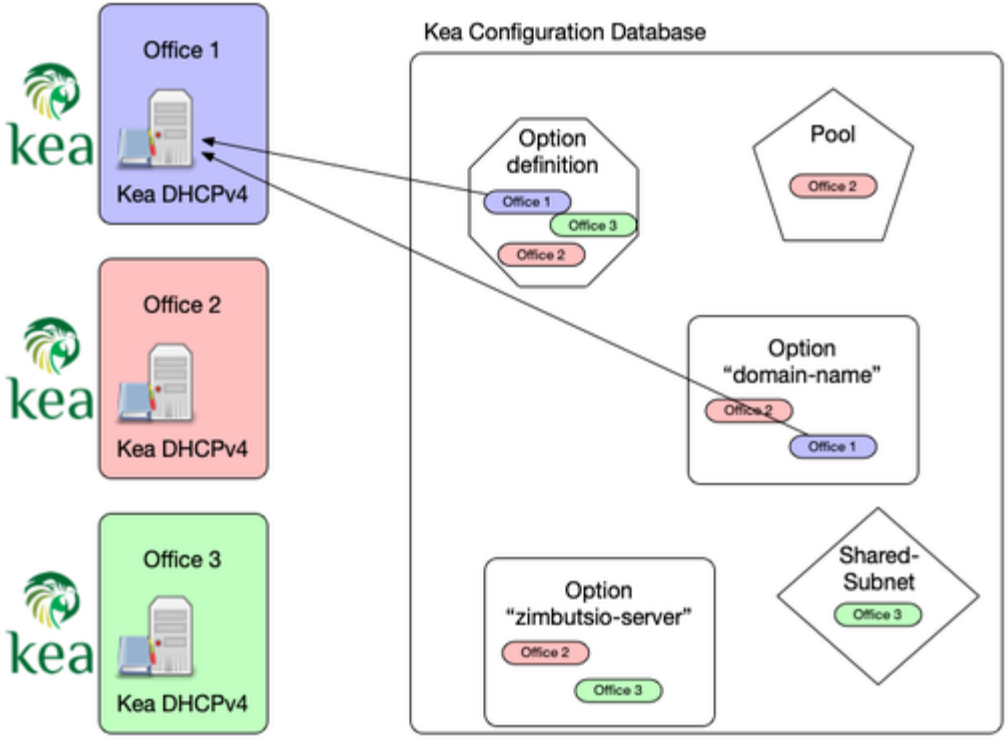
The role of server tags (2/2)

- 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

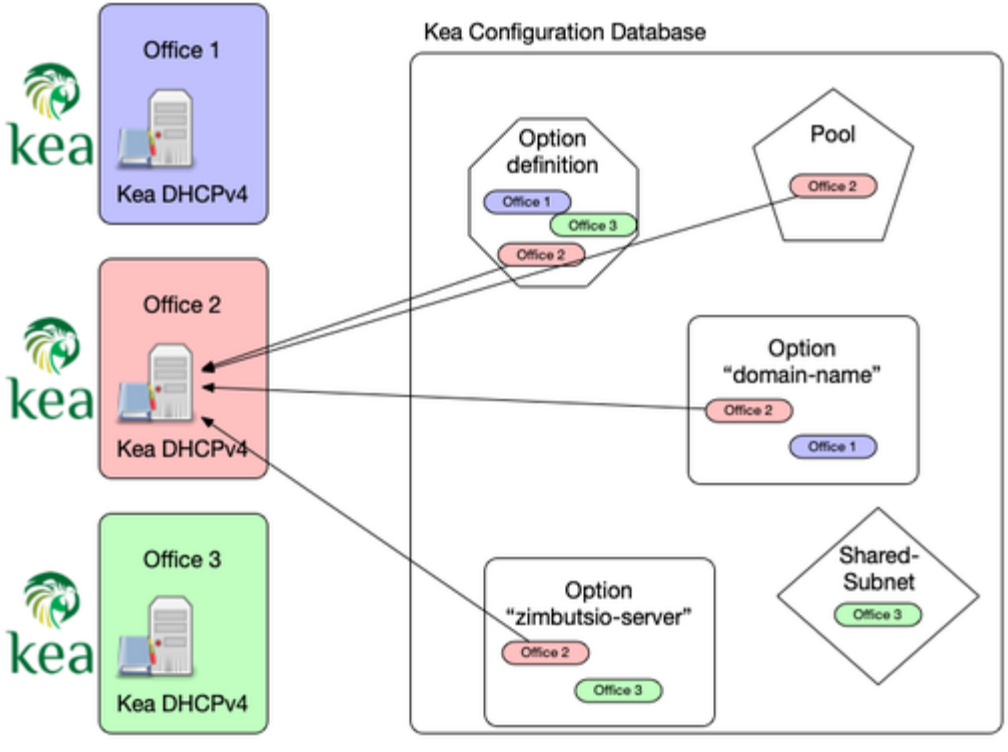
Use of Server-Tags in the configuration backend (1/4)



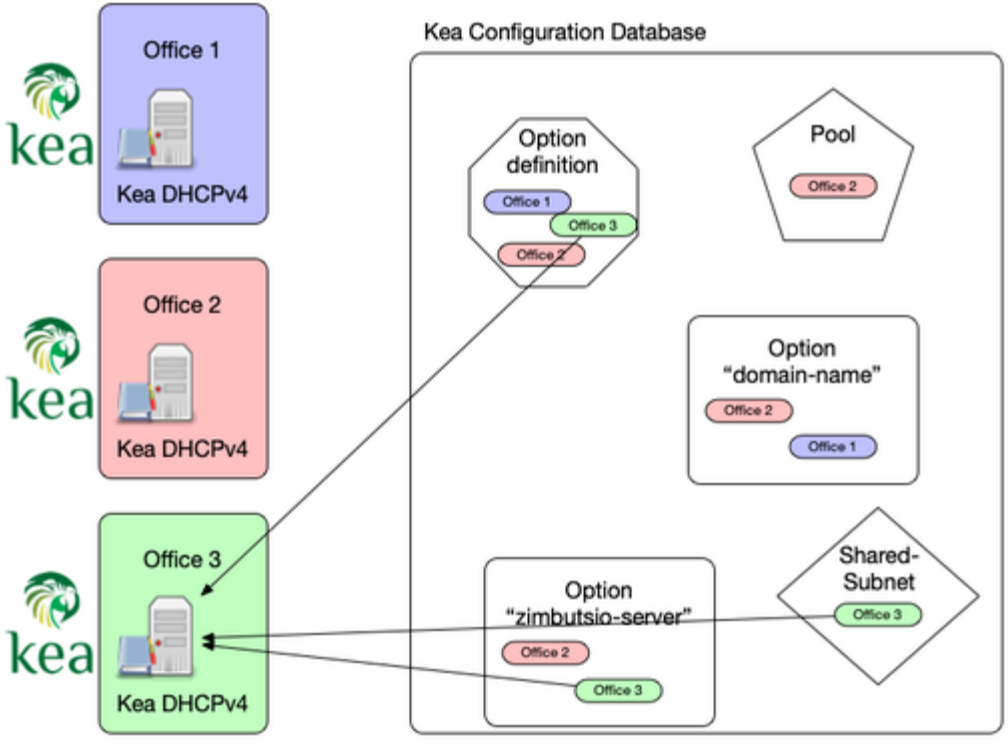
Use of Server-Tags in the configuration backend (2/4)



Use of Server-Tags in the configuration backend (3/4)



Use of Server-Tags in the configuration backend (4/4)



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=gnVE04ThE10

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 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

Resources

- Kea Performance Optimization https://kb.isc.org/docs/en/kea-performance-optimization
- MariaDB 10.x and Kea https://kb.isc.org/docs/en/maria-10x-and-kea
- Using the Kea Configuration Backend https://kb.isc.org/docs/en/using-the-kea-configuration-backend