Kea Documentation

Mark Fiers

Aug 26, 2019

Table of contents

1 Installation **3**
- 1.1 Use Postgres . . . 3

2 Metadata **5**
- 2.1 Using Kea to store metadata . . . 5
- 2.2 Metadata fields . . . 6
- 2.3 Multiple copies . . . 6

3 kea **7**
- 3.1 kea package . . . 7

4 Introduction **9**

5 Indices and tables **11**

Kea aims to be a lightweight system to track file and transactional metadata and on top of this a simple scripting template system.

CHAPTER 1

Installation

Kea3 required Python 3.6+. Apart from that, installation should be simple:

```
pip install kea3
```

1.1 Use Postgres

The default database is sqlite. If you want to use Postgresql instead, follow your system's instructions to install. For example on Ubuntu and derivates:

```
sudo apt install postgresql
```

Also, ensure the python bindings are installed:

```
pip install psycopg2
```

Properly configure Postregsql for use with Sqlalchemy (specifically psycopg2) (I expect it will need to listen to the network - see *listen_addresses* in the postgresql configuration). Create a user and database for kea, assign the correct permissions. Something like:

```
$ sudo -u postgres psql

postgres=# CREATE DATABASE keadb;
postgres=# CREATE USER kea WITH ENCRYPTED PASSWORD 'keapass';
postgres=# GRANT ALL PRIVILEGES ON DATABASE keadb TO kea;
```

Now, you need to tell kea to use this database by creating an approriate sqlalchemy connection string, and tell kea of this:

```
k3 conf set dbconnection 'postgresql+psycopg2://kea:KeaKea42@localhost/keadb'
```

CHAPTER 2

Metadata

Tracking file metadata is not simple. There are many different solutions proposed, but none seems to work for all situations (and neither will Kea). A few solutions that are out there are:

- Within a file, for example JPEG images have exif data
- In the filesystem. All filesystems store some metadata (as the filename, location on the filesystem, etc). Some filesystems allow for storage of more metadata (e.g. *xattr* - extended attributes).
- As sidecar files, i.e. each file has a accompanying file that stores metadata.
- In a database

Kea links metadata (basically, any key value pair that you like) to the *sha256* checksum of a file. The Kea implementation has the following advantages.

- It does not interfere with your files, they do not change. None of your software needs to be aware that you are storing metadata.
- No sidecar files, no forgetting to copy, move or rename them.
- Independent of OS & filesystem. Xattr is nice, but not universal and does (for example) not work across NFS.
- Easy to export, you can dump a file with checksum & metadata that you can process.
- Sha256 is a strong, cross platform, algorithm.

There are downsides, though:

- It can be slow - calculating the checksum of a large file takes time. Kea does cache checksums and only recalculates if modification time, size or path change.
- You need to backup & maintain your database.

2.1 Using Kea to store metadata

Kea can be operated as a command line tool called *k3* (It's the third iteration of this tool).

For this part of the documentation we will work with one demo file. To create a file to work on:

```
$ echo "Nestor notabilis" > kea.txt
```

For later reference, lets check the sha256 checksum:

```
$ sha256sum kea.txt
25802ac7e79b7134ca8968790909264e9d7a23a6a0a4ed6c00caff7c818b83de  kea.txt
```

If you want to associate metadata to a file, you can use *k3*:

```
$ k3 set test any_test_value file.txt
```

You can then check if it worked by running:

```
$ k3 show file.txt
sha256     : a23e5fdcd7b276bdd81aa1a0b7b963101863dd3f61ff57935f8c5ba462681ea6
sha1       : a5c341bec5c89ed16758435069e3124b3685ad93
short      : foj5f3Neyd
md5        : 4d93d51945b88325c213640ef59fc50b
# Files (mtime, hostname, path)
2018-08-31T15:16:50.132572 gbw-d-l0067 /home/luna.kuleuven.be/u0089478/project/kea3/
↪docs/file.txt **
# Metadata:
 - size       : 10
 - test       : any_value
```

There are a few things to unpack here:

- First of all, you see under the header *# Metadata* the field we just set. That's good.
- Secondly, at the top there are a number of checksums. *sha256*. Kea does also store the *sha1* and *md5* (mostly for cross referencing with an older system using *sha*). There is one extra checksum: *short* which always starts with a *f* and then has the first 9 characters of a base64 conversion of sha256 checksum. The *short* checksum is used only for convencience, within *Kea*. It is easier to copy paste short ids than the long sha256 ones.
- Given that Kea links metadata to a checksum, not a file, kea essentially show data associated with a checksum, not a file. For the filename (*file.txt*) on the command line the checksum is retrieved, and the database is queried. Kea then also output all files it knows of with that checksum - they're shown under *# Files*

2.2 Metadata fields

2.3 Multiple copies

Given that Kea stores data linked to

CHAPTER 3

kea

3.1 kea package

3.1.1 Subpackages

kea.plugin package

Submodules

kea.plugin.metadata module

kea.plugin.run module

kea.plugin.simple_executor module

kea.plugin.simple_job_checker module

kea.plugin.transaction module

Module contents

3.1.2 Submodules

3.1.3 kea.cli module

3.1.4 kea.kmeta module

3.1.5 kea.util module

3.1.6 kea.workflow module

3.1.7 Module contents

CHAPTER 4

Introduction

Kea aims to:

- **Track file metadata**: Kea tracks file metadata based on a file's *sha256* checksum. This means that if you delete a file and recreate it the metadata wil still be there. Similarly, if you have multiple copies of a file, all of them will share the metadata. *More..*
- **Track relationships between files.**: Relationships between files are tracked by storing a record with the checksums of the related files. For example if one file is created from another file, Kea creates a record with the checksums the two files (and additional information). You can later query the database for any transaction related to a certain file. If you consequently store transactions (as part of your scripts), you will have full provenance. Again, by storing checksums, you can move files around without losing information.
- **Provide simple script templates**: File and transaction metadata can be stored individualy per file and transaction. However, to make things easier, Kea provides a simple templating system where you can build a template, inidicate what the in-, and output files are, and upon exeuction, Kea stores the transaction automatically. Additionally, the script engine provides the ability to run across many files at once.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search