ETCD

→ etcd V3 is the recent version, so I am not going to worry about older version.

→ To install:
  → etcd binary can be downloaded online which is the easiest way to install

Getting Started:

→ It is quite easy to put in data and fetch data from etcd

→ for storing data

```
etcdctl put <my key> "value" ↵
```

→ for fetching data

```
etcdctl get <my key> ↵
```

→ Flatten binary key-value space:

  → keys are treated as simple byte strings
  → no hierarchy in keys
  → even if the key can have slashes to create hierarchical appearance, they do not actually mean anything

[illegible]

→ makes it simple

→ Both keys and values are stored as raw, binary byte array. Allows for storing not just strings but serialized data such as protocol buffers, JSON or any other format

→ Benefit:

→ quick lookups: Don't have to parse complex hierarchical key

→ lexicographically sorted: easy for range queries, for example, to query all keys with certain prefix; <my key>

→ compaction

→ ETCD uses MVCC (multi version concurrency control) model

→ This means every update of a data is stored as a new revision

→ This creates large space consumption even without new object creation. Just update of key value would overrun the space

→ Compaction marks older version of data for deletion

→ The disk space is not reclaimed back, though, through compaction

That must be done through "Defragment"

[illegible] ations"

→ etcd stores all historical versions of keys and data. Over time, this creates a lot of tombstoned (deleted) or superseded (old versions) of data

→ Compaction prunes these old data; keeps in-use database size in check

→ What happens during compaction?

→ every write operation increments global revision

→ keys are stored in tuples of (key, revision, value) in BoltDB backend

→ compaction is triggered with a revision number 'r'. Any key older than 'r' is marked as compactable

→ MVCC removes all the keys that are before 'r' except the most recent version that is <= 'r'

→ Any client watching from a compacted version will get "Error compacted" and must reestablish their watch from a newer version

→ The actual underlying DB rewriting only occurs upon "Defragmentation"