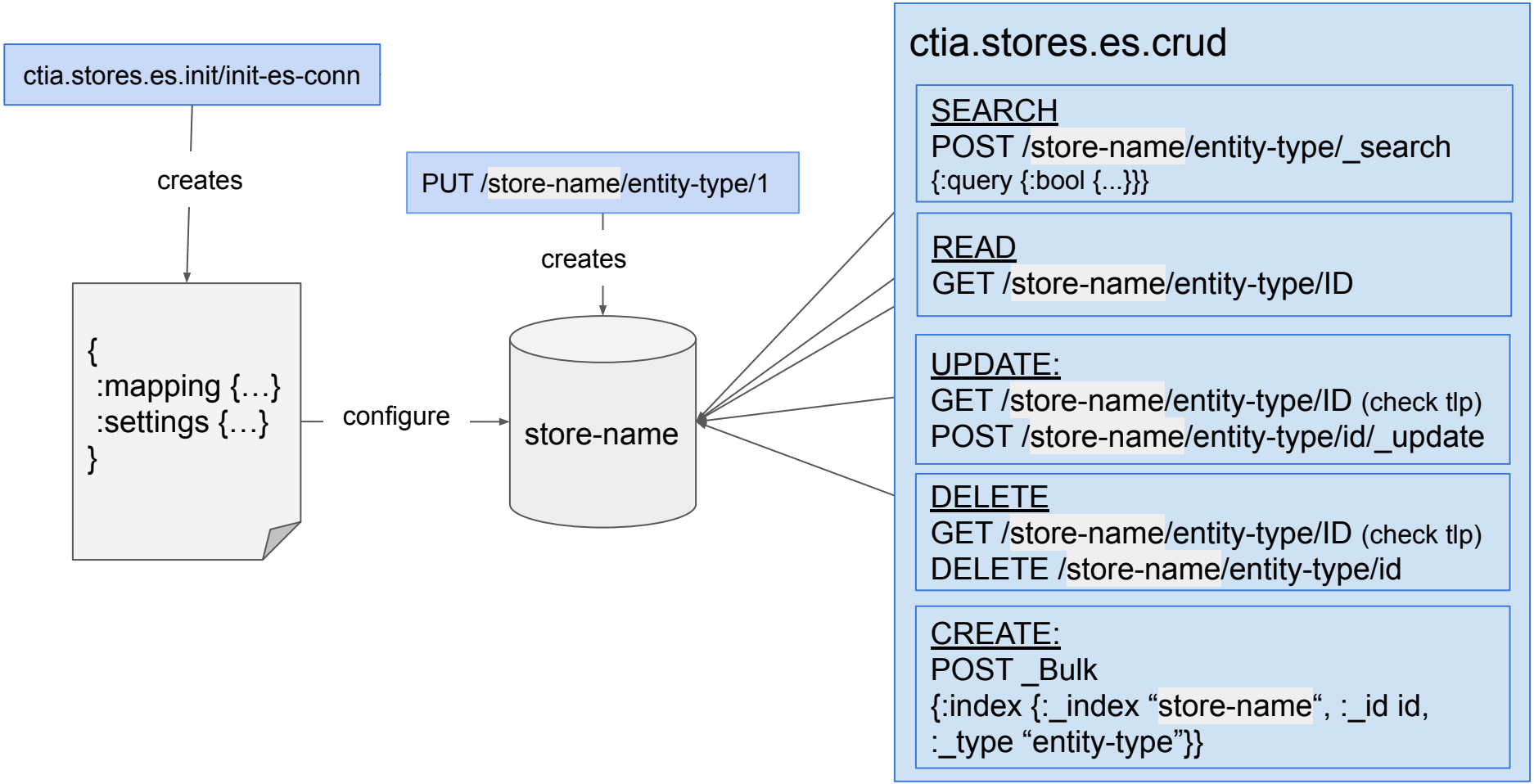
CTIA Elasticsearch stores

Managing big indices

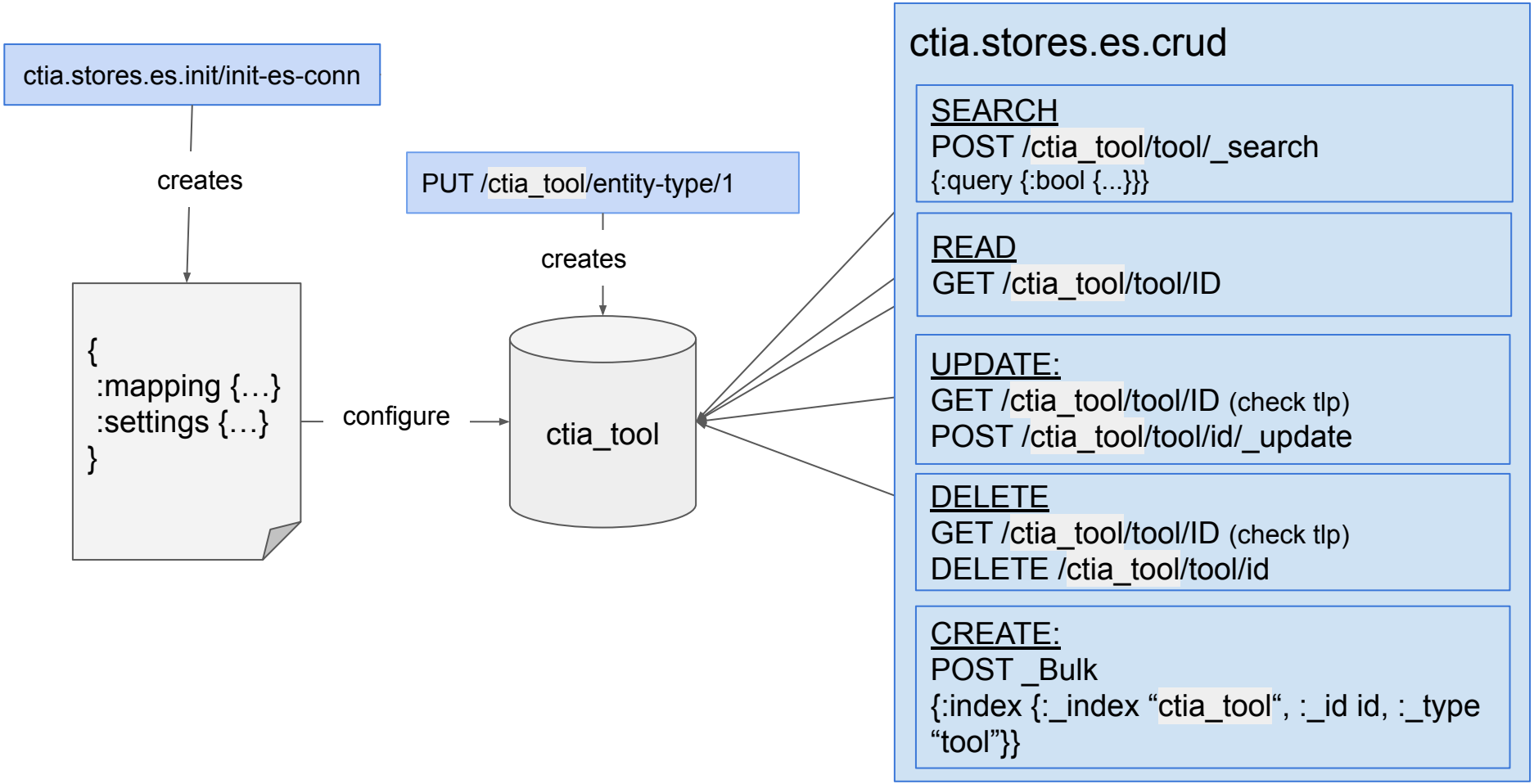
Unaliased stores: 1 store = 1 index

- Directly write on configured index name
- Initialization:
  - Create one template with proper index settings and type mapping.
  - Index is created with first document insertion, with proper configuration thanks to template
- CRUD operations and Search request directly target the index.
- \+ ⇒ simpler code
- \- ⇒ limits horizontal scaling since number of shards cannot be increased.

unaliased stores: 1 store = 1 index



example: unaliased tool store



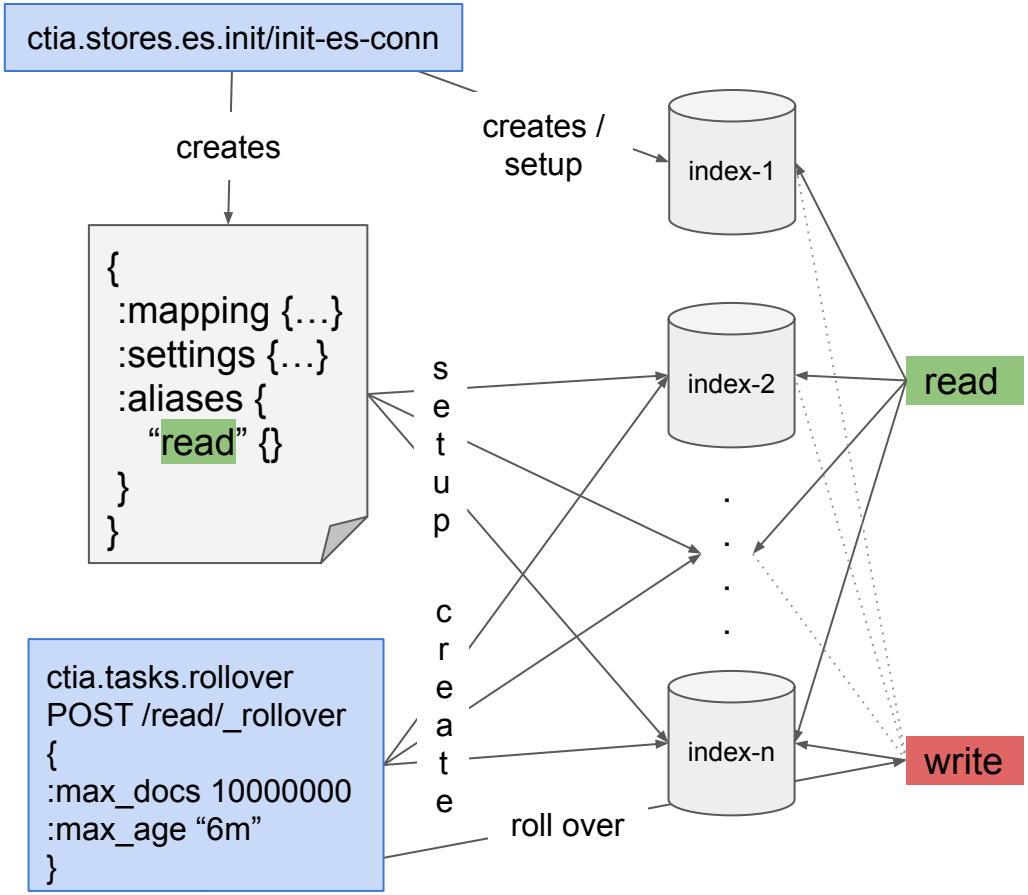
Aliased stores: 1 store / n indices, read/write aliases

- multiple indices, with *read* alias on all indices and *write* alias on most recent index. read alias is the store name, write alias is the store name suffixed by *-write*, index names are the store name suffixed by a number.
- Initialization: generates a template with a read alias and the first index with the write and read aliases.
- ES CRUD operations do not work on multiple indices:
  - GET is replaced by a `_search` request with an `ids` query on *read* alias
  - CREATE are done on the most recent index using *write* alias
  - UPDATE and DELETE require 2 steps:

    1) a `_search` with an `ids` query on *read* alias to retrieve the real index of the targeted document.

    2) update / delete request on that retrieved index.
- `_rollover` api is used to create a new index and to “roll over” *write* alias on that new index, when some rules (`max_docs` and / or `max_age`) are matched.
- \+ ⇒ unlimited horizontal scaling
- \- ⇒ more complex code

Aliased stores: 1 store / n indices, read/write aliases



ctia.stores.es.crud

SEARCH
POST /read/entity-type/_search
{:query {:bool {...}}}

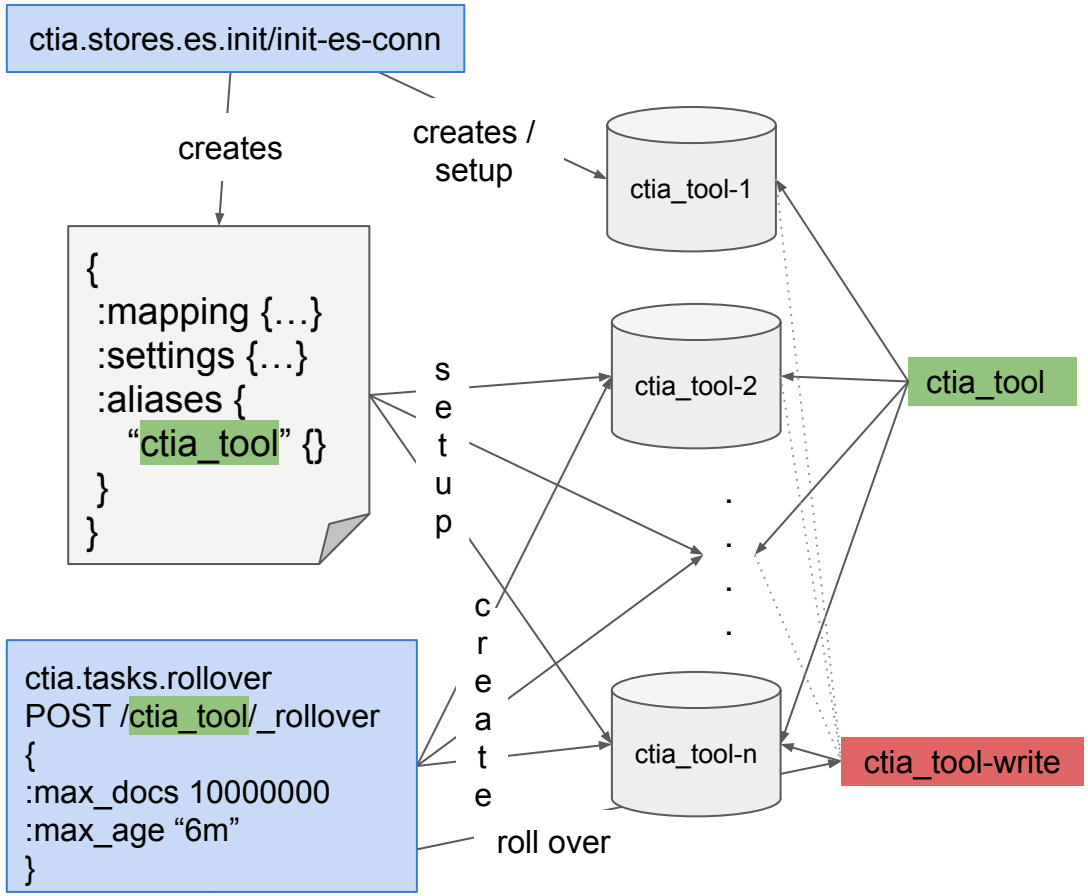
READ
POST /read/entity-type/_search
{“query”: {“ids”: { “values”: [id]}}}

UPDATE:
POST /read/entity-type/_search
{“query”: {“ids”: { “values”: [id]}}}
POST /index-%d/entity-type/id/_update

DELETE
POST /read/entity-type/_search
{“query”: {“ids”: { “values”: [id]}}}
DELETE /index-%d/entity-type/id

CREATE
POST _Bulk
{:index {:_index “write“, :_id id, :_type “entity-type”}}

example: aliased tool store



ctia.stores.es.crud

SEARCH
POST /ctia-tool/entity-type/_search
{:query {:bool {...}}}

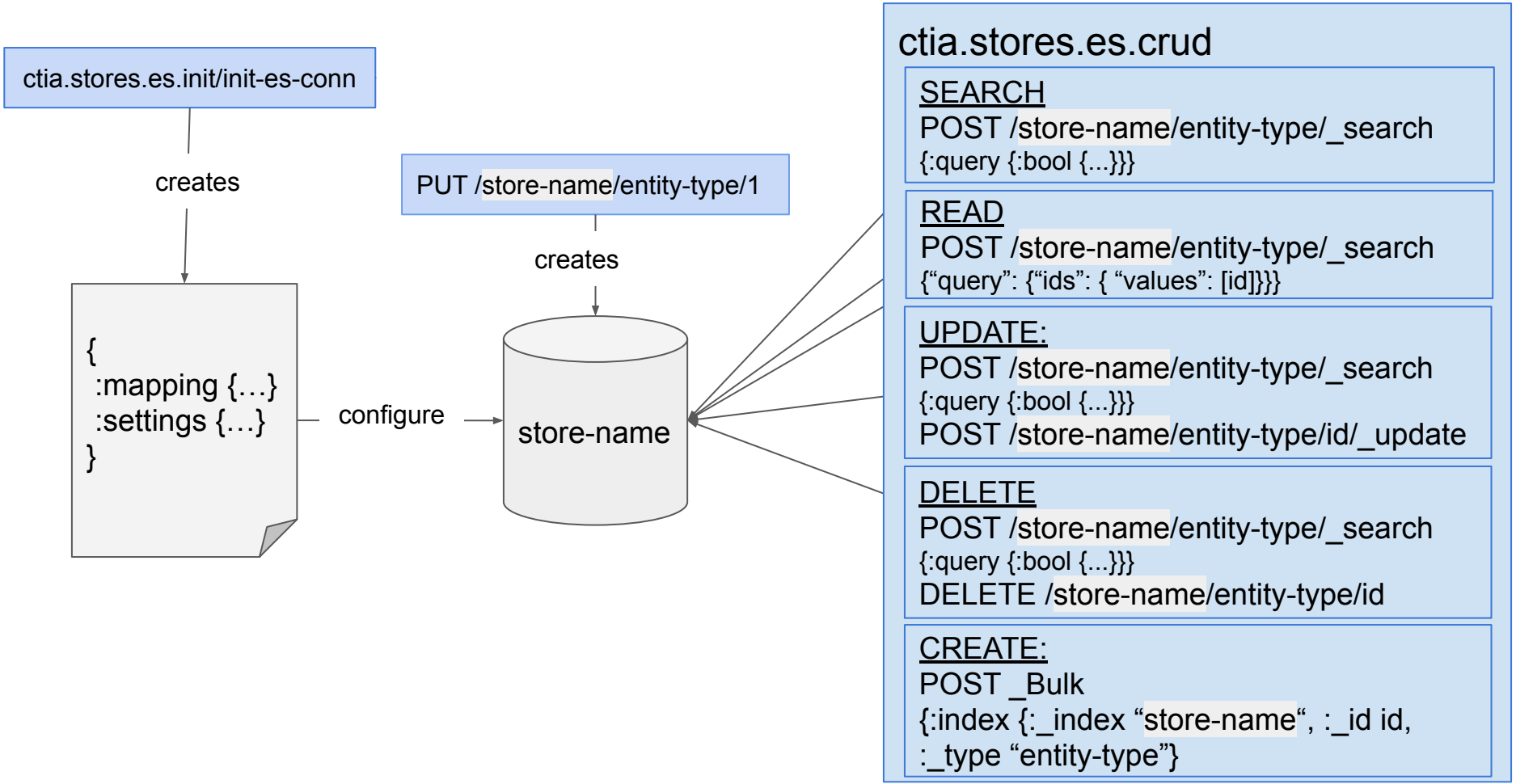
READ
POST /ctia-tool/entity-type/_search
{“query”: {“ids”: { “values”: [id]}}}

UPDATE:
POST /ctia-tool/entity-type/_search
{“query”: {“ids”: { “values”: [id]}}}
POST /ctia-tool-%d/entity-type/id/_update

DELETE
POST /ctia-tool/entity-type/_search
{“query”: {“ids”: { “values”: [id]}}}
DELETE /ctia-tool-%d/entity-type/id

CREATE
POST _Bulk
{:index {:_index “ctia_tool-write“, :_id id,
:_type “tool”}}

New unaliased stores: 1 store = 1 index + shared code



Migrating stores

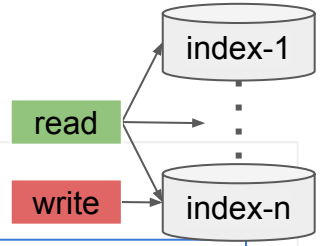
- We have 4 main cases when migrating a store:
  - unaliased ⇒ unaliased
  - unaliased ⇒ aliased
  - aliased ⇒ unaliased
  - aliased ⇒ aliased
- In every case the problem is twofold:
  - Reading from source aliased or not.
  - Writing into target aliased or not.
- The migration can be started while CTIA still running, stopped and restarted if necessary. Thus it must detect updates and deletes in source store and apply it to target store:
  - We detect updates by sorting documents by modification date.
  - We detect deletes from event store.
- Moreover it must roll over target when an aliased index is wanted.

Migration: reading source.

Unaliased

Retrieve created and modified docs:
POST /store-name/entity-type/_search
```
{
:sort [{:modified :desc},
       {:created :desc},
       {:uuid :desc}]
 :search_after: [..., …, ...]
}
```

retrieve deleted docs from events
POST /event/event/_search
```
{
:sort [{:modified :asc},
       {:created :asc},
       {:uuid :asc}]
 :search_after: [..., …, ...]
}
```

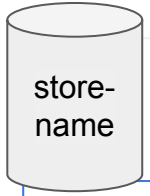


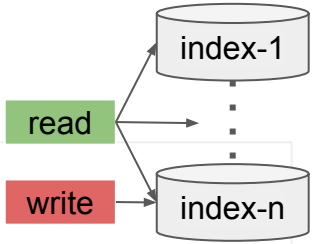
Aliased

Retrieve created and modified docs:
POST /store-name/entity-type/_search
```
{
:sort [{:modified :desc},
       {:created :desc},
       {:uuid :desc}]
 :search_after: [..., …, ...]
}
```

retrieve deleted docs from events
POST /event/event/_search
```
{
:sort [{:modified :asc},
       {:created :asc},
       {:uuid :asc}]
 :search_after: [..., …, ...]
}
```

Migration: writing target





Unaliased

Bulk upsert for created and modified
POST /_bulk
{
{:_index “v1.0.0-store-name“, :mapping “entity-type”, …}
{:_index “v1.0.0-store-name“, :mapping “entity-type”, …}
{:_index “v1.0.0-store-name“, :mapping “entity-type”, …}
}

Delete_by_query deleted docs in source:
POST /v1.0.0-store-name/entity-type/_delete_by_query
{
  :query {
    :ids {:values [...]}
  }
}

Aliased

Bulk upsert for created and modified
POST /_bulk
{
{:_index “store-name-write“, :mapping “entity-type”, …}
{:_index “index-1“, :mapping “entity-type”, …}
{:_index “index-3“, :mapping “entity-type”, …}
}

Delete_by_query deleted docs in source
(works on multiple indices)
POST /store-name/entity-type/_delete_by_query
{
  :query {
    :ids {:values [...]}
  }
}

Rollover Target Store

When migrating, a *_rollover* request is sent after an insert of a batch when:

- the store is configured as *aliased*
- the size of current index exceed a multiple of *max_docs* by less than *batch-size*. (*max_age* is not applicable).
- Code:

```
(s/defn rollover? [aliased? max_docs batch-size migrated-count]
   (and aliased?
        max_docs
        (> migrated-count max_docs)
        (<= (mod migrated-count max_docs)
            batch-size)))
```

Migration state to restart and trigger roll over

- The prefix used for target store, used to build the target index name from source name, *v{prefix}_source-name*.
- for each store it records the `started` and `completed` date. The start date is used to search deletes that occurred during migration in event store.
- State of reads in source and writes in target.
  - the `total` number of red documents in source and actually `migrated` in target (some document might be corrupted and thus not migrated, some documents might be red and migrated twice when updated during migration)
  - the `index` name of the source (the target index name is built using prefix).
  - the `search_after` value to use for next search in source

Migrating with limited downtime

https://github.com/threatgrid/ctia/blob/master/migration.md

Migrating with limited downtime but temporary inconsistency

https://github.com/threatgrid/ctia/blob/master/migration.md

lein run -m
ctia.task.migration.migrate-es-stores
--migrations 1.0.0
--prefix 1.1.0
--stores tool,malware
--batch-size 3000
--id migration-1
--confirm

- STOP CTIA
- lein run -m
  ctia.task.migration.migrate-es-stores
  --batch-size 3000
  --id migration-1
  --confirm
  --restart

- Modify CTIA configuration with new store names
- Restart CTIA