NDBI040: PRACTICAL CLASS 3

RIAKKV

Based on NDBI040 practical class materials created by Martin Svoboda; Tutor: Pavel Čontoš; November 4th 2020

(RECOMMENDED) REQUIREMENTS

- macOS / Linux command line or PuTTy / WinSCP on Windows
- TextEdit, nano, Notepad or any other simple text editor

SERVER ACCESS

CONNECT TO NOSQL SERVER

- `ssh` on macOS / Linux
- `PuTTy` on Windows

- nosql.ms.mff.cuni.cz:42222
- Login and password send by e-mail
- Change your initial password (if not yet changed) by `passwd`

TRANSFER FILES

- `scp` on macOS / Linux
- `WinSCP` on Windows

RIAKKV

- Highly available distributed key-value store
- http://basho.com/products/riak-kv/

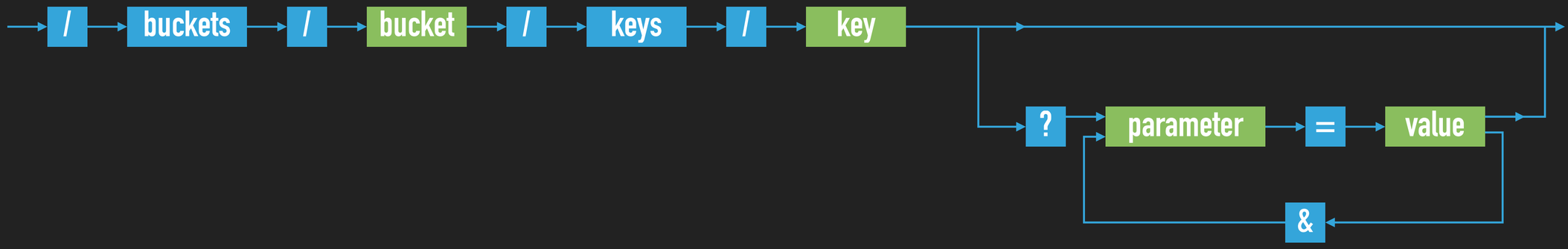
DATA MODEL

- `Instance` (→ bucket types) → `buckets` → `objects`
- `Bucket` is a logical collection of objects
- `Object` is a key-value pair with metadata
  - `Key` is a Unicode string, unique within a bucket
  - `Value` can be anything (text, binary object, image, ...)
  - Each object is also associated with metadata

CRUD OPERATIONS

HTTP API

- All the user requests are submitted as `HTTP requests` with appropriately selected / constructed methods, URLs, headers and data
- URL pattern of HTTP requests for all the `CRUD operations`



- `Optional parameters` (depending on the operation)

CRUD OPERATIONS

BASIC OPERATIONS ON OBJECTS

- Create: `POST` or `PUT` methods
  - Inserts a key-value pair into a given bucket
  - Key is specified manually, or will be generated automatically
- Read: `GET` method
  - Retrieves a key-value pair from a given bucket
- Update: `PUT` method
  - Updates a key-value pair in a given bucket
- Delete: `DELETE` method
  - Removes a key-value pair from a given bucket

HTTP API

- cURL tool
  - Allows to transfer data from / to a server using HTTP (or other supported protocols)

OPTIONS

- `-X command, --request command`
  - HTTP request method to be used (GET, ...)
- `-d data, --data data`
  - Data to be sent to the server (implies the POST method)
- `-H header, --header header`
  - Extra headers to be included when sending the request
- `-i, --include`
  - Prints both headers and (not just) body of a response

FIRST STEPS

- Check Riak cluster status
  - `curl -v http://localhost:10011/ping`

- And with higher permissions...
  - `riak ping`
  - `riak-admin test`
  - `riak-admin status`
  - `riak-admin status | grep ring_members`

READ AND WRITE OPERATIONS

INSERT OBJECT FOR A NEW ACTOR

Do not access your buckets directly
Use $(whoami) instead of M201LOGIN

- Prefix all the bucket names with your M201LOGIN
- `curl -i -X PUT -H 'Content-Type: text/plain' -d 'Ivan Trojan, 1964' http://localhost:10011/buckets/$(whoami)_actors/keys/trojan`

RETRIEVE THE PREVIOUSLY INSERTED ACTOR

- Examine both response body and headers
- `curl -i -X GET http://localhost:10011/buckets/$(whoami)_actors/keys/trojan`

BUCKET OPERATIONS

LIST ALL THE BUCKETS

- Only buckets with at least one object will be included
- `curl -i -X GET http://localhost:10011/buckets?buckets=true`

LIST ALL THEY KEYS IN THE BUCKET OF ACTORS

- Note that this operation cannot be executed efficiently
- `curl -i -X GET http://localhost:10011/buckets/$(whoami)_actors/keys?keys=true`

UPDATE AND DELETE OPERATIONS

UPDATE OUR ACTOR OBJECT

- `curl -i -X PUT -H 'Content-Type: application/json' -d '{ "name" : "Ivan Trojan", "year" : 1964 }' http://localhost:10011/buckets/$(whoami)_actors/keys/trojan`

CHECK THE UPDATED ACTOR OBJECT

- Use different virtual nodes as well
- `localhost:10011`, `localhost:10012`, `localhost:10013`

REMOVE THE ACTOR OBJECT

- `curl -i -X DELETE http://localhost:10011/buckets/$(whoami)_actors/keys/trojan`

EXERCISE 1: SAMPLE DATA

- Insert objects for new actors
  - Put the data into `$(whoami)_actors` bucket
  - Use `application/json` content type
  - Make sure that suffixes recognizable by the JSON extractor were added (we will need later)
  - Do not use Czech accented characters

```
{ "name_s" : "Ivan Trojan", "year_i" : 1964 }
{ "name_s" : "Jiri Machacek", "year_i" : 1966 }
{ "name_s" : "Jitka Schneiderova", "year_i" : 1973 }
{ "name_s" : "Zdenek Sverak", "year_i" : 1936 }
```

SAVE YOUR COMMANDS
YOU WILL NEED THEM LATER AGAIN

EXERCISE 1: ADDITIONAL SAMPLE DATA

- Insert objects for new movies
  - Put the data into `$(whoami)_movies` bucket
  - Use `application/json` content type once again

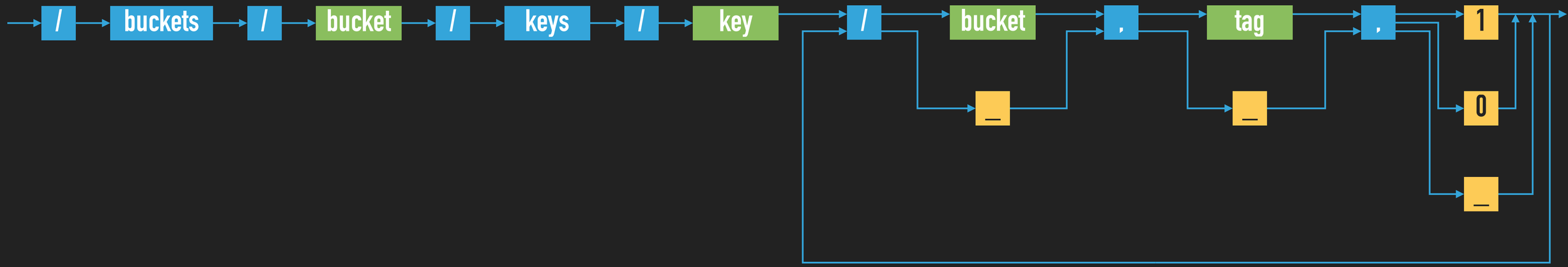
```
{ "title" : "Vratne lahve", "year" :  2006,
  "actors" : [ "Zdenek Sverak" , "Jiri Machacek" ] }

{ "title" : "Samotari", "year" :  2000,
  "actors" : [ "Jitka Schneiderova" , "Ivan Trojan", "Jiri Machacek" ] }

{ "title" : "Medvidek", "year" :  2007,
  "actors" : [ "Jiri Machacek" , "Ivan Trojan" ] }
```

LINKS AND LINK WALKING

- `Links` are directed relationships between objects



PARAMETERS

- `Bucket` assumes only a given target bucket
- `Tag` considers only a given link tag
- `Keep` whether the objects should be included in the result

EXERCISE 2: LINKS AND LINK WALKING

- Create new links `actor` → `movie` for all the actors
- `curl -i -X PUT -H 'Content-Type: application/json' -H 'Link: </buckets/m201_login_movies/keys/samotari>; riaktag="tmovie"' -H 'Link: </buckets/m201_login_movies/keys/medvidek>; riaktag="tmovie"' -d '{ "name_s" "Ivan Trojan", "year_i" : 1964 }' http://localhost:10011/buckets/$(whoami)_actors/keys/trojan`

- Check the updated actor object
- Verify the presence of links in particular

TRAVERSE THE LINKS FROM THE ACTOR

- `curl -i -X GET http://localhost:10011/buckets/$(whoami)_actors/keys/trojan/$(whoami)_movies,tmovie,1`

EXERCISE 3

- First, add all the links movie → actor

- Next, express a more complicated link walking query:
  - Find all the actors who appeared in movies where Trojan stared

SEARCH 2.0

CREATE A FULL-TEXT INDEX FOR THE BUCKET OF ACTORS

- `curl -i -X PUT -H 'Content-Type: application/json' -d '{ "schema" : "_yz_default" }' http://localhost:10011/search/index/$(whoami)_iactors`

- `curl -i -X PUT -H 'Content-Type: application/json' -d '{ "props" : { "search_index" : "m201_login_iactors" } }' http://localhost:10011/buckets/$(whoami)_actors/props`

VERIFY THE NEW BUCKET PROPERTIES

- `curl -i -X GET http://localhost:10011/buckets/$(whoami)_actors/props`

SEARCH 2.0

- Reinsert objects for all the actors
  - see page #12 and your notepad (or history of commands)

```
{ "name_s" : "Ivan Trojan", "year_i" : 1964 }
{ "name_s" : "Jiri Machacek", "year_i" : 1966 }
{ "name_s" : "Jitka Schneiderova", "year_i" : 1973 }
{ "name_s" : "Zdenek Sverak", "year_i" : 1936 }
```

- Find all the actors born in 1964
  - `curl -i -X GET 'http://localhost:10011/search/query/m201_login_iactors?wt=json&omitHeader=true&q=year_i:1964'`

EXERCISE 4

- Express a more complicated full-text query
  - Find all the actors who were born in 1960 or later and their name contains substring de

REFERENCES

- Riak documentation
  - https://docs.riak.com/riak/kv/latest/
- Search queries (Apache Solr query syntax)
  - https://docs.riak.com/riak/kv/latest/developing/usage/search/
  - https://lucene.apache.org/solr/guide/6_6/the-standard-query-parser.html