Database Lab

Checkpoint 0: Project Updates

There are real complications of Open Source Software that arise from needing to manage community rather than just the technical details. Three (kind of recent?) well-publicized events highlight these complexities. We will discuss these during class on August 12. Below are some resources for you to review. Google around to find more. You should make one (or more) of these topics the subject of your blog this week. Put a link to your blog entry into your lab report and be ready to discuss what you write (in some format ... a debate, round table discussion, or maybe groups of single topics) during class on the twelfth.

Consider different aspects of the cases ... Maybe what other members of the community might think about the actions? Were the actions justified? Do they ultimately help or hurt? What are the unintended consequences. There are no real right or wrong answers.

I encourage you to discuss these topics with your neighbors. Then each of you please write at least one well-formed paragraph on any part of this and post it to your Blog. Put a link to your blog in your Lab Report for today.

- MongoDB License change (You can find many examples, these are just a few. Google it!)
  - Percona immediately after the change
  - Same company a year later
  - License and some analysis
  - One more for good luck!
- Faker.js and Color.js
  - Summary
  - Additional Details
- Node IPC
  - Summary
- GitHub Copilot
  - Software Freedom Conservancy
  - OSI
  - ZDNet
  - Deep Dive AI

For those of you who are conspiracy minded, you might want to consider what steps you could take to undermine the legitimacy of the Open Source movement and to discredit it. Are those steps any different from the first three bullets? For Copilot, there are several questions. Is Copilot trained and used legally? Ethically? Does it violate copyright or licensing?

Checkpoint 1: Install CouchDB

For the first checkpoint of this lab, we'll work on just getting a proper running instance of CouchDB. CouchDB has some good documentation for Linux, OSX and Windows here:

https://docs.couchdb.org/en/stable/install/index.html

I have personally installed it in my WSL2 Ubuntu using the Linux instructions in Section 1.1.1 and using the instructions for Docker. I have not tried the rest. Going by what I found, I expect that the install instructions will *almost* work. You may need to play a little.

In particular, for **WSL2 Ubuntu**, the installation assumes that it is installed in the directory `/home/couchdb/bin/couchdb` when it actually installs it in `/opt/couchdb/bin/couchdb`. And for **Docker**, be sure to export the port 5984 using `-p 5984:5984`. Note that by default any data you put into couchdb in a Docker container will go away when you kill the image. If you want it to persist, you will need to use `-v /opt/couchdb/data:<mydatadirectory>` to bind the docker database directory to actual storage on your machine.

I expect similar minor issues are in most of the install instructions.

Whichever, way you choose to install couchdb, I recommend you do it stand-alone for simplicity and that you enable replication. Make sure you complete the setup and if you don't give an administrator name, it is probably *admin*.

Once you are running, open a browser and go to `http://localhost:5984`

Add a screenshot of the message you get to your lab report.

It should look something like:

```
{"couchdb":"Welcome",
 "version":"3.2.1",
 "git_sha":"244d428af",
 "uuid":"a3cc56d36b644ee376e45661aeacf43b",
 "features":["access-ready","partitioned","pluggable-storage-engines",
             "reshard","scheduler"],
 "vendor":{"name":"The Apache Software Foundation"}
}
```

Checkpoint 2: Quick Tour

Now you are going to walk through the first part of the *couchdb* Tutorial at https://docs.couchdb.org/en/stable/intro/tour.html. This is the demo I did in class. It requires cURL and a browser. If you don't have cURL you can install it from: https://curl.se/download.html.)

Grab screenshots and capture your output from cURL and Fauxton as you go and put them in your lab report.

Checkpoint 3: Now Complete the API Tutorial

Walk through https://docs.couchdb.org/en/stable/intro/api.html

You will run into an issue when working with the `artwork.jpg` example. It looks like the example was not updated when the security model of ***CouchDB*** was modified. Regardless of the reason, the line

```
http://127.0.0.1:5984/albums/6e1295ed6c29495e54cc05947f18c8af/artwork.jpg
```

will probably not work without some modifications to the instructions. To see the attached file you can either browse to the record in the database using `fauxton` or you can use `fauxton` to modify the permissions on the ***albums*** database to allow anyone to access it and then retry the above link. I expect there are other ways as well, so feel free to experiment.

Similarly, you may have problems with the line

```
curl -vX POST http://admin:password@127.0.0.1:5984/_replicate \
     -d '{"source":"http://127.0.0.1:5984/albums",\
          "target":"http://127.0.0.1:5984/albums-replica"}' \
     -H "Content-Type: application/json"
```

With the new permissions structure, you will likely need to give an account name and password for each of the *source* and *target* repositories.

```
curl -vX POST http://admin:password@127.0.0.1:5984/_replicate \
     -d '{"source":"http://admin:password@127.0.0.1:5984/albums",\
          "target":"http://admin:password@127.0.0.1:5984/albums-replica"}' \
     -H "Content-Type: application/json"
```

Finally, stop after you get local replication working. Don't worry about remote replication.

Grab screenshots and capture your output as you go and put them in your lab report.

Checkpoint 4: What Did We Miss?

We covered most of the necessary database operations, but one thing we didn't cover was finding selected documents via **cURL**. If you look at the Fauxton Tutorial, you were able to do a find using *Mango* and two *json* structures: an **index** and a **selector**. Technically, the *index* is optional, but for large data bases it is necessary to make efficient queries. Let's see what happens if we don't have it first.

1. The following cURL command can be used to select movies from our **hello-world** database based on the given year:

   ```
   curl -X POST admin:password@localhost:5984/hello-world/_find -d '{
      "selector": {
         "year": {
   ```

```
        "$gt": 1987
    }
  }
}' -H 'Content-Type: application/json'
```

Note that this should run correctly and not give an error messages because we previously defined an index for movies based on year in Fauxton. **Run it and paste the results in your lab report.**

2. Modify the above command to find all movies whose titles come after "L" in alphabetic order. (You can be more creative here if you would like, but this is a simple change of indices.) **Run your command and paste it and your results, including any warnings, into your lab report.**

3. The warning you got in 2 is telling you that this is an inefficient operation. Couchdb wants to optimize your queries so that it can do them fast. In order to do this, it needs to generate a B-Tree based on whichever fields we may want to search on. However, since documents can have arbitrarily large numbers of fields, and databases can be huge, this "indexing" can take a long time and eat up a lot of space. We need to tell couchdb which fields we are interested in. Below is the cURL call to create the "year-json-index" index you created in Fauxton.

```
curl -X POST admin:password@localhost:5984/hello-world/_index -d '{
   "index": {
      "fields": [
         "year"
      ]
   },
   "name": "year-json-index",
   "type": "json"
}' -H 'Content-Type: application/json'
```

Based on the above command, you should be able to create a new commmand to generate an index based on title. **Run your command and paste it and your results into your lab report.**

4. **Finally, rerun your movie title query and paste it and your results into your lab report.** It won't be any faster because we only have 3 movies in our database, but it won't complain any more, and it will remain fast as our database grows.

When you are finished, push your report to your lab notebook and submit a link to the repository on Submitty.