Lab 11.1: Introduction to Riak
IN705 Databases Three

Introduction

Riak is a distributed key-value database that is designed to provide scalability and fault-tolerance. Because it is a simple key-value store, it doesn't provide any of the machinery that a relational database provides for maintaining data consistency or performing arbitrary queries (although is provides powerful capabilities of a different type, as we shall see later this week). Keys are simply strings, and the values are basically arbitrary bits from Riak's point of view. Riak can store formatted text like JSON or XML, but it can just as easily store binary file data, like images.

1 Installing and Starting Riak

Install Riak on your ec2 server with the commands:

```
curl https://packagecloud.io/install/repositories/basho/riak/script.deb | sudo bash
sudo apt-get install riak
```

Before we start the Riak service, we need to modify its start stop script. Add the following line to the file `/etc/init.d/riak` at line 40, right about the line that says `start-stop-daemon -start`.

File: /etc/init.d/riak

```
ulimit -n 65536
```

Then you can start the Riak server with the command

```
sudo service riak start
```

2 Adding and retreiving data with cURL

Riak provides a REST interface that is supported by libraries in many languages. For simplicity, we'll just interact with it directly using cURL.

First, put something in your database with the following command:

```
user@riak:~$curl -v -X PUT  http://localhost:8098/riak/hello/world \
> -H "Content-Type: text/html" \
> -d "<html><body><h1>Hello, world</h1></body></html>"
```

We used cURL to make an HTTP put request of our Riak cluster. We told Riak that our value to be stored was HTML formatted text, and then we supplied the text. Riak organises data using *buckets* and *keys*. In the above request we used the bucket **hello** and the key **world**.

You can see your data pointing a browser at `http://<your-server-ip>:8098/riak/hello/world` to perform an HTTP get request.

Now let's create a new bucket to store information about animals.

```
user@riak:~$curl -v -X PUT  http://localhost:8098/riak/animals/bob \
> -H "Content-Type: application/json" \
> -d '{"name":"Bob", "species":"possum"}'
```

Add a few more animals on your own.

Besides retrieving individual records, you can also view your buckets:

```
user@riak:$ curl http://localhost:8098/riak?buckets=true
```

or see all the values in a bucket

```
user@riak:~$ curl http://localhost:8098/buckets/hello/keys?keys=true
```

3 Links

A key-value store like Riak doesn't handle relationships with foreign keys, but it can store metadata about a value. One important kind of metadata is a *link* that can represent relationships between data values.

Let's put our possum, Bob, in a cage.

```
user@riak:~/dev$ curl -v -X PUT http://localhost:8098/riak/cages/1 \
> -H "Content-Type: application/json" \
> -H "Link: </riak/animals/bob>; riaktag=\"contains\"" \
> -d '{"room":"D311"}'
```

I guess the cage is in my office.

Now we can see what animal is on cage 1 using a technique called *link walking*:

```
user@riak:~/dev$ curl http://localhost:8098/riak/cages/1/_,contains,_
```

The comma-separated values on the end of the URL above indicate which links we want to follow. Underscores indicate wildcards. The first position indicates what bucket we want linked values from. Putting "animals" in this position will retrieve only links to animals. The second postion indicates what riaktags we want to follow. In the example above we looked for the links tagged "contains". The third value may be 0 or 1 and indicates if we want to keep intermediate values when we follow a chain of links.

4 Other MIME types and binary data

Riak is perfectly happy to store binary data. Find a picture of a possum online and copy it to your Riak server. Suppose it is named "bob.jpg". We can store the picture and link it to Bob like this:

```
user@riak:~/dev$ curl -v -X PUT http://localhost:8098/riak/photos/bob \
> -H "Content-Type: image/jpeg" \
> -H "Link: </riak/animals/bob>; riaktag=\"photo\"" \
> --data-binary @bob.jpg
```

5 Exercises

1. Create another animal. Link it to Bob the possum and tag the link `friend-of`.
2. Now create a new cage and place your animal in the cage with the appropriate tag. You can now walk the links back to Bob with a URL like
   `http://localhost:8098/riak/cages/2/_,contains,1/_,friend-of,_`
3. What happens if you change the 1 to a 0 in the URL above?
4. Can you change the link-walking parameters above and get the same results?
5. Find an image of some bush and download it to your Riak server. Create a new bucket called habitats and upload your image into the bucket with the appropriate MIME type. Link the image to Bob the possum.
6. Read the Riak documentation online at basho.com to see how to delete a value and try it with your database.