LEARNING rethinkdb

#rethinkdb

Table of Contents

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: rethinkdb

It is an unofficial and free rethinkdb ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official rethinkdb.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with rethinkdb

Remarks

What is RethinkDB?

- **Open-source** database for building realtime web applications
- **NoSQL** database that stores schemaless JSON documents
- **Distributed** database that is easy to scale
- **High availability** database with automatic failover and robust fault tolerance

RethinkDB is the first open-source scalable database built for realtime applications. It exposes a new database access model -- instead of polling for changes, the developer can tell the database to continuously push updated query results to applications in realtime. RethinkDB allows developers to build scalable realtime apps in a fraction of the time with less effort.

Versions

Version	Release Date
2.3.5	2016-08-27
2.3.4	2016-06-03
2.3.3	2016-05-31
2.3.2	2016-05-05
2.3.1	2016-04-22
2.3.0	2016-04-05

Examples

Installation on OS X

Using Homebrew

Prerequisites: Make sure you're on OS X 10.9 (Mavericks) or above, and have Homebrew installed.

Run the following in your terminal:

```
brew update && brew install rethinkdb
```

Compile from source

Building RethinkDB from source requires OS X 10.9 (Mavericks) or greater. Xcode is required to build from source.

Get the source code

Download and extract the archive:

```
wget https://download.rethinkdb.com/dist/rethinkdb-2.3.4.tgz
tar xf rethinkdb-2.3.4.tgz
```

Build RethinkDB

Kick off the build process:

```
cd rethinkdb-2.3
./configure --allow-fetch --fetch openssl
make
```

You will find the `rethinkdb` binary in the `build/release/` subfolder.

Installation on Ubuntu

Ubuntu binaries are available for both 32-bit and 64-bit architectures

```
source /etc/lsb-release && echo "deb http://download.rethinkdb.com/apt $DISTRIB_CODENAME main"
| sudo tee /etc/apt/sources.list.d/rethinkdb.list
wget -qO- https://download.rethinkdb.com/apt/pubkey.gpg | sudo apt-key add -
sudo apt-get update
sudo apt-get install rethinkdb
```

Installation on Windows

Downloading

Prerequisites: We provide native 64-bit binaries for Windows 7 and above. A 64-bit version of Windows is required.

Download the ZIP archive and unpack it in a directory of your choice.

The Windows port of RethinkDB is a recent addition and hasn't received as much tuning as the Linux and OS X versions yet. Please report any performance issues on GitHub.

Running RethinkDB

The Windows version of RethinkDB, like the Linux/OS X versions, is executed from the command line. You'll need to start the Windows command shell.

- Press `Win` + `X` and click "Command Prompt"; or
- Open the Start Menu, click "Run," and type "cmd" `ENTER`

Use the `cd` command to go to the directory that you unpacked `rethinkdb.exe` in.

```
C:\Users\Slava\>cd RethinkDB
C:\Users\Slava\RethinkDB\>
```

Then, you can start RethinkDB with its default options.

```
C:\Users\Slava\RethinkDB\>rethinkdb.exe
```

You can also use any of the command line options to control configuration (as well as specify a configuration file).

To start with a specific data directory:

```
rethinkdb.exe -d c:\RethinkDB\data\
```

To specify a server name and another cluster to join:

```
rethinkdb.exe -n jarvis -j cluster.example.com
```

Read Getting started with rethinkdb online: https://riptutorial.com/rethinkdb/topic/2498/getting-started-with-rethinkdb

Chapter 2: Getting started with Node

Examples

Installing the RethinkDB package from NPM

```
npm install -g rethinkdb
```

Making a connection to RethinkDB

const r = require("rethinkdb");

```
r.connect({host: 'localhost', port: 28015}, (conn) => console.log(conn))

// Or as a promise

let rdb_conn;
r.connect({host: 'localhost', port: 28015}).then((conn) => {
  rdb_conn = conn;
}).then(() => {
  // Continue to use rdb_conn
});
```

Listing all databases

```
r.connect({host: 'localhost', port: 28015})
.then((conn) => {
    return r.dbList().run(conn);
}).then((result) => {
    // Prints out list of databases on the RethinkDB instance
    console.log(result);
});
```

Create a new database

```
r.connect({host: 'localhost', port: 28015})
.then((conn) => {
    return r.dbCreate("stackoverflow").run(conn);
}).then((result) => {
    console.log(result);
});
```

Create a new table on a database

```
r.connect({host: 'localhost', port: 28015})
.then((conn) => {
    return r.db("stackoverflow").tableCreate("examples").run(conn);
}).then((result) => {
    console.log(result);
});
```

Insert a document into a table

```
r.connect({host: 'localhost', port: 28015})
.then((conn) => {
    return r.db("stackoverflow").table("examples")
        .insert({
            // If `id` is not set, will automatically generate a UUID
            id: 1,

            name: 'Thinker',

            // Will translate Date types.
            creationDate: new Date(),

            // Embedded array
            tags: ['rethinkdb', 'rethinkdb-javascript', 'rethinkdb-python'],

            // Will evaluate `r.now()` using server time
            dateWithServerTime: r.now(),

            // Embedded document
            location: {
                // Using geospatial example
                coordinates: r.point(-122.423246,37.779388),
                name: 'San Francisco'
            },
        }).run(conn);
}).then((result) => {
    // Returns results object which includes array of generated UUIDs
    //  for inserted documents
    console.log(result);
});
```

Querying a document from a table

```
r.connect({host: 'localhost', port: 28015})
.then((conn) => {
    // Can also use .get({id: 1})
    return r.db("stackoverflow").table("examples").get(1).run(conn)
}).then((result) => {
    console.log(result);
})
```

Read Getting started with Node online: https://riptutorial.com/rethinkdb/topic/6475/getting-started-with-node

Chapter 3: Using RethinkDB with Docker

Examples

Basic usage

By default, RethinkDB binds all services to `127.0.0.1`. So this following example will persist data to the `host_data_path` on the container's host machine and available to `127.0.0.1` on the standard ports.

Service	Flag	Default Port
Driver	`--driver-port`	28015
Cluster	`--cluster-port`	29015
HTTP WebUI	`--http-port`	8080

```
docker run -d -v host_data_path:/data rethinkdb
```

To open up the driver and cluster port to the external traffic you must specify the address of local interfaces or provide `all`.

```
docker run -d -v host_data_path:/data rethinkdb --bind all
```

Binding WebUI to localhost or disabling

When deploying RethinkDB in production, you want to either turn off or lock down the WebUI. This will only respond to `localhost` to access the WebUI allowing you to SSH Tunnel to the host machine and access it for diagnostics and troubleshooting.

```
docker run -d \
   -v host_data_path:/data \
   rethinkdb \
   rethinkdb --bind-cluster all --bind-driver all --bind-http 127.0.0.1 -d /data
```

If you'd like to completely turn off the WebUI:

```
docker run -d \
  -v host_data_path:/data \
  rethinkdb \
  rethinkdb --bind-cluster all --bind-driver all --no-http-admin -d /data
```

Read Using RethinkDB with Docker online: https://riptutorial.com/rethinkdb/topic/4499/using-rethinkdb-with-docker

Chapter 4: using thinky.io with RethinkDB

Examples

Starting thinky in node.js

thinky is a lightweight node.js ORM for RethinkDB.

First you need to have RethinkDB running on your server.

Then install the thinky.io npm package into your project.

```
npm install --save thinky
```

Now import thinky into your model file.

```
const thinky = require('thinky)();
const type = thinky.type
```

Next create a model.

```
const User = thinky.createModel('User' {
    email:type.string(),
    password: type.string()
});
```

You can now create and save a user.

```
const user = new User({
    email: 'test@email.com',
    password: 'password'
});

user.save();
```

The user will be given a unique id.

You can chain a promise onto the save function to use the resulting user, or catch an error.

```
user.save()
    .then(function(result) {
        console.log(result);
    })
    .catch(function(error) {
        console.log(error);
    });
```

Find your user using functions such as filter, and use promises to use the results.

```
User.filter({ email: 'test@email.com }).run()
    .then(function(result) {
        console.log(result);
    })
    .catch(function(error) {
        console.log(error);
    });
```

Or search for a specific user by the unique id.

```
User.get(id)
    .then(function(user) {
        console.log(user);
    })
    .catch(function(error) {
        console.log(error);
    });
```

Read using thinky.io with RethinkDB online: https://riptutorial.com/rethinkdb/topic/4068/using-thinky-io-with-rethinkdb

Credits

S. No	Chapters	Contributors
1	Getting started with rethinkdb	Community, dalanmiller, Matteo
2	Getting started with Node	dalanmiller
3	Using RethinkDB with Docker	dalanmiller
4	using thinky.io with RethinkDB	alexi2