Installing MongoDB

https://docs.mongodb.org/v3.0/installation/
What is MongoDB?

Most popular NoSQL database program.
Why use MongoDB?
Why use MongoDB?

- Very efficient when you need to do a lot of writes
Why use MongoDB?

- Very efficient when you need to do a lot of writes
- You have a lot of data (partitioning and sharding is easy)
Why use MongoDB?

- Very efficient when you need to do a lot of writes
- You have a lot of data (partitioning and sharding is easy)
- Schema is very prone to change
Structure

- MongoDB Instance

```
echentw@pudding:~$ sudo mongod
mongod --help for help and startup options
2017-01-12T02:49:49.729-0500 [initandlisten] MongoDB starting : pid=30368 port=27017 dbpath=/data/db 64-bit host=pudding
2017-01-12T02:49:49.730-0500 [initandlisten] db version v2.6.10
2017-01-12T02:49:49.730-0500 [initandlisten] git version: nogitversion
2017-01-12T02:49:49.730-0500 [initandlisten] OpenSSL version: OpenSSL 1.0.2d 9 Jul 2015
2017-01-12T02:49:49.730-0500 [initandlisten] allocator: tcmalloc
2017-01-12T02:49:49.730-0500 [initandlisten] options: {}
2017-01-12T02:49:49.845-0500 [initandlisten] journal dir=/data/db/journal
2017-01-12T02:49:49.845-0500 [initandlisten] recover: no journal files present, no recovery needed
2017-01-12T02:49:49.910-0500 [initandlisten] waiting for connections on port 27017
```
Structure

● MongoDB Instance
  ○ Database
Structure

- MongoDB Instance
  - Database
    - Collections
Structure

- MongoDB Instance
  - Database
    - Collections
  - Documents

```json
[{
  "name": "David",  "age": 12,
},
{
  "name": "Hunter",  "age": 14,
},
{
  "name": "Aashish",  "age": 10
}]
```
MongoDB Shell

- Interactive interface to MongoDB
- Can read and write to your database
- Uses ‘test’ database by default
Start a MongoDB Instance

- A MongoDB instance exists within a directory
- In your terminal:
  - mkdir data (only directory doesn’t exist)
  - mongod --dbpath path/to/data
- OR
  - mongod (will use the default directory)
Mongo Shell

- To start the Mongo shell (in your terminal):
  - `mongo`

- (If you are a Windows user, call `mongo` from within the `bin` folder where MongoDB is installed)
Mongo Shell Commands

- `show collections` - list all the collections in the database
- `db.<collection name>.find()` - list all the documents in the collection
- `db.<collection name>.find(obj)` - list all the documents in the collection which match the key value pairs in `obj`
Mongo Shell Commands (cont’d)

- `db.<collection name>.insert(obj)` - inserts `obj` into the collection
  - If the collection doesn’t exist, also creates the collection
- `db.<collection name>.remove(obj)` - removes all documents in the collection matching the fields in `obj`
- `db.<collection name>.drop()` - deletes the collection from the database
Demo!
Mongoose
What is Mongoose?

- A NodeJS library
- Allows you to integrate MongoDB with your app
Installing Mongoose for Your App

`npm install --save mongoose`

Installs in `node_modules/` and saves to `package.json`!
Setting up the DB with Mongoose

```javascript
var mongoose = require('mongoose');

// connect Mongoose to MongoDB
mongoose.connect(process.env.MONGOLAB_URI || 'mongodb://localhost/test');

var connection = mongoose.connection;

// error handler
connection.on('error', console.error.bind(console, 'connection error:'));

// runs when the connection is successful
connection.on('connected', function() {
  console.log('database connected!');
});
```

Schemas

- Structure for your documents
- Each collection should have a schema
Schemas

- Structure for your documents
- Each collection should have a schema

```json
[
  {
    "name": "David", "age": 12,
  },
  {
    "name": "Hunter", "age": 14,
  },
  {
    "name": "Aashish", "age": 10
  }
]
```
Schemas

- Structure for your documents
- Each collection should have a schema

```json
[
  {"name": "David", "age": 12},
  {"name": "Hunter", "age": 14},
  {"name": "Aashish", "age": 10}
]
```

Schema: {"name": string, "age": number}
Schemas

```javascript
var mongoose = require('mongoose');

var studentSchema = new mongoose.Schema({
  name: String,
  course: Number
});

var Student = mongoose.model('Student', studentSchema);

module.exports = Student;
```
Schemas

```javascript
var mongoose = require('mongoose');

var studentSchema = new mongoose.Schema({
  name: String,
  course: Number
});

var Student = mongoose.model('Student', studentSchema);

module.exports = Student;
```

Specifies the collection: “students”
```javascript
var mongoose = require('mongoose');

var studentSchema = new mongoose.Schema({
  name: {type: String, required: true, index: {unique: true}},
  course: {type: Number, required: true}
});

var Student = mongoose.model('Student', studentSchema);

module.exports = Student;
```
Creating a Document

```javascript
1 var Student = require('./student.js');
2
3 var newStudent = new Student({
4    'name': 'Eric',
5    'course': 6
6 });
7
8 newStudent.save();
9 ```
Finding documents

```javascript
var Student = require('./student.js');

Student.findOne({'name': 'Eric'}, function(err, student) {
  if (err) {
    console.log('An error occurred!');
  }

  console.log('The name of the student is ' + student.name);
  console.log('The course # of the student is ' + student.course);
});
```
Finding documents (pt 2)

```
var Student = require('./student.js');

Student.find({'name': 'Eric'}, function(err, students) {
  if (err) {
    console.log('An error occurred!');
  }

  // 'students' is an array of all the students with name 'Eric'!
  console.log('Number of students named Eric: ' + students.length);
});
```
Finding documents (pt 3)

```javascript
var Student = require('./student.js');

Student.find({
  'name': 'Eric'
  'course': 6
}, function(err, students) {
  if (err) {
    console.log('An error occurred!');
  }

  // 'students' is an array of all the students with name 'Eric'
  // who are course 6!
  console.log('There are ' + students.length + ' course 6 Erics.');
});
```
Editing documents

You can first retrieve the document, and then update.

```javascript
var Student = require('./student.js');

Student.findOne({name: 'Eric'}, function(err, student) {
  if (err) {
    console.log('An error occurred!');
  }

  student.course = 6;
  student.save(); // Update is automatically triggered by calling save.
});
```
Editing documents (pt 2)

Or you can update directly.

```javascript
var Student = require('./student.js');

Student.update({
  'name': 'Eric',
}, {
  $set: {'course': 8}
}, function(err, result) {
  if (err) {
    console.log('An error occurred!');
  }

  // result is an object containing information about the update
});
```
Removing documents

```javascript
1 var Student = require('./student.js');
2
3 // removes all documents with name === 'Eric'
4 Student.remove({'name': 'Eric'}, function(err, result) {
5   if (err) {
6     console.log('An error occurred!');
7   }
8 }
9 // result is an object containing information about the removal
10 });
11```
Mongoose Documentation

https://mongoosejs.com/docs/guide.html
Workshop: go.6148.io/mongodb