

Cloud and Web Application

Part 4: Lab

Build a Course Material Sharer

Table of Contents

- Project Design
- Install Tools
- Implementation
- Deployment

➤ The source code pieces showed in this slide can be found at :

https://github.com/lidangz/2015CourseMaterials/blob/master/Code_pieces_Part4_Lab.txt

➤ The source code files can be downloaded from:

https://github.com/lidangz/2015CourseMaterials/blob/master/Code_Part4_Lab.zip

➤ The full project can be found at:

<https://github.com/lidangz/MaterialSharer>

➤ The tools used in the lab can also be downloaded from:

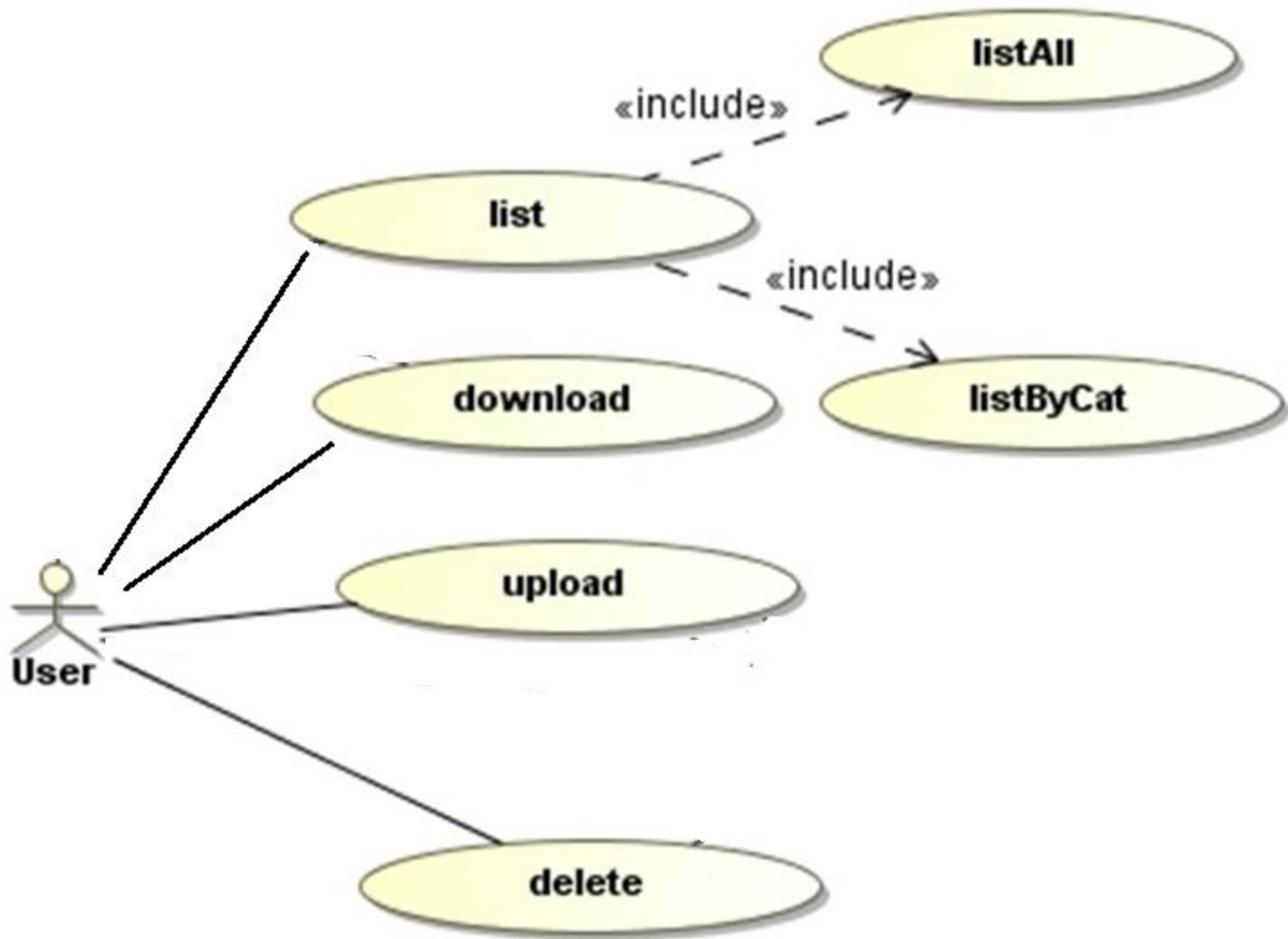
<http://pan.baidu.com/s/1AWzZg#path=%252FCourseTools>

PROJECT DESIGN

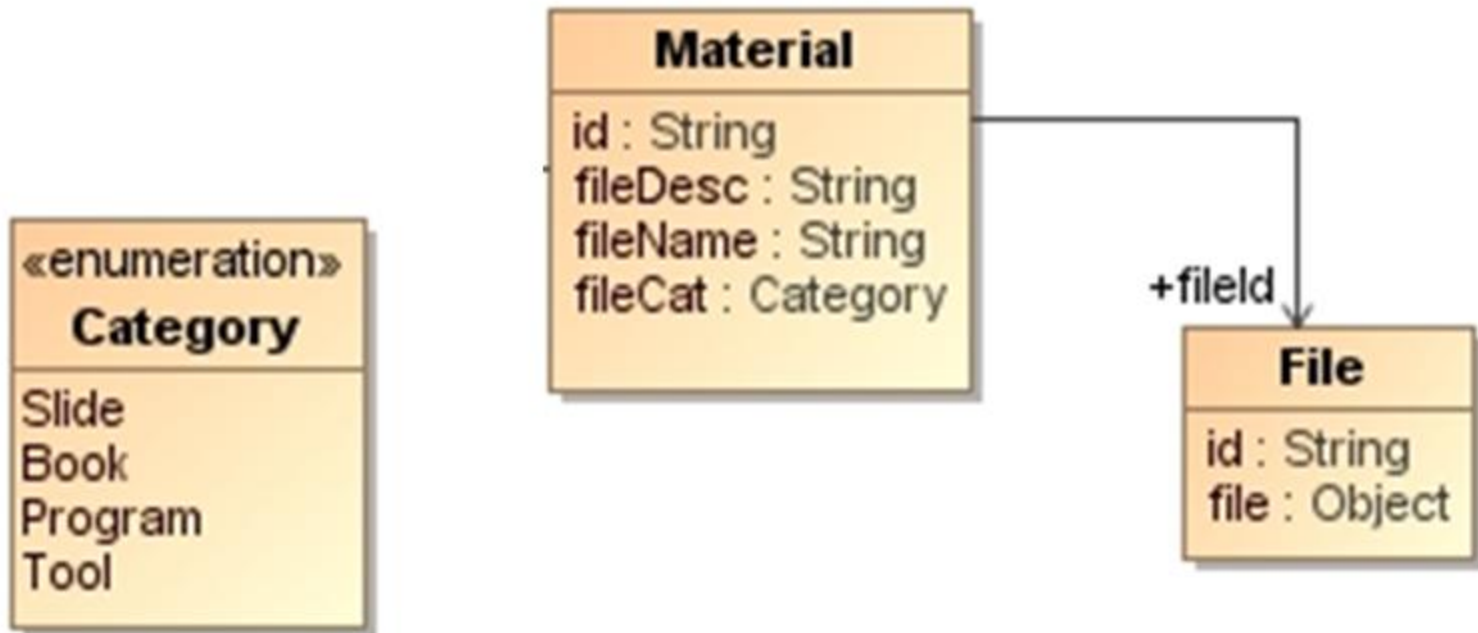
Course Material Sharer : Ideas

- We want to build a course material sharing app. and deploy it in cloud
- A user can **list** the all the sharing materials or a **category** of materials, and **download** the interested one.
- A user can **upload** new material and **delete** the materials.

App Design : Use Case



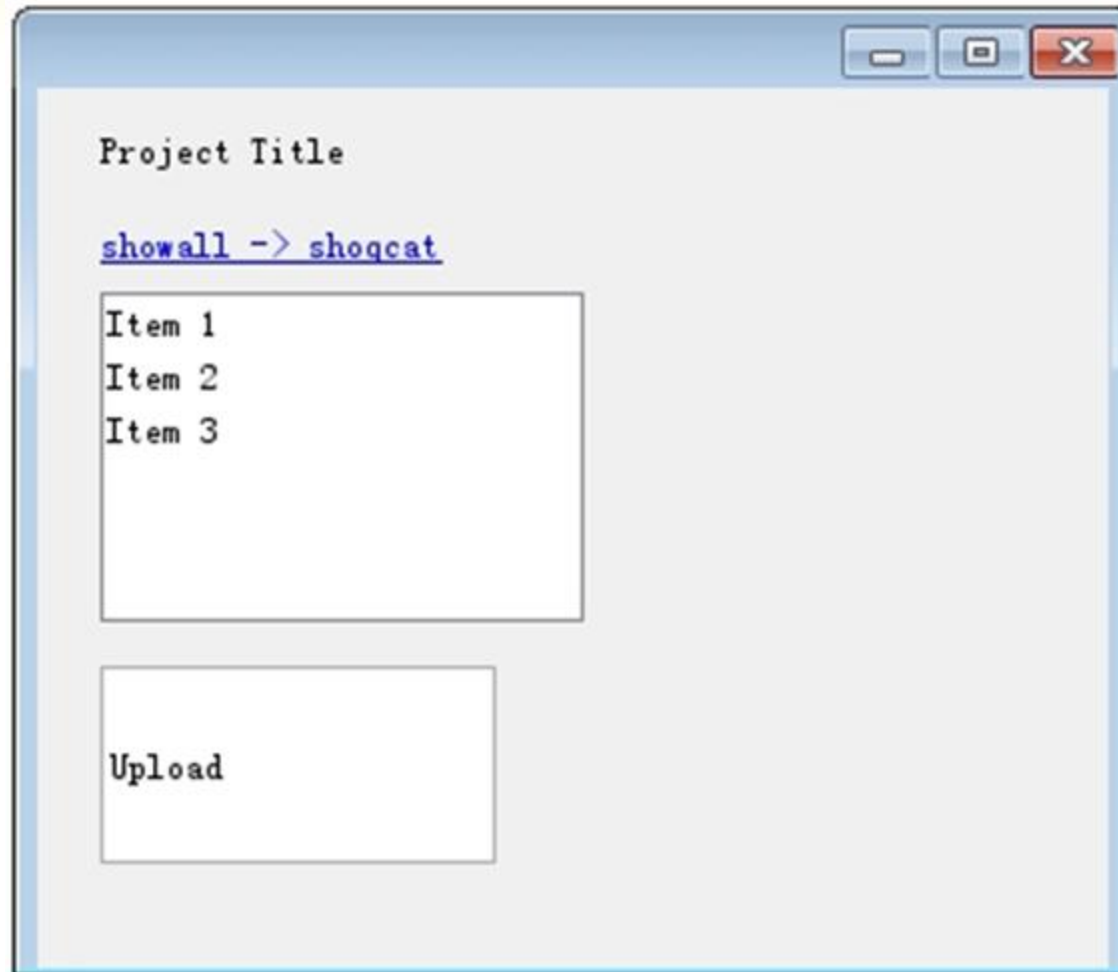
App Design : Class Diagram



Technical Stacks

- Use **HTML** + **JavaScript** + **Ajax** + **REST** in the front-end;
- Use **Node.js** + **MongoDB** + **Express** in the back-end;
- The developing version will be installed on **localhost** using the **8080** port.
- The app will be deployed on the OpenShift online.

UI Design of Single-page App



REST API Design

Action	HTTP Method	URL
listAll	GET	http://localhost:8080/materials
listByCat	GET	http://localhost:8080/materials/{catalog}
upload	POST	http://localhost:8080/materials
delete	DELETE	http://localhost:8080/materials/{materialId}
download	GET	http://localhost:8080/files/{fileId}

INSTALL TOOLS

Install MongoDB

- Download mongoDB from <http://www.mongodb.org/downloads>
(msi for windows)
- Create a new folder *c:\mongodb*, and install mongodb to the folder
- Create a new folder *data* inside the folder *mongodb* to store data

Start MongoDB

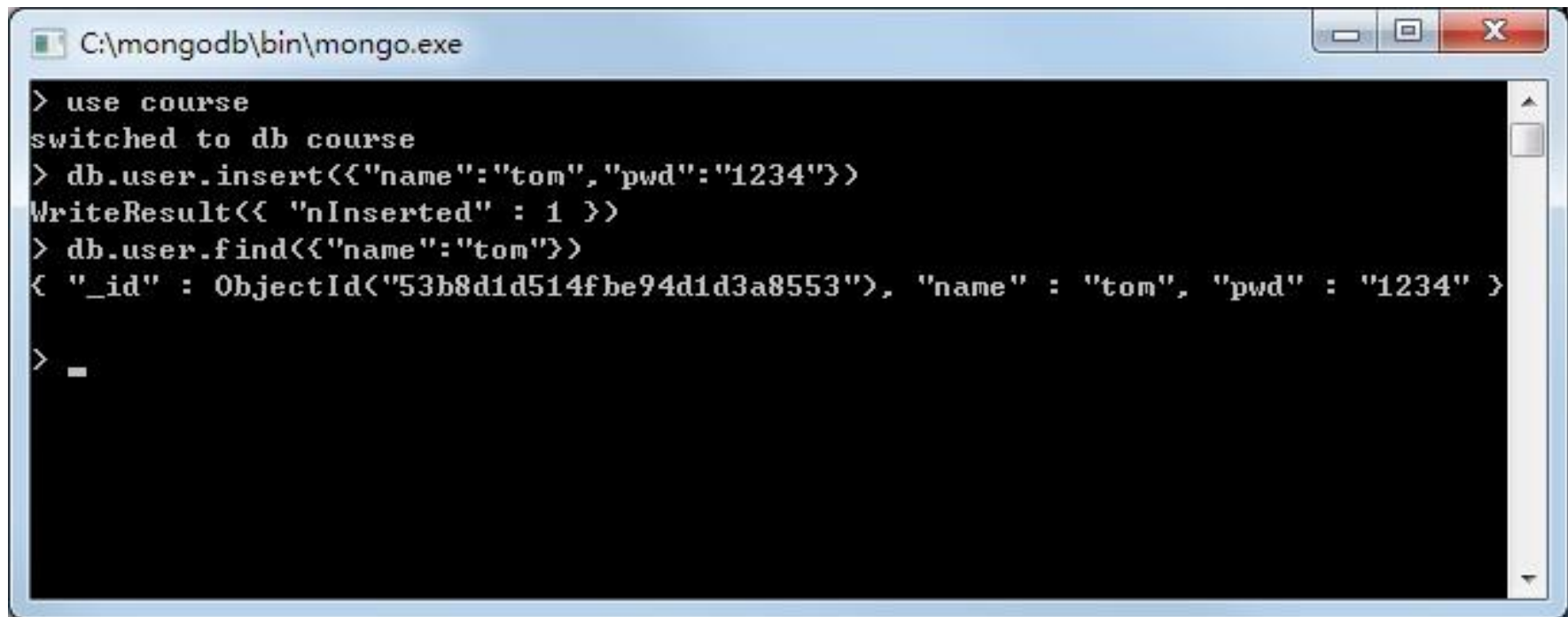
- Open a cmd, goto folder *c:\mongodb\bin*
- Start monfoDB by typing

```
mongod.exe --dbpath c:\mongodb\data
```

- Open another cmd, goto folder *c:\mongodb\bin* , and type *mongo.exe* to open a management window.

Add a User to MongoDB

- Go to the management window.
 - Open database `course`
 - Add to collection `user` a user
- ```
> use course
> db.user.insert({"name":"tom","pwd":"1234"})
> db.user.find({name:"tom"})
```

A screenshot of a Windows command prompt window titled "C:\mongodb\bin\mongo.exe". The window has a black background with white text. The text shows the following commands and their outputs: 1. "> use course" followed by "switched to db course". 2. "> db.user.insert(<{"name":"tom","pwd":"1234"}>)" followed by "WriteResult(< { 'nInserted' : 1 } >)". 3. "> db.user.find(<{"name":"tom"}>)" followed by "< { '\_id' : ObjectId(<'53b8d1d514fbe94d1d3a8553'>), 'name' : 'tom', 'pwd' : '1234' }". 4. "> \_" followed by a cursor. The window has standard Windows window controls (minimize, maximize, close) in the top right corner.

```
C:\mongodb\bin\mongo.exe
> use course
switched to db course
> db.user.insert(<{"name":"tom","pwd":"1234"}>)
WriteResult(< { 'nInserted' : 1 } >)
> db.user.find(<{"name":"tom"}>)
< { '_id' : ObjectId(<'53b8d1d514fbe94d1d3a8553'>), 'name' : 'tom', 'pwd' : '1234' }
> _
```

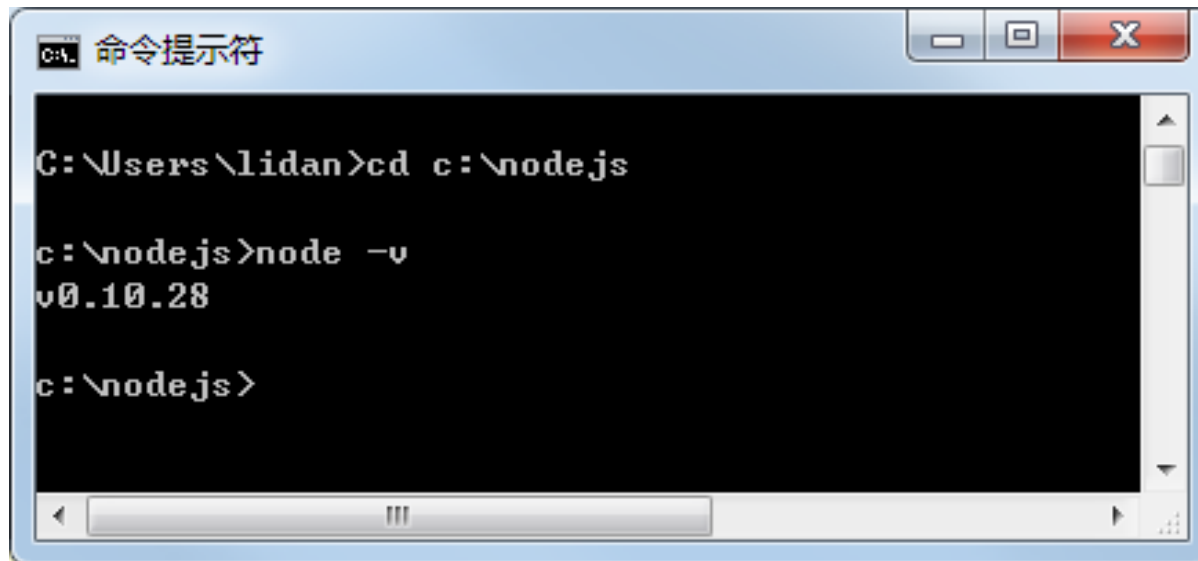
# Install Node.js

- Create a new folder `c:\nodejs`
- Download and install Node.js : <http://nodejs.org/>
  - Click 'Install' or go to the 'Downloads' page
  - Once downloaded, run the installer
  - Install Node.js in directory `c:\nodejs`



# Test Node.js Installation

- In a cmd window, go to `c:\nodejs` directory
- Type `node -v` in the command line
- If the version information is shown, Node.js is correctly installed.



A screenshot of a Windows Command Prompt window titled "命令提示符" (Command Prompt). The window shows the following commands and output:

```
C:\Users\lidan>cd c:\nodejs

c:\nodejs>node -v
v0.10.28

c:\nodejs>
```

# Install OpenShift Client Tools

Installing the client tools (**rhc**) on Windows requires three steps:

- **Step 1:** Install **Ruby** with RubyInstaller
- **Step 2:** Install **Git** version control
- **Step 3:** Install the **rhc** Ruby gem
- <https://developers.openshift.com/en/getting-started-windows.html> for details.



# RubyInstaller



**RubyInstaller**  
for Windows

[About](#) [Download](#) [Help](#) [Contribute](#)

## The easy way to install Ruby on Windows

This is a **self-contained Windows-based installer** that includes the **Ruby language**, an execution environment, important **documentation**, and more.

**Download**

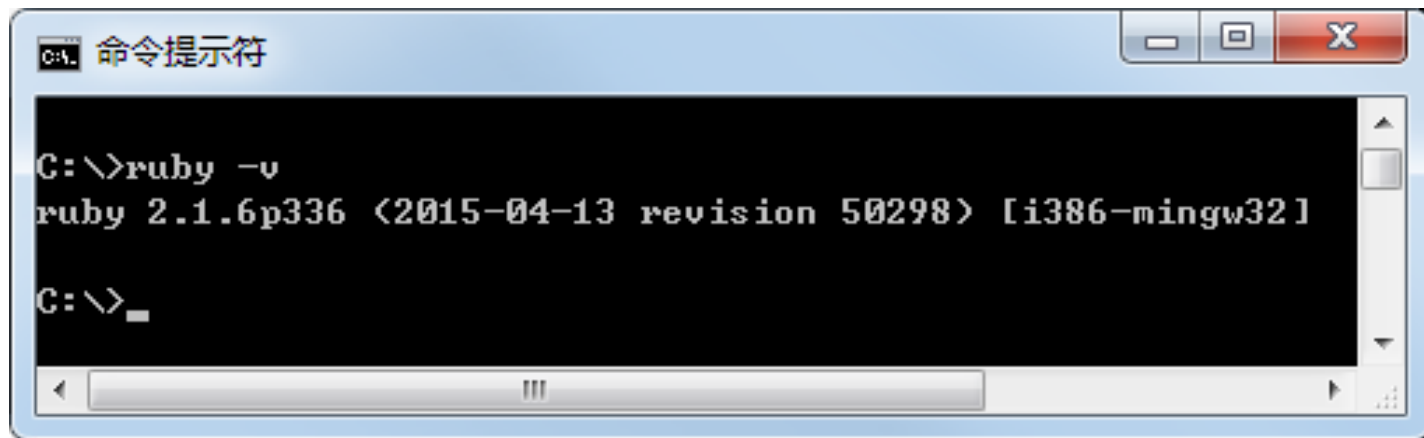
**Add-ons**



RubyInstaller provides the best experience for installing Ruby on Windows.

# Step 1: Install Ruby

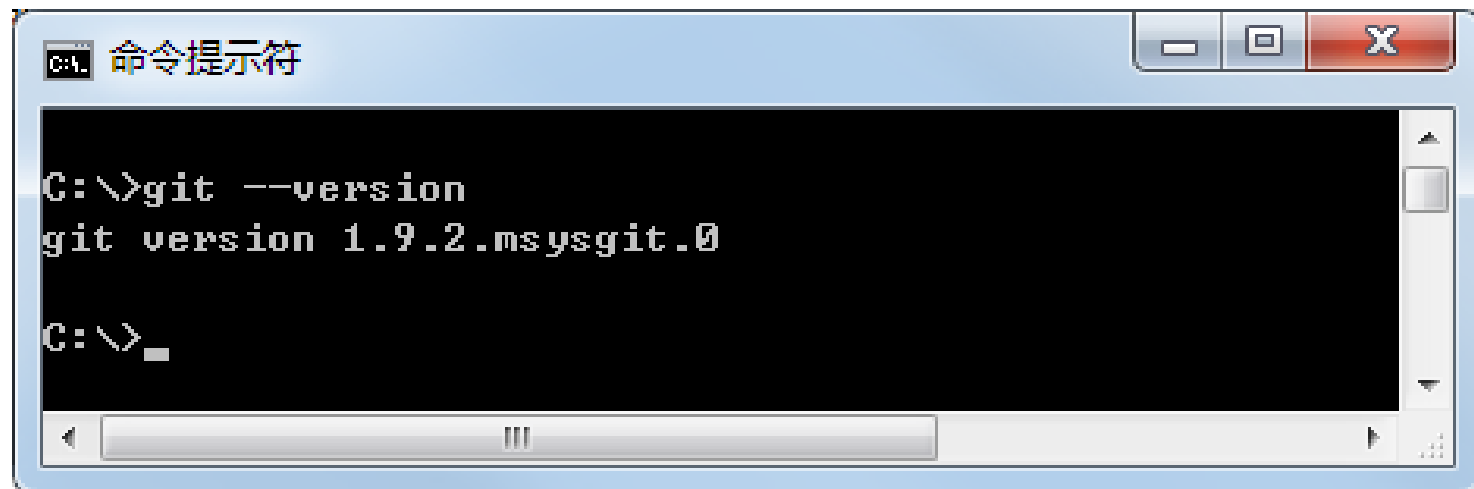
- Download the 2.1.xx version of RubyInstaller from <http://rubyinstaller.org/>)
- Launch the installer , select the **Add Ruby executables to your PATH** check box.
- Verify that the installation:



A screenshot of a Windows Command Prompt window titled "命令提示符" (Command Prompt). The window shows the command `C:\>ruby -v` being executed, resulting in the output `ruby 2.1.6p336 (2015-04-13 revision 50298) [i386-mingw32]`. The prompt then shows `C:\>_` with a cursor, indicating the command has finished.

# Step 2: Install Git

- Download Git for Windows from <http://msysgit.github.io/>
- Install the Git, selecting the ***Run Git from the Windows Command Link Prompt*** checkbox, also, selecting **Checkout Windows-style, commit Unix-style line endings**.
- Verify the installation:



A screenshot of a Windows Command Prompt window titled "命令提示符" (Command Prompt). The window shows the command `C:\>git --version` being executed, followed by the output `git version 1.9.2.msysgit.0`. The prompt then shows `C:\>_` with a cursor, indicating the command has finished.

# Git Basics

- Git is a distributed revision control system. Git keeps a **repository** in your working directory. It can push/pull the source code to/from remote server.
- **git clone <url>** Clone an existing repository from remote server. It creates a working directory indicated by the url, and pulls down all the data for that repository.
- **git add <files>** Specify the files you want to track.
- **git commit -a "<msg>"** commit the changes to local git repository and provide an explanation as <msg>
- **git push** Upload the changed files of local repository to remote repository.
- **git status** Checking the status of local repository

More about git:

<http://git-scm.com/book/en/v2/Getting-Started-Git-Basics>

# Step 3: Install OpenShift gem

- Download rhc gem and all its dependency gems from <https://rubygems.org/gems/rhc> into a folder (using http instead of https)\*.
- Goto that folder, install rhc locally:

```
gem install rhc --local ./rhc-1.35.3.gem
```

\* The gems package can also be downloaded from <http://pan.baidu.com/s/1qWLtjB2>

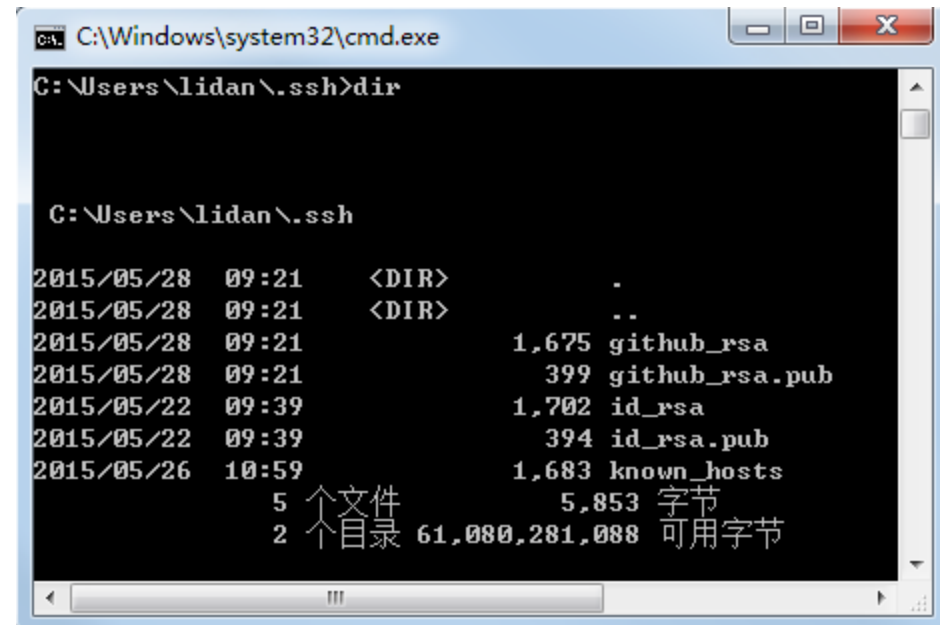
# Configure OpenShift gem

- Open a cmd and run:

```
rhc setup
```

- Input OpenShift username and password;
- In this process, a pair of SSH keys will be generated and kept in `.ssh` folder in your personal folder.

Answering yes to generate the keys, and yes to upload the public key.



```
C:\Windows\system32\cmd.exe
C:\Users\lidan\.ssh>dir

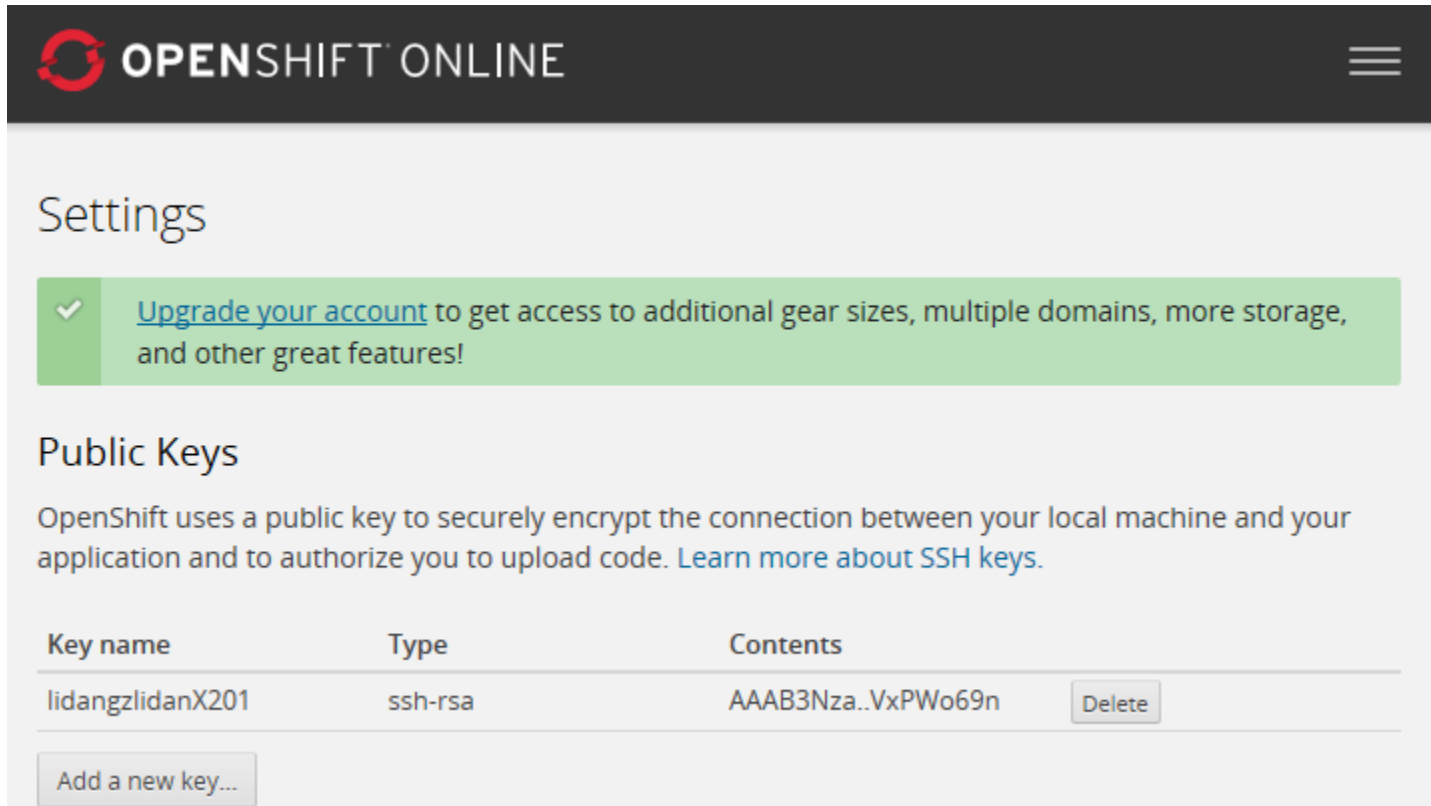
C:\Users\lidan\.ssh
2015/05/28 09:21 <DIR> .
2015/05/28 09:21 <DIR> ..
2015/05/28 09:21 1,675 github_rsa
2015/05/28 09:21 399 github_rsa.pub
2015/05/22 09:39 1,702 id_rsa
2015/05/22 09:39 394 id_rsa.pub
2015/05/26 10:59 1,683 known_hosts
 5 个文件 5,853 字节
 2 个目录 61,080,281,088 可用字节
```

# Secure Shell (SSH)

- A cryptographic network protocol client-server communication in a secure way.
- SSH uses automatically generated public-private key pairs to encrypt a network connection.
- The public key is present on the server end and the matching private key is present on the local machine.

# Check SSH Key in OpenShift

- Login the web console (<https://www.openshift.com/>)
- Click on *Settings* to check the *Public Keys*. If there are more than one, delete all others except the one you just uploaded.



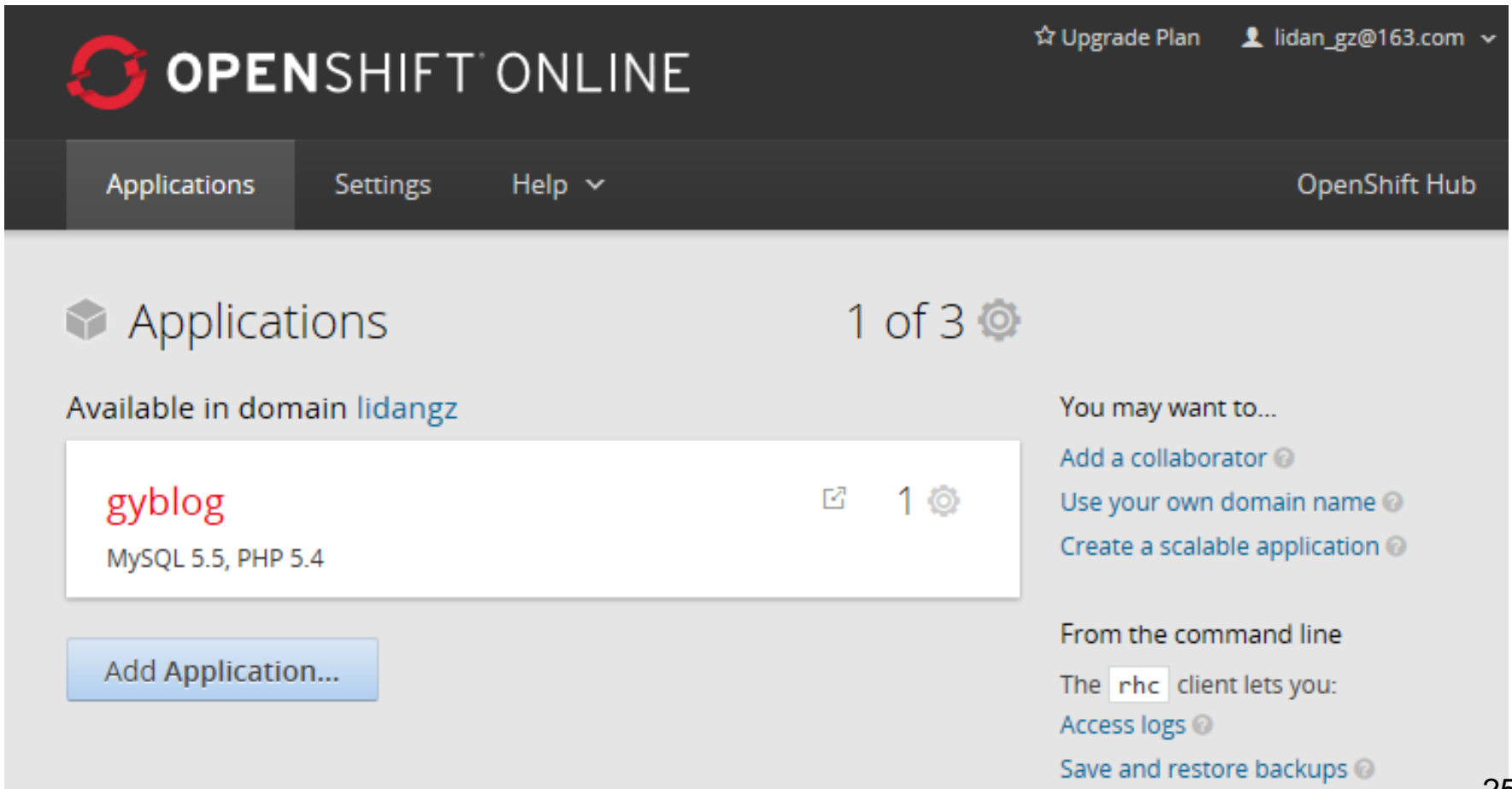
The screenshot shows the OpenShift Online web console interface. At the top is a dark header with the OpenShift logo and the text "OPENS SHIFT ONLINE". Below the header, the "Settings" page is displayed. A green banner with a checkmark icon contains the text: "Upgrade your account to get access to additional gear sizes, multiple domains, more storage, and other great features!". Below this, the "Public Keys" section is shown, with a descriptive paragraph: "OpenShift uses a public key to securely encrypt the connection between your local machine and your application and to authorize you to upload code. [Learn more about SSH keys.](#)". A table lists the public keys, with one key named "lidangzlidanX201" of type "ssh-rsa". The table has columns for "Key name", "Type", and "Contents". A "Delete" button is next to the key. At the bottom of the table is a button labeled "Add a new key...".

| Key name         | Type    | Contents          |
|------------------|---------|-------------------|
| lidangzlidanX201 | ssh-rsa | AAAB3Nza..VxPW69n |



# Create an App

- Click on [Applications](#) and
- [Add Application...](#)



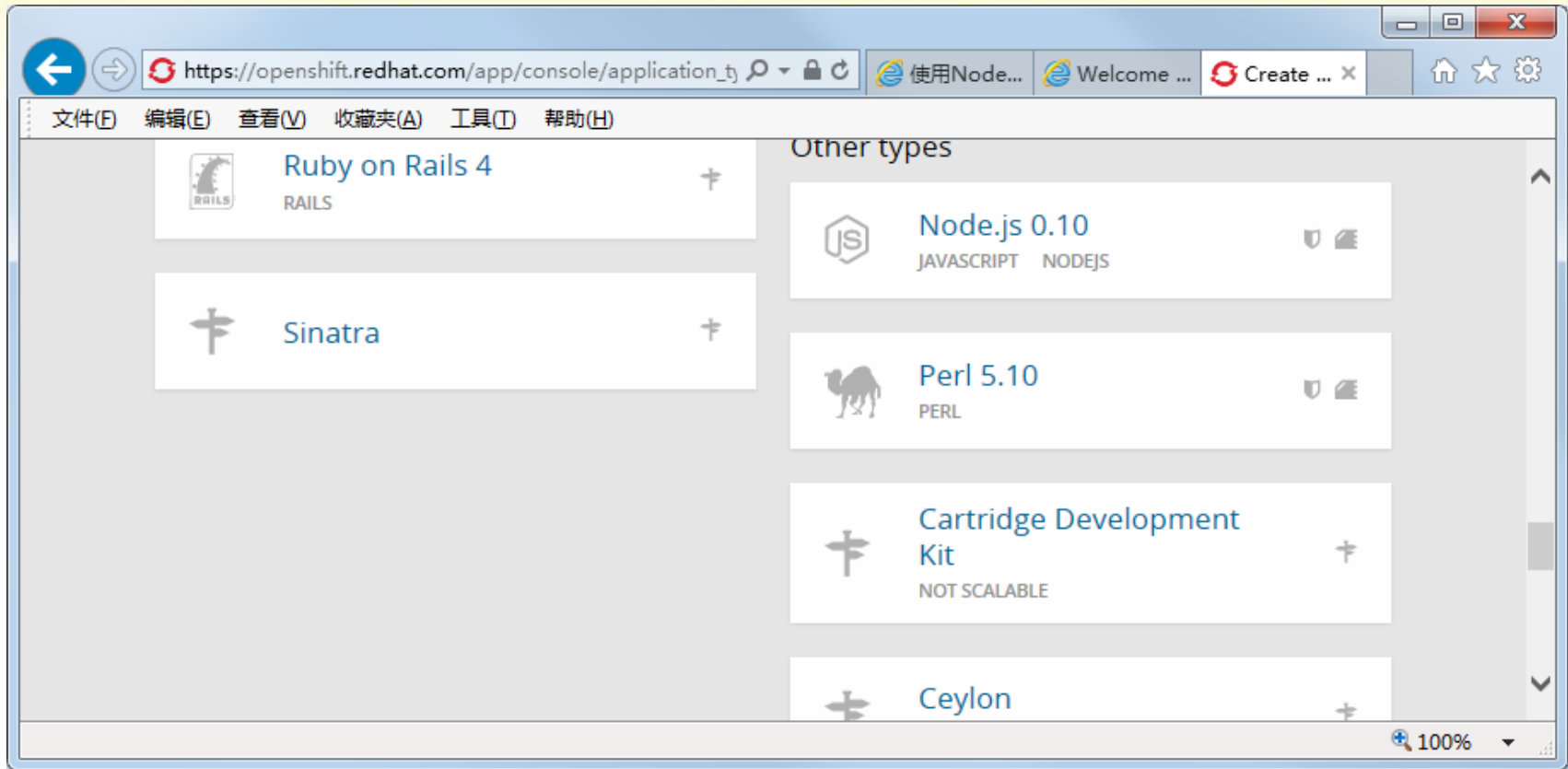
The screenshot shows the OpenShift Online web interface. At the top, there's a dark header with the OpenShift logo and the text "OPENSIFT ONLINE". To the right of the header, there are links for "Upgrade Plan" and a user profile "lidan\_gz@163.com". Below the header is a navigation bar with "Applications" (selected), "Settings", and "Help". On the right of the navigation bar is a link to "OpenShift Hub".

The main content area is titled "Applications" with a cube icon. It shows "1 of 3" applications. Below this, it says "Available in domain lidangz". A card for an application named "gyblog" is displayed, with the details "MySQL 5.5, PHP 5.4". To the right of the card are icons for a link and a gear, with the number "1" between them. Below the card is a blue button labeled "Add Application...".

On the right side of the main content area, there's a section titled "You may want to..." with three links: "Add a collaborator", "Use your own domain name", and "Create a scalable application". Below this is a section titled "From the command line" with the text "The `rhc` client lets you:" followed by two links: "Access logs" and "Save and restore backups".

# Choose Type of the Application

- Select cartridge *Node.js 0.10*



- In the next page, input, for example “mycourse” to *Public URL*
- Press button *Create Application*

# Configure the Application

- Input **mycourse** in *Public URL*
- Press button *Create Application*

☆ OpenShift maintained  
🛡️ Receives automatic security updates

Public URL

OpenShift will automatically register this domain name for your application. You can add your own domain name later.

Source Code

We'll create a Git code repository in the cloud, and populate it with a set of reasonable defaults. If you provide a Git URL, your application will start with an exact copy of the

# Add NongoDB to the App

- *Continue to the application overview page*
- Click [Add MongoDB 2.4](#), and [Add Cartridge](#) in next page

The screenshot shows the OpenShift Online interface. At the top, the header includes the OpenShift logo, the text "OPENSIFT ONLINE", and links for "Upgrade Plan" and a user profile "lidan\_gz@163.com". Below the header is a navigation bar with "Applications", "Settings", and "Help". The main content area displays the application name "mycourse-lidangz.rhcloud.com" with a "change" link, the status "Started", and a gear icon. Below this, the "Cartridges" section shows a table with one cartridge: "Node.js 0.10" with status "Started", "1 small" gears, and "1 GB" storage. To the right of the cartridges is the "Source Code" section with an SSH URL and instructions to clone the repository. At the bottom left, the "Databases" section lists "Add MongoDB 2.4", "Add MySQL 5.5", and "Add PostgreSQL 9.2". To its right, the "Continuous Integration" section has a link to "Enable Jenkins". At the bottom right, there is a "Remote Access" section with a login prompt and a "Delete this application..." button.

OPENSIFT ONLINE

Upgrade Plan lidan\_gz@163.com

Applications Settings Help OpenShift Hub

mycourse-lidangz.rhcloud.com change Started 1

Created 9 minutes ago in domain lidangz and the aws-us-east-1 region

Cartridges

|              | Status  | Gears   | Storage |
|--------------|---------|---------|---------|
| Node.js 0.10 | Started | 1 small | 1 GB    |

Source Code

ssh://5563d521e0b8cd9e7d0000a

Pass this URL to 'git clone' to copy the repository locally.

Remote Access

Want to log in to your application?

Delete this application...

Databases

- Add MongoDB 2.4
- Add MySQL 5.5
- Add PostgreSQL 9.2

Continuous Integration

- Enable Jenkins

*App's url*

# Overview of your App.

*ssh url*

The screenshot shows the Heroku application overview page. At the top, the application name 'mycourse-lidangz.rhcloud.com' is displayed with a 'change' link. Below it, a note states 'Created 35 minutes ago in domain lidangz and the aws-us-east-1 region'. To the right, the status 'Started' is shown with a count of '1' and icons for settings and refresh. The 'Cartridges' section lists two components: 'Node.js 0.10' and 'MongoDB 2.4'. The MongoDB entry includes details: 'Database: mycourse', 'User: admin', and 'Password: show'. The 'Source Code' section displays an SSH URL: 'ssh://5563d521e0b8cd9e7d0000a'. Below this, instructions state 'Pass this URL to 'git clone' to copy the repository locally.' and 'Remote Access' with the question 'Want to log in to your application?'. A 'Delete this application...' button is at the bottom right. The bottom left contains 'Continuous Integration' with an 'Enable Jenkins' link. The bottom right contains 'Tools and Support' with links to 'Add 10gen Mongo Monitoring Service Agent' and 'Add RockMongo 1.1'.

mycourse-lidangz.rhcloud.com [change](#)  
Created 35 minutes ago in domain lidangz and the aws-us-east-1 region

Started 1

### Cartridges

| Icon | Name         | Status             | Gears       | Storage                        |
|------|--------------|--------------------|-------------|--------------------------------|
|      | Node.js 0.10 | Started            | 1 small     | 1 GB                           |
|      | MongoDB 2.4  | Database: mycourse | User: admin | Password: <a href="#">show</a> |

### Source Code

```
ssh://5563d521e0b8cd9e7d0000a
```

Pass this URL to 'git clone' to copy the repository locally.

### Remote Access

Want to log in to your application?

[Delete this application...](#)

### Continuous Integration

[Enable Jenkins](#)

### Tools and Support

- [Add 10gen Mongo Monitoring Service Agent](#)
- [Add RockMongo 1.1](#)

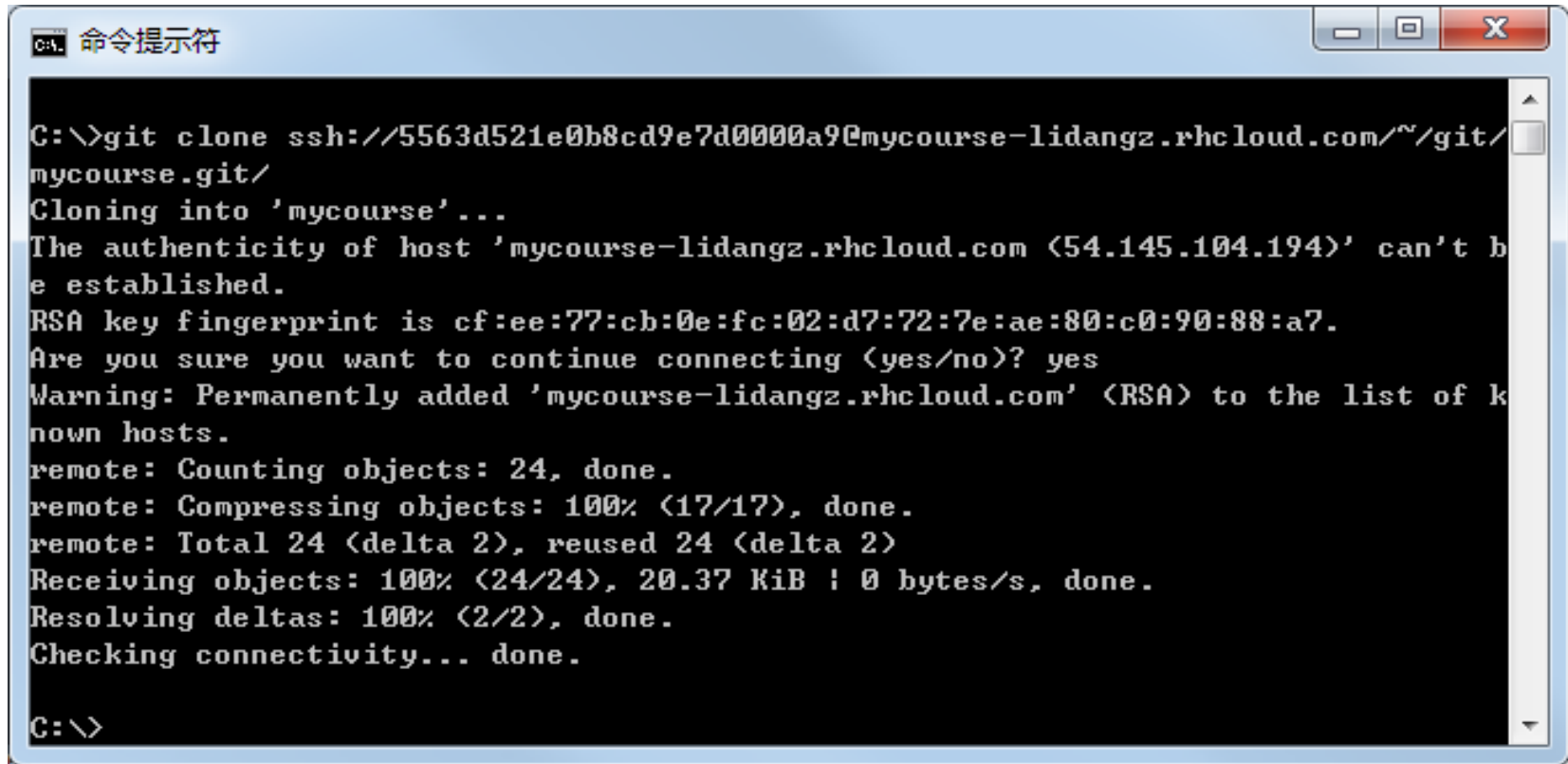
*Name of your database*

- Copy your ssh url under the **Source Code**

# Download Source Code

- Open a cmd, goto `c:\`, and run

```
git clone <sshUrl>
```



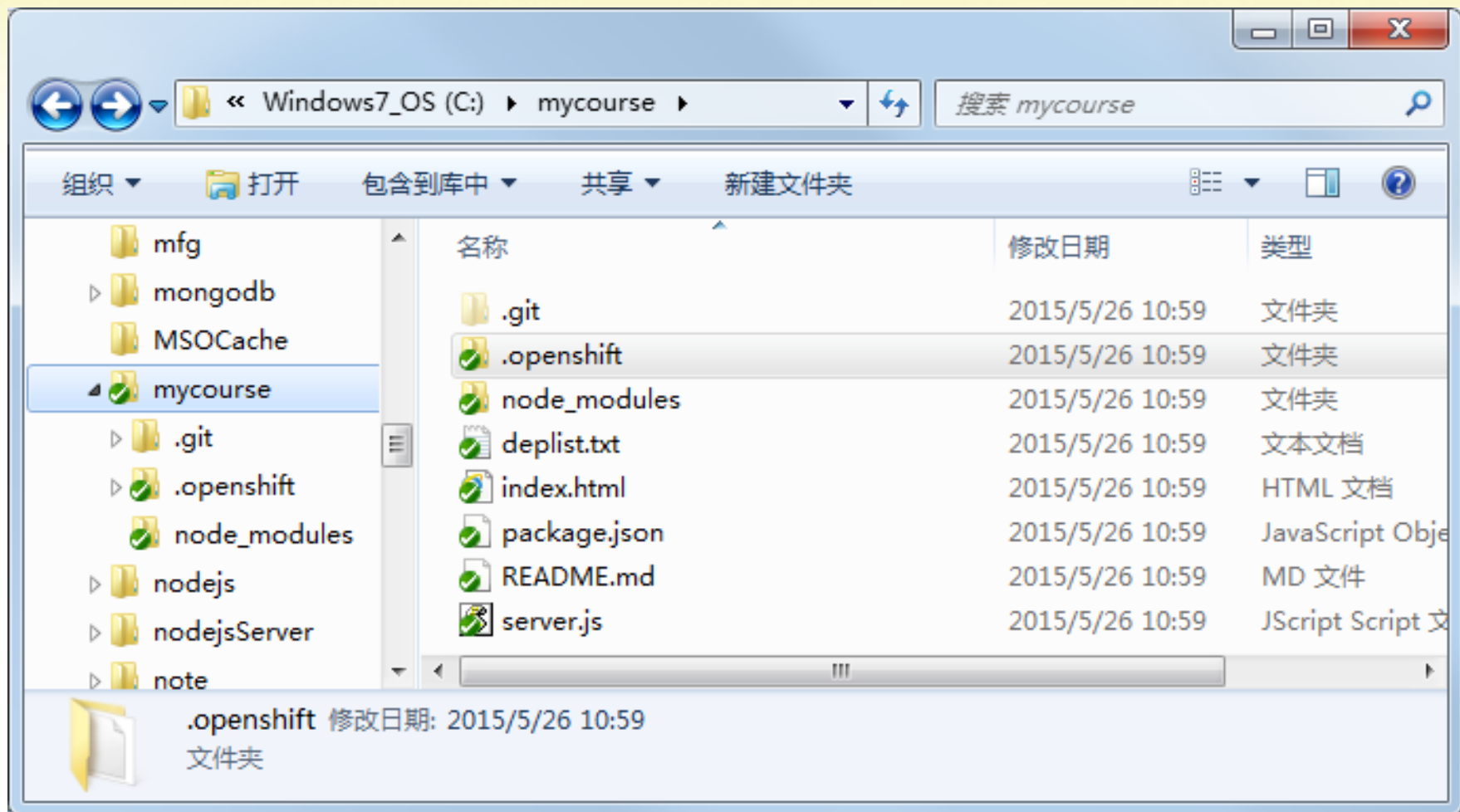
The screenshot shows a Windows Command Prompt window titled "命令提示符" (Command Prompt). The command entered is `git clone ssh://5563d521e0b8cd9e7d0000a9@mycourse-rhcloud.com/~git/mycourse.git/`. The output shows the cloning process, including a warning about the host's authenticity and the successful completion of the clone.

```
C:\>git clone ssh://5563d521e0b8cd9e7d0000a9@mycourse-rhcloud.com/~git/mycourse.git/
Cloning into 'mycourse'...
The authenticity of host 'mycourse-rhcloud.com (54.145.104.194)' can't be established.
RSA key fingerprint is cf:ee:77:cb:0e:fc:02:d7:72:7e:ae:80:c0:90:88:a7.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'mycourse-rhcloud.com' (RSA) to the list of known hosts.
remote: Counting objects: 24, done.
remote: Compressing objects: 100% (17/17), done.
remote: Total 24 (delta 2), reused 24 (delta 2)
Receiving objects: 100% (24/24), 20.37 KiB | 0 bytes/s, done.
Resolving deltas: 100% (2/2), done.
Checking connectivity... done.

C:\>
```

- Now, the `c:\mycourse` is your project folder.

# Structure of the Project Folder



# Modify *package.json*

- Open the `package.json` file with an editor, and change it so it looks like this:

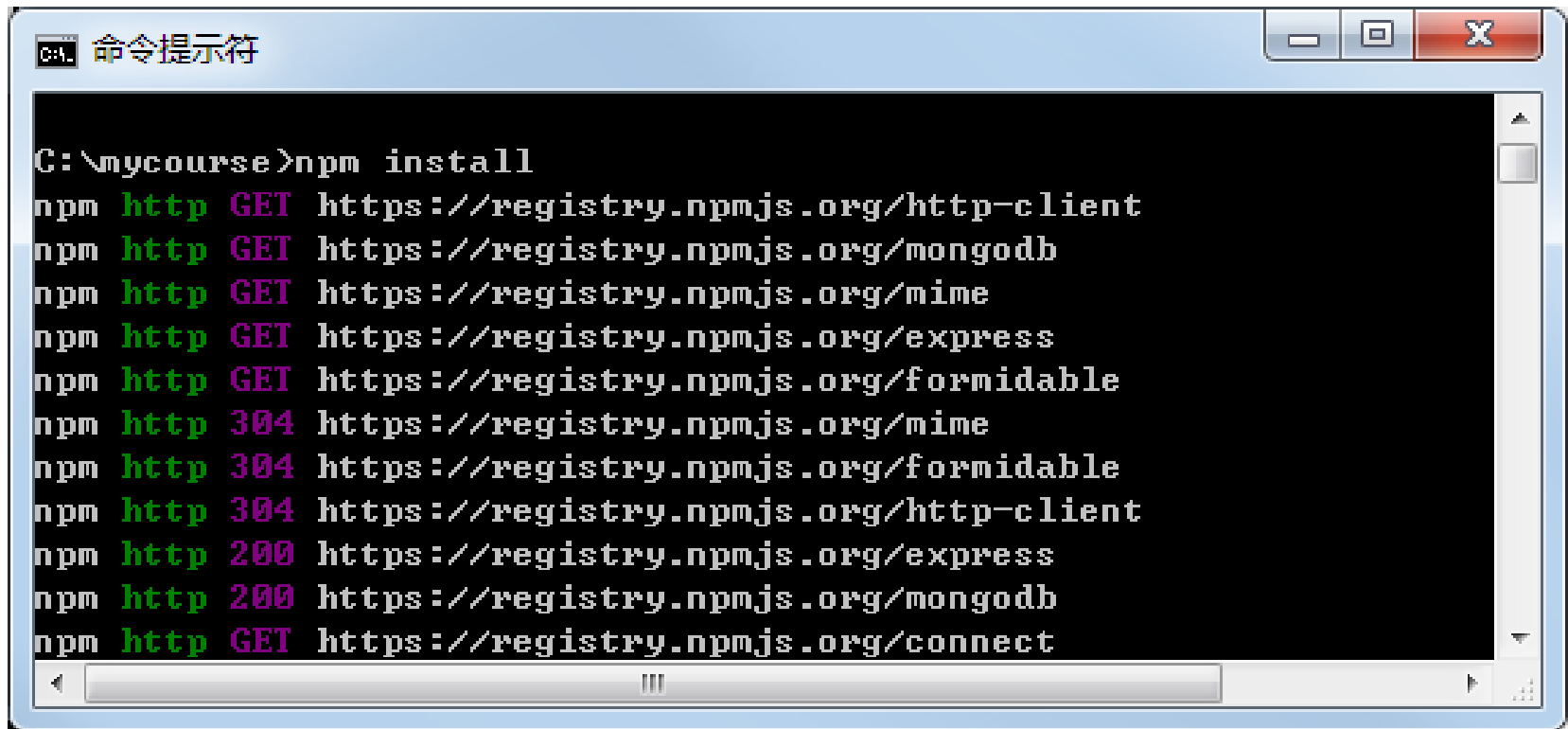
```
"dependencies": {
 "express": "~3.4.4",
 "formidable": "~1.0.15",
 "mime": "~1.2.11",
 "mongodb": "~1.4.7",
 "http-client": "~1.0.0"
},
"devDependencies": {},
"bundleDependencies": [],
"private": true,
"scripts": {
 "start": "supervisor server.js"
},
"main": "server.js"
}
```



# Install Dependencies

- In the cmd, goto the project folder, run:

```
npm install
```

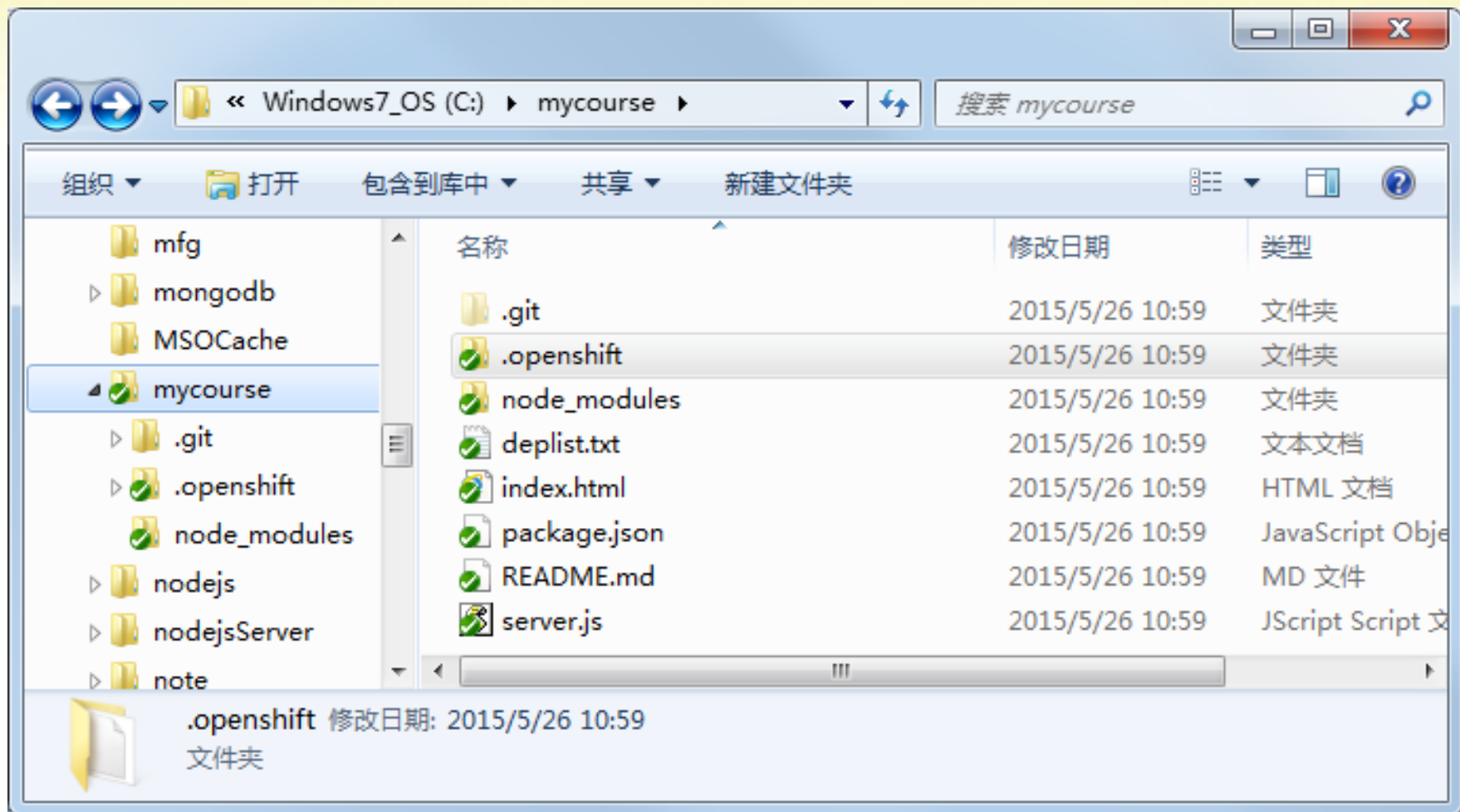


A screenshot of a Windows Command Prompt window titled "命令提示符" (Command Prompt). The window shows the execution of the command `npm install` in the directory `G:\mycourse`. The output displays a series of HTTP requests to the npm registry for various packages. The status codes for the requests are color-coded: green for successful GET requests (200) and red for failed requests (304). The packages listed are `http-client`, `mongodb`, `mime`, `express`, and `formidable`. The output also includes a final `connect` package request.

```
G:\mycourse>npm install
npm http GET https://registry.npmjs.org/http-client
npm http GET https://registry.npmjs.org/mongodb
npm http GET https://registry.npmjs.org/mime
npm http GET https://registry.npmjs.org/express
npm http GET https://registry.npmjs.org/formidable
npm http 304 https://registry.npmjs.org/mime
npm http 304 https://registry.npmjs.org/formidable
npm http 304 https://registry.npmjs.org/http-client
npm http 200 https://registry.npmjs.org/express
npm http 200 https://registry.npmjs.org/mongodb
npm http GET https://registry.npmjs.org/connect
```

# IMPLEMENTATION

# Files in the Project Folder



# Start the Development

- Create the following two files in the project folder.

*handlers.js*

*dbutils.js*

- Now the project folder has the following files.

- Server-side

- *server.js* -- app main entry, route the requests
    - *handlers.js* -- actually deal with the requests
    - *dbutils.js* -- database connection
    - *package.json* -- configuration file

- Client-side

- *Index.html* -- front-end html file

# Design an Empty Client-side Html

- Replace the contents of [index.htm](#) with the following code:

```
<html>
 <head>
 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
 <title>Course material sharer</title>
 <script type="text/javascript">

 </script>
 </head>
 <body>
 <h2>Training Course Material Sharer</h2></p>
 <div id="catDiv">

 </div>
 <div id="listDiv">

 </div>
 <p> =====
 <div id="upLoadDiv">

 </div>
 </body>
</html>
```

# Server-side Design: *server.js*

- Here we start the app, and route the requests to functions in *handler.js*. Replace the contents of *server.js* with the following code:

```
var express = require('express');
var handlers = require('./handlers');

var ip_address = process.env.OPENSIFT_NODEJS_IP || '127.0.0.1';
var port = process.env.OPENSIFT_NODEJS_PORT || 8080;

var app = express();
app.configure(function () {
 app.use(express.logger('dev'));
 app.use(express.bodyParser());
});
```

# *server.js (2)*

- Followed by the code below:

```
console.log ('registering event routes with express');

app.get('/', handlers.start);
app.get('/materials', handlers.listAll);
app.get('/materials/:cat', handlers.listByCat);
app.post('/materials', handlers.upload);
app.get('/files/:fileId/:fileName', handlers.download);
app.delete('/materials/:id', handlers.delete);

console.log ('About to start listening');
app.listen(port,ip_address);
console.log('Listening on port: ', port, ' of ', ip_address);
```

# *handlers.js*

- Here we write the logic to handle the requests .  
Update *handlers.js* as below:

```
var fs = require('fs');
var mime = require('mime');
var formidable = require("formidable");
var mongo = require('mongodb');
var BSON = mongo.BSONPure;
var dbutils = require('./dbutils');
var mongoUrl = dbutils.getMongoUrl();

exports.start = function(req, res) {
 console.log("Request handler 'start' was called.");
 fs.readFile('index.html', 'utf-8', function (err, data) {
 if (err) {return console.dir(err);}
 res.setHeader('Content-Type', 'text/html');
 res.send(data);
 });
}
```



# Deal with *upload* in *handlers.js*

```
exports.upload = function(req,res) {
 console.log("Request handler 'upload' was called.");

 var form = new formidable.IncomingForm();
 form.parse(req, function(err, fields, files) {
 mongo.Db.connect(mongoUrl, function (err, db) {
 var idobj=new mongo.ObjectID();
 var fileId=idobj.toString();
 var gridStore = new mongo.GridStore(db, fileId, 'w');
 gridStore.writeFile(files.upload.path, function(err, fileInfo) {
 db.collection('material',{safe:true},function(err,collection){
 fields["fileName"]=files.upload.name;
 fields["fileId"]=fileId;
 collection.insert(fields,{safe:true},function(err,result){
 fields["_id"]=result._id;
 db.close();
 res.setHeader('Content-Type', 'text/html');
 res.send(JSON.stringify(fields));
 });
 });
 });
 });
 });
};
```

# Functions listAll, listByCat in *handlers.js*

```
exports.listAll = function(req, res) {
 console.log("Request handler 'listAll' was called.");

 readList(res,null);
}

exports.listByCat = function(req, res) {
 console.log("Request handler 'listByCat' was called.");

 var cat = req.params.cat;
 readList(res,cat);
}
```

# Function Called by listAll, listByCat

```
function readList(res,cat) {
 var qstr = {};
 if (cat!=null) {
 qstr=eval('({"fileCat" : "'+ cat + '"})');
 };

 mongo.Db.connect(mongoUrl, function (err, db) {
 db.collection('material', function(err, collection) {
 collection.find(qstr).toArray(function(err, items) {
 res.setHeader('Content-Type', 'text/html');
 res.send(JSON.stringify(items));
 db.close();
 });
 });
 });
}
```

# Deal with *download* in *handlers.js*

```
exports.download = function(req,res) {
 console.log("Request handler 'download' was called.");

 var fileId = req.params.fileId;
 var fileName = req.params.fileName;
 mongo.Db.connect(mongoUrl, function (err, db) {
 var gridStore = new mongo.GridStore(db, fileId, 'r');
 gridStore.open(function(err, gridStore) {
 var stream = gridStore.stream(true);
 stream.on("end", function(err) {
 db.close();
 res.end();
 });
 var contentType=mime.lookup(fileName).toString();
 res.setHeader('Content-Type', contentType);
 stream.pipe(res);
 });
 });
}
```


# Deal with *delete* in *handlers.js*

```
exports.delete = function(req, res) {
 console.log("Request handler 'delete' was called.");
 var id = req.params.id;
 mongo.Db.connect(mongoUrl, function (err, db) {
 db.collection('material', {safe:true}, function(err, collection) {
 collection.findOne({'_id':new BSON.ObjectId(id)}, function(err, doc) {
 var fileName=doc.fileName;
 collection.remove({'_id':new BSON.ObjectId(id)}, {safe:true}, function(err, result) {
 if (doc.fileId) {
 var gridStore=mongo.GridStore;
 gridStore.unlink(db, doc.fileId, function(err) {
 console.log('delete '+fileName);
 db.close();
 readList(res,null);
 });
 } else { db.close();
 readList(res,null);
 };
 });
 });
 });
 });
 });
}
```

# Add the Follows to *dbutils.js*

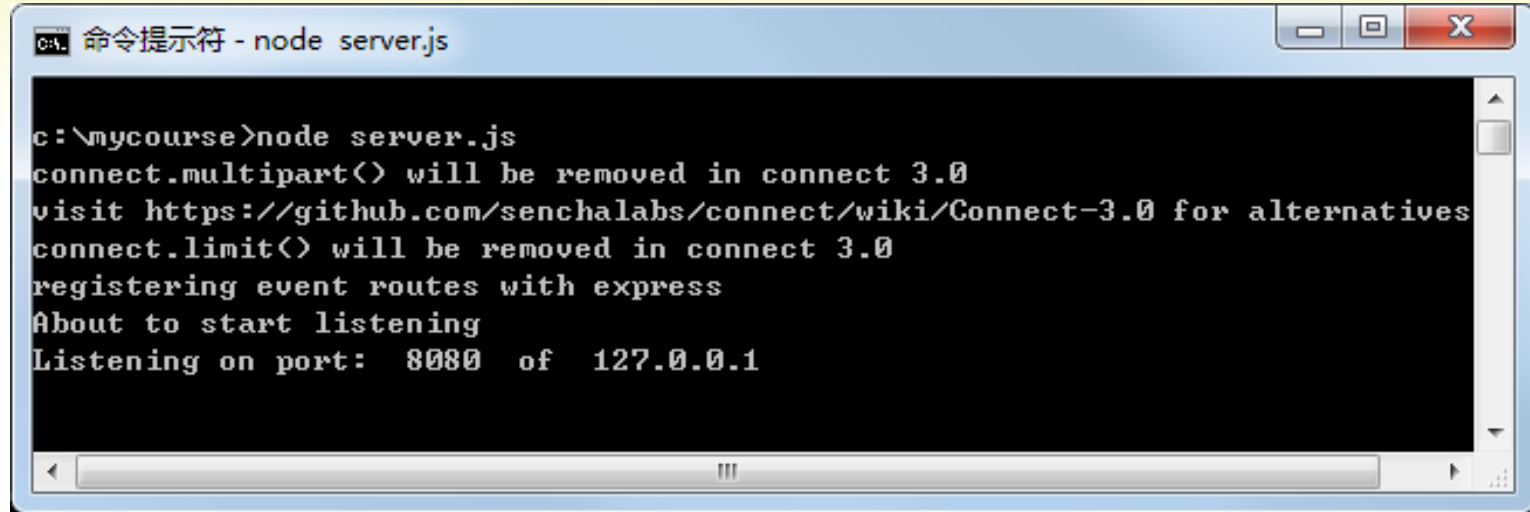
```
var mongostr = { // local machine
 "hostname":"localhost",
 "port":27017,
 "username":"tom",
 "password":"1234",
 "name": "",
 "db":"course"
}
if(process.env.OPENSIFT_NODEJS_PORT){ // OpenShift
 mongostr = {
 "hostname":process.env.OPENSIFT_MONGODB_DB_HOST,
 "port":process.env.OPENSIFT_MONGODB_DB_PORT,
 "username":process.env.OPENSIFT_MONGODB_DB_USERNAME,
 "password": process.env.OPENSIFT_MONGODB_DB_PASSWORD,
 "name": "",
 "db":"mycourse"
 }
}
exports.getMongoUrl = function() {
 return "mongodb://" + mongostr.username + ":" +
 mongostr.password + "@" + mongostr.hostname + ":" +
 mongostr.port + "/" + mongostr.db;
}
```

*Change to name of your database*



# Test the App in Local

- Cd to the project folder, and start the server



```
c:\mycourse>node server.js
connect.multipart() will be removed in connect 3.0
visit https://github.com/senchalabs/connect/wiki/Connect-3.0 for alternatives
connect.limit() will be removed in connect 3.0
registering event routes with express
About to start listening
Listening on port: 8080 of 127.0.0.1
```

- Access <http://localhost:8080/> from a browser
- Check the log message showed by Nodejs

# Client-side Design

## Upload in `index.html`

- Replace the `uploadDiv` by :

```
<div id="upLoadDiv">
 <fieldset><legend>Select a file to upload:</legend>
 <p>File Cat:
 <select size="1" id="fileCat">
 <option selected>Slide</option>
 <option>Book</option>
 <option>Program</option>
 <option>Tool</option>
 </select> </p>
 <p>File Desc. : <input type="text" id="fileDesc" size="40"> </p>
 <p>Select File: <input type="file" id="upload"></p>
 <p><input type="button" value="Upload File" onclick="uploadFile()"/>
 </p>
 </fieldset>
</div>
```



# Design Upload in [index.html](#) (2)

- Add function `uploadFile` to the script section:

```
function uploadFile(){
 var urlstr = "/materials";
 var fileObj = document.getElementById("upload").files[0];
 var fileName=document.getElementById("upload").value;
 var fileDesc=document.getElementById("fileDesc").value;
 var fileCat=document.getElementById("fileCat").value;

 if (validate(fileName,fileDesc)){
 var form = new FormData();
 form.append("fileDesc", fileDesc);
 form.append("fileCat", fileCat);
 form.append("fileName", document.getElementById("upload").value);
 form.append("upload", fileObj);
 }
}
```

# Design Upload in `index.html` (3)

- Second part of function `uploadFile`:

```
xmlhttp = new XMLHttpRequest();
xmlhttp.open("post", urlstr, true);
xmlhttp.setRequestHeader("Content-type", "multipart/form-data");
xmlhttp.onreadystatechange = function(){
 if(xmlhttp.readyState === 4){
 if(xmlhttp.status === 200){
 var json = JSON.parse(xmlhttp.responseText);
 var ihtm = getOneLi(json);
 document.getElementById("listDiv").innerHTML += ihtm;
 document.getElementById("upload").value = null;
 fileDesc=document.getElementById("fileDesc").value=null;
 }else{
 alert('Error: '+xmlhttp.status);
 }
 }
}
xmlhttp.send(form);
}
```

# Design Upload in [index.html](#) (4)

- Two more functions to the script section:

```
function validate(fileName, fileDesc) {
 if (fileName == null || fileName == "") {
 alert("Please select a file to upload.");
 return false;
 }
 if (fileDesc == null || fileDesc == "") {
 alert("Please input the file description.");
 return false;
 }
 return true;
}

function getOneLi(json) {
 var fpath= '/files/'+json["fileId"]+'/'+json["fileName"];
 var str='<u>'+
 json["fileCat"]+'</u> : '+
 json["fileDesc"]+ ' (' +json["fileName"]+') '+
 '<a onclick="delItem(\''+json["_id"]+\'',\''+json["fileName"]+
 '\')">[delete]';
 return str;
}
```

# Design List in `index.html`

- Replace the `catDiv` by :

```
<div id="catDiv">
 <u>All </u>
</div>
```

- Add event to the *body*:

```
<body onload="listAll(null)">
```

# Design List in [index.html](#) (2)

- Add function **listAll** to the script section:

```
function listAll(cat){
 var urlstr = "/materials";
 if (cat!=null && cat !='') {
 urlstr = urlstr+"/"+cat;
 }
 xmlhttp = new XMLHttpRequest();
 xmlhttp.open("get", urlstr, true);
 xmlhttp.onreadystatechange = function(){
 if(xmlhttp.readyState === 4){
 if(xmlhttp.status === 200){
 showItems(xmlhttp.responseText,cat);
 }else{
 alert('Error: '+xmlhttp.responseText); // An error occurred
 }
 }
 };
 xmlhttp.send(null);
}
```

# Design List in [index.html](#) (3)

- Add function [showItems](#) to the script section:

```
function showItems(jsontxt,cat) {
 var alljson = JSON.parse(jsontxt);
 var ihtm = '';
 for(var i=0;i<alljson.length;i++){
 ihtm += getOneLi(alljson[i]);
 }
 var titlelstr='<u>All</u>';
 if (cat!=null && cat !='') {
 titlelstr += '<u>-->'+cat+'</u>';
 }

 document.getElementById("listDiv").innerHTML=ihmt+'';
 document.getElementById("catDiv").innerHTML=titlelstr;
}
```

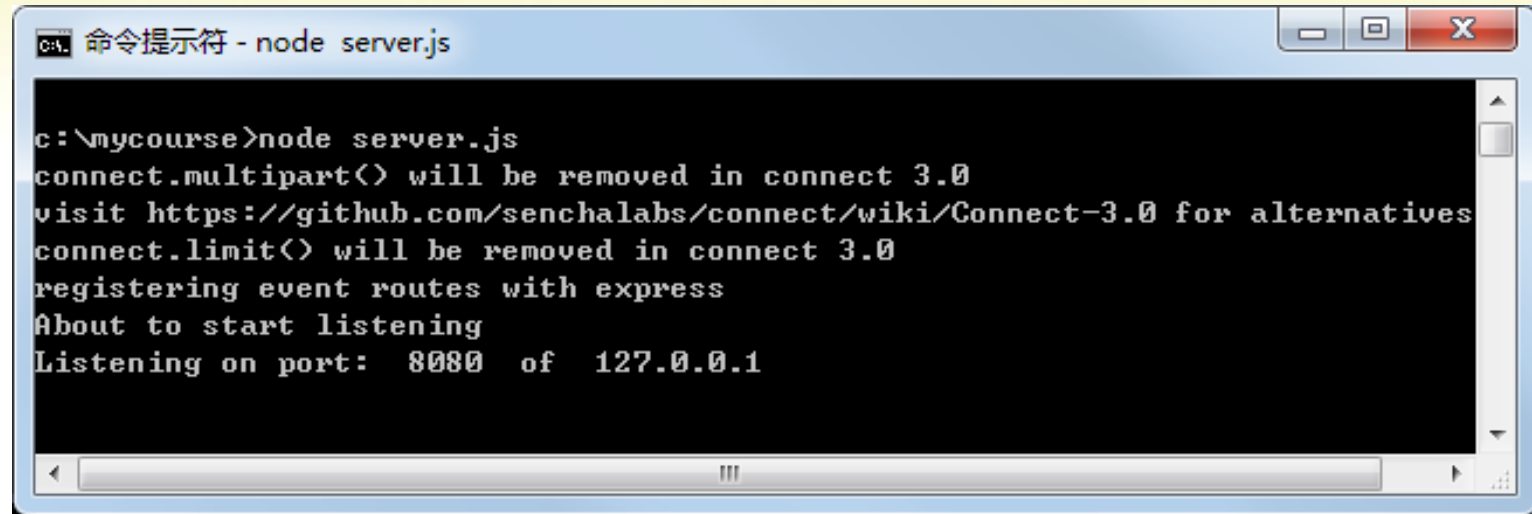
# Design Delete in [index.html](#)

- Add function `delItem` to the script section:

```
function delItem(mid, fileName) {
 if(confirm('Do you want to delete file '+ fileName + ' ?')){
 var urlstr = '/materials'+ '/' + mid;
 xmlhttp = new XMLHttpRequest();
 xmlhttp.open("delete", urlstr, true);
 xmlhttp.send();
 xmlhttp.onreadystatechange = function(){
 if(xmlhttp.readyState === 4){
 if(xmlhttp.status === 200){
 showItems(xmlhttp.responseText, null);
 } else {
 alert('Error: ' + xmlhttp.responseText);
 }
 }
 }
 }
};
```

# Again Test the App

- Cd to the project folder, and start the server



```
c:\mycourse>node server.js
connect.multipart() will be removed in connect 3.0
visit https://github.com/senchalabs/connect/wiki/Connect-3.0 for alternatives
connect.limit() will be removed in connect 3.0
registering event routes with express
About to start listening
Listening on port: 8080 of 127.0.0.1
```

- Access <http://localhost:8080/> from a browser
- Check the log message showed by Nodejs



# DEPLOYMENT

# Upload code to OpenShift

- Open a cmd, cd to the project folder, runing

1. Add the two new js files to git control

```
git add handlers.js dbutils.js
```

2. Commit the changes to local git repository

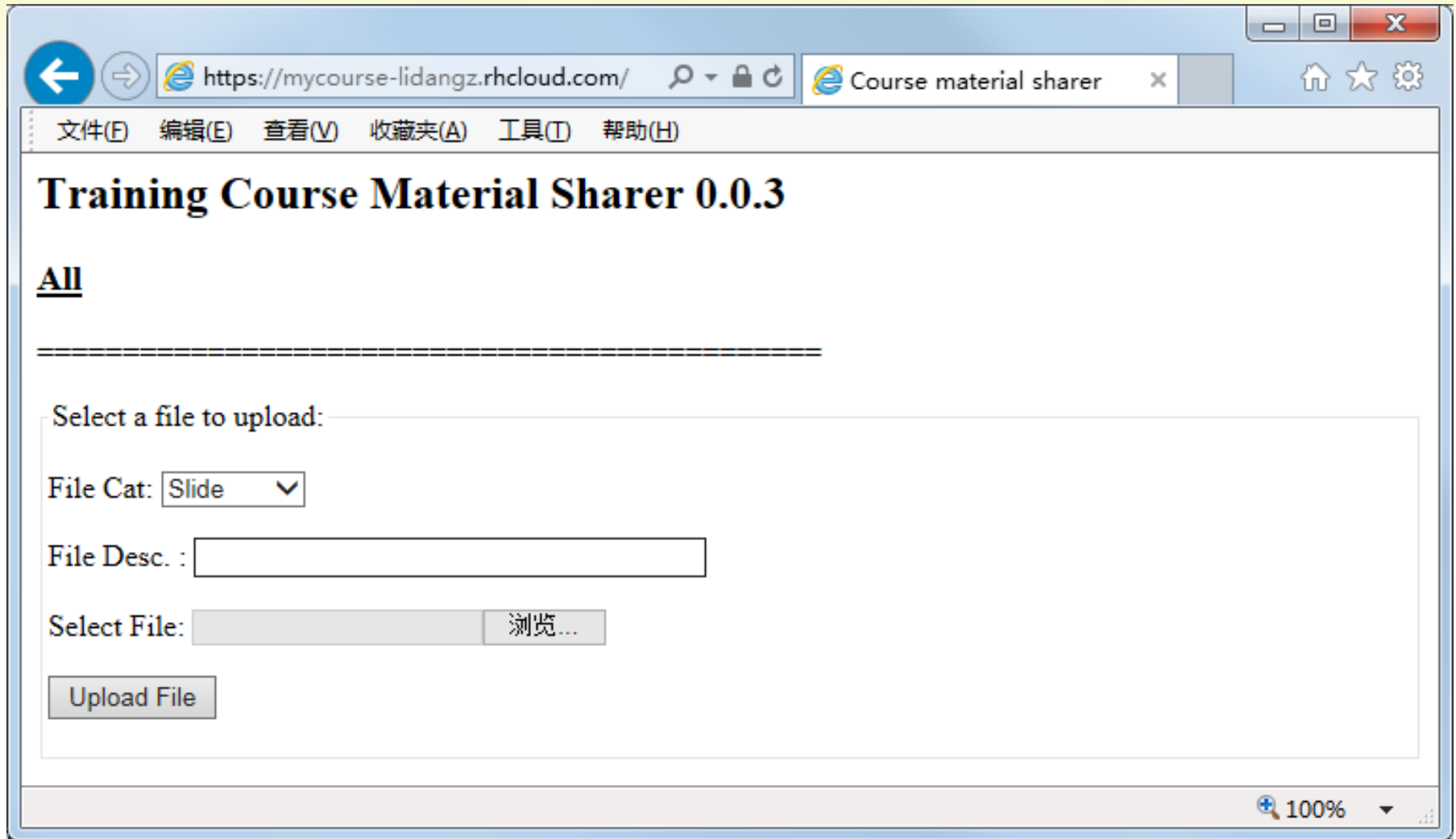
```
git commit -am "My first change"
```

3. Push the code to OPENSHIFT server\*

```
git push
```

\* Try several times if errors

# Test the App in OpenShift



# References

- Manuel Kiessling, “The Node Beginner Book”, <http://leanpub.com/nodebeginner>