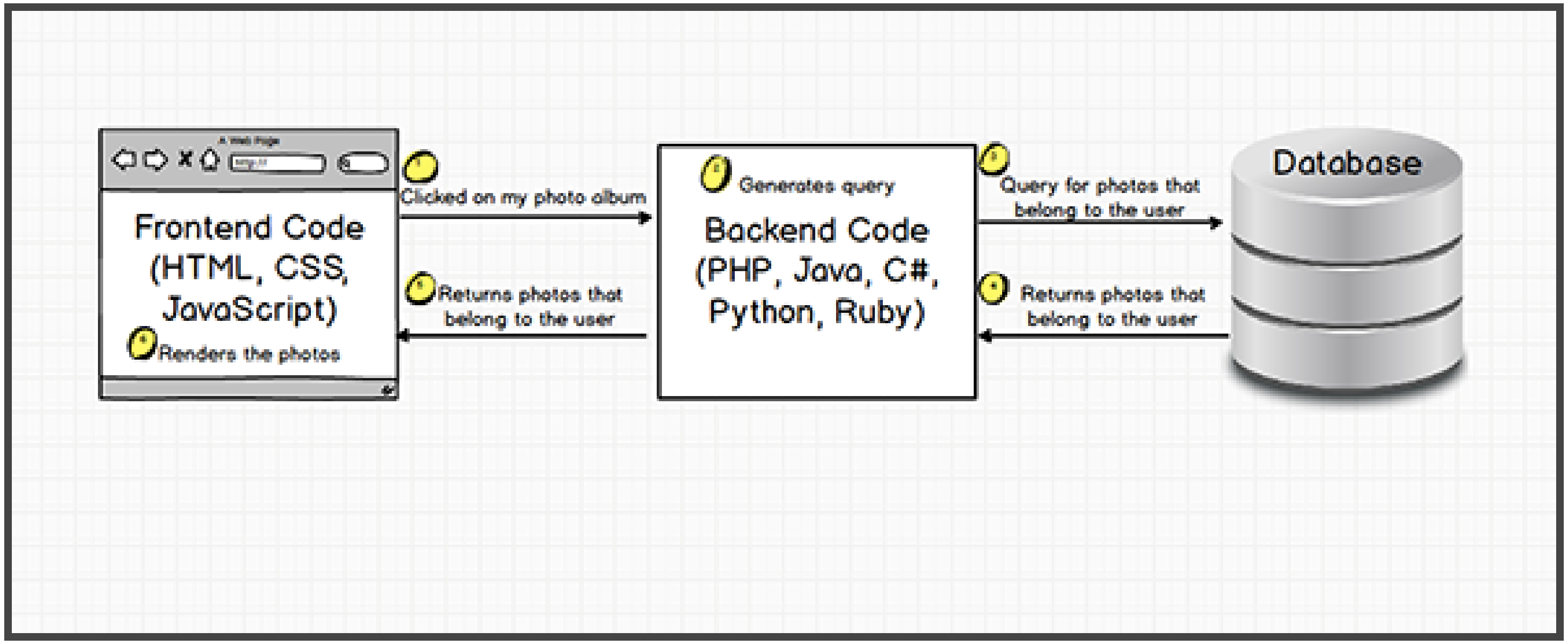
Node.js, Express, and MongoDB

SOSEL
Service-Oriented Software Engineering Lab

NTOU CSE
2022

Frontend and Backend


A Web Page
Frontend Code (HTML, CSS, JavaScript)
Renders the photos
Clicked on my photo album
Returns photos that belong to the user
Generates query
Backend Code (PHP, Java, C#, Python, Ruby)
Query for photos that belong to the user
Returns photos that belong to the user
Database

http://felixthea.com/frontend-vs-backend/

Why Backend?

- Connect database
- Connect legacy systems
- Let users communicate
- Push messages to users
- Protect your business logic code

Learning Backend Skills

- PHP (Hypertext Preprocessor)
  - https://www.w3schools.com/pHP/default.asp
- Node.js
  - https://www.w3schools.com/nodejs/
- Python Flask
  - https://www.tutorialspoint.com/flask/index.htm
- ASP.NET
  - https://www.w3schools.com/asp/webpages_intro.asp
- Spring Boot
  - https://spring.io/projects/spring-boot

Node.js

- Node.js is an open source server environment.
- Node.js allows you to run JavaScript on the server.
- https://www.w3schools.com/nodejs/default.asp

Node.js File

- What is a Node.js File?
  - Node.js files have extension ".js“.
  - Node.js files contain tasks that will be executed on certain events.
  - A typical event is someone trying to access a port on the server.
  - Node.js files must be initiated on the server before having any effect.

Installation and First Example1

- Download and install node.js
  - https://nodejs.org/en/

- Create a Node.js file named "myfirst.js“:

```
const http = require('http');

http.createServer(function (req, res) {
    res.writeHead(200, { 'Content-Type': 'text/html' });
    res.end('Hello World!');
}).listen(8080);
```

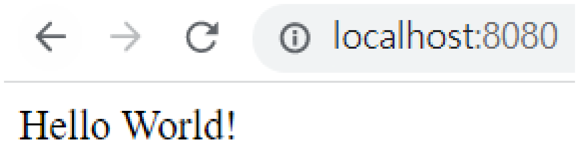
The code tells the computer to write "Hello World!" if anyone (e.g. a web browser) tries to access your computer on port 8080.

Installation and First Example2

- The file you have just created must be initiated by Node.js before any action can take place.
  - Start your command line interface (CMD in Windows or "終端機" in MAC), switch to the code folder, and input the following command:

```
node myfirst.js
```

  - Now, your computer works as a server!
    - Start your internet browser, and type in the address: http://localhost:8080



Hello World!

Node.js Modules1

- What is a Module in Node.js?
  - Consider modules to be the same as JavaScript libraries.
  - A set of functions you want to include in your application.
- Built-in Modules
  - Node.js has a set of built-in modules which you can use without any further installation.
  - Look at our Built-in Modules Reference for a complete list of modules.

Node.js Modules2

- Include Modules
  - To include a module, use the require() function with the name of the module:

```
const http = require('http');
```

  - Now your application has access to the HTTP module, and is able to create a server:

```
//myFirst.js
const http = require('http');

http.createServer(function (req, res) {
    res.writeHead(200, { 'Content-Type': 'text/html' });
    res.end('Hello World!');
}).listen(8080);
```

Node.js HTTP Module1

- Node.js has a built-in module called HTTP, which allows Node.js to transfer data over the Hyper Text Transfer Protocol (HTTP).
- To include the HTTP module, use the require() method:

```
const http = require('http');
```

Node.js HTTP Module2

- Node.js as a Web Server
  - The HTTP module can create an HTTP server that listens to server ports and gives a response back to the client.
  - Use the createServer() method to create an HTTP server:

```
//create a server object:
http.createServer(function (req, res) {
    res.write('Hello World!'); //write a response to the client
    res.end(); //end the response
}).listen(8080); //the server object listens on port 8080
```

The function passed into the `http.createServer()` method, will be executed when someone tries to access the computer on port 8080.

Node.js HTTP Module3

- If the response from the HTTP server is supposed to be displayed as HTML, you should include an HTTP header with the correct content type:

```
//demo_http.js
http.createServer(function (req, res) {
    res.writeHead(200, { 'Content-Type': 'text/html' });
    res.writeHead('Hello World!');
    res.end();
}).listen(8080);
```

The first argument of the `res.writeHead()` method is the status code, 200 means that all is OK, the second argument is an object containing the response headers (代表輸出型態是HTML).

https://restfulapi.net/http-status-codes/

Node.js HTTP Module4

- Read the Query String
  - The function passed into the http.createServer() has a req argument that represents the request from the client, as an object (http.IncomingMessage object).
  - This object has a property called "url" which holds the part of the **url** that comes after the domain name:

```
//demo_http_url
http.createServer(function (req, res) {
    res.writeHead(200, { 'Content-Type': 'text/html' });
    res.write(req.url);
    res.end();
}).listen(8080);
```

  - Start your internet browser, and type in the address: http://localhost:8080/hello or http://localhost:8080/hi

Node.js HTTP Module5

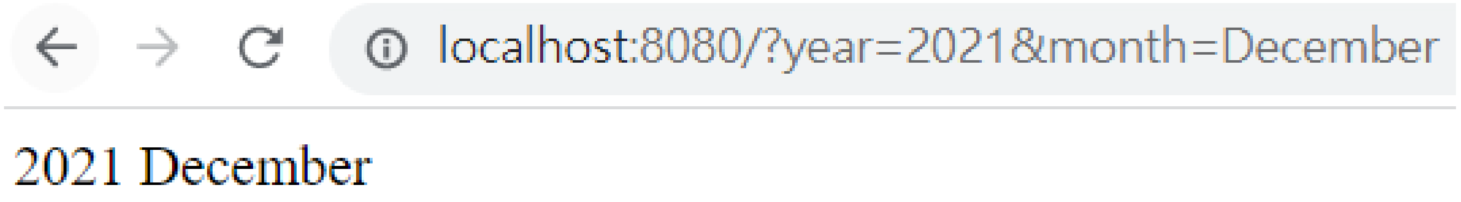
- Split the Query String
  - There are built-in modules to easily split the query string into readable parts, such as the URL module.

```
//demo_querystring.js
const http = require('http');
const url = require('url');

http.createServer(function (req, res) {
    res.writeHead(200, { 'Content-Type': 'text/html' });
    let q = url.parse(req.url, true).query;
    let txt = q.year + " " + q.month;
    res.end(txt);
}).listen(8080);
```

Node.js HTTP Module6

- Start your internet browser, and type in the address: http://localhost:8080/?year=2021&month=December



localhost:8080/?year=2021&month=December

```
2021 December
```

Node.js File System Module1

- The Node.js file system module allows you to work with the file system on your computer.
- To include the File System module, use the require() method:

```
const fs = require('fs');
```

Node.js File System Module2

Assume we have the HTML file (demofile1.html)

```
<html><body>
  <h1>My Header</h1>
  <p>My paragraph.</p>
</body></html>
```

Create a Node.js file that reads the HTML file, and return the content:

```
const http = require('http');
const fs = require('fs');
http.createServer(function (req, res) {
    fs.readFile('demofile1.html', function (err, data) {
        res.writeHead(200, { 'Content-Type': 'text/html' });
        res.write(data);
        return res.end();
    });
}).listen(8080);
```

Node.js + JSON

- You can easily convert JavaScript objects/arrays to JSON objects/arrays.

```
const http = require('http');
let myObj = {name: 'Stephen Curry', team: "GSW"};

http.createServer(function (req, res) {
    res.writeHead(200, { 'Content-Type': 'application/json' });
    res.end(JSON.stringify(myObj));
}).listen(8080);
```

Node.js NPM1

- What is NPM?
  - NPM is a package manager (套件/函式庫管理工具) for Node.js packages (modules).
  - www.npmjs.com hosts thousands of free packages to download and use.
  - The NPM program is installed on your computer when you install Node.js.

Node.js NPM₂

- What is a Package?
  - A package in Node.js contains all the files you need for a module.
  - Modules are JavaScript libraries you can include in your project.
- Download a Package
  - Downloading a package is very easy.
  - Open the command line interface and tell NPM to download the package you want.
  - If downloading a package called “upper-case”:

```
npm install upper-case
```

(會在node_modules下建立 upper-case子目錄)

Node.js NPM3

- Include the "upper-case" package the same way you include any other module:

```
const http = require('http');
const uc = require('upper-case');
http.createServer(function (req, res) {
    res.writeHead(200, { 'Content-Type': 'text/html' });
    res.write(uc.upperCase("Hello World!"));
    res.end();
}).listen(8080);
```

MongoDB

SOSEL
Service-Oriented Software Engineering Lab

MongoDB

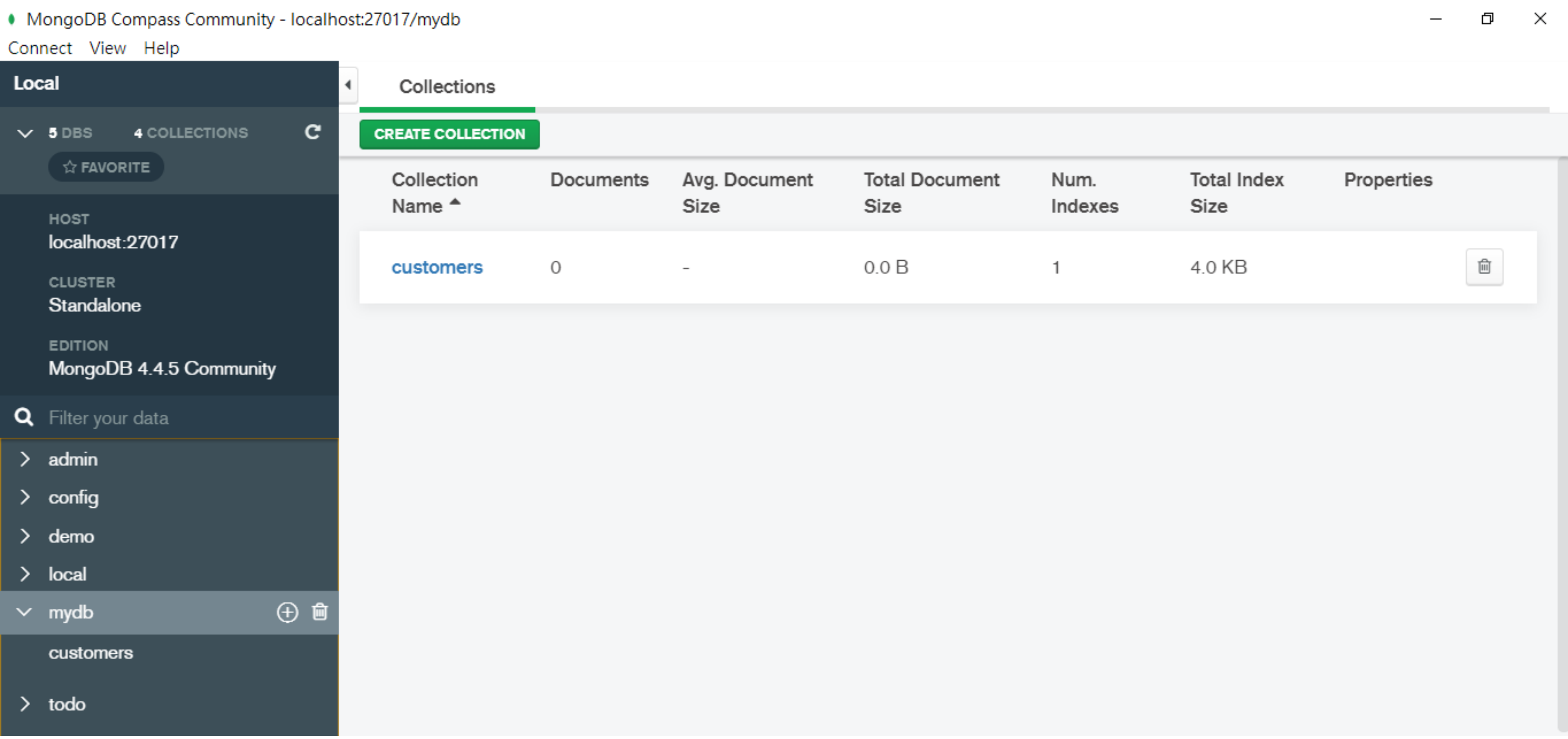
- One of the most popular NoSQL database is MongoDB.
  - You can download a free MongoDB database at https://www.mongodb.com.
  - You can also get started with a MongoDB cloud service at https://www.mongodb.com/cloud/atlas.
- Install MongoDB Driver
  - To download and install the official MongoDB driver, open the Command Terminal and execute the following:

```
npm install mongodb
```

MongoDB

- Creating a Database
  - To create a database in MongoDB, start by creating a MongoClient object, then specify a connection URL with the correct ip address (如果是本地端就是localhost) and the name of the database you want to create.
  - MongoDB will create the database if it does not exist, and make a connection to it.

MongoDB Compass



HOST
localhost:27017

CLUSTER
Standalone

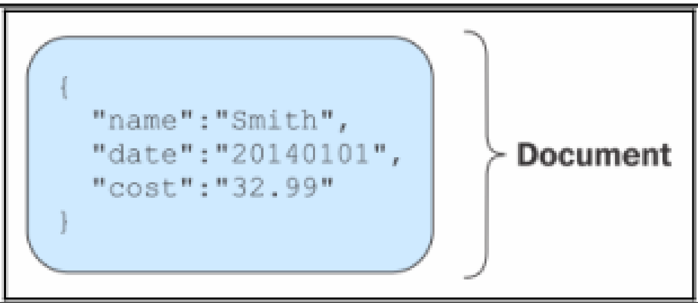
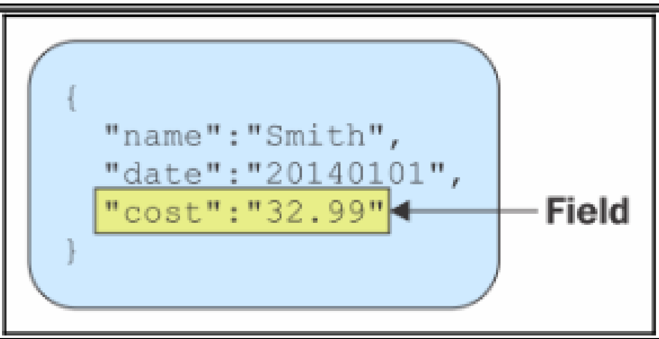
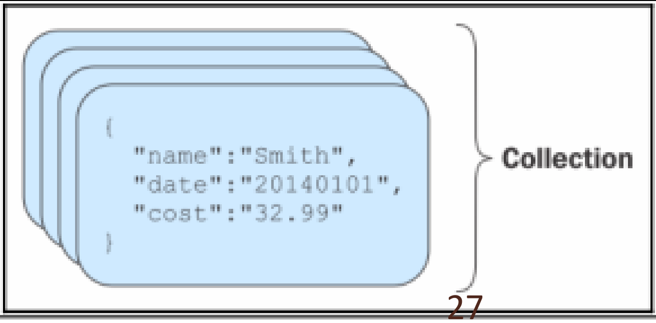
EDITION
MongoDB 4.4.5 Community

Collection Name	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size	Properties
customers	0	-	0.0 B	1	4.0 KB	

此為本地端的MongoDB管理工具，可以在上面創建Database

SOSEL
Service-Oriented Software Engineering Lab

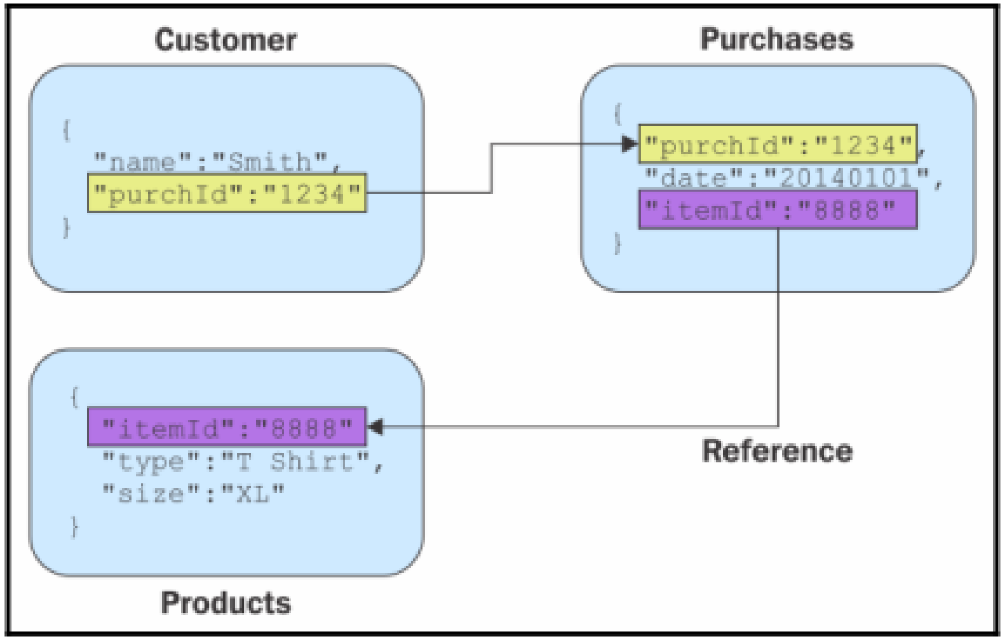
MongoDB and RDBMS

MongoDB	RDBMS	Illustration
Document	Row	
Field	Column	
Collection	Table	

```
{
  "name":"Smith",
  "date":"20140101",
  "cost":"32.99"
}
```

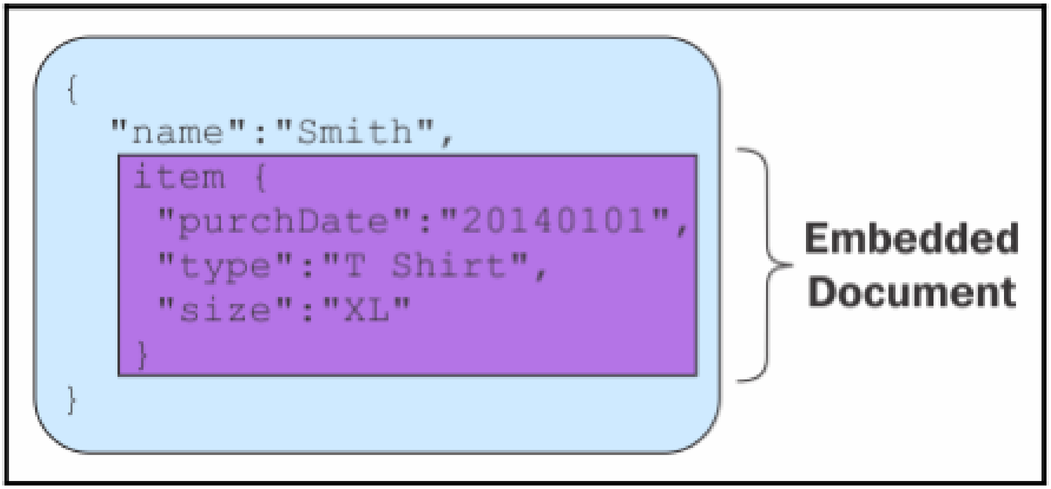
References

- Using references in MongoDB, it is possible to create a series of related collections in order to establish a normalized data model
  - https://docs.mongodb.com/manual/reference/database-references/



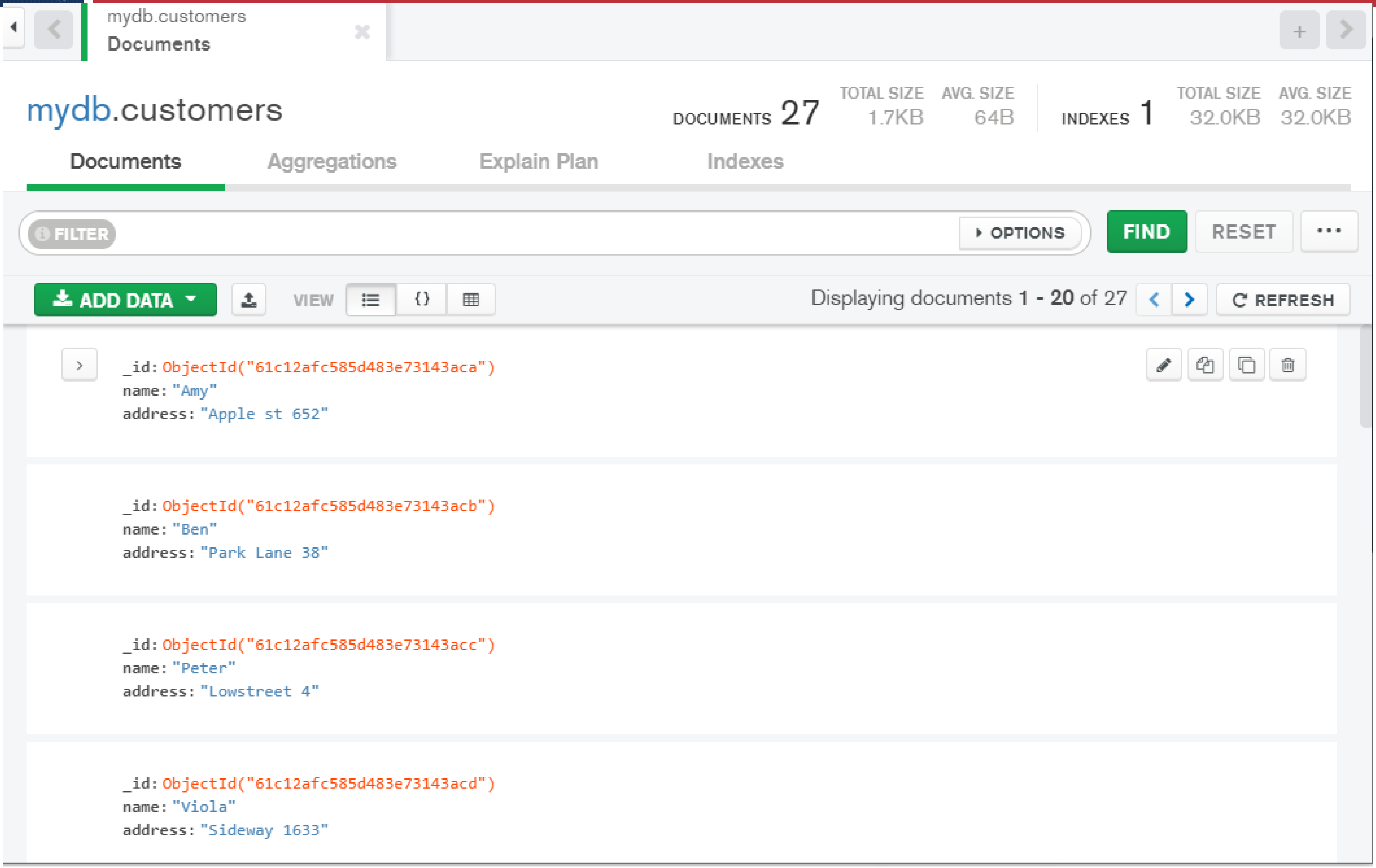
Embedded Documents

- A data-modeling solution in MongoDB would be simply to collapse the normalized relationships and fold the related information into embedded documents.
  - https://docs.mongodb.com/manual/core/data-model-design/#data-modeling-embedding



```
{
  "name":"Smith",
  item {
   "purchDate":"20140101",
   "type":"T Shirt",
   "size":"XL"
  }
}
```

MongoDB Collection Example (in Compass)



```
_id: ObjectId("61c12afc585d483e73143aca")
name: "Amy"
address: "Apple st 652"

_id: ObjectId("61c12afc585d483e73143acb")
name: "Ben"
address: "Park Lane 38"

_id: ObjectId("61c12afc585d483e73143acc")
name: "Peter"
address: "Lowstreet 4"

_id: ObjectId("61c12afc585d483e73143acd")
name: "Viola"
address: "Sideway 1633"
```

Node.js MongoDB Insert1

- To insert a document in MongoDB, into a collection, we use the insertOne() method.

```
//demo_mongodb_insert.js
const MongoClient = require('mongodb').MongoClient;
const url = "mongodb://localhost:27017/";

MongoClient.connect(url, function (err, db) {
    if (err) throw err;
    const dbo = db.db("mydb");
    let myobj = { name: "Company Inc", address: "Highway 37" };
    dbo.collection("customers").insertOne(myobj, function (err, res) {
        if (err) throw err;
        console.log("1 document inserted");
        db.close();
    });
});
```

The first parameter is an object containing the name(s) and value(s) of each field in the document you want to insert. It also takes a callback function where you can work with any errors, or the result of the insertion.

Node.js MongoDB Insert2

- To insert multiple documents into a collection in MongoDB, we use the insertMany() method.

```
const options = { ordered: true };
dbo.collection("customers").insertMany(myobjs, options,
    function (err, res) {
        if (err) throw err;
        console.log(`multiple documents were inserted`);
        db.close();
});
```

https://www.w3schools.com/nodejs/shownodejs_cmd.asp?filename=demo_mongodb_insert

The first parameter of the insertMany() method is an array of objects, containing the data you want to insert.

It also takes a callback function where you can work with any errors, or the result of the insertion:

Node.js MongoDB Find1

- To select data from a collection in MongoDB, we can use the findOne() method.
  - The findOne() method returns the first occurrence.

```
MongoClient.connect(url, function (err, db) {
    if (err) throw err;
    let dbo = db.db("mydb");
    dbo.collection("customers").findOne({ name: 'Amy' },
    function (err, result) {
        if (err) throw err;
        console.log(result.name + ": " + result.address);
        db.close();
    });
});
```

Node.js MongoDB Find₂

- To select data from a collection in MongoDB, we can also use the find() method.
  - The find() method returns all occurrences in the selection.

```
MongoClient.connect(url, function (err, db) {
    if (err) throw err;
    var dbo = db.db("mydb");
    dbo.collection("customers").find({}).
    toArray(function (err, result) {
        if (err) throw err;
        console.log(result);
        db.close();
    });
});
```

Node.js MongoDB Delete

- To delete document, we use the deleteOne() method.
  - The first parameter is a query object defining which document to delete.

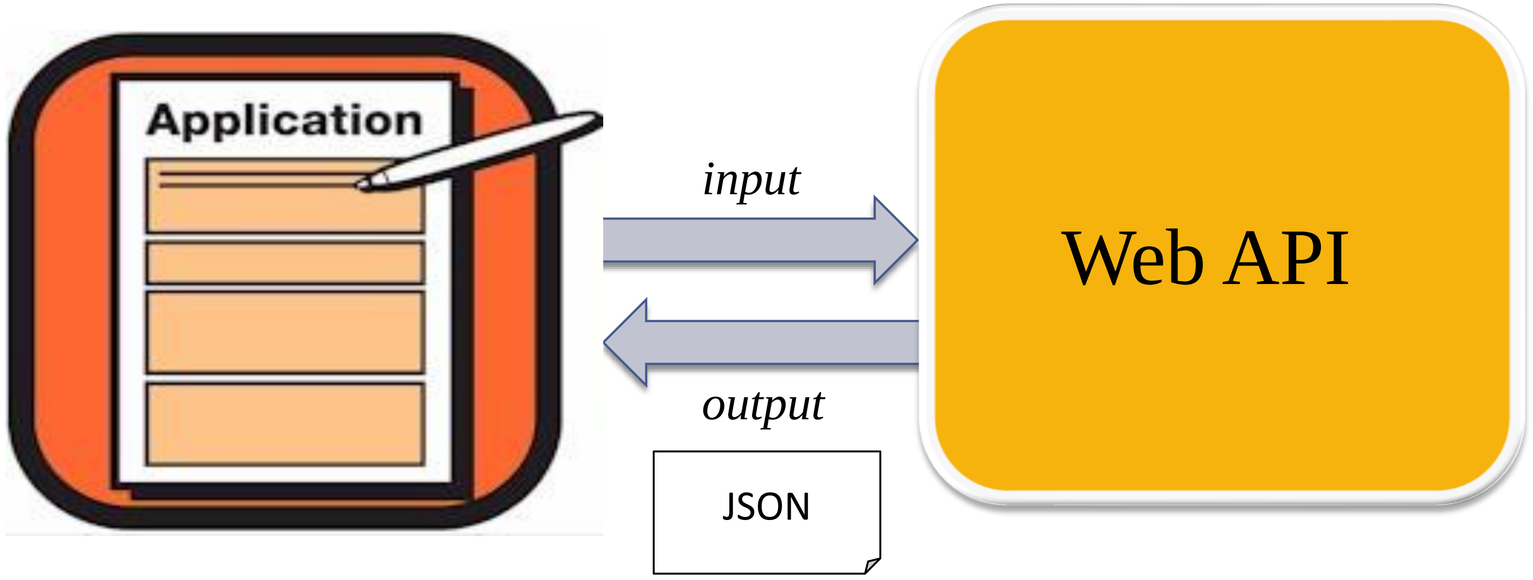
```
MongoClient.connect(url, function (err, db) {
  if (err) throw err;
  const dbo = db.db("mydb");
  let myquery = { address: 'Mountain 21' };
  dbo.collection("customers").deleteOne(myquery, function (err, obj) {
    if (err) throw err;
    console.log("1 document deleted");
    db.close();
  });
});
```

Concepts of Web APIs

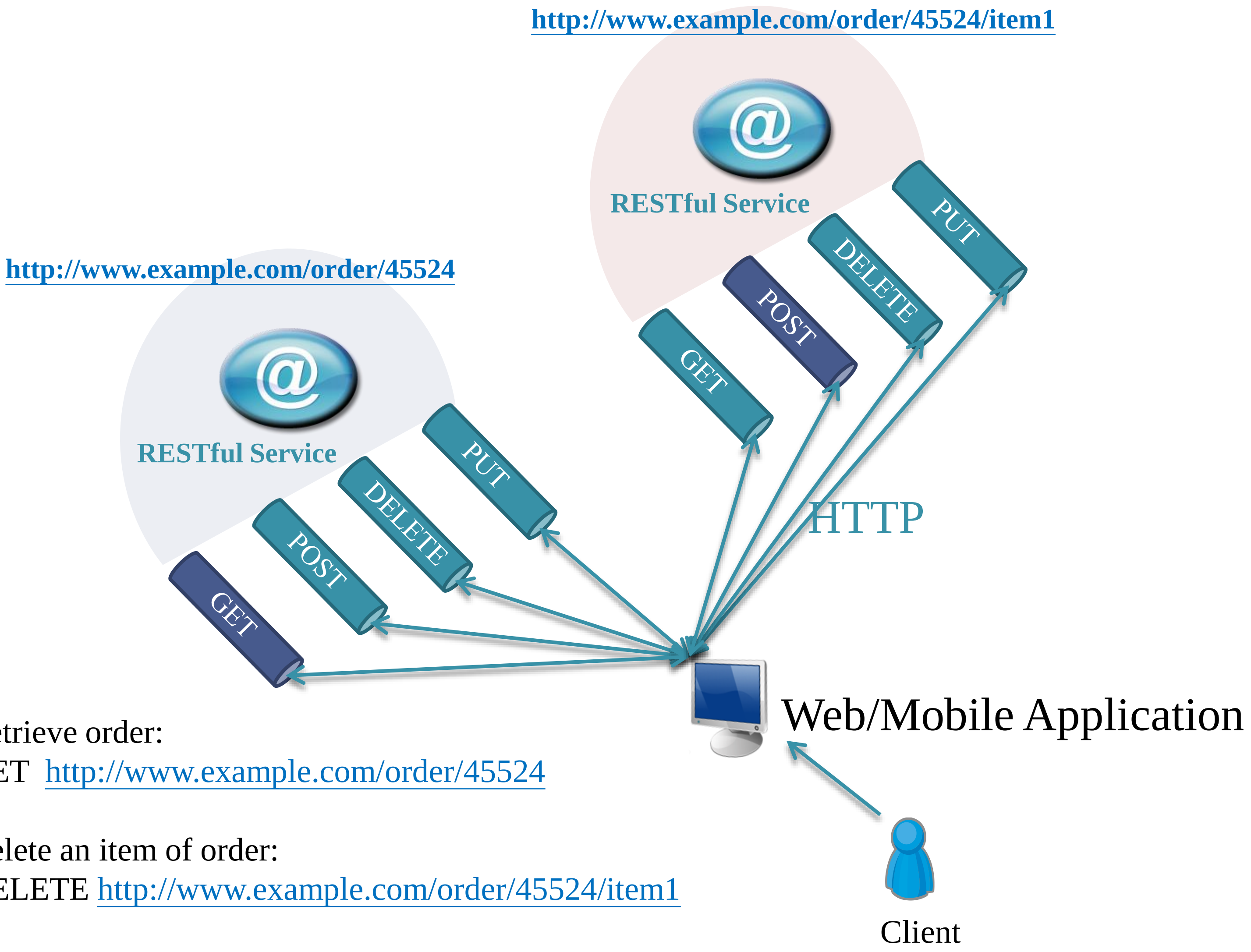
SOSEL
Service-Oriented Software Engineering Lab

Web API

- A Web API is a development in Web services where emphasis has been moving to simpler **REST-**based communications. (also called RESTful service)



RESTful Web Service



Retrieve order:
GET http://www.example.com/order/45524

Delete an item of order:
DELETE http://www.example.com/order/45524/item1

HTTP Methods (Cornerstone of RESTful Services)

- GET: to get the thing (resource / file) at the requested URL. (讀取)
- POST: to ask the server to accept the body info attached to the request, and give it to the thing at the requested URL. (新增)
- PUT: to put the enclosed info (the body) at the requested URL. (更換)
- Patch: to apply partial modifications to a resource. (更新)
- DELETE: to delete the thing (resource / file) at the requested URL. (刪除)

Express 101

express

- express 是小型 Node.js Web 應用程式架構。
  - https://expressjs.com/
  - https://expressjs.com/zh-tw/

建立開發環境

- 安裝node.js
  - https://nodejs.org/en/
- 安裝express
  - 建立專案目錄(如myapp)，並切換至此目錄
  - 於目錄下開啟console視窗 (windows為cmd視窗)，輸入底下指令：

```
npm init
```

    - 這個指令會提示設定一些事項，如應用程式的名稱和版本，我們可接受大部分的預設值，除了entry point要改為app.js
  - 接著將 Express 安裝在 myapp 目錄中，並儲存在相依關係清單中

```
npm install express --save
```

Hello World1

- 於目錄下建立app.js程式：

```
const express = require('express’);
const app = express();
const port = 3000;

app.get('/', (req, res) => {
    res.send('Hello World!')
});

app.listen(port, () => {
    console.log(`Example app listening at http://localhost:${port}`)
});
```

應用程式會啟動伺服器，並在埠 3000 接聽連線。

應用程式對指向根 URL (/) 或路由的要求，以 “Hello World!” 回應。(對於其他每一個路徑，它的回應是 404 找不到。)

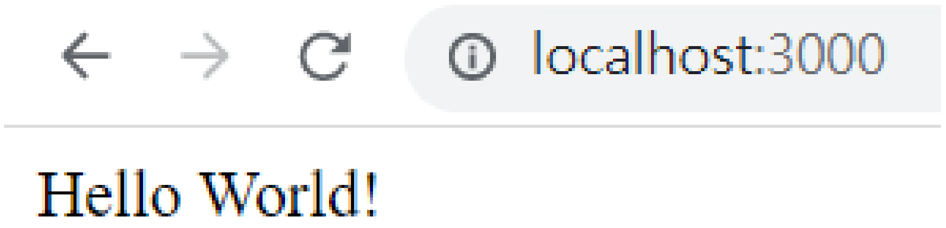
- 箭頭函式(Arrow Function): https://www.w3schools.com/js/js_arrow_function.asp
- Template Literals: https://www.w3schools.com/js/js_string_templates.asp

Hello World₂

- 使用下列指令來執行應用程式：

```
node app.js
```

然後在瀏覽器中載入 http://localhost:3000/ ，以查看輸出。應會看到頁面上出現Hello World。



```
Hello World!
```

Express 應用程式產生器

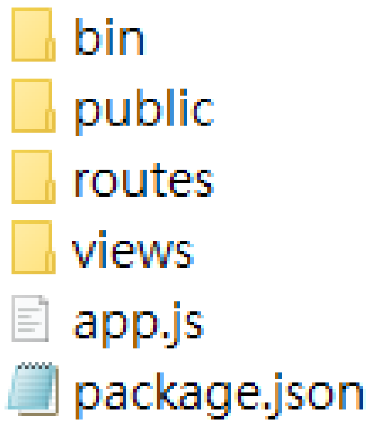
- 可使用應用程式產生器工具express，快速建立應用程式架構。
  - 使用下列指令來安裝 express：

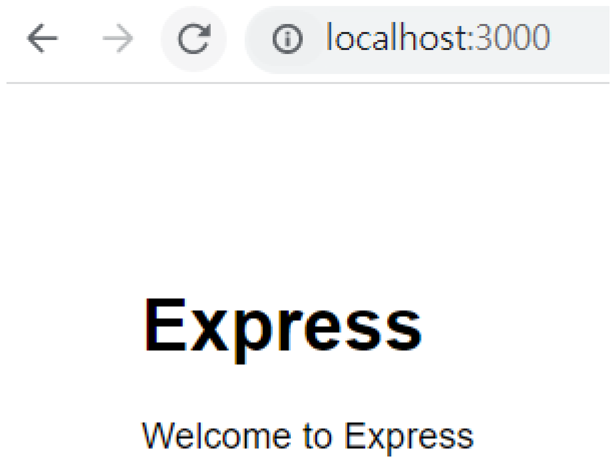
    npm install express-generator -g
  - 接著即可在現行工作目錄中建立一個 Express 應用程式的架構。

    express myapp (建立一個myapp專案)
  - 然後安裝相依項目：

    cd myapp
    npm install
  - 最後即可執行

    npm start





Express

Welcome to Express

Express專案結構1

- bin:
  - 專案啟動的位置，www為專案的進入點，負責啟動Node.js server，並呼叫專案底下的app.js，以啟動程式。
  - 可修改www去調整port (例如從3000改為8888)。
- node_modules:
  - 專案中必要的模組以及npm安裝的模組均在此目錄。通常不會手動改這個目錄的內容。
- public:
  - 儲存靜態檔案(如HTML、CSS、前端JavaScript、圖片檔等)。
  - 可以透過前端JavaScript去呼叫後端JavaScript提供的API。

Express專案結構2

- routes:
  - 存放路由文件，負責傳遞資料、設定API路由路徑等。
- views:
  - 存放顯示畫面用的樣板文件。預設格式是Jade樣板引擎。(如是前後端完全分離架構，可先忽略此塊)
- app.js:
  - 程式進入點
- package.json:
  - 專案組態設定檔，如專案描述、自訂指令、安裝套件等。
  - 預設會有一個自訂指令start。

基本路由1

- 路由是指判斷應用程式如何回應用戶端對特定端點(endpoint)的要求，而這個特定端點是一個 URI 與一個特定的 HTTP 要求方法(GET、POST 等)。
- 每一個路由可以有一或多個處理程式函式，當路由相符時，就會執行這些函式。
- 路由定義的結構如下：

```
app.METHOD(PATH, HANDLER)
```

- app 是 express 的實例。
- METHOD 是 HTTP 要求方法。
- PATH 是伺服器上的路徑。
- HANDLER 是當路由相符時要執行的函數。

基本路由2

- 可修改routes子目路底下的index.js，增加路由設定與處理函式 (底下均為simple-express範例)
  - 例子：對 /hello路由發出 GET 要求時的回應(JSON)：

```
router.get('/hello', function (req, res) {
  res.send({message: 'Hello World!'});
});
```

  - 例子：對根路由(/)發出POST 要求時的回應(JSON)：

```
router.post('/', function (req,
res) {
  res.send({status: 'OK!'});
});
```

基本路由3

- 例子：對 /hello路由發出 PUT 要求時的回應(JSON)：

```
router.put('/hello', function (req, res) {
  res.send({message: 'Hello New World!'});
});
```

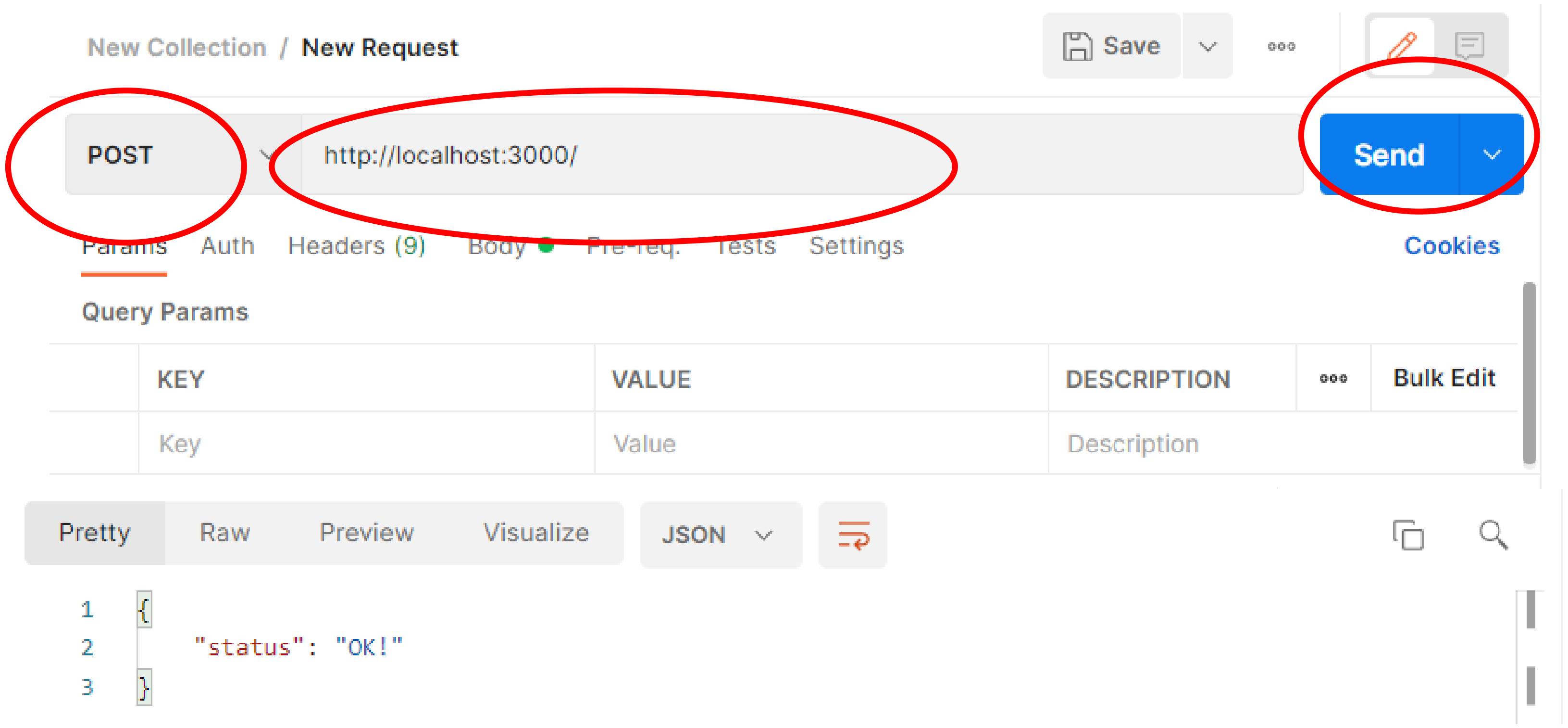
- 例子：對 /hello路由發出 DELETE 要求時的回應(JSON)：

```
router.delete('/hello', function (req, res) {
  res.send({status: 'Done!'});
});
```

這樣我們就建立了最為簡單的API

使用Postman去測試API

- Postman: https://www.postman.com/downloads/
- 可使用Postman去測試以各API端點 (method + URL)。



```
{
    "status": "OK!"
}
```

在 Express 中提供靜態檔案

- 如果要提供影像、CSS 檔案和 JavaScript 檔案等之類的靜態檔案，請使用 Express 中的 express.static 內建中介軟體函數。
  - 將含有靜態資產的目錄名稱傳遞給 express.static 中介軟體函數，就能直接開始提供檔案。
  - 舉例來說，使用下列程式碼在名稱是 public 的目錄中，提供影像、CSS 檔案和 JavaScript 檔案：

```
app.use(express.static('public'));
```

新增前端頁面

- 於"public"目錄下新增html檔案(如index.html)
  - CSS連結至stylesheets子目錄下的style.css
  - JavaScript連結至javascripts子目錄下的.js檔(需新增)

前端頁面: HTML1

- 引入CSS樣式檔、jQuery函式庫(optional)、你的JavaScript程式連結。(此範例使用jQuery函式庫)

```
<head>
    <link rel=stylesheet type="text/css" href="stylesheets/style.css">
    <script type="text/javascript" src="https://code.jquery.com/jquery-
3.6.1.min.js">
    </script>
    <script type="text/javascript" src="javascripts/myapp.js"></script>
    <title>TodoList</title>
</head>
```

前端頁面: HTML2

- 加入HTML主體內容。以此範例而言，是四個button (點擊後會呼叫四個不同的function)，並增加一個div區塊用以顯示資訊。

```
<body>
    <button onclick="getTest()">GET Test</button>
    <button onclick="postTest()">Post Test</button>
    <button onclick="putTest()">PUT Test</button>
    <button onclick="deleteTest()">DELETE Test</button>
    <hr>
    <div id="display"></div>
</body>
```

前端頁面: JavaScript1

- 透過jQuery的get/post函式去異步呼叫(非同步呼叫)後端API，結果再呈現於頁面中(以此案例為div區塊)。

```
function getTest() {
    $.get("hello", function (res) {
        document.getElementById("display").innerHTML = res.message;
    });
}

function postTest() {
    $.post("/", { name: "Curry" }, function (res) {
        document.getElementById("display").innerHTML = res.status;
    }); //假裝有要發布的資料
}
```

前端頁面: JavaScript2

透過jQuery的ajax函式(更一般化的用法)去呼叫後端API，結果再呈現於頁面中(以此案例為div區塊)。

```
function putTest() {
    $.ajax({
        method: "PUT",
        url: "hello",
        data: { name: "Curry" } //假裝有更新的資料
    }).done(function (res) {
        document.getElementById("display").innerHTML = res.message;
    });
}

function deleteTest() {
    $.ajax({
        method: "DELETE",
        url: "hello"
    }).done(function (res) {
        document.getElementById("display").innerHTML = res.status;
    });
}
```

jQuery: AJAX

- AJAX is the art of exchanging data with a server, and update parts of a web page - without reloading the whole page.

Method	Description
$.ajax()	Performs an async AJAX request

$.get()	Loads data from a server using an AJAX HTTP GET request
$.getJSON()	Loads JSON-encoded data from a server using a HTTP GET request

$.post()	Loads data from a server using an AJAX HTTP POST request

$.ajax()可以發送GET、POST、PUT、DELETE等各種請求

https://www.w3schools.com/jquery/jquery_ref_ajax.asp

Mongoose

What is Mongoose?

- Mongoose provides a schema-based solution to model your application data.
  - It includes built-in type casting, validation, query building, business logic hooks and more, out of the box.
  - It realizes ODM (Object Data Model).
  - https://mongoosejs.com/

Mongoose Example

- Mongoose中文參考資料：https://hackmd.io/@Agry/HJu3KSZoc
  - 範例：todo-moongoose專案
- 前置安裝程序
  - 先使用Express Generator生成目錄 (*express yourapp*)
  - 安裝mongoose
    - *npm install mongoose*
  - 啟動程式
    - *npm start*

Mongoose Example: 設定app.js

```
const express = require('express');
const mongoose = require("mongoose");
const app = express();

// 引入設定檔
const config = require("./config/config");

// 與資料庫連線
mongoose.connect(config.db.url);
const db = mongoose.connection;

// 與資料庫連線發生錯誤時
db.on('err', err => console.log(err));

// 與資料庫連線成功連線時
db.once('open', () => console.log('connected to database...'));

// 引入Router 一個Router基本上處理
一個資料表
const todoRouter =
require("./routes/todo");

// 此處的/todo代表連線到該Router的
基本路徑為 http://localhost:3000/todo
app.use("/todo", todoRouter);

module.exports = app;
```

Mongoose Example: 設定組態檔

- 在根目錄新增一個config資料夾，並新增config.json作為組態設定檔。

```
// config.js
const config = {
    app: {
        port: 3000
    },
    db: {
        url: "mongodb://localhost/test"
    }
};

module.exports = config;
```

Mongoose Example: 新增Model

- 在根目錄新增一個models資料夾，並新增todo.js以設置model。
  - 此model對應之MongoDB Collection名稱為設定名稱的複數(此案例為todos)。

https://mongoosejs.com/docs/models.html

```
const todoSchema = new mongoose.Schema({
    thing: { //事項名稱
        type: String, //設定該欄位的格式
        required: true //設定該欄位是否必填
    },
    isDone: { //是否已完成
        type: Boolean,
       default: false //設定預設值
    },
    createdDate: { //新增的時間
        type: Date,
        default: Date.now,
        required: true
    },
})
//匯出該Model類別
module.exports = mongoose.model("todo", todoSchema);
```

Mongoose Example: 設定Router1

- 於routes/todo.js設定router相關邏輯。

```
const Todo = require("../models/todo");

// 使用非同步，才能夠等待資料庫回應
router.get("/", async (req, res) => {
    try {
        // 找出Todo資料資料表中的全部資料
        const todo = await Todo.find();
        // 將回傳的資訊轉成Json格式後回傳
        res.json(todo);
    } catch (err) {
        // 如果資料庫出現錯誤時回報 status:500 並回傳錯誤訊息
        res.status(500).json({ message: err.message })
    }
});
```

Mongoose Example: 設定Router2

```
// 新增待辦事項，將Method改為Post
router.post("/", async (req, res) => {
    // 從req.body中取出資料
    const todo = new Todo({
        thing: req.body.thing,
        isDone: req.body.isDone,
    });
    try {
        // 使用.save()將資料存進資料庫
        const newTodo = await todo.save();
        // 回傳status:201代表新增成功 並回傳新增的資料
        res.status(201).json(newTodo);
    } catch (err) {
        // 錯誤訊息發生回傳400 代表使用者傳入錯誤的資訊
        res.status(400).json({ message: err.message })
    }
});
```

Mongoose Example: 設定Router3

```
// 檢視是否有指定ID之待辦事項 (作為Middleware)
async function getTodo(req, res, next) {
    let todo;
    try {
        todo = await Todo.findById(req.params.id);
        if (todo == undefined) {
            return res.status(404).json({ message: "Can't find todo" })
        }
    } catch (err) {
        return res.status(500).json({ message: err.message })
    }
    // 如果有該事項 則將他加入到res中
    res.todo = todo
    // 在router中執行middleware後需要使用next()才會繼續往下跑
    next();
}
```

Middleware: https://expressjs.com/zh-tw/guide/using-middleware.html

Mongoose Example: 設定Router4

```
// 在網址中傳入id用以查詢，會先執行getTodo後才繼續裡面的內容
router.get("/:id", getTodo, (req, res) => {
   // 取出res.todo並回傳
   res.send(res.todo);
});

// 刪除待辦事項，先使用getTodo取得該待辦資訊
router.delete("/:id", getTodo, async (req, res) => {
   try {
      // 將取出的待辦事項刪除
      await res.todo.remove();
      // 回傳訊息
      res.json({ message: "Delete todo succeed" })
   } catch (err) {
      // 資料庫操作錯誤將回傳500及錯誤訊息
      res.status(500).json({ message: "remove todo faild" })
   }
});
```

MongoDB Atlas

SOSEL

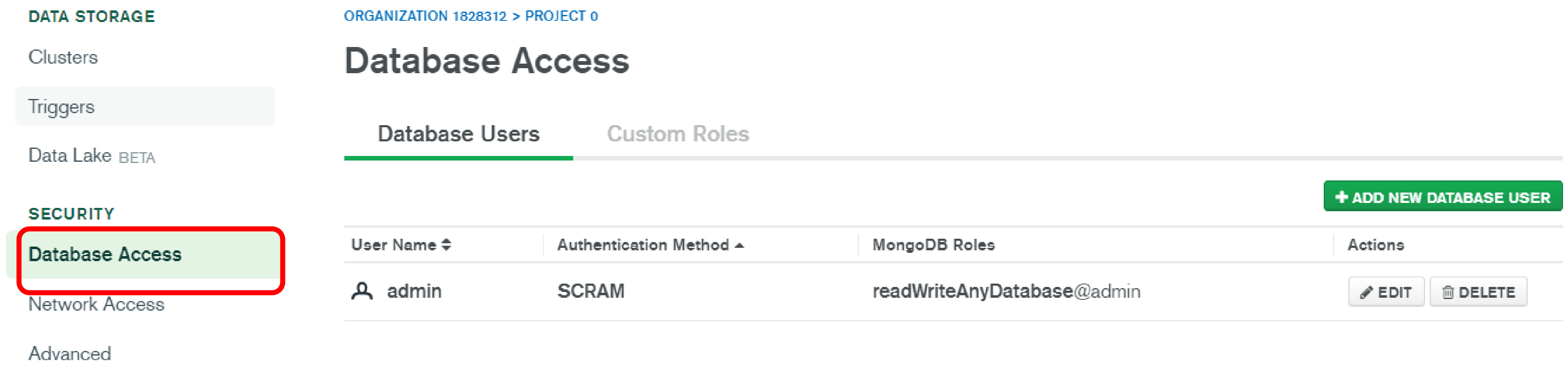
Service-Oriented Software Engineering Lab

MongoDB Atlas

- MongoDB Atlas is a cloud MongoDB.
  - https://www.mongodb.com/cloud/atlas
  - Atlas handles deploying, managing, and healing your deployments on the cloud (AWS, Azure, and GCP).
  - Avoiding the deployment issue of DB.

Configure Security in Atlas1

- Database Access: create a user to be the database administrator. (設定可存取雲端DB之帳密)



DATA STORAGE
- Clusters
- Triggers
- Data Lake BETA

SECURITY
- Database Access
- Network Access
- Advanced

ORGANIZATION 1828312 > PROJECT 0

Database Access

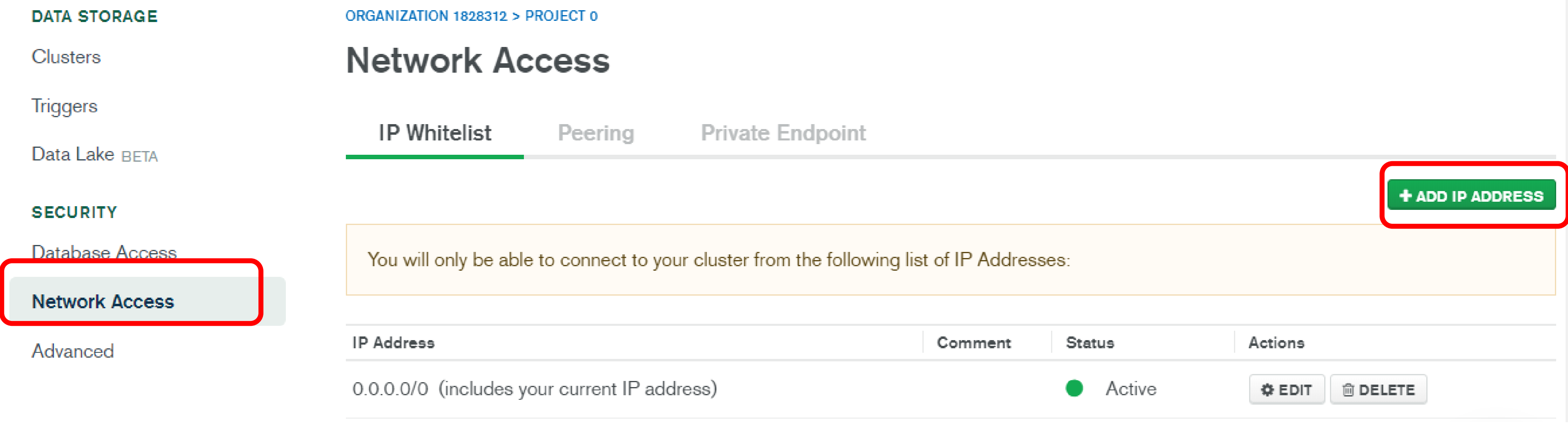
Database Users | Custom Roles

+ ADD NEW DATABASE USER

User Name	Authentication Method	MongoDB Roles	Actions
admin	SCRAM	readWriteAnyDatabase@admin	EDIT DELETE

Configure Security in Atlas₂

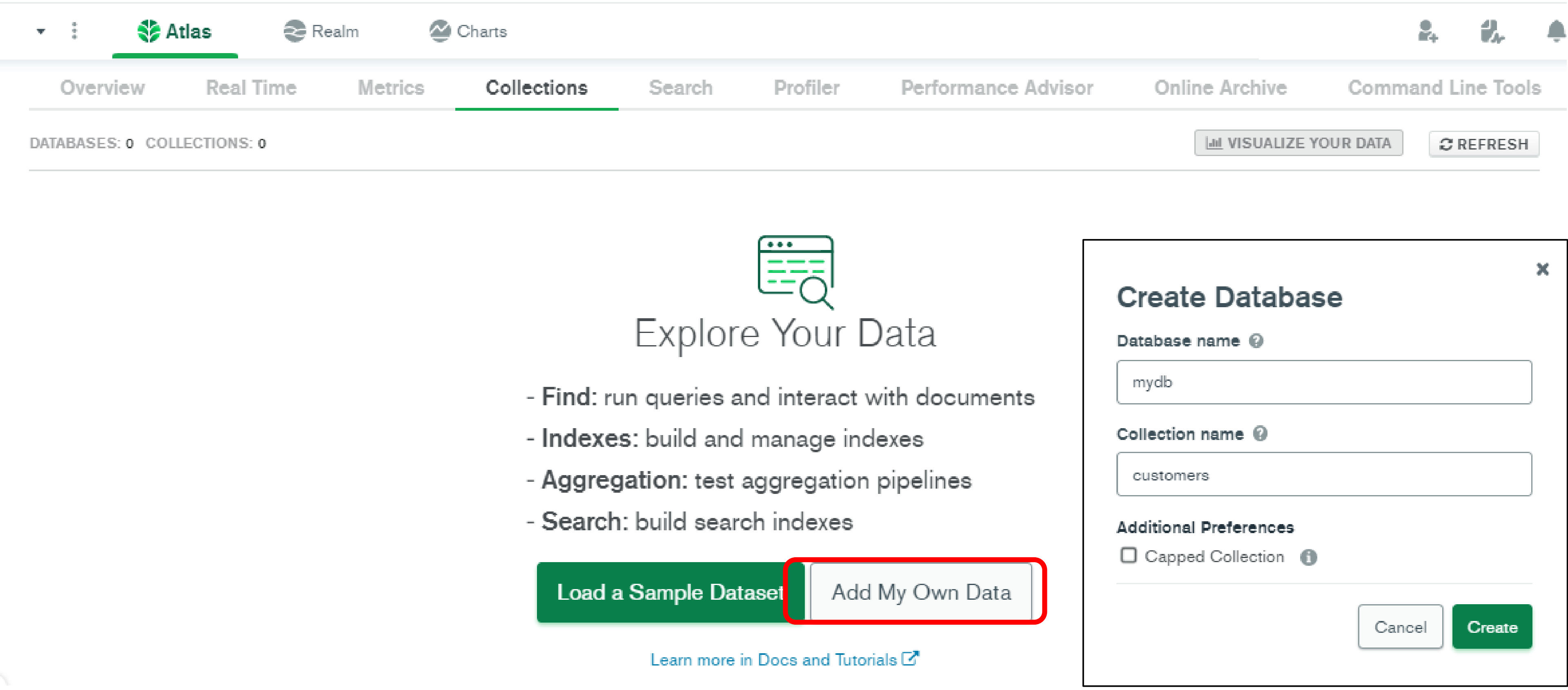
- Atlas only allows client connections to a cluster from entries in the project's whitelist.
- You can set the IP to 0.0.0.0/0 to allows access from anywhere. (若設定固定IP更為安全)
  - https://docs.atlas.mongodb.com/security-whitelist/



You will only be able to connect to your cluster from the following list of IP Addresses:

IP Address	Comment	Status	Actions
0.0.0.0/0 (includes your current IP address)		● Active	EDIT DELETE

Create Database



Explore Your Data

- **Find:** run queries and interact with documents
- **Indexes:** build and manage indexes
- **Aggregation:** test aggregation pipelines
- **Search:** build search indexes

Create Database

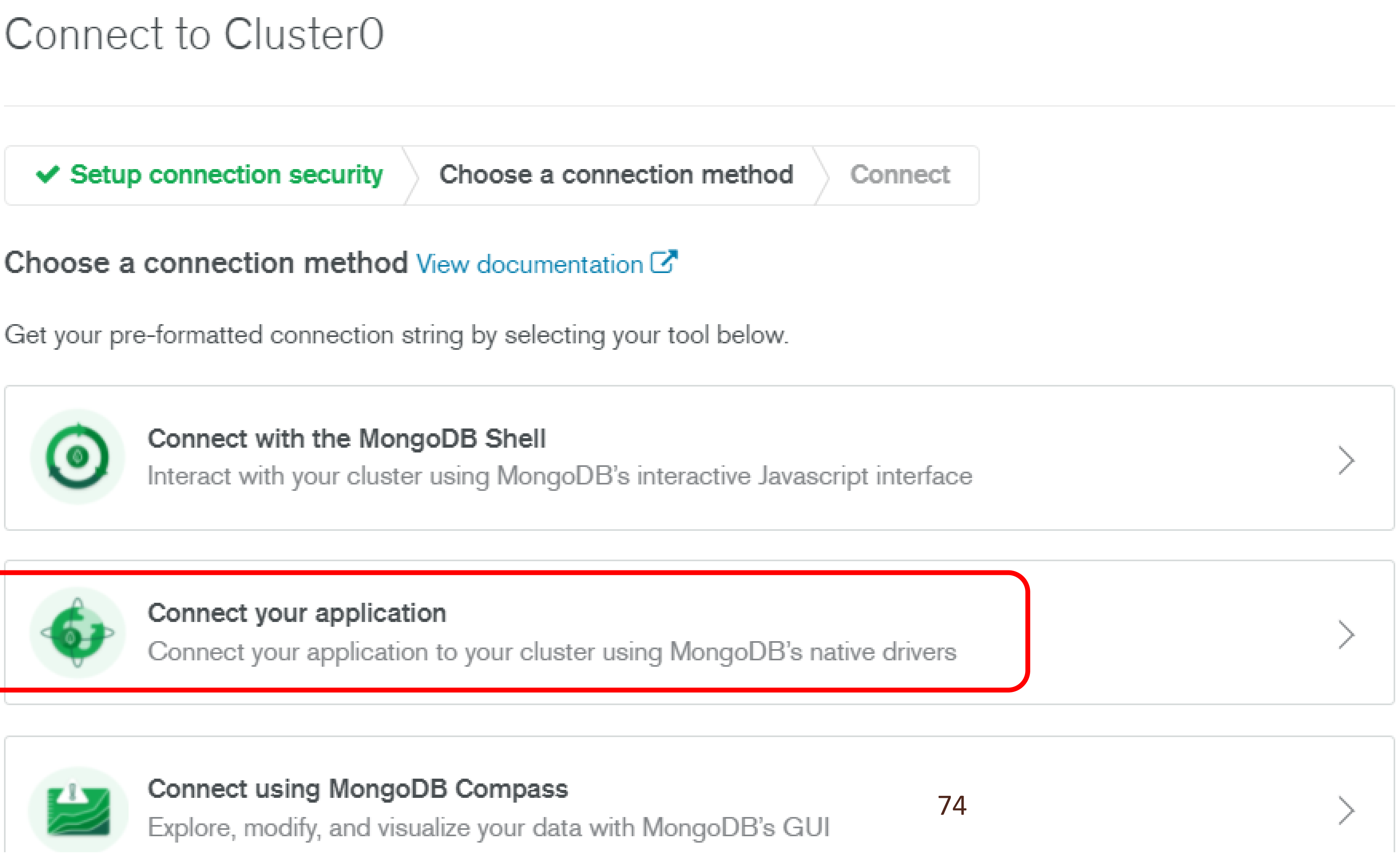
Database name: mydb

Collection name: customers

Additional Preferences

☐ Capped Collection

Connect Node.js to Atlas[1]



Connect to Cluster0

Choose a connection method View documentation

Get your pre-formatted connection string by selecting your tool below.

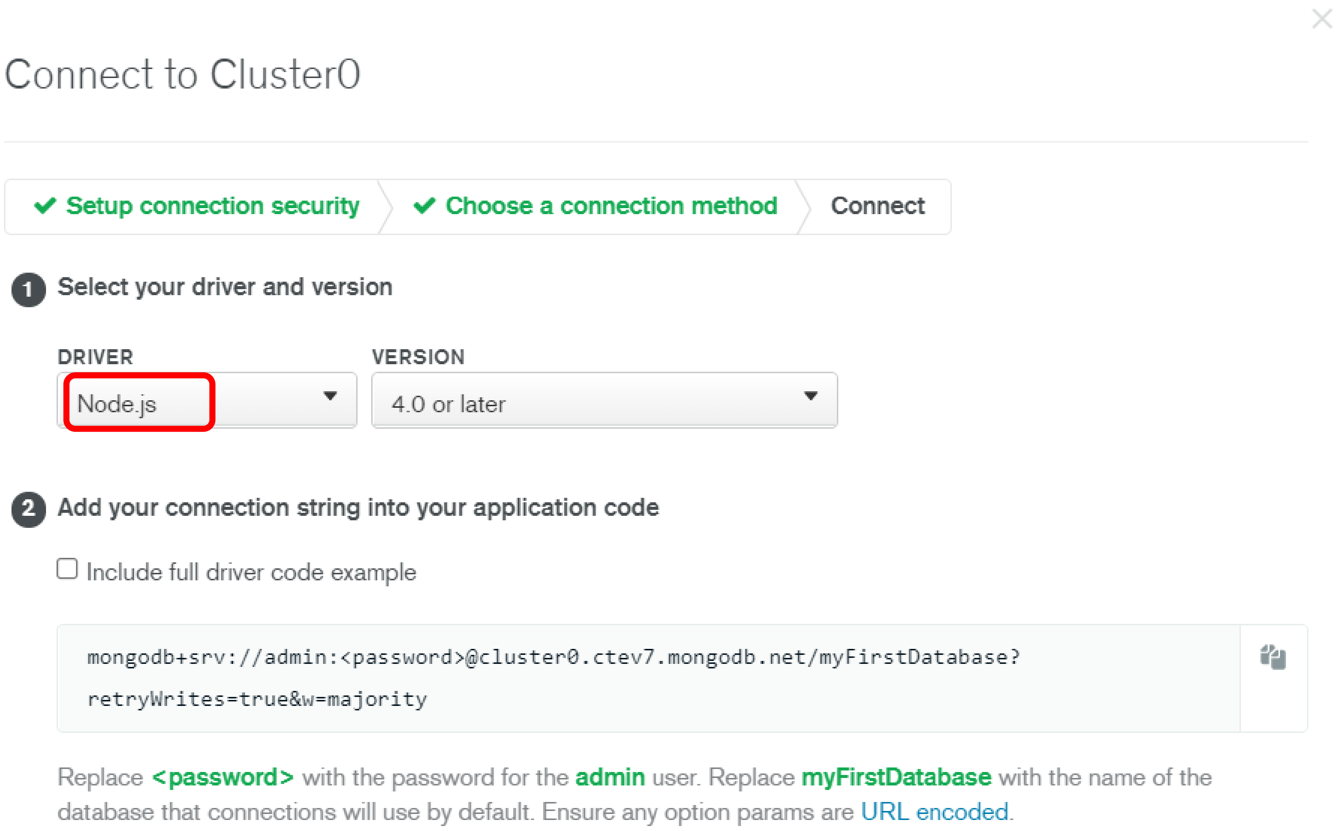
Connect with the MongoDB Shell
Interact with your cluster using MongoDB's interactive Javascript interface

Connect your application
Connect your application to your cluster using MongoDB's native drivers

Connect using MongoDB Compass
Explore, modify, and visualize your data with MongoDB's GUI

Connect Node.js to Atlas$_2$

- Copy the connection string for Node.js


Connect to Cluster0

1 Select your driver and version

DRIVER: Node.js
VERSION: 4.0 or later

2 Add your connection string into your application code

☐ Include full driver code example

```
mongodb+srv://admin:<password>@cluster0.ctev7.mongodb.net/myFirstDatabase?
retryWrites=true&w=majority
```

Replace **<password>** with the password for the **admin** user. Replace **myFirstDatabase** with the name of the database that connections will use by default. Ensure any option params are URL encoded.

Connect Node.js to Atlas3

- We can simply change the url of database to the connection string in the previous slide, we can connect Mongo Atlas in the cloud, rather than the local MongoDB.
  - Example of connection string: such as "*mongodb+srv://admin:<password>@cluster0.gt34a.mongodb.net/myFirstDatabase?retryWrites=true&w=majority*".
  - 若以先前mongoose之範例，則將設定檔中的url改為上述之connection string即可。

雲端部署node.js應用程式：Cyclic.sh

SOSEL
Service-Oriented Software Engineering Lab

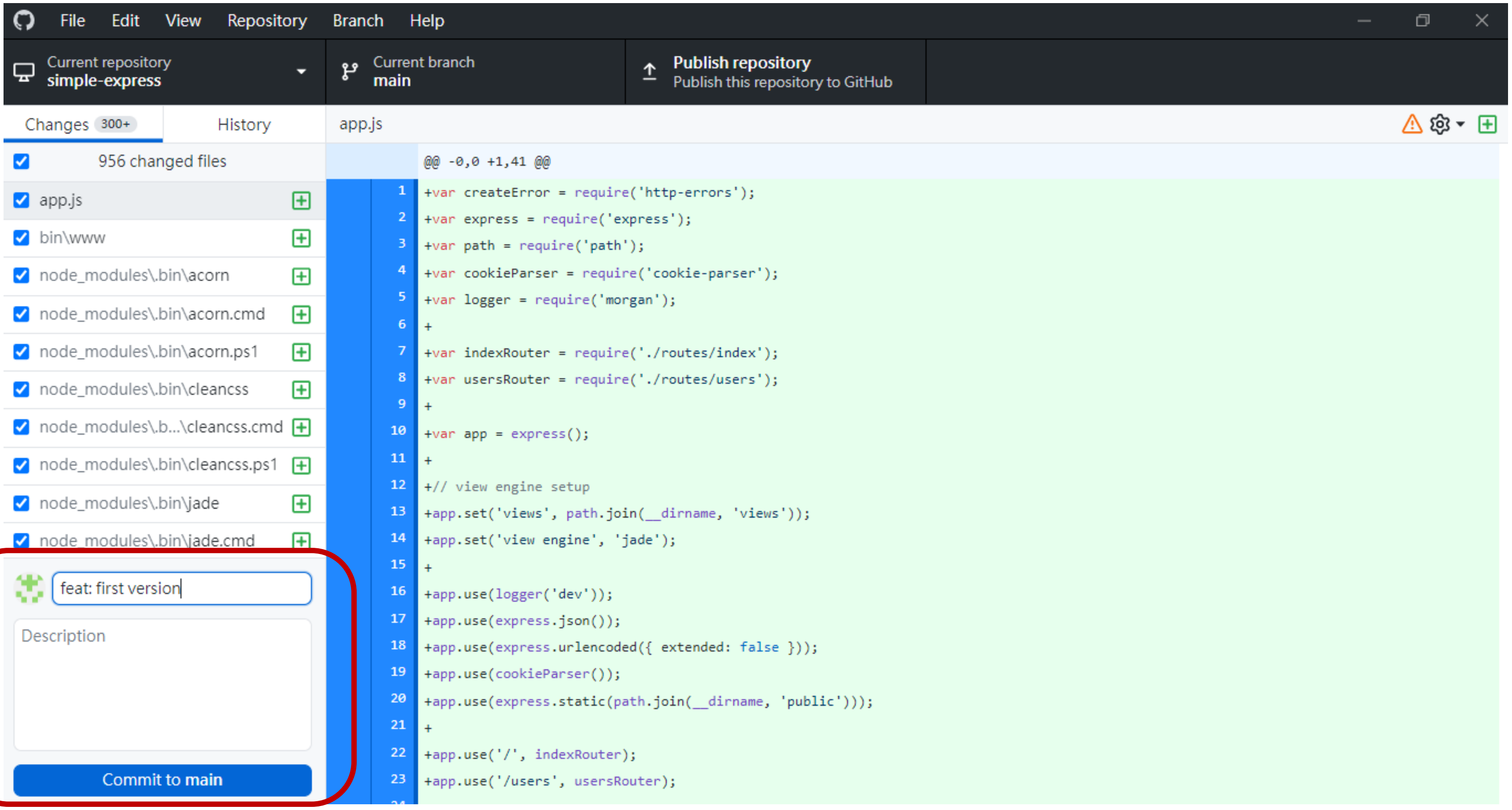
What is Cyclic.sh?

- https://www.cyclic.sh/
- 是一種PaaS (Platform as a Service)。
- 透過GitHub帳號登入後，授權Cyclic.sh應用程式，日後即可於專案push後自動部署至Cyclic.sh雲端。

如何運用GitHub Desktop進行版控？

- 先安裝GIT: https://git-scm.com/
- 再安裝GitHub Desktop: https://desktop.github.com/
- 於專案目錄下先透過GitBash或CMD執行git init，使目錄可受到GIT控管。
- 使用GitHub Desktop之[Add local repository]將專案加入工具之管理。
  - 記得先新增.gitignore檔案，排除對“node_modules”目錄之版控。
    - https://gitbook.tw/chapters/using-git/ignore

GitHub Desktop: Commit



956 changed files

- app.js
- bin\www
- node_modules\.bin\acorn
- node_modules\.bin\acorn.cmd
- node_modules\.bin\acorn.ps1
- node_modules\.bin\cleancss
- node_modules\.b...\cleancss.cmd
- node_modules\.bin\cleancss.ps1
- node_modules\.bin\jade
- node_modules\.bin\jade.cmd

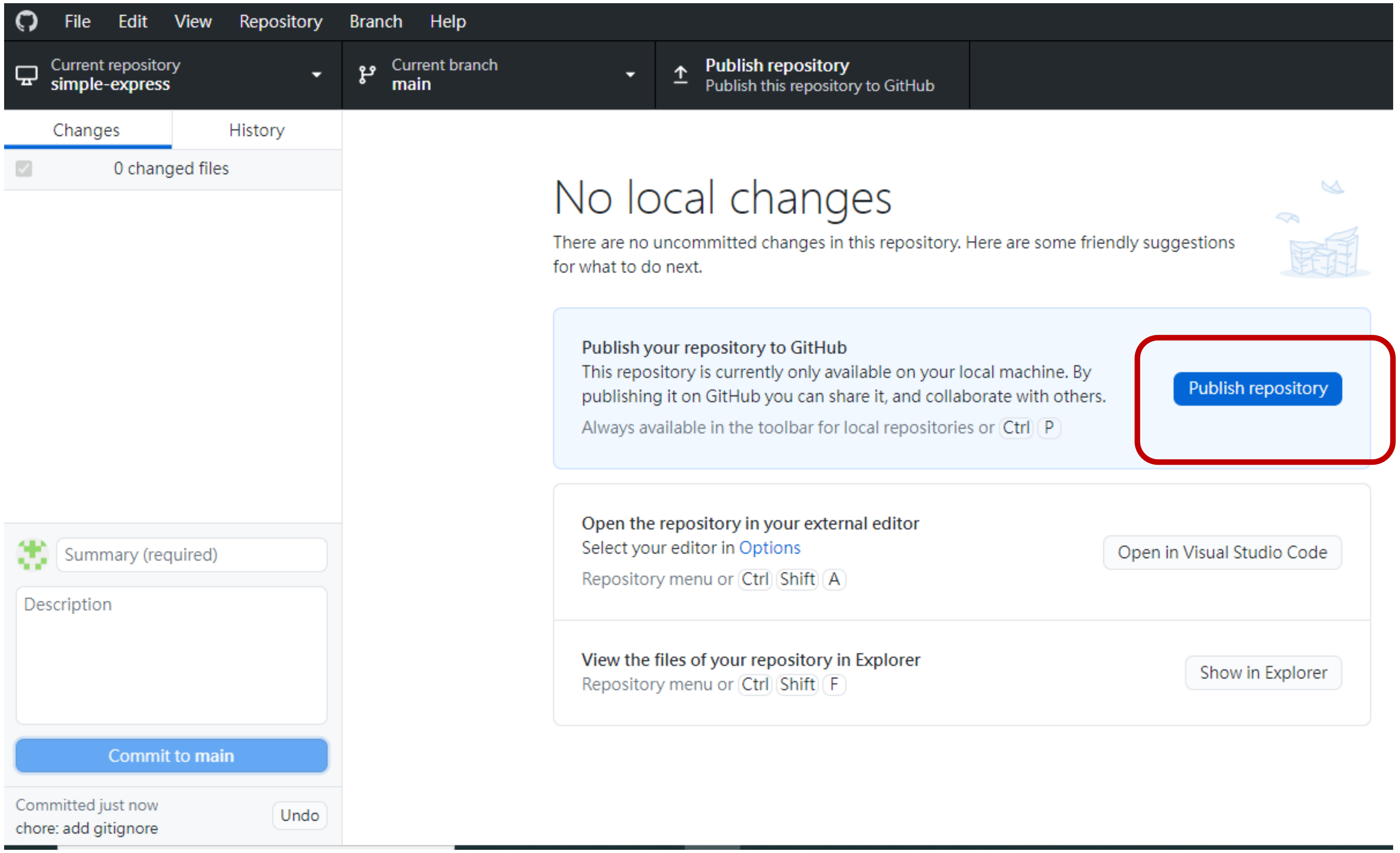
feat: first version

Description

Commit to main

```
@@ -0,0 +1,41 @@
+var createError = require('http-errors');
+var express = require('express');
+var path = require('path');
+var cookieParser = require('cookie-parser');
+var logger = require('morgan');
+
+var indexRouter = require('./routes/index');
+var usersRouter = require('./routes/users');
+
+var app = express();
+
+// view engine setup
+app.set('views', path.join(__dirname, 'views'));
+app.set('view engine', 'jade');
+
+app.use(logger('dev'));
+app.use(express.json());
+app.use(express.urlencoded({ extended: false }));
+app.use(cookieParser());
+app.use(express.static(path.join(__dirname, 'public')));
+
+app.use('/', indexRouter);
+app.use('/users', usersRouter);
```

GitHub Desktop: Publish/Push



No local changes

There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next.

Publish your repository to GitHub
This repository is currently only available on your local machine. By publishing it on GitHub you can share it, and collaborate with others.
Always available in the toolbar for local repositories or Ctrl P

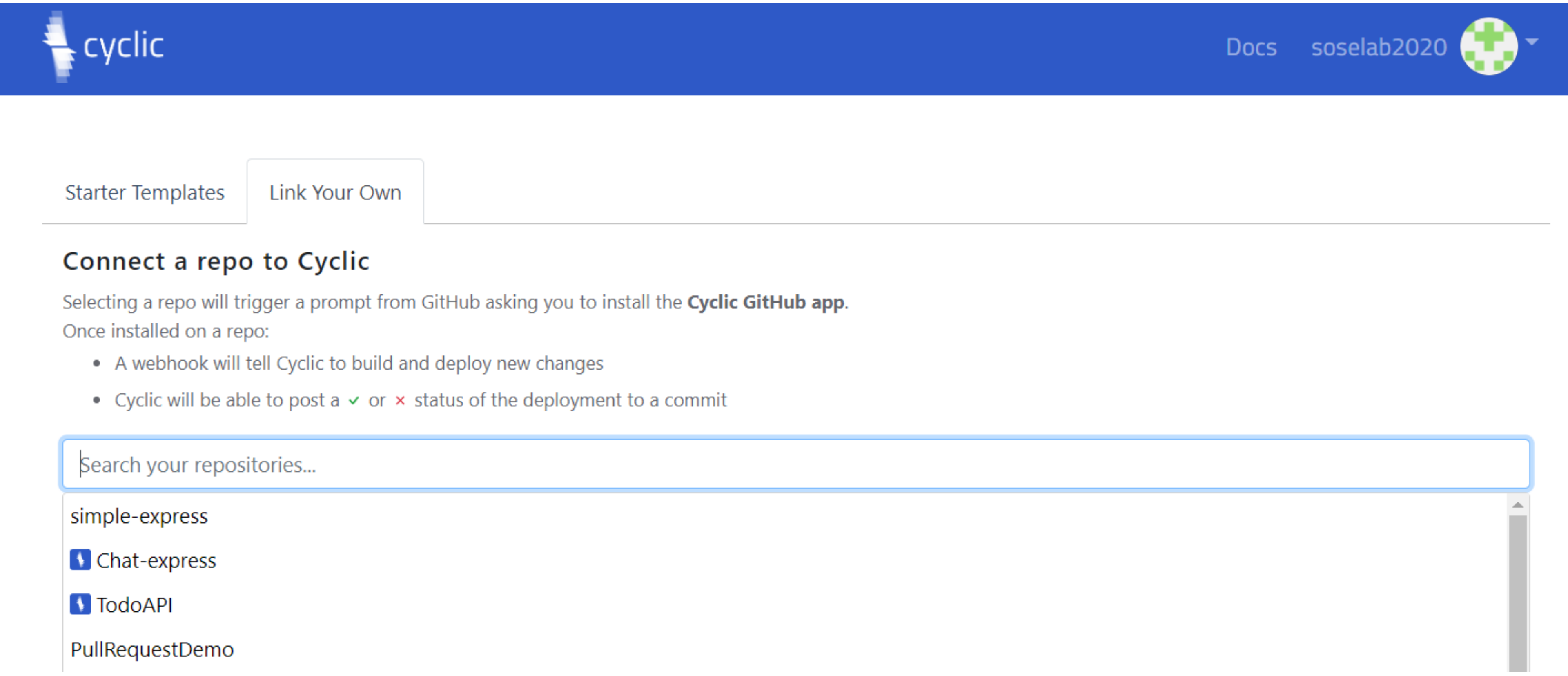
Open the repository in your external editor
Select your editor in Options
Repository menu or Ctrl Shift A

View the files of your repository in Explorer
Repository menu or Ctrl Shift F

Committed just now
chore: add gitignore

這樣即可推送專案至GitHub的同名repo中。

透過Cyclic.sh將應用程式部署至雲端



Connect a repo to Cyclic

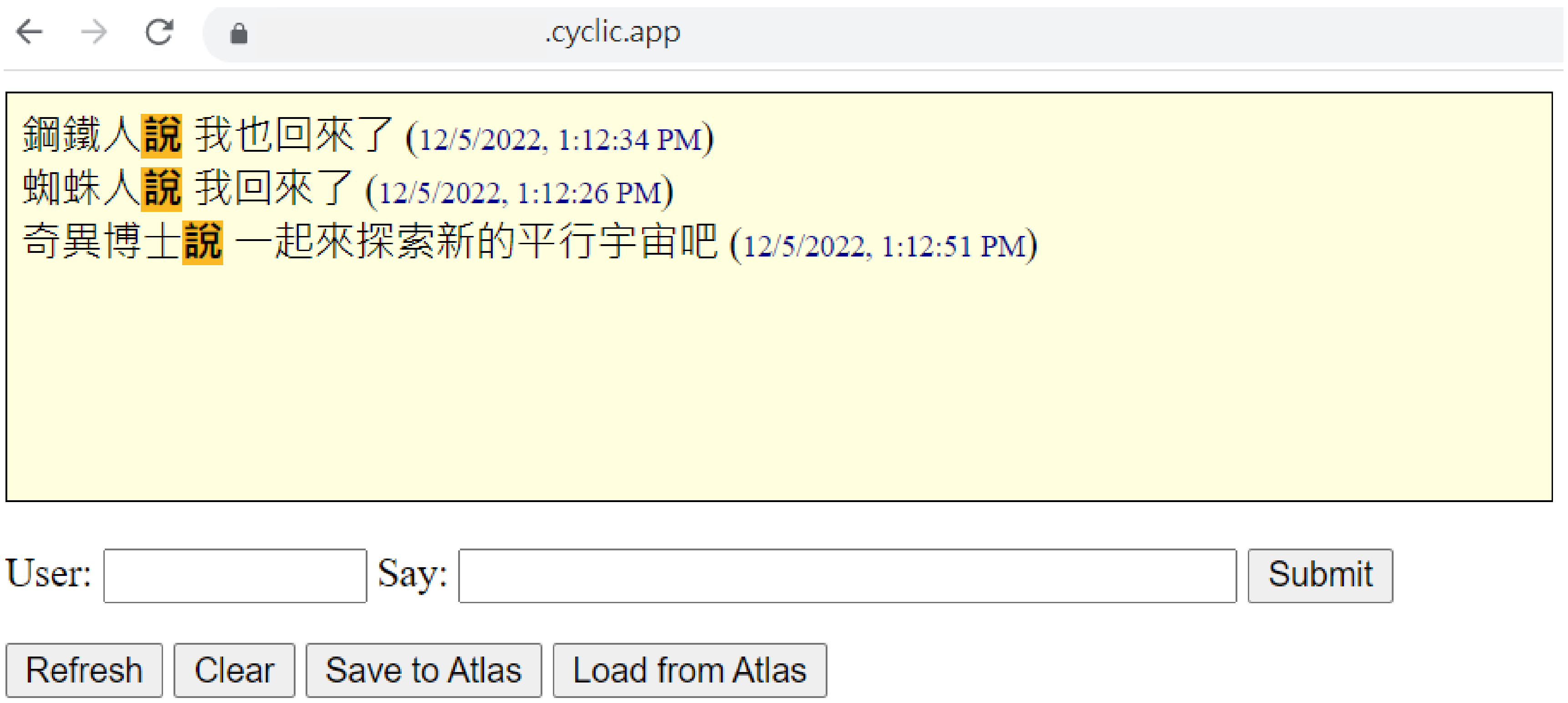
Selecting a repo will trigger a prompt from GitHub asking you to install the **Cyclic GitHub app**.
Once installed on a repo:

- A webhook will tell Cyclic to build and deploy new changes
- Cyclic will be able to post a ✓ or × status of the deployment to a commit

Search your repositories...

simple-express
Chat-express
TodoAPI
PullRequestDemo

部署好之預期結果畫面



.cyclic.app

鋼鐵人**說** 我也回來了 (12/5/2022, 1:12:34 PM)
蜘蛛人**說** 我回來了 (12/5/2022, 1:12:26 PM)
奇異博士**說** 一起來探索新的平行宇宙吧 (12/5/2022, 1:12:51 PM)

Any Question?