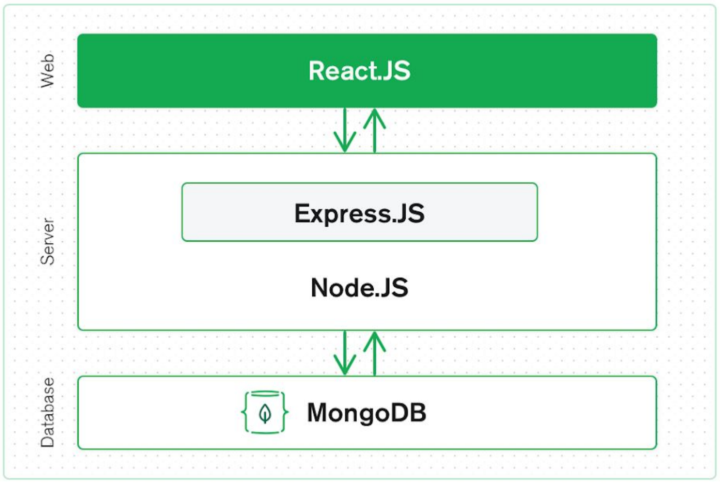
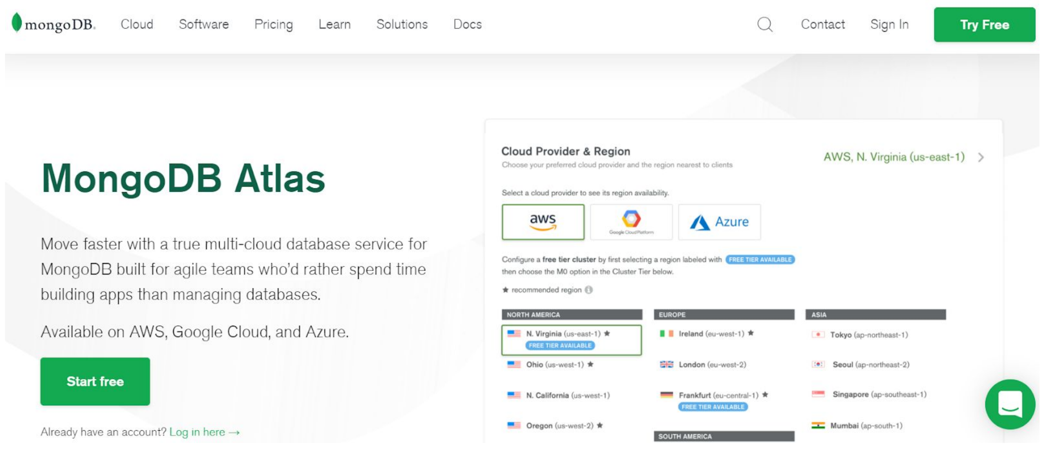
B06902127 資工四 鄭人愷

1. Framework
    a. Frontend: html and react.js
    b. Backend: node.js
    c. Database: MongoDB
    d. Middleware: express.js and websocket



2. Database Setup
    a. Mongo Atlas

        MongoDB Altas is built for agile teams who'd rather spend time building apps than managing databases. Available on AWS, Google Cloud, and Azure.



MongoDB Atlas

Move faster with a true multi-cloud database service for MongoDB built for agile teams who'd rather spend time building apps than managing databases.

Available on AWS, Google Cloud, and Azure.

Already have an account? Log in here →

Cloud Provider & Region
Choose your preferred cloud provider and the region nearest to clients

AWS, N. Virginia (us-east-1)

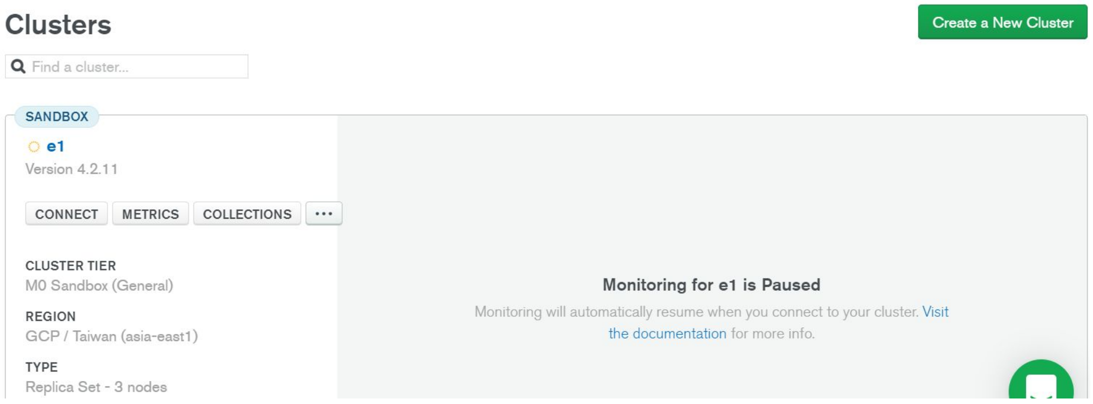
Select a cloud provider to see its region availability.

Configure a **free tier cluster** by first selecting a region labeled with FREE TIER AVAILABLE then choose the M0 option in the Cluster Tier below.

★ recommended region

NORTH AMERICA	EUROPE	ASIA
N. Virginia (us-east-1) ★ FREE TIER AVAILABLE	Ireland (eu-west-1) ★	Tokyo (ap-northeast-1)
Ohio (us-west-1) ★	London (eu-west-2)	Seoul (ap-northeast-2)
N. California (us-west-1)	Frankfurt (eu-central-1) ★ FREE TIER AVAILABLE	Singapore (ap-southeast-1)
Oregon (us-west-2) ★	SOUTH AMERICA	Mumbai (ap-south-1)

b. Create a cluster



CLUSTER TIER
M0 Sandbox (General)

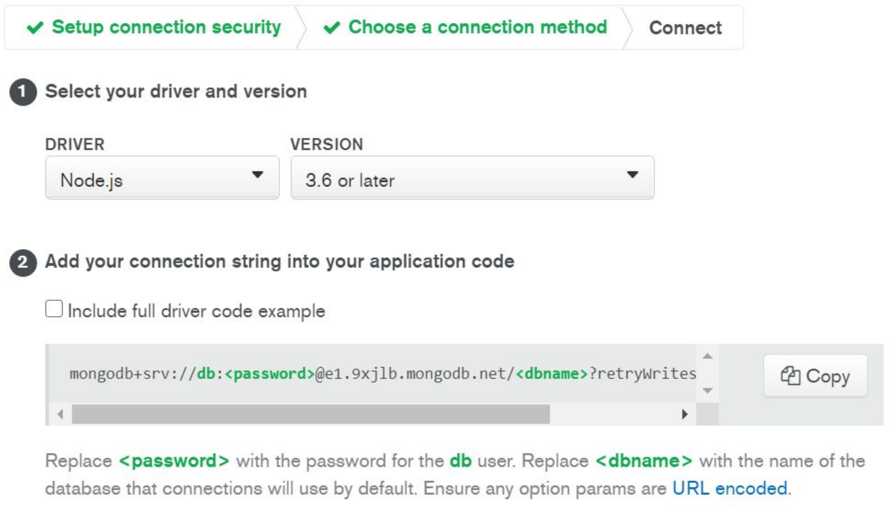
REGION
GCP / Taiwan (asia-east1)

TYPE
Replica Set - 3 nodes

Monitoring for e1 is Paused

Monitoring will automatically resume when you connect to your cluster. Visit the documentation for more info.

c. Access the database via a given url



1 Select your driver and version

DRIVER: Node.js
VERSION: 3.6 or later

2 Add your connection string into your application code

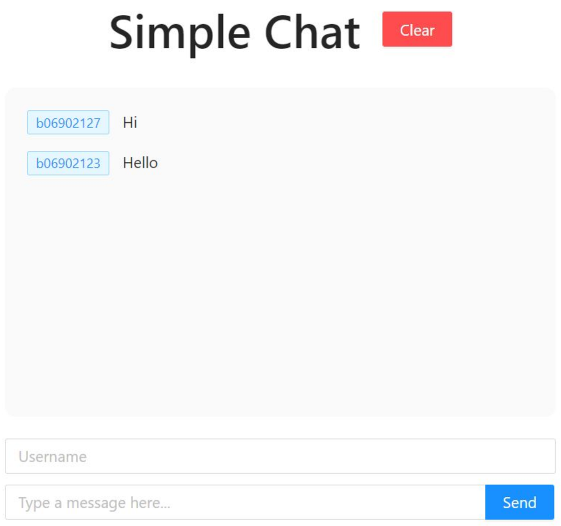
☐ Include full driver code example

```
mongodb+srv://db:<password>@e1.9xjlb.mongodb.net/<dbname>?retryWrites
```

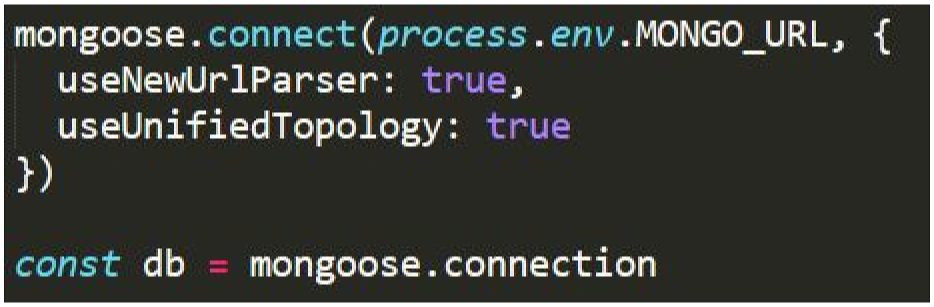
Replace <password> with the password for the db user. Replace <dbname> with the name of the database that connections will use by default. Ensure any option params are URL encoded.

3. Application: Simple Chat Service

a. outlook

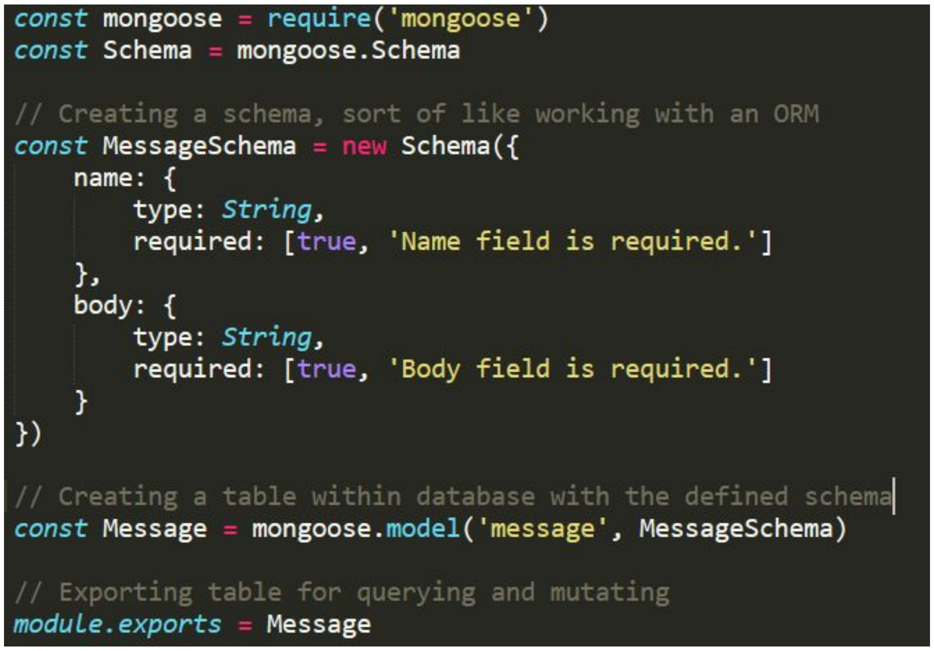


b. Function
    i. Enter the username and message to broadcast the message to the channel.
    ii. Clear all the messages
c. Database
    i. Connection



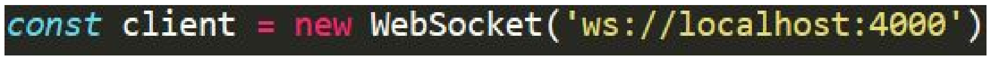
```
mongoose.connect(process.env.MONGO_URL, {
  useNewUrlParser: true,
  useUnifiedTopology: true
})

const db = mongoose.connection
```

    ii. Schema

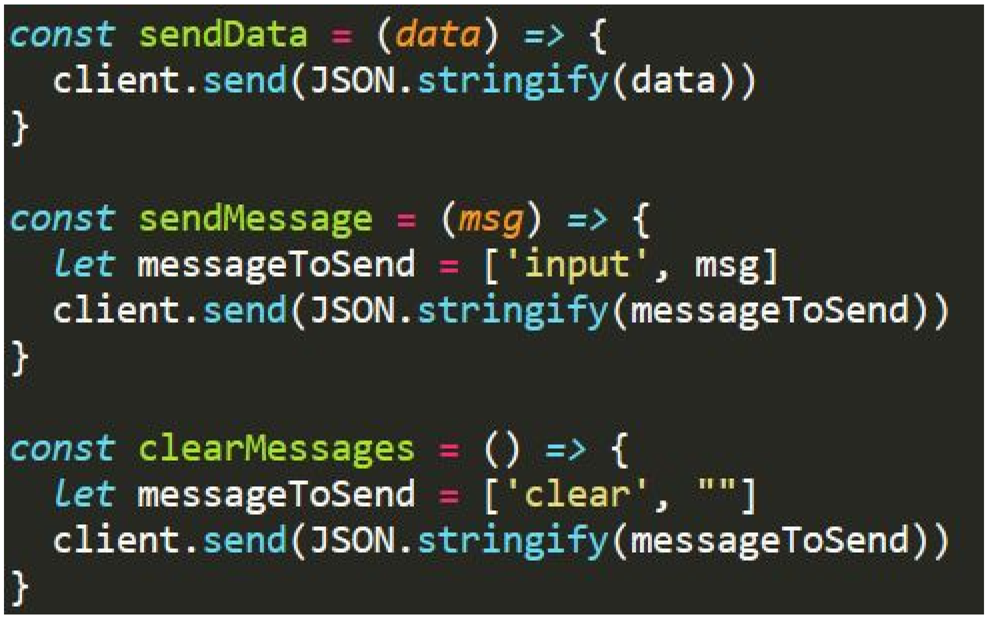


```
const mongoose = require('mongoose')
const Schema = mongoose.Schema

// Creating a schema, sort of like working with an ORM
const MessageSchema = new Schema({
    name: {
        type: String,
        required: [true, 'Name field is required.']
    },
    body: {
        type: String,
        required: [true, 'Body field is required.']
    }
})

// Creating a table within database with the defined schema
const Message = mongoose.model('message', MessageSchema)

// Exporting table for querying and mutating
module.exports = Message
```

4. Procedure
    a. Send a message
        i. Set the message
        ii. Send the message via WebSocket to the server



```
const client = new WebSocket('ws://localhost:4000')
```



```
const sendData = (data) => {
  client.send(JSON.stringify(data))
}

const sendMessage = (msg) => {
  let messageToSend = ['input', msg]
  client.send(JSON.stringify(messageToSend))
}

const clearMessages = () => {
  let messageToSend = ['clear', ""]
  client.send(JSON.stringify(messageToSend))
}
```

iii. The server is triggered and preserves the message in the database.

b. Clear messages

i. Clear the message on the website.

ii. Send the clear request via WebSocket to the server.

iii. Clear all the messages in the database.

5. Github repo

   https://github.com/RenKaiZheng/CloudCompute2020Fall