



The background of the slide is a dark blue gradient. On the right side, there is a large, faint, light blue circular shape that is partially cut off by the edge of the frame. The text 'MongoDB' is positioned on the left side of the slide.

MongoDB

← → ↻ mongodb.com/2 ☆ 📁 N ⋮

 Register for MongoDB.live today! →


 **mongoDB.** ✓

MongoDB

The application data platform

Accelerate development, address diverse data sets, and adapt quickly to change with a proven application data platform built around the database most wanted by developers 4 years running.

[Start free](#)


MONGODB ATLAS

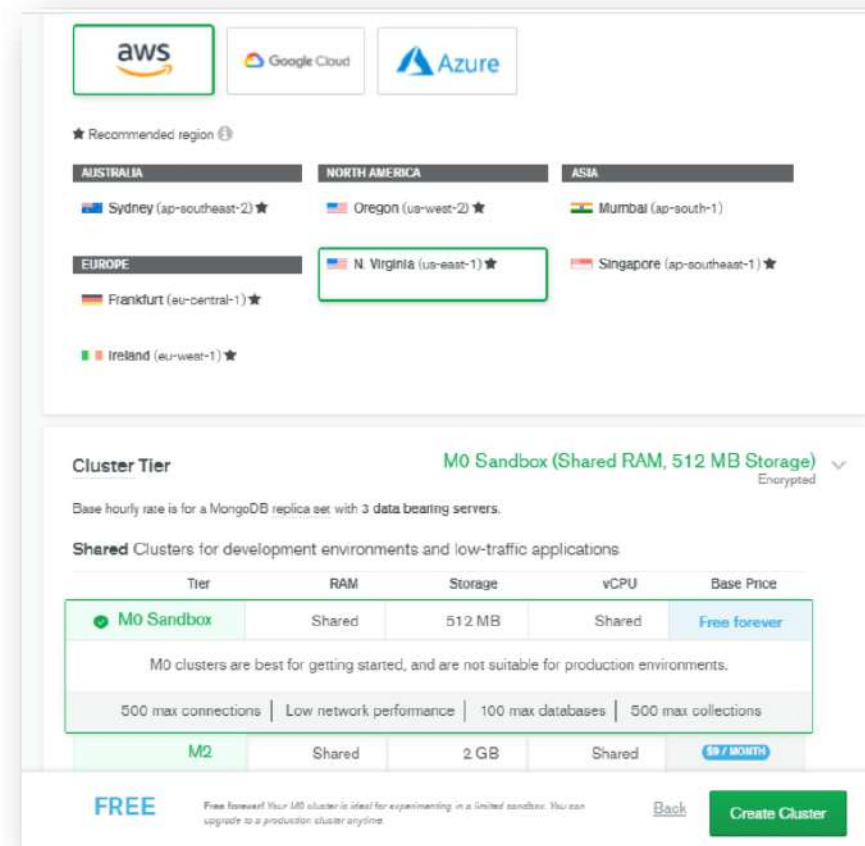
Choose a path. Adjust anytime.

Available as a fully managed service across 60+ regions on AWS, Azure, and Google Cloud

Dedicated Multi-Cloud & Multi-Region Clusters	Dedicated Clusters	Shared Clusters
For teams developing world-class applications that require multi-region resiliency or ultra-low latency.	For teams building applications that need advanced development and production-ready environments.	For teams learning MongoDB or developing small applications.
<ul style="list-style-type: none">✓ Includes all features from Shared and Dedicated Clusters✓ Replicate data across clouds and regions✓ Globally distributed read and write operations✓ Control data residency at the document level	<ul style="list-style-type: none">✓ Includes all features from Shared Clusters✓ Auto-scaling✓ Network isolation✓ Realtime performance metrics	<ul style="list-style-type: none">✓ Highly available auto-healing cluster✓ End-to-end encryption✓ Role-based access control
Create a cluster	Create a cluster	Create a cluster
Starting at \$0.13/hr* <small>*estimated cost \$99.85/month</small>	Starting at \$0.08/hr* <small>*estimated cost \$69.84/month</small>	Starting at FREE

MongoDB Atlas Setup

1. goto: cloud.mongodb.com
2. Register
3. Create a project : VacQ
4. build a cluster
5. free tier
6. Settings
 - a. aws >> Region: North America
 - b. Cluster Tier: free plan
 - c. cluster name: **VacQCluster**
 - d. Click 'Create a cluster'
7. Database Access >> Add New User
8. Network Access >> ADD CURRENT IP ADDRESS
9. Clusters >> Collections>>No data
10. Clusters >> Connect >> Connect with Compass >> copy the Connection String



Database Deployments | Atlas: M x +

cloud.mongodb.com/v2/610b4d7d6553a53beff0ca8d#clusters

Nuengwong's Org - ... Access Manager

SWArchREST Atlas

DEPLOYMENT

Databases

Triggers

Data Lake

SECURITY

Database Access

Network Access

Advanced

NUENGWONG'S ORG - 2021-05-01 > SWARCHREST

Database Deployment

Find a database deployment...

restaurant Connect

R 0
W 0
Last 10 minutes
100.0/s

VERSION REGION
4.4.6 AWS / N. Virginia (us-east-1)

System Status: All Good

Connect to restaurant

✓ Setup connection security Choose a connection method Connect

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

Go Back Close

+ Create

FREE SHARED

Enhance Your Experience

For production throughput and richer metrics, upgrade to a dedicated cluster now!

Upgrade

Database Deployments | Atlas: M x +

cloud.mongodb.com/v2/610b4d7d6553a53beff0ca8d#clusters/connect?clusterId=restaurant

COVID-19: AI Deco... Chula email Zoom สมัยก่อนถึง งาน Thai... ONE-STOP COMM... CU CAS : main/index 75 Beautiful Small... บัญชีมาร์กอีเมล แจ้งรอดอ่าน

Nuengwong's Org - ... Access Manager ...

SWArchREST Atlas

DEPLOYMENT

Databases

Triggers

Data Lake

SECURITY

Database Access

Network Access

Advanced

Find a database deployment...

restaurant Connect

● R 0
● W 0
Last 10 minutes
100.0/s

VERSION REGION
4.4.6 AWS / N. Virginia (us-east-1)

System Status: All Good

Connect to restaurant

✓ Setup connection security ✓ Choose a connection method Connect

1 Select your driver and version

DRIVER VERSION
Node.js 3.7 or later

2 Add your connection string into your application code

☐ Include full driver code example

`mongodb+srv://nuengwong:<password>@restaurant.xgrqz.mongodb.net/myFirstDatabase?
retryWrites=true&w=majority`

Replace <password> with the password for the **nuengwong** user. Replace **myFirstDatabase** with the name of the database that connections will use by default. Ensure any option params are [URL encoded](#).

Having trouble connecting? [View our troubleshooting documentation](#)

Go Back Close

copy

FREE SHARED

Enhance Your Experience

For production throughput and richer metrics, upgrade to a dedicated cluster now!

Upgrade

4.19 Connecting to Database With Mongoose

1. `> npm i mongoose`
2. `> npm run dev`
3. @config.env file: add a variable



```
MONGO_URI=mongodb+srv://...username:password@clustername.o0hi2.mongodb.net/myDatabaseName?retryWrites=true
&w=majority
```

//copy Connection String from MongoDB Atlas>> Connect Your Application

4. create a file: config/db.js
5. add code to config/db.js file
6. Add code to server.js to call database

3. Add a variable to config/config.env

```
PORT=5000
NODE_ENV=development

MONGO_URI=mongodb://localhost:27020/your-database?retryWrites=true
MONGO=
```

5. Add code to config/db.js

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

const connectDB = async () => {
  mongoose.set('strictQuery', true);
  const conn = await mongoose.connect(process.env.MONGO_URI);

  console.log(`MongoDB Connected: ${conn.connection.host}`);
}

module.exports = connectDB;
```

only to suppress deprecated warning! No special meaning.



5. Add code to config/db.js

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

const connectDB = async () => {
  mongoose.set('strictQuery', true);
  const conn = await mongoose.connect(process.env.MONGO_URI);

  console.log(`MongoDB Connected: ${conn.connection.host}`);
}

module.exports = connectDB;
```

only to suppress deprecated warning! No special meaning.



6. Add code to server.js

```
const express = require('express');
const dotenv = require('dotenv');
const connectDB = require('./config/db');

//Load env vars
dotenv.config({path: './config/config.env'});

//Connect to database
connectDB();

//Route files
const hospitals = require('./routes/hospitals')

const app=express();

//Body parser
app.use(express.json());

app.use('/api/v1/hospitals',hospitals);

const PORT=process.env.PORT || 5000;

const server = app.listen(PORT, console.log('Server running in ', process.env.NODE_ENV, ' mode on port ', PORT));

//Handle unhandled promise rejections
process.on('unhandledRejection',(err,promise)=>{
  console.log(`Error: ${err.message}`);
  //Close server & exit process
  server.close(()=>process.exit(1));
});
```

Creating our First Model

1. create a folder: `models/`
2. create a file: `models/Hospital.js`

models/Hospital.js

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

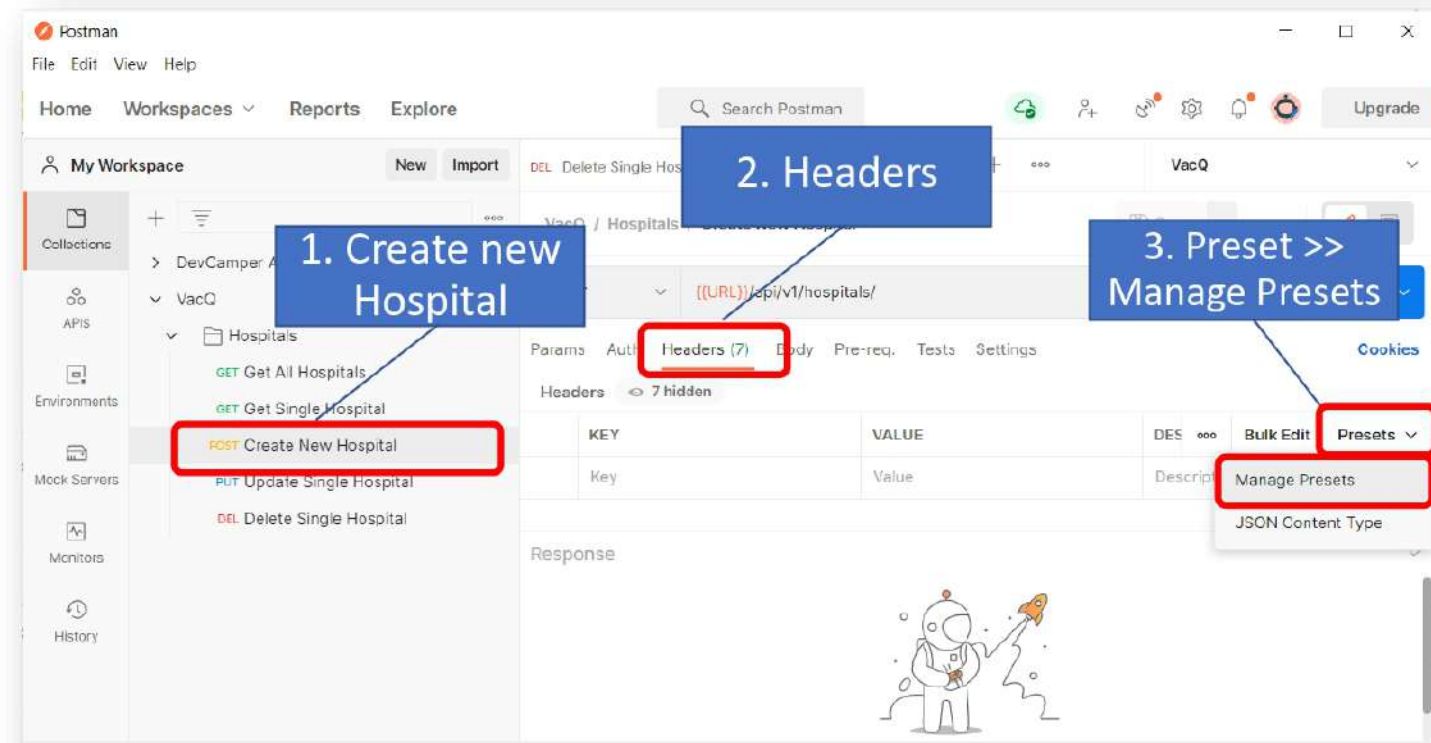
const HospitalSchema = new mongoose.Schema({
  name: {
    type: String,
    required: [true, 'Please add a name'],
    unique: true,
    trim: true,
    maxlength: [50, 'Name can not be more than 50 characters'],
  },
  address: {
    type: String,
    required: [true, 'Please add an address'],
  },
  district: {
    type: String,
    required: [true, 'Please add a district'],
  },
  province: {
    type: String,
    required: [true, 'Please add a province'],
  },
  postalcode: {
    type: String,
    required: [true, 'Please add a postalcode'],
    maxlength: [5, 'Postal Code can not be more than 5 digits'],
  },
  tel: {
    type: String,
  },
  region: {
    type: String,
    required: [true, 'Please add a region'],
  }
});

module.exports = mongoose.model('Hospital', HospitalSchema);
```

Create Hospital - POST

1. @controllers/hospitals.js
 2. require('../models/Hospital.js');
 3. @Postman
- >> Create new Hospital
>> Headers
>> Presets
>> Manage Presets
>> Add

1. @Postman >> Body



Create Hospital - POST (cont.)

The image shows a two-step process in a REST client interface. The first step, 'MANAGE HEADER PRESETS', allows creating a new preset named 'JSON Content Type' with the key 'Content-Type', value 'application/json', and description 'JSON Type'. The second step shows the main interface where this preset is selected from a dropdown menu to be added to the request headers.

4. Add Header Preset: JSON Content Type

5. Enter Key, Value, and Description

KEY	VALUE	DESCRIPTION		Bulk Edit
<input checked="" type="checkbox"/> Content-Type	application/json	JSON Type		
Key	Value	Description		

6. Add

7. Presets

8. JSON Content Type

9. Key, Value automatically added

KEY	VALUE	DE		Bulk Edit	Presets
<input checked="" type="checkbox"/> Content-Type	application/json	JSON T			Manage Presets JSON Content Type
Key	Value				

Postman

File Edit View Help

Home Workspaces Reports Explore

Search Postman

Invite Upgrade

My Workspace

New Import

DEL Delete Single Hos... POST Create New Hos...

VacQ / Hospitals / Create New Hospital

POST {{URL}}/api/v1/hospital

10. Body

11. raw

13. Send

12. Enter Data.

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1 {
2   "name": "BNH",
3   "address": "Changwattana Pakkret",
4   "district": "Pakkret",
5   "province": "Nonthaburi",
6   "postalcode": "10110",
7   "tel": "02-8369999",
8   "region": "กรุงเทพมหานคร (Bangkok)"
9 }
10
```

Response

Find and Replace Console

Bootcamp Runner Trash

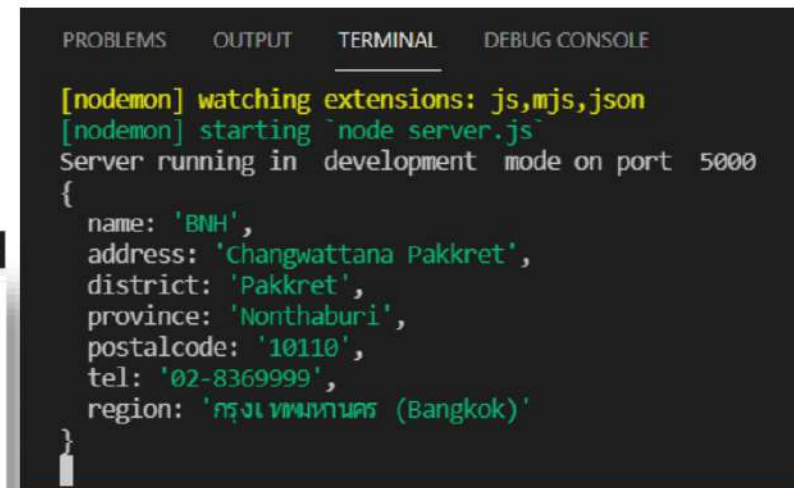
Create Hospital - POST (cont.)

@server.js: add Body parser under `const app = express();`

```
//Body parser
app.use(express.json());
```

@controller/hospitals.js: add `console.log(req.body)` at `exports.createHospital()`

```
exports.createHospital=(req,res,next)=>{
  console.log(req.body);
  res.status(200).json({success:true, msg:'Create new hospital'});
};
```

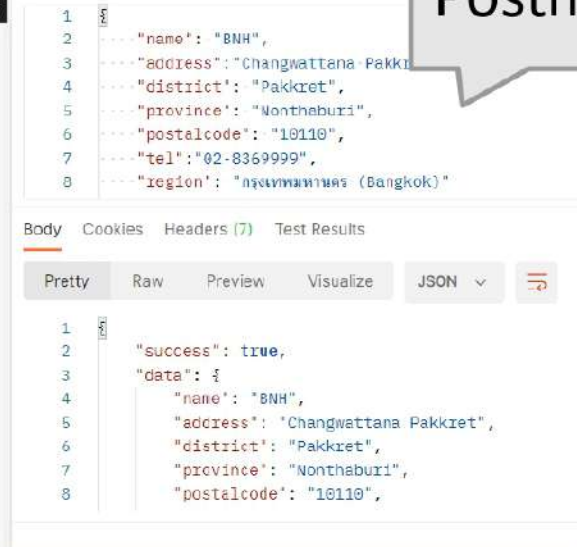


Create Hospital - POST (cont.)

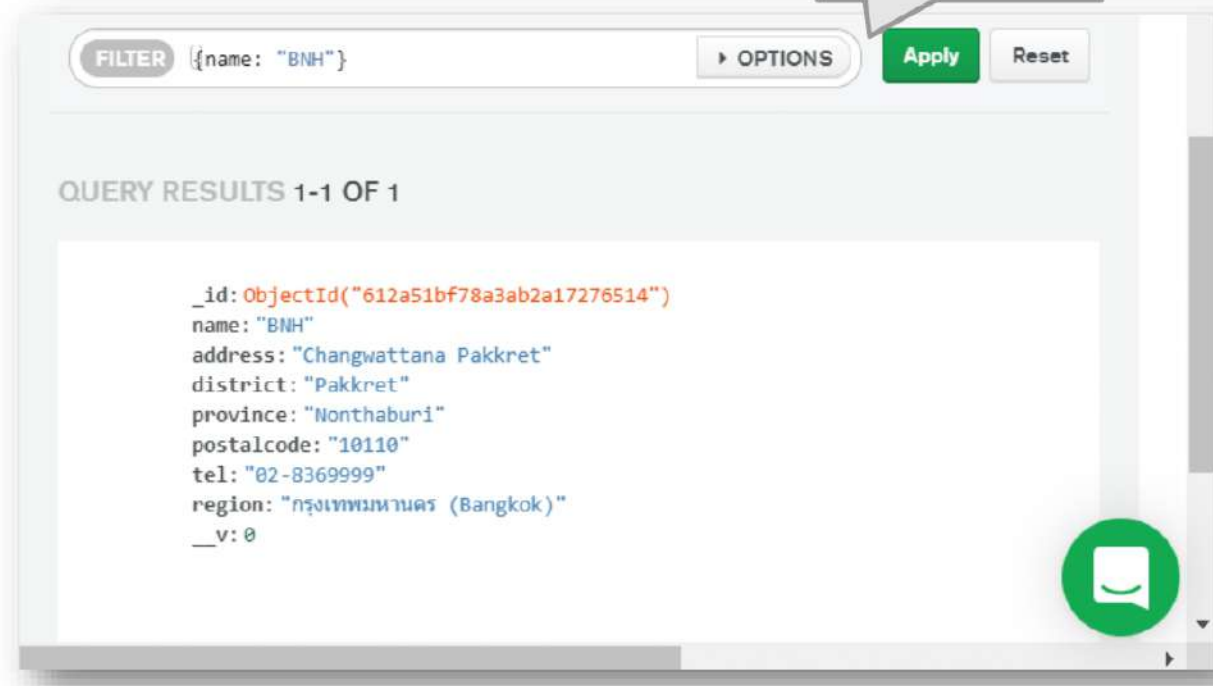
@controller/hospitals.js: change exports.createHospital() to this:

```
exports.createHospital= async (req,res,next)=>{  
  const hospital =await Hospital.create(req.body);  
  res.status(201).json({  
    success: true,  
    data:hospital  
  });  
};
```

Postman



MongoDB



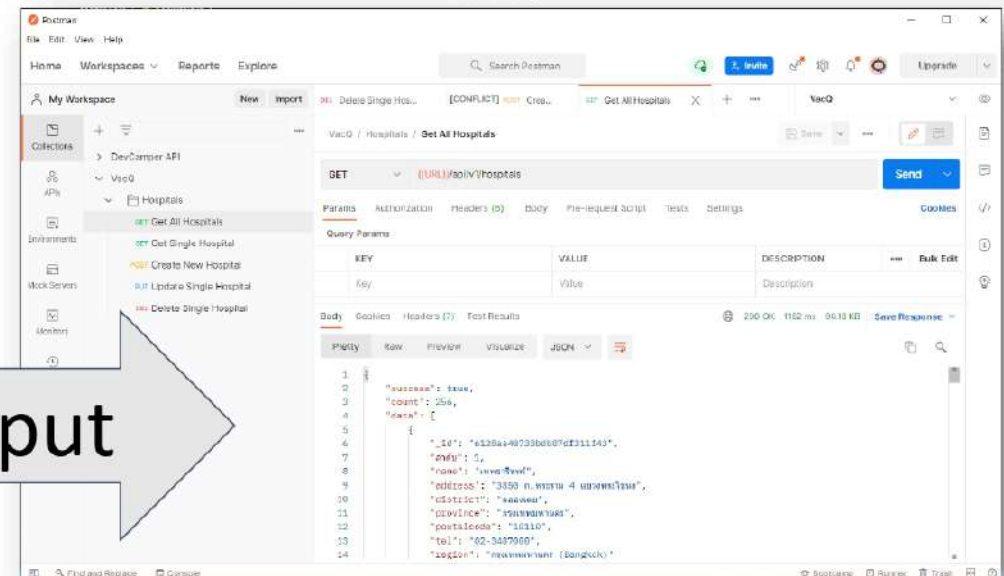
Fetching Hospitals - GET ALL Hospitals

1. @controllers/hospitals.js Add code at exports.getHospitals()

```
exports.getHospitals= async (req,res,next)=>{  
  try{  
    const hospitals = await Hospital.find();  
  
    res.status(200).json({success:true,count:hospitals.length, data:hospitals});  
  } catch(err){  
    res.status(400).json({success:false});  
  }  
};
```

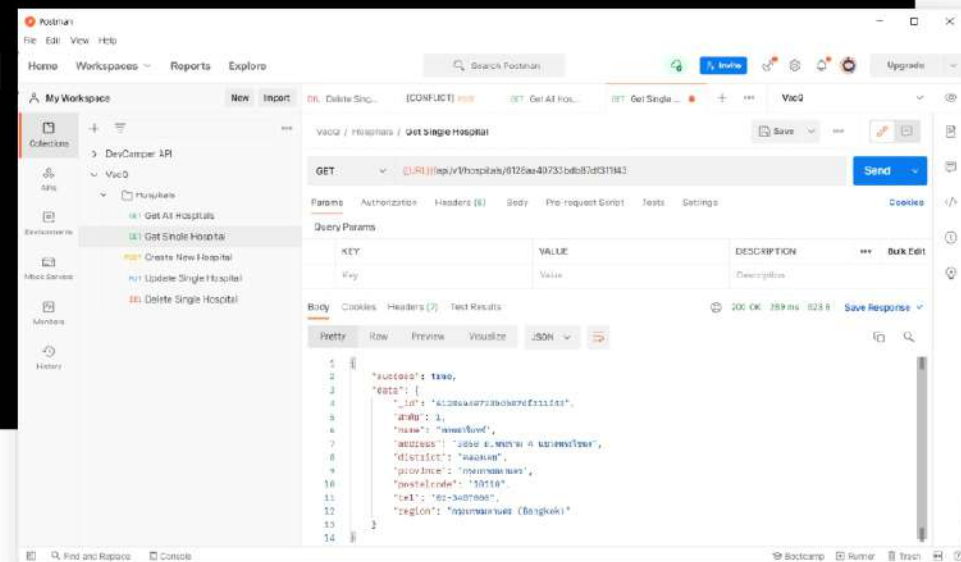
1. @Postman: Get All Hospitals

output



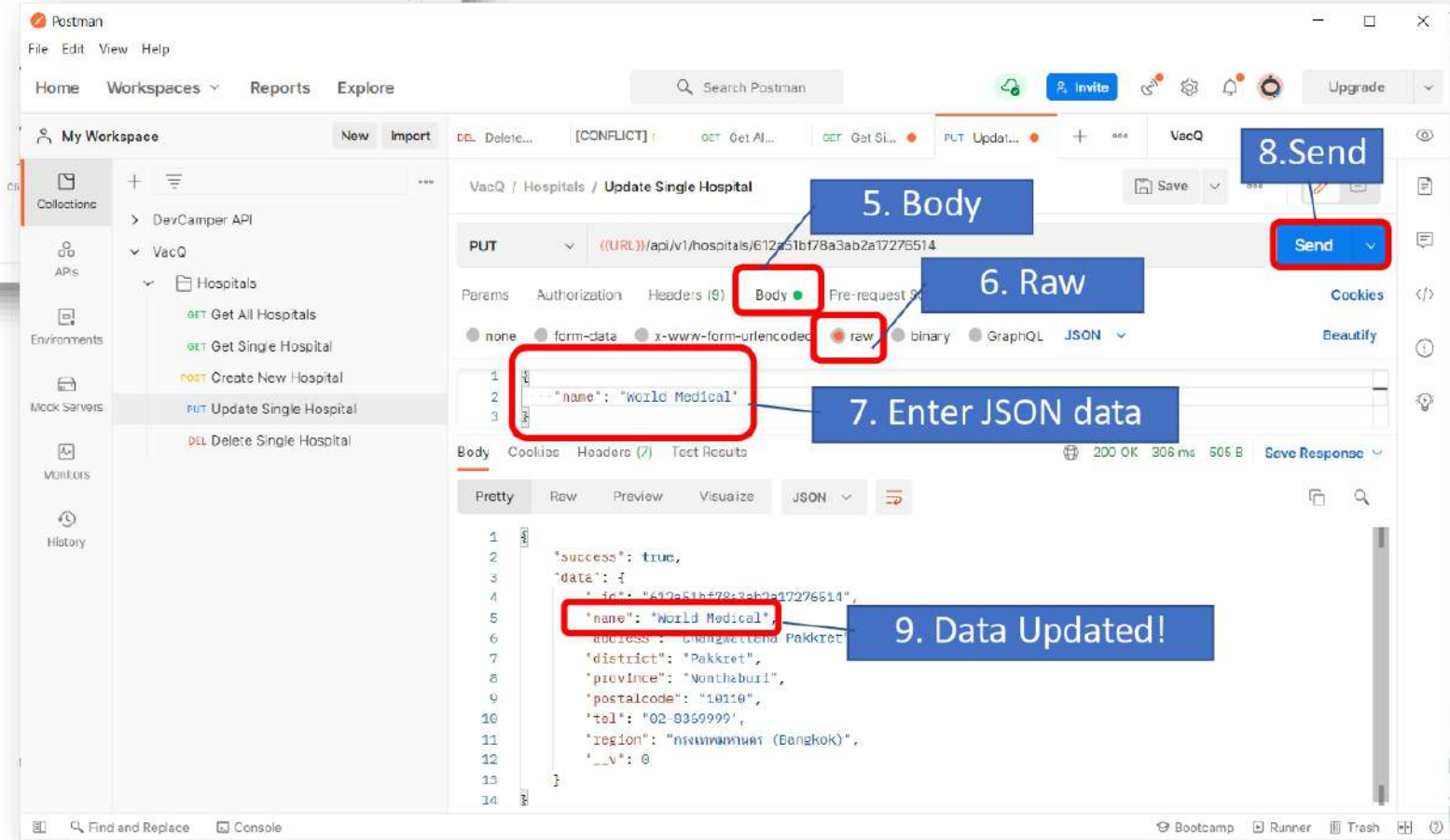
Fetching One Hospital - GET ONE Hospital

```
exports.getHospital= async (req,res,next)=>{  
  try{  
    const hospital = await Hospital.findById(req.params.id);  
  
    if(!hospital){  
      return res.status(400).json({success:false});  
    }  
  
    res.status(200).json({success:true,data:hospital});  
  } catch(err){  
    res.status(400).json({success:false});  
  }  
};
```



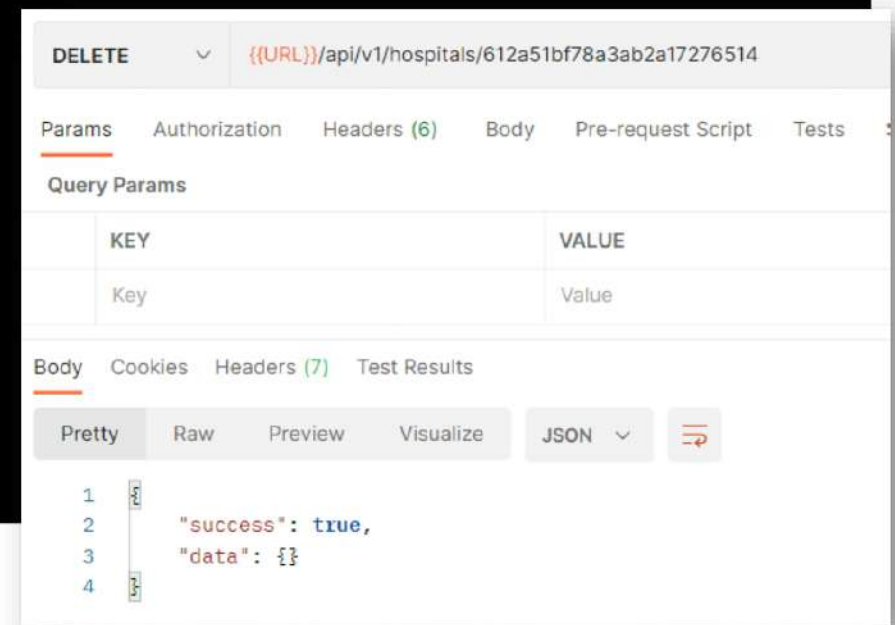
Updating Hospital - PUT

```
exports.updateHospital= async (req,res,next)=>{  
  try{  
    const hospital = await Hospital.findByIdAndUpdate(req.params.id, req.body, {  
      new: true,  
      runValidators:true  
    });  
  
    if(!hospital){  
      return res.status(400).json({success:false});  
    }  
  
    res.status(200).json({success:true, data: hospital});  
  }catch(err){  
    res.status(400).json({success:false});  
  }  
};
```

Deleting Hospital - DELETE

```
exports.deleteHospital= async (req,res,next)=>{  
  try{  
    const hospital = await Hospital.findByIdAndDelete(req.params.id);  
  
    if(!hospital){  
      return res.status(400).json({success:false});  
    }  
  
    res.status(200).json({success:true, data: {}});  
  }catch(err){  
    res.status(400).json({success:false});  
  }  
};
```



Conclusions

Basic REST APIs

- GET
- POST
- PUT
- DELETE

Develop APIs with Node.JS and MongoDB Atlas (on cloud)

- Controller Files
- Model Files

Test APIs with Postman