Zappos Data Science – Software Engineering Intern Challenge

Github: https://github.com/Anuj2512/Restaurant-REST-API

Anuj Chaudhari
chaudhari.anuj93@gmail.com

Restaurant-REST-API

Technology	Version
npm	5.5.5
Node.js	6.11.5
MongoDB	3.4.9
Redis	4.0.8

How to Run the Project

1. Install NodeJs, MongoDB, Redis locally. [Tested with MacOSX 10.13.3]
2. Clone or download the repository.
3. Make sure mongodb server and redis database server are running locally.
4. Go to api folder. (package.json file is located here)
5. run following command, to install node modules locally

```
npm install
```

6. run following command to start node server which exposes restaurant REST api

```
npm start
```

7. run following command to run unit tests

```
npm test
```

Database Schema

```
Restaurants
{
    _id         : <restaurant_id>,
    name        : <restaurant_name>,
    description : <description>,
    cuisines    : [<cuisines-1>, <cuisines-2>, <cuisines-3>],
    contact_no  : <contact_no>,
    address     : <address>,
    menus       : [
                    {
                        menu_id     : <menu_id>,
                        menu_type   : <breakfast, lunch, dinner, drinks>,
                        menu_items  : [
                                        {
                                            item_name   : <item_name>,
                                            description : <item_description>,
                                            price       : <price>,
                                            image_url   : <image_url>
                                        }
                                    ]
                    }

                ]
}
```

REST API Endpoints

BaseURL : localhost:3000/zappos/api/v1

GET	/restaurants
POST	/restaurants
GET	/restaurants/:RestaurantID
DELETE	/restaurants/:RestaurantID
POST	/restaurants/:RestaurantID/menus
GET	/restaurants/:RestaurantID/menus
GET	/restaurants/:RestaurantID/menus/:MenuID
DELETE	/restaurants/:RestaurantID/menus/:MenuID
POST	/restaurants/:RestaurantID/menus/:MenuID/items
GET	/restaurants/:RestaurantID/menus/:MenuID/items
GET	/restaurants/:RestaurantID/menus/:MenuID/items/:ItemID
DELETE	/restaurants/:RestaurantID/menus/:MenuID/items/:ItemID

REST API Requests

- Here is the sharable link for POSTMAN to use all of this request. Just Import Request from the given link to POSTMAN.

  https://www.getpostman.com/collections/0833962d50247e7f69a0

1. Adding new restaurant

POST http://localhost:3000/zappos/api/v1/restaurants

```
POST http://localhost:3000/zappos/api/v1/restaurants


201 Created
{
    "name" : "Starbucks",
    "description": "Starbucks is considered the main representative of second wave coffee, initially distinguishing itself from other
coffee-serving venues in the US by taste, quality, and customer experience while popularizing darkly roasted coffee",
    "contact_no": "(408) 275-9368",
    "area": "San Jose",
    "cuisines" : ["cafe"],
    "menu": [{
                "menu_type" :   "Coffee",
                "menu_items":   [{
                                    "item_name": "Coffee Latte",
                                    "Description": "coffee with the best flovour of latte",
                                    "price" : 3
                                },
                                {
                                    "item_name": "Caffe Mocha",
                                    "Description": "coffee with the best flovour of mocha",
                                    "price" : 4
                                }]
            },
            {
                "menu_type" :   "Tea",
                "menu_items":   [{
                                    "item_name": "Classic Chai Tea Latte",
                                    "Description": "Classic Chai Tea with the best flovour of latte",
                                    "price" : 2
                                }]
            }
        ]
```

```
  }
```

},

2. Get restaurant details

GET http://localhost:3000/zappos/api/v1/restaurants/

```
  GET http://localhost:3000/zappos/api/v1/restaurants/starbucks-san-jose

  200 OK
  {
      "description": "Starbucks is considered the main representative of second wave coffee, initially distinguishing itself from other
coffee-serving venues in the US by taste, quality, and customer experience while popularizing darkly roasted coffee",
      "contact_no": "(408) 275-9368",
      "area": "San Jose",
      "cuisines": [
          "cafe"
      ],
      "menu": [
          {
              "menu_items": [
                  {
                      "decription": "",
                      "item_name": "Coffee Latte",
                      "price": 3
                  },
                  {
                      "decription": "",
                      "item_name": "Caffe Mocha",
                      "price": 4
                  }
              ],
              "menu_type": "Coffee",
              "menu_id": "coffee"
          } ,
        {
          "menu_items": [
            {
              "decription": "",
              "item_name": "Classic Chai Tea Latte",
              "price": 2
            }
          ],
          "menu_type": "Tea",
          "menu_id": "tea"
        }

      ],
      "locality_count": 0,
      "timestamp": "Tue Feb 06 2018 01:45:33 GMT-0800 (PST)",
      "id": "starbucks-san-jose",
      "name": "Starbucks",
    }
```

3. **`Delete restaurant`**

   `DELETE http://localhost:3000/zappos/api/v1/restaurants/<restaurant-id>`

```
DELETE http://localhost:3000/zappos/api/v1/restaurants/starbucks-san-jose

204 No Content
```

4. **`Add Menu`**

   `POST http://localhost:3000/zappos/api/v1/restaurants/<restaurant-id>/menus`

```
POST http://localhost:3000/zappos/api/v1/restaurants/starbucks-san-jose/menus

201 Created
{
  "menu_type" :   "Hot Drinks",
  "menu_items":   [{
                    "item_name": "Coffee Latte",
                    "Description": "coffee with the best flovour of latte",
                    "price" : 3
                },
                {
                    "item_name": "Caffe Mocha",
                    "Description": "coffee with the best flovour of mocha",
                    "price" : 4
                }]
  }
```

5. Get Menu

`GET http://localhost:3000/zappos/api/v1/restaurants/<restaurant-id>/menus/<menu-type>`

```
GET http://localhost:3000/zappos/api/v1/restaurants/starbucks-san-jose/menus/coffee

200 OK
{
   "menu_items": [
      {
         "decription": "",
         "item_name": "Coffee Latte",
         "price": 3
      },
      {
         "decription": "",
         "item_name": "Caffe Mocha",
         "price": 4
      }
   ],
   "menu_type": "Coffee",
   "menu_id": "coffee"
}
```

6. Delete Menu

`DELETE http://localhost:3000/zappos/api/v1/restaurants/<restaurant-id>/menus/<menu-type>`

```
DELETE http://localhost:3000/zappos/api/v1/restaurants/starbucks-san-jose/menus/coffee


204 No Content
```

7. Add MenuItem

`POST http://localhost:3000/zappos/api/v1/restaurants/<restaurant-id>/menus/<menu-type>/items`

```
POST http://localhost:3000/zappos/api/v1/restaurants/starbucks-san-jose/menus/coffee/items


201 Created
{
   "item_name": "Hot Chocolate",
   "Description": "hot milk drink with chocolate cyrup",
   "price" : 3
}
```

8. Get MenuItem

`GET http://localhost:3000/zappos/api/v1/restaurants/<restaurant-id>/menus/<menu-type>/items/<item-id>`

```
GET http://localhost:3000/zappos/api/v1/restaurants/starbucks-san-jose/menus/coffee/items/ 5a7c16156d887752a071b3a2


 200 OK
{
    "decription": "",
    "_id": "5a7c16156d887752a071b3a2",
    "item_name": "Coffee Latte",
    "price": 3
  }
```

9. **`Delete MenuItem`**

   ```
   DELETE http://localhost:3000/zappos/api/v1/restaurants/<restaurant-id>/menus/<menu-type>/items/<item-id>
   ```

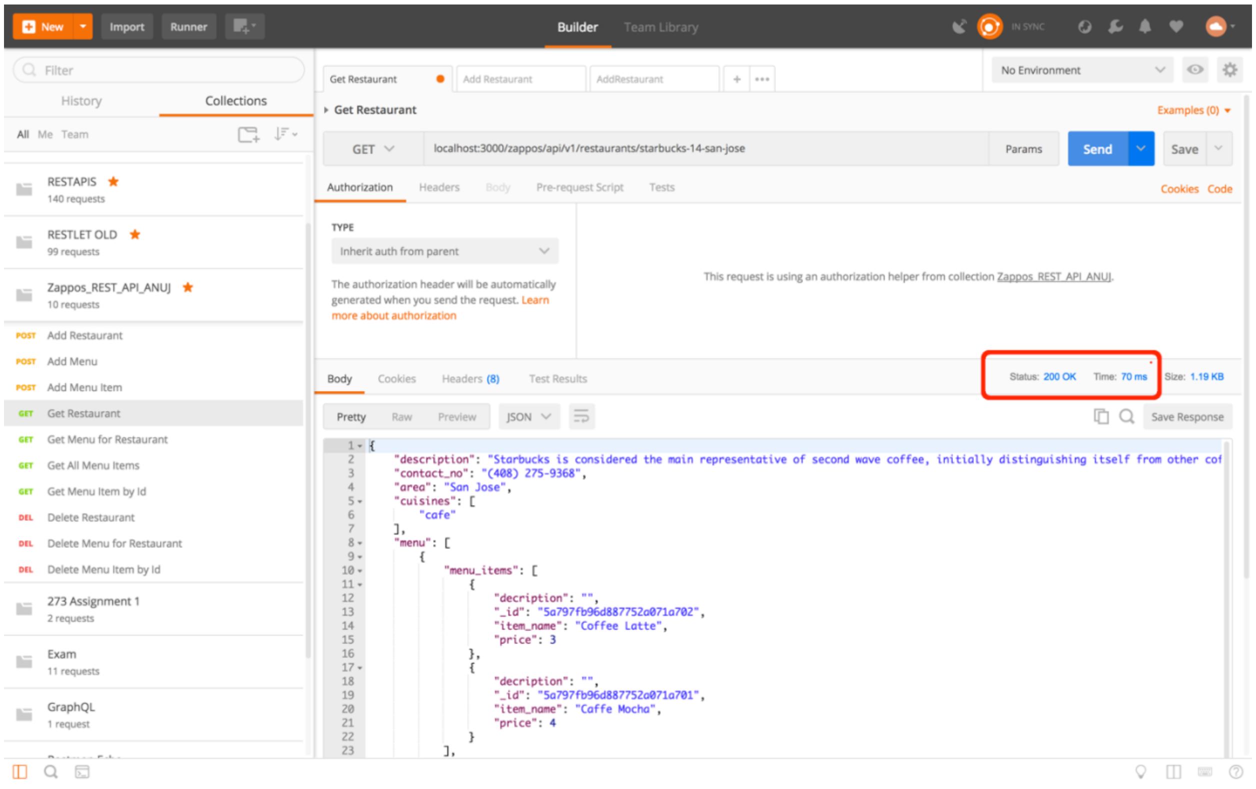
```
DELETE http://localhost:3000/zappos/api/v1/restaurants/starbucks-san-jose/menus/coffee/items/ 5a7c16156d887752a071b3a2
204 No Content
```

Redis for Caching

- To compare the request performace of redis and direct mongodb access, I have added approximatly 30,000 dummy data to the mongodb.
  Import dummy data to mongodb

  ```
  cd db_backup
  mongorestore -d anuj_restaurant_api anuj_restaurant_api
  ```

- So, that when randomly sending a GET request for the first time, I can measure response time and sending the same request again, it will hit the redis cache, I can see the response time improvement in the request.
- I have attached some screenshots displaying the response time for both the scenario.
  1. MongoDB Access : **70ms**
  2. Redis Cache : **5ms**



Get Restaurant

GET localhost:3000/zappos/api/v1/restaurants/starbucks-14-san-jose

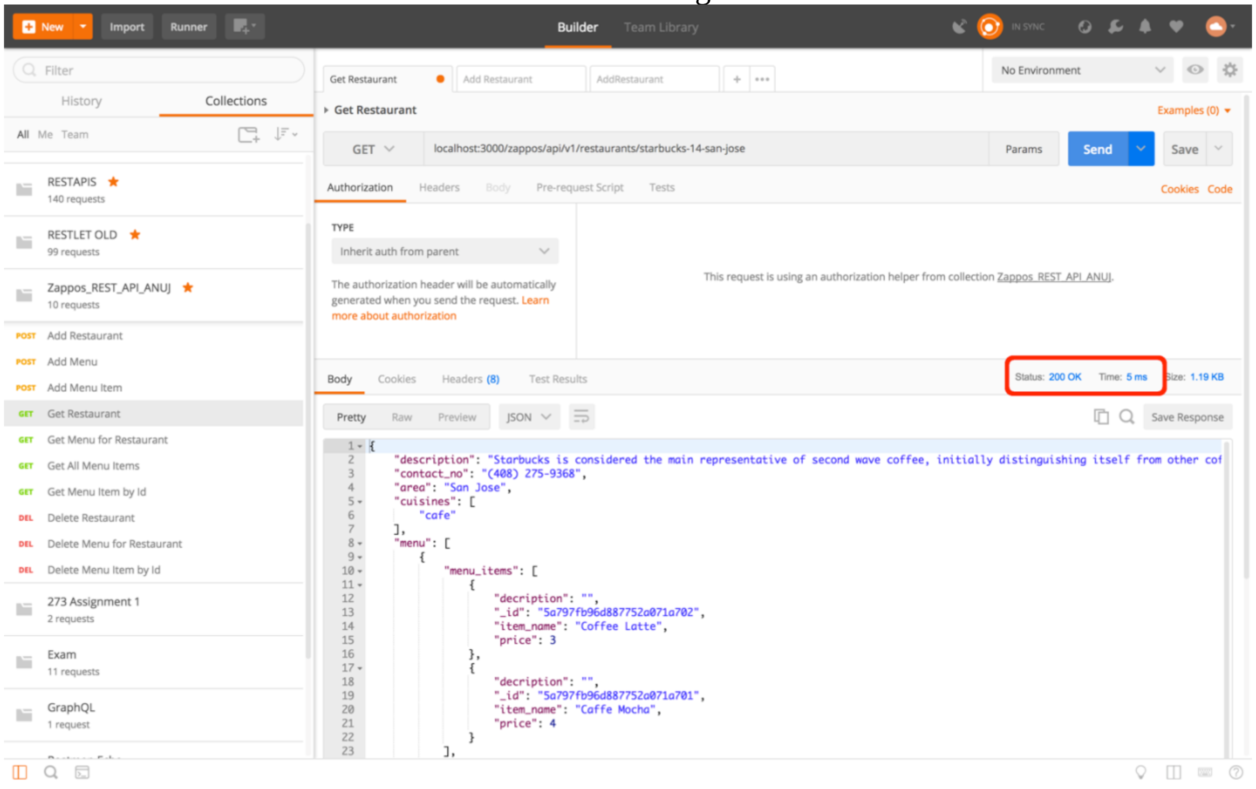
The authorization header will be automatically generated when you send the request. Learn more about authorization

This request is using an authorization helper from collection Zappos_REST_API_ANUJ.

Status: 200 OK Time: 70 ms Size: 1.19 KB

```
{
    "description": "Starbucks is considered the main representative of second wave coffee, initially distinguishing itself from other cof
    "contact_no": "(408) 275-9368",
    "area": "San Jose",
    "cuisines": [
        "cafe"
    ],
    "menu": [
        {
            "menu_items": [
                {
                    "decription": "",
                    "_id": "5a797fb96d887752a071a702",
                    "item_name": "Coffee Latte",
                    "price": 3
                },
                {
                    "decription": "",
                    "_id": "5a797fb96d887752a071a701",
                    "item_name": "Caffe Mocha",
                    "price": 4
                }
            ],
```

MongoDB hit



Get Restaurant

GET localhost:3000/zappos/api/v1/restaurants/starbucks-14-san-jose

The authorization header will be automatically generated when you send the request. Learn more about authorization

This request is using an authorization helper from collection Zappos_REST_API_ANUJ.

Status: 200 OK Time: 5 ms Size: 1.19 KB

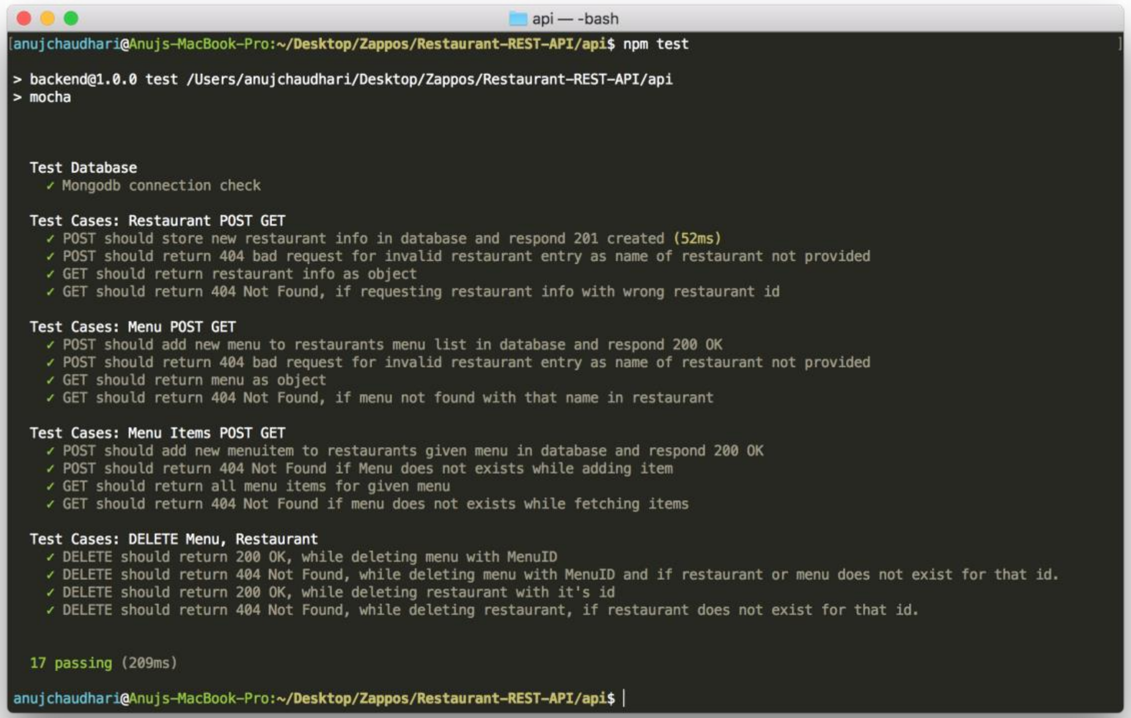
```
{
    "description": "Starbucks is considered the main representative of second wave coffee, initially distinguishing itself from other cof
    "contact_no": "(408) 275-9368",
    "area": "San Jose",
    "cuisines": [
        "cafe"
    ],
    "menu": [
        {
            "menu_items": [
                {
                    "decription": "",
                    "_id": "5a797fb96d887752a071a702",
                    "item_name": "Coffee Latte",
                    "price": 3
                },
                {
                    "decription": "",
                    "_id": "5a797fb96d887752a071a701",
                    "item_name": "Caffe Mocha",
                    "price": 4
                }
            ],
```

Redis Hit

Unit Tests

Run following command to run unit tests

```
npm test
```



```
api — -bash
anujchaudhari@Anujs-MacBook-Pro:~/Desktop/Zappos/Restaurant-REST-API/api$ npm test

> backend@1.0.0 test /Users/anujchaudhari/Desktop/Zappos/Restaurant-REST-API/api
> mocha



  Test Database
    ✓ Mongodb connection check

  Test Cases: Restaurant POST GET
    ✓ POST should store new restaurant info in database and respond 201 created (52ms)
    ✓ POST should return 404 bad request for invalid restaurant entry as name of restaurant not provided
    ✓ GET should return restaurant info as object
    ✓ GET should return 404 Not Found, if requesting restaurant info with wrong restaurant id

  Test Cases: Menu POST GET
    ✓ POST should add new menu to restaurants menu list in database and respond 200 OK
    ✓ POST should return 404 bad request for invalid restaurant entry as name of restaurant not provided
    ✓ GET should return menu as object
    ✓ GET should return 404 Not Found, if menu not found with that name in restaurant

  Test Cases: Menu Items POST GET
    ✓ POST should add new menuitem to restaurants given menu in database and respond 200 OK
    ✓ POST should return 404 Not Found if Menu does not exists while adding item
    ✓ GET should return all menu items for given menu
    ✓ GET should return 404 Not Found if menu does not exists while fetching items

  Test Cases: DELETE Menu, Restaurant
    ✓ DELETE should return 200 OK, while deleting menu with MenuID
    ✓ DELETE should return 404 Not Found, while deleting menu with MenuID and if restaurant or menu does not exist for that id.
    ✓ DELETE should return 200 OK, while deleting restaurant with it's id
    ✓ DELETE should return 404 Not Found, while deleting restaurant, if restaurant does not exist for that id.


  17 passing (209ms)

anujchaudhari@Anujs-MacBook-Pro:~/Desktop/Zappos/Restaurant-REST-API/api$ |
```

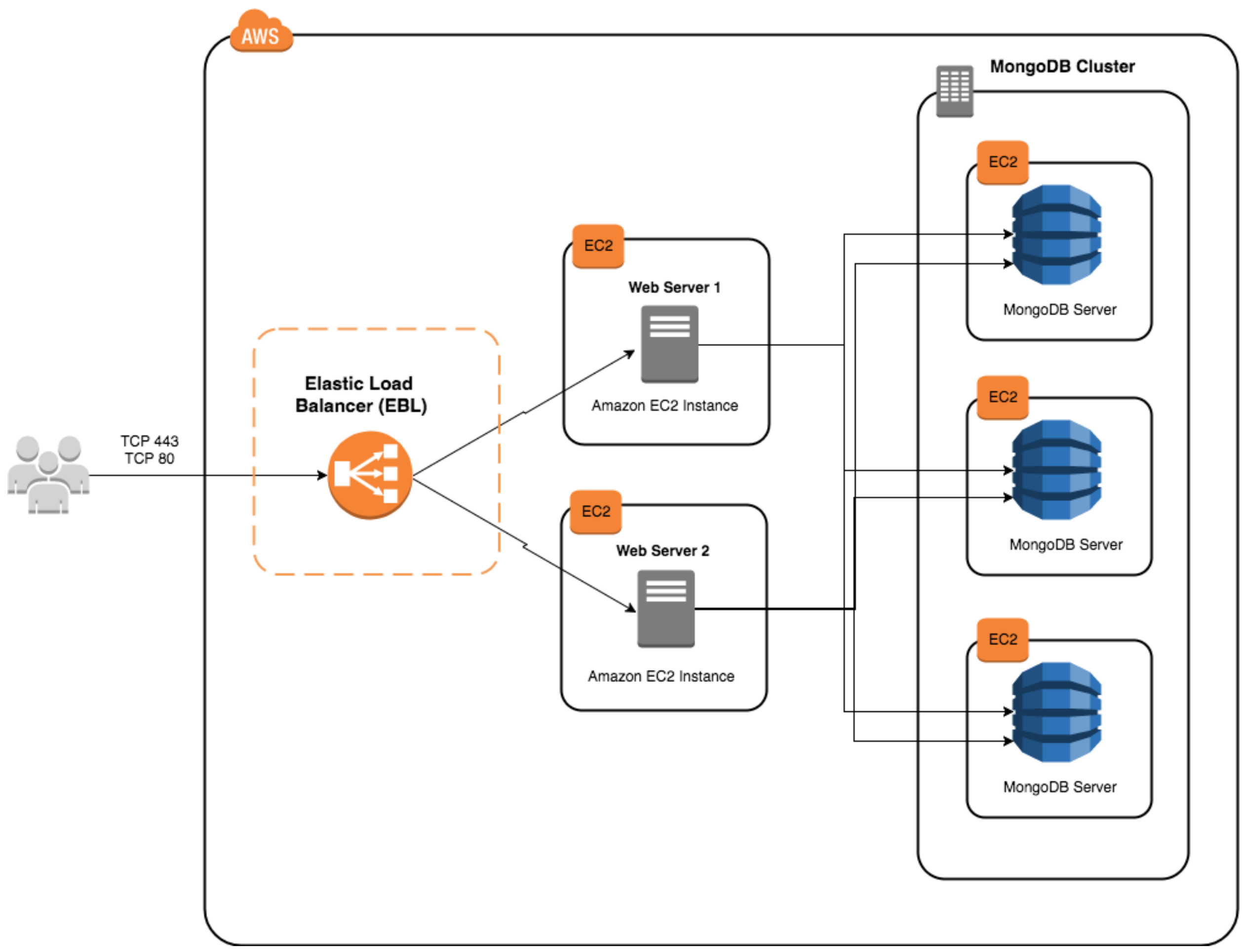
Handling millions of Requests at once

For handling these many request at once we can use multiple servers running the same service and we can put the mongodb database on different machine and not where the node server is currently running.

For this we can use mongodb replicaset as well as load balancer on top of the node server.

If we use AWS services for handling scaling for this type of problem, Here is the server architecture diagram. I have hosted this service on my AWS EC2 instance.

Here is the link for the API server. http://ec2-54-215-224-120.us-west-1.compute.amazonaws.com:3000


AWS
MongoDB Cluster
EC2
MongoDB Server
EC2
Web Server 1
Amazon EC2 Instance
Elastic Load Balancer (EBL)
TCP 443
TCP 80
EC2
MongoDB Server
EC2
Web Server 2
Amazon EC2 Instance
EC2
MongoDB Server