SUDOKU:

ANGULAR-NODE-MOCHA-KARMA-NGINX-DOCKER

TABLE OF CONTENTS

INTRODUCTION

This Sudoku web application is built using Angular for frontend User Interface and Express, Node for handling the API calls. We use Karma for testing the frontend component and Mocha for testing our API calls. We also make use of Nginx for Reverse Proxy and Docker for deploying images for containers.

SPECIFICATIONS

A web service that returns a randomized 9x9 grid of integers in the range [1-9] representing a Sudoku board:

- Calls should be accessible via '/sudoku/board' for loading the Sudoku board in the default mode. i.e. no cell in the board is selected. This may or may not take the size of the board as its parameter in the request header.
- Calls should also be accessible via '/sudoku/boardCell' for loading the Sudoku board in case a cell in the board is selected. This call takes the selected cell information as its header
- The resulting array of values must be valid abiding all the rules of the Sudoku game
- Content type of the request/response is supposed to be application/json
- Request must return the array of numbers generated for the Sudoku board in its response body.
- The entire reception of request to sending of the response must take less than 500ms
- Programming language : Typescript, Express, Node
- Build tools: npm
- Testing tools: Karma/Mocha

A Single Page Application which loads and displays a Sudoku board with the values sent by the API server:

- When the web page loads initially, a spinner should be displayed until the board values are fetched from the server.
- Upon reception of the values, the spinner disappears and the board is displayed.
- RELOAD button which when clicked refreshes the board values by fetching the fresh values from the server.
- RELOAD button is deactivated until the spinner exists.
- If a cell is selected on the board and the Reload button is clicked then fresh board is generated with new values without altering the value and the position of the selected cell.
- At a time only one cell must be selected.
- The resulting Sudoku board must be valid abiding all the rules of the Sudoku game
- Programming language: Typescript, Angular
- Build tools: npm

- Libraries & tools: HTML5, CSS, Angular, Karma and Bootstrap

Nginx is used to act as a reverse proxy to communicate with the server API calls so that the SPA (Single Page Application) and the web service appear as one website on the local machine.

Dockerize the web-service, frontend components and the entire application:

- Web service docker container must be run via the command
  docker run –it –p 8080:8080 sudoku-ws:level-4
- When the web service is running, **curl http://loaclhost:8080/sudoku/board** should return the Sudoku array of values
- Single Page Application docker container must be built with one command generating a docker image called **sudoku-spa** with **level-4** as its tag.
- Nginx should be used to act as a reverse proxy to talk between the SPA and web service.
- Technology: Docker

SOFTWARES REQUIRED

The software that are needed for the application to be edited or executed

- *OPERATING SYSTEM UTILIZED:* **UBUNTU-18.04**
- '*backend*' → Node JS
- *'API* calls' → Express*(not needed to install explicitly- gets auto installed while installing frontend)*.
- '*frontend*' → Angular (4/6), Bootstrap.
- '*testing*' → Karma, Mocha, Jasmine
- *'reverse proxy'* → Nginx.
- *'docker images'* → *Docker*

SOURCE CODE

The Source code for the application is in the '*sudokuApp*' folder

- '*sudokuServer*' → This folder has the source code for the API which generate the values for Sudoku Board.
- '*sudokuBoard*' → This folder has the source code for the Front User Interface which displays the values received from the API.
- *'dockerfile'* → Each folder have the Dockerfiles needed for building our docker image **(Please fetch 'master-docker' git Repo)**
- *'docker-compose.yml'* → For running our SPA and web service as a single container image **(Please fetch 'master-docker' git Repo)**

INSTALLATION NOTES

All the execution of this application will be done via command prompt terminal. You can skip the Installation step if you have already Nginx installed. But you are still required to follow the *underlined* steps in-order to setup the project properly and be able to run it locally.

Installing Nginx in Ubuntu:

- Go to Sudo nano /etc/apt/sources.list
- Add the following lines:

  deb http://nginx.org/packages/mainline/ubuntu/ bionic nginx
- Import the Repo Keys:

  sudo wget http://nginx.org/keys/nginx_signing.key

  sudo apt-key add nginx_signing.key
- Install Nginx

  sudo apt update

  sudo apt install nginx
- Enable and Run Nginx

  sudo systemctl start nginx

  sudo systemctl enable nginx
- Enable firewall

  sudo ufw enable
- Create a sudokuapp.conf file for the server in /etc/nginx/conf.d/ location

  sudo nano /etc/nginx/conf.d/sudokuapp.conf
- Copy the contents of ‘sudokuapp.conf’ file into the sudokuapp.conf file created at /etc/nginx/conf.d/ location
- Disable the default conf file

  sudo mv /etc/nginx/conf.d/default.conf etc/nginx/conf.d/default.conf.disabled
- Test and Reload nginx

  sudo nginx –t

  sudo nginx -s reload

‘Nginx Setup’

- *Check the applications that the firewall has given access to by default:*

  sudo ufw app list

- *Allow the ‘Nginx Full’(Both encrypted and unencrypted Traffics are allowed), ‘Nginx HTTP’(Unencrypted Traffics are allowed) or ‘Nginx HTTPS’(Only encrypted Traffics are allowed) as per your needs.*

  sudo ufw allow ‘Nginx Full’

  sudo ufw allow ‘Nginx HTTPS’

  sudo ufw allow ‘Nginx HTTPS’

- *Or you can enable the specifically required Ports as per your needs.*

  sudo ufw allow 8080/tcp

  sudo ufw allow 80/tcp

  sudo ufw allow 21/ftp

- *Check the status of the recently allowed traffic issue the following command. You should see a full list of all the traffic ports enabled*

  sudo ufw status

- *Active status is displayed if Nginx is running with the help of following command.*

  sudo systemctl status nginx

- *Copy your Git Cloned Code into /var/www/*

  sudo cp -r /home/YOUR_PC_NAME/YOUR_GIT_FOLDER/. /var/www/

- *Then point your domain name to our new server. So create a record in hosting providers’ DNS setting by pointing the domain name (sudoku.com) to our server IP address(0.0.0.0).*

  sudo nano /etc/hosts

- *Add the domain name - IP address mapping*

  0.0.0.0 suduko.com

- *Navigate into the /var/www/sudokuapp/sudokuBoard and give permissions:*

sudo chmod –R 777 *

- *Move the Static files to the Server*

  scp -r * 0.0.0.0:/var/www/sudokuapp/sudukoBoard

- *(OPTIONAL: If you set up the Nginx properly, you dont need this step) To configure nginx server to point to our new location navigate to nginx folder Create 'sites-available' and 'sites-enabled' folder if not created by default:*

  cd /etc/nginx && mkdir sites-available && mkdir sites-enabled

- *(OPTIONAL: If you set up the Nginx properly, you dont need this step) Create the configuration files to our files in 'sites-available' and 'sites-enabled' has the symbolic link which we will tell nginx to run. Open 'sudokuapp' and copy its contents into the file you open in terminal using the command below*

  sudo nano sites-available/sudokuapp

- *(OPTIONAL: If you set up the Nginx properly, you dont need this step) Create a system link the 'sudokuapp' file to the 'sites-enabled' folder:*

  sudo ln -s /etc/nginx/sites-available/sudokuapp /etc/nginx/sites-enabled/sudokuapp

- *(OPTIONAL: If you set up the Nginx properly, you dont need this step) Remove the 'default' file from 'sites-enabled' folder if created by default*

  sudo mv /etc/nginx/sites-enabled/default /home/YOUR_DESIRED_LOCATION/default

- *Restart nginx:*

  sudo systemctl restart nginx

'Backend'

- *Navigate into your /var/www/sudokuapp/sudokuServer/ and install then run the Server:*

  cd /var/www/sudokuapp/sudokuServer

  sudo npm install

  nodemon

  Or

  node index.js

- *Test the sudokuServer:* **npm test**

'Frontend'

- *Navigate into your /var/www/sudokuapp/sudokuBoard/ and install then run the Sudoku Board UI*

  cd /var/www/sudokuapp/sudokuBoard

  sudo npm install

  ng serve

  Or

  npm start

- *Test the sudokuBoard:*

  ng test

  Or

  npm test

'Docker Images'

- *Clone the repo and checkout to 'master-docker' branch and navigate into 'sudokuapp' folder:*

  git clone https://github.com/mattasridhar/sudokuapp.git

  git checkout master-docker

- *Have Docker installed up and running. Also from /etc/hosts file comment the Domina-IP mapping we did earlier.*

  # 0.0.0.0 sudoku.com

- *Build the docker image for the server, navigate to the 'sudokuServer' folder and issue the following command to create a docker image.*

  docker build -t sudoku-ws:level-4 .

- *Run the docker image we just built for the server, issue the following command to run the docker image. This command maps the server to run on 8080.*

  docker run -p 8080:8080 -d sudoku-ws:level-4

  Or

  docker run -it -p 8080:8080 sudoku-ws:level-4

- *Navigate to localhost:8080 in browser or issue the curl command(requires 'curl' installed).*

  curl http://localhost:8080/sudoku/board

- *Build the docker image for front-end, navigate to the 'sudokuBoard' folder and issue the following command to create a docker image.*

  docker build -t sudoku-spa:level-4 .

- *Run the docker image we just built for the front-end, issue the following command to run the docker image. This command maps the front-end to run on 4200.*

  docker run -d -v ${PWD}:/app -v /app/node_modules -p 2305:4200 --name sudoku-spa --rm sudoku-spa:level-4

- *To stop running the docker image we just built for the front-end, issue the following command.*

  docker stop sudoku-spa

- *To stop running the docker image we just built for the server, get the container id from the output of the first command and then issue the second command specifying the retrieved container id.*

  docker ps

  docker stop <container-ID>

- *Move a directory up (**cd ..**) i.e. inside the 'sudokuapp' folder and issue the following command to see the whole sudoku application be deployed using nginx. The website should be accessible on localhost:9093 and server on localhost:9094*

  docker-compose up --build

ABOUT THE APPLICATION

This application can be accessed by visiting the url '*http://sudoku.com/*' after you have installed and initiated the step given in the Nginx section of *'INSTALLATION NOTES'* section(except the Docker section if you are running it locally).

The application loads and sends a query to the server via the *'/sudoku/board'* service call and until it receives the data, a spinner is shown to the user. The reload button is also displayed until the data is populated. Figure 1 shows the screen before loading of the board and Figure 2 displays the screen when the data is received and Sudoku board is populated with values.

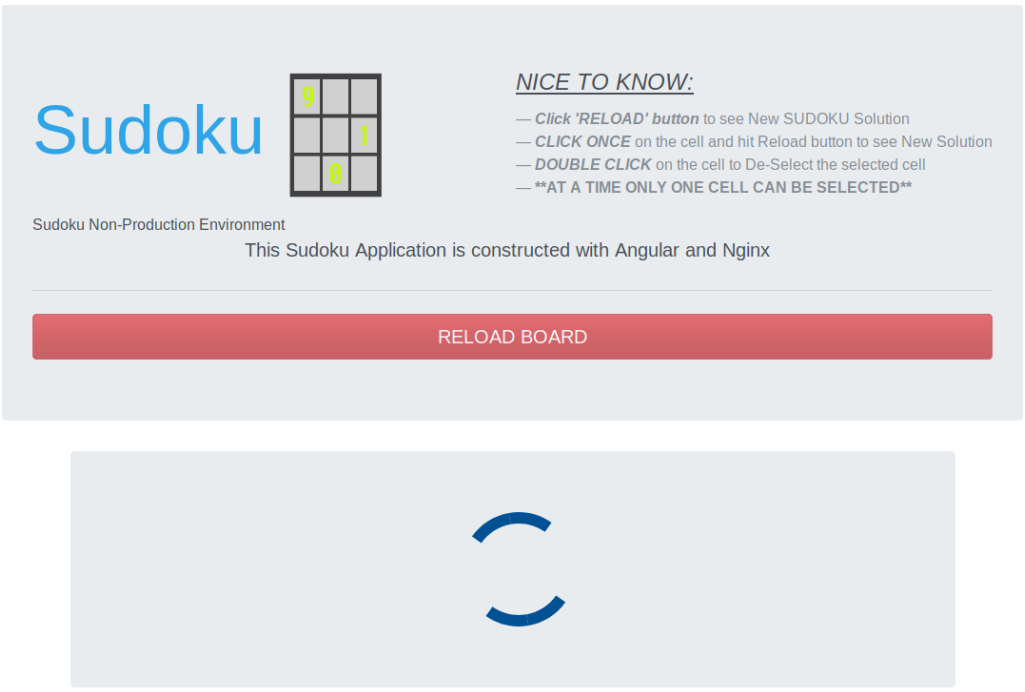

Figure 1: While the UI is waiting for the server response

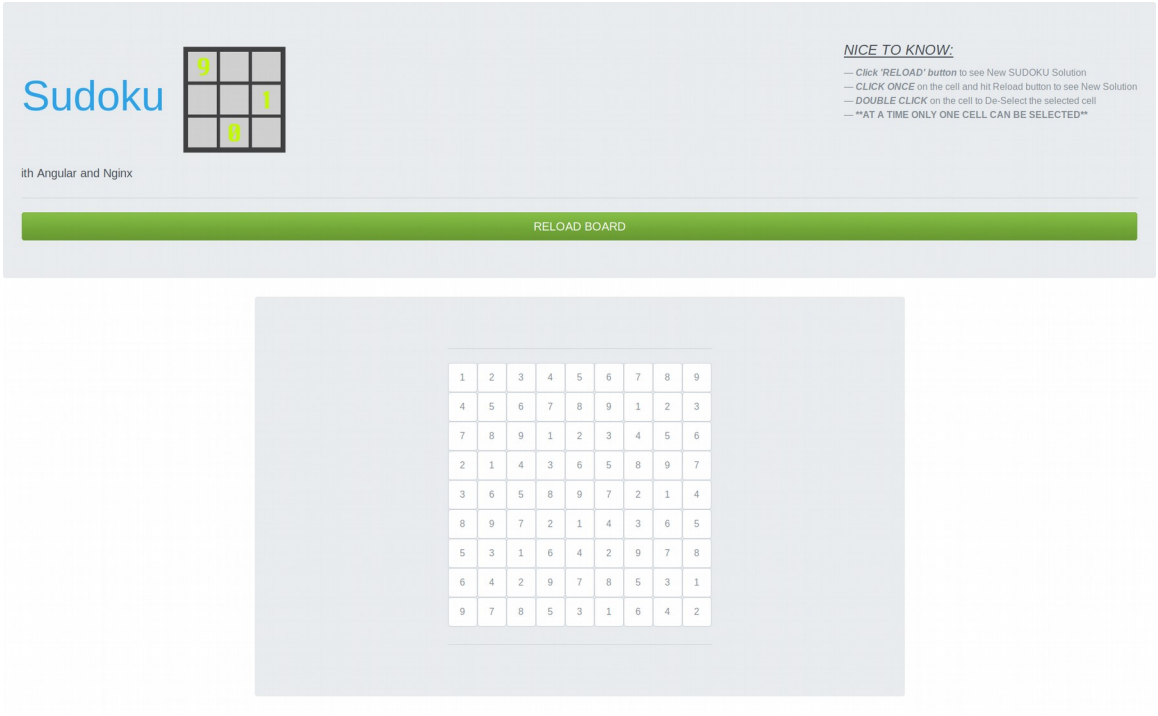

Figure 2: Display of the screen after reception of the Board values from the server.

The user can toggle the cells in the UI. Upon selection of a cell any previously selected cell will be un-selected. The selected cell (darker shade) and un-selected cell is differentiated by a difference in their shade as shown in Figure 3.

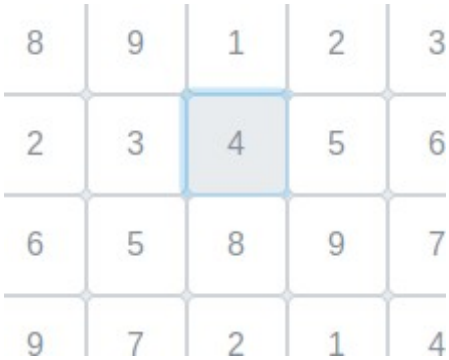

Figure 3: Selected and Un-selected cell.

If for some reason, the application never receives any message then the Figure 1 screen will be shown to the user. But if the server responds with an empty array due to network issues or malformed url formation then the error message of screen at Figure 4 is shown.

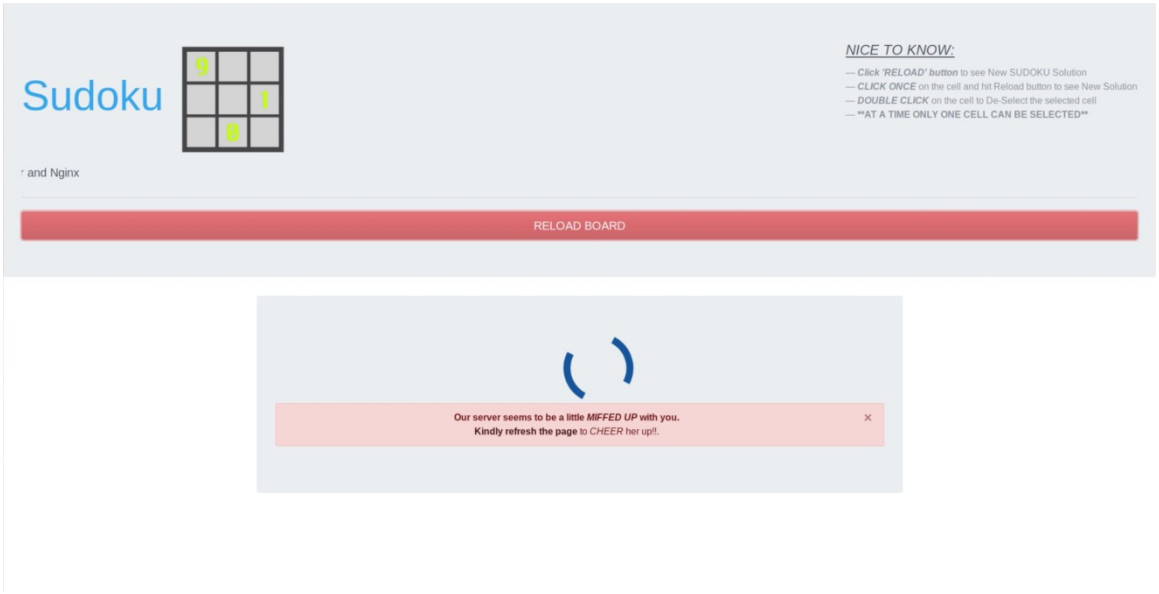

Figure 4: Error Screen