Working with Containerization with complete Authorization –Login, Logout, Forget-Reset Password & Admin Functionality

Raghav Goel

UG student, IT, Maharaja Agrasen Institute Of Technology, Delhi, India.

Abstract: Are you a software engineer/developer/coder or maybe even a tech enthusiast who is thinking of agility, parallel development and reducing cost. In the early twentieth century, we witnessed the rise of Service Oriented Architecture (SOA), which is a software architecture pattern that allows us to construct large-scale enterprise applications that require us to integrate multiple services, each of which is made over different platforms and languages through a commoncommunication mechanism, where we write code and multiple services talk to each other's for a business use case, but sometimes we end up with one big monolithic code base whose maintenance becomes difficult. Nowadays clients are using cloud and paying for on-demand services without effectively utilizing resources. These problems invite micro-services. In this paper, I am going to discuss how one should use scale application in a production environment and local machine

DOI of the Article: https://doi.org/10.46501/IJMTST0706053

Available online at: http://www.ijmtst.com/vol7issue06.html

As per UGC guidelines an electronic bar code is provided to seure your paper

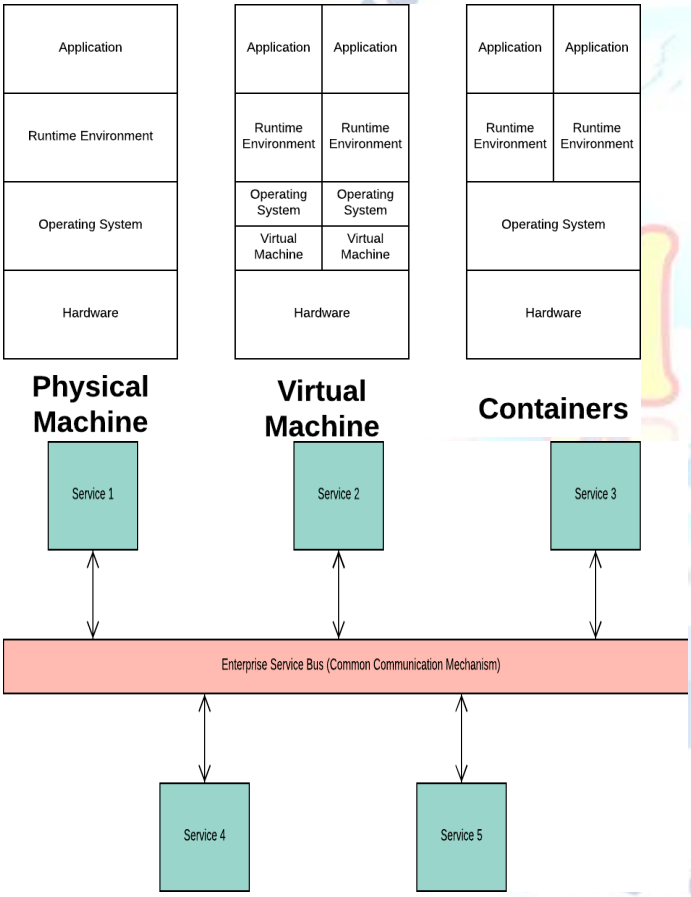
To Cite this Article: Raghav Goel. Working with Containerization with complete Authorization –Login, Logout, Forget-Reset Password & Admin Functionality. *International Journal for Modern Trends in Science and Technology* 2021, *7*, 0706112, pp. 313-323. https://doi.org/10.46501/IJMTST0706053

Article Info. Received: 15 May 2021; Accepted: 14 June 2021; Published: 22 June 2021

INTRODUCTION

Needofthestudy

Build a dynamic containerised platform which in real time deployed with help of Travis CI using MERN stack and containerized technology. A person can see Food Plans hosted on platform with all the real world details and Login his/her personal account form anywhere, anytime. A user can also register himself and use GUI to login or register with his account details. Deployed with help Travis CI, it is hosted on any Cloud Computing platform. Also a user will be directed to the payments page automatically if he wants to book any plan and pay there.



register himself and use GUI to login or register with his account details. Deployed with help Travis CI, it is hosted on any Cloud Computing platform. Also a user will be directed to the payments page automatically if he wants to book any plan and pay.

OBJECTIVE AND SCOPE OF PROJECT

A full-fledged platform where any user can come and see different Plans with all details and can register/ login themselves first and then can book any Tour and make payments. Also the user can change their account details, profile photo. Also they can register and use 'Contact us' forum to get in touch with the developer team. It is completed tested by Travis CI in the real world environment and it is hosted on any cloud computing platform (AWS, GCP, Azure) SimpleTransaction:ConventionalVSBlockchainSystems

(a) What is SOA?

KeyPoints

1. SOA is preferred for large-scale softwareproductssuchasenterpriseapplications
2. SOA focuses on integrating multiple servicesin a single application rather thanemphasizingonmodularizingtheapplication
The common communication mechanismused for interaction

3. between various servicesiscalledEnterprise Service Bus(ESB)

(b) WhatisMonolithicArchitecture?

Forexampleinanonlinetravelshoppingcompanywebsite multiple devices can access most sites, sothey have various user interfaces for laptop andmobileviews.

Also multiple services are running with each other toensure the regular functioning of application servicesare like account creation, displaying travel plancatalog, building and validating your shopping cart,generating bill, order confirmation, paymentmechanismetc.

B. MicroserviceTheRescue

Monolithicarchitecturedrawbackscanbeovercomewith Microservice architecture which focuses onmodularizing the application by dividing it intosmaller standalone services that can be built,deployed, scaled and even

maintainedindependently of other existing services or theapplication itself as a whole. These independentservices arecalled microservices

Highlights

Microservice Architecture is considered to asspecialization of SOA. Or SOA can beconsidered to be a superset of MicroservicesArchitecture

C.Advantages

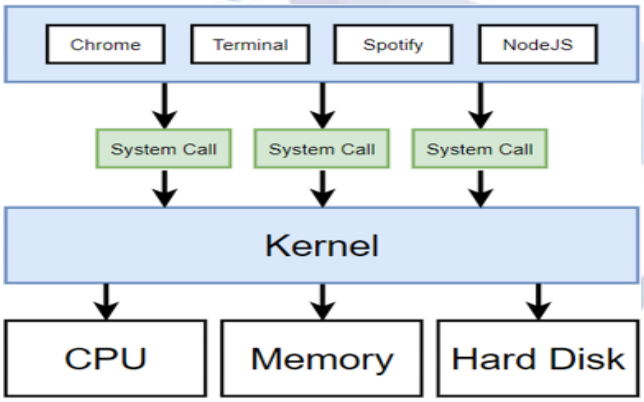
1.Introduces the philosophy ofSeparation ofConcerns and ensures Agile Development ofsoftwareapplications

2. The independent nature of microservicesopendoorsforfollowingbenefits:

- Reduces complexity by allowingdeveloperstobreakintosmallteams,each of which builds/maintains manyservices

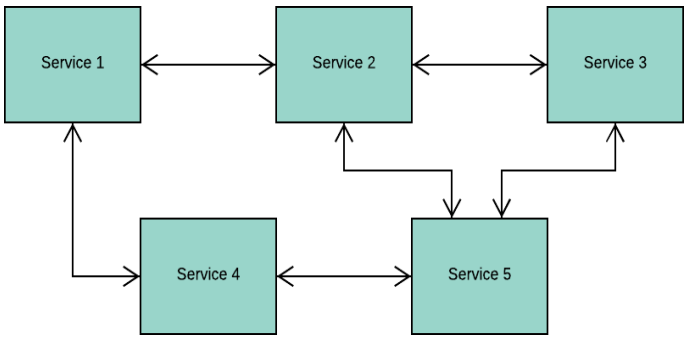
Easy maintenance by allowingflexibility to incrementallyupdate/upgradethetechnologystackfor many services, rather than theentire application in a single point in time.

Also a fully automated deploymentmechanism can be build for ensuringindividual service deployments, servicemanagement and autoscaling of theapplication



METHODOLOGY

EvolutionOfTechnologies



The first picture shows a physical machine. Typically, when we build and deploy applications, we use the resources provided by our host. But if we want to scale the application? At some point, you might want another hardware server and as the number keeps increasing, so does your cost and other resources like hardware and energy consumptions. In the second diagram, VMs have their guest OS which runs over a single physical machine or host OS which allow to run multiple applications without needing installing numerous physical machines. Though VMs make software more accessible to maintain and reduced costs, more optimization was still possible. These problems led to the next innovation: containerization. Unlike virtual machines which were more operating system specific, containers are application specific, making them far lighter. This lead to two things:

1. Multiple containers can run on a single VM. In either case, it solves application related problems

2. Containerization is not in competition with Virtualization, but rather a complementary factor to

It is containerization technology. "Dockercontainers wrap a piece of software in acomplete file system that containseverything needed to run: code, runtime,systemtools,systemlibraries—anythingthat can be installed on a server whichguarantees that the software will alwaysrun the same, regardless of its setup ordependencies."

Docker Client - Tool that we are going to issuecommandsto.

DockerServer
-Toolthatisresponsibleforcreatingimages,runningcontainers,etc

The New Way is to deploy containers basedonOS-levelvirtualizationratherthanhardwarevirtualization.Thesecontainersareisolated from each other and from the hosttheyhavetheirownfilesystems.. containers are isolated from each other and from the host they have their own file systems

Chrome, Terminal, spotify, NodeJS are Processes running

- on your computer.

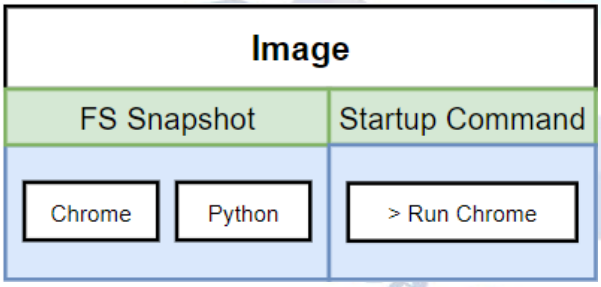
System calls – These are the calls that running programs issues request to kernel to interact with piece of hardware.

Representation of a container

Hard Drive, Network, RAM, CPU are the portions of each made available to process of a container.

What is a Docker Image?

A Docker Image consists of two portions, one is the File System Snapshhot and the other one is the Startup command which can be anything.



Docker – reference the Docker client
Run – Try to create and run a container
<Image Name>Name of the image to use for this container
Command – Default command override.

A. GenerateListallrunningcontainers

To generate the list of all the containers in therunning processes we use the command "dockerps"where psstandsfor process status.

Whereas,theDockerruncommandisthecombinationof the2 commands.i.e.

Dockerrun =dockercreate+dockerstart
Create–triestocreatethe container.
<image-name>-Nameoftheimagetouseforthiscontainer

Start–triesto startthecontainer.

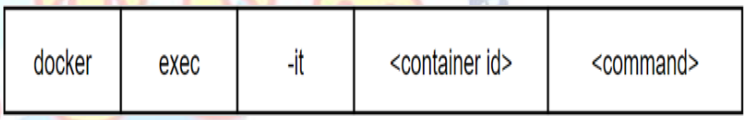
<container id> - gives the ID of the container to thestart the container

B.

C.Executeanadditional commandinacontainer

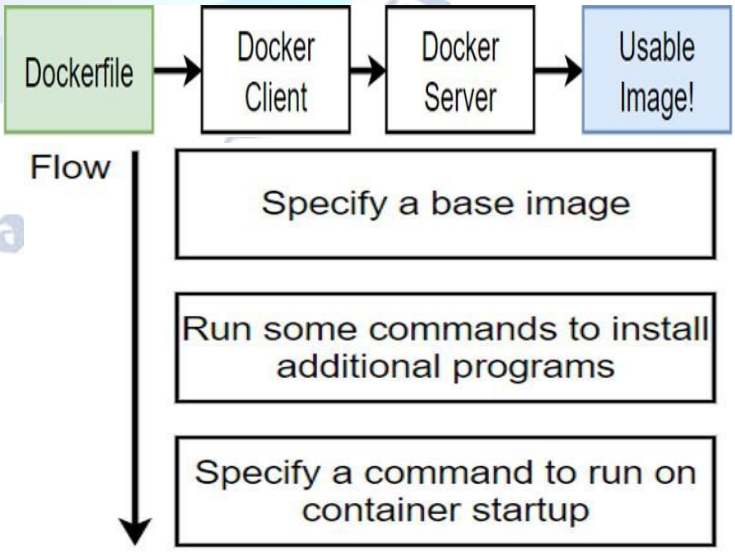
Exec– itisusedtorunanother command.

-it flag - It is Used to provide input to the Container



ForCreateaDockerFile

"Dockerfile" – It is the configuration used to definehowourcontainersshouldbehave.



docker	build	-t	.

Taggingan Image

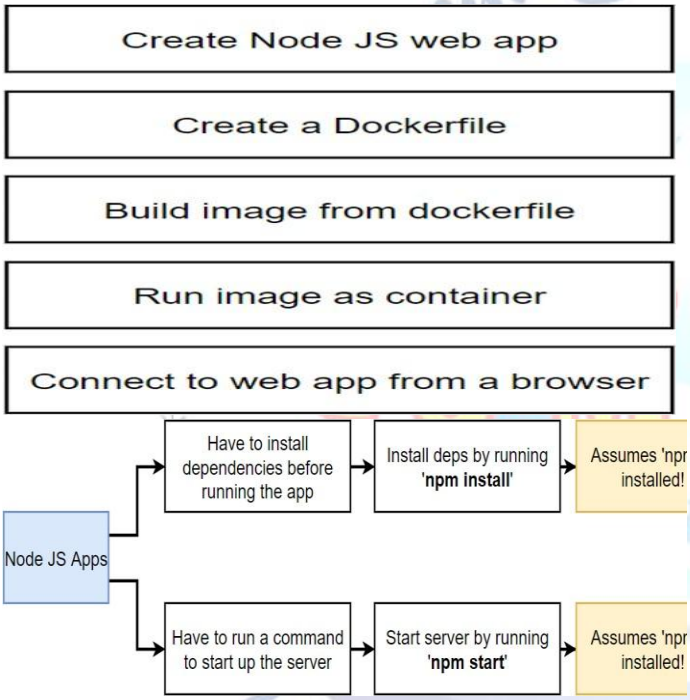
"-t<name>"–Tagetheimage

.Specifies the directory of files/folder to use for thebuildprocess

D. HowtocreateasimplewebserverusingNodeJS?

Itcanbecreated inthe5stepsbelowinthesamesequence.

These are simple 5 steps which leds us to create asimpleweb serverusingNodeJS.



ThischartillustratesthatwithNodejsappswehaveto install the dependencies before running the appwith command "npm install" and to start up theserver withthe command"npm start"

About Database

It is organized collection of data, stored and accessed electronically from a computer. They are often developed using formal design and modeling techniques.

Most databases can be categorized as either:

- Relational
- Non-relational

The main difference between these is how they store their information.

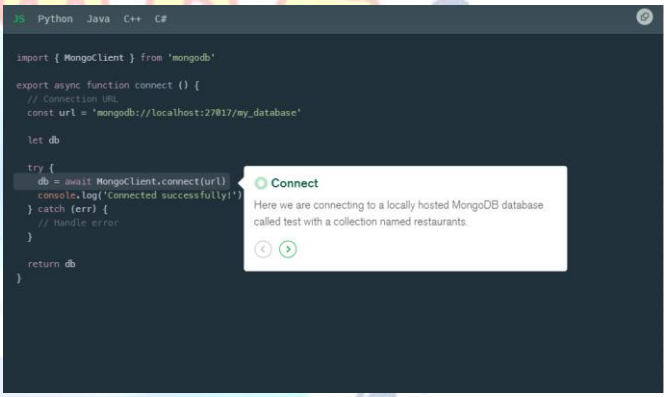
A non-relational database stores data in a non-tabular form, and tends to be more flexible than the traditional, SQL-based, relational database structures. It does not follow the relational model provided by traditional relational database management systems.

I have used Non-Relational Database (NoSQL) which provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases.

DataBase Used – MongoDB

MongoDB is a document database with the scalability and flexibility that you want with the querying and indexing. MongoDB's document model is simple for developers to learn and use, while still providing all the capabilities needed to meet the most complex requirements at any scale.

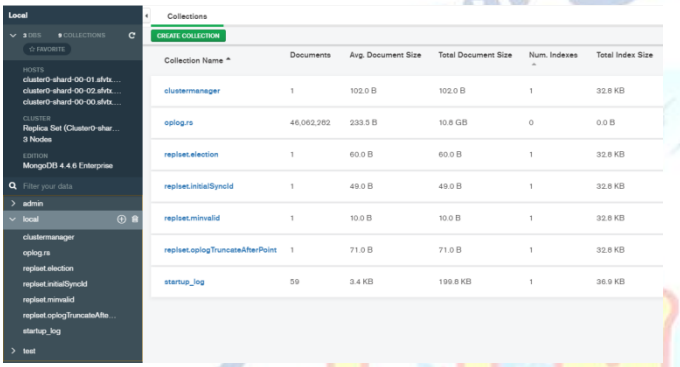
It can be connected easily.

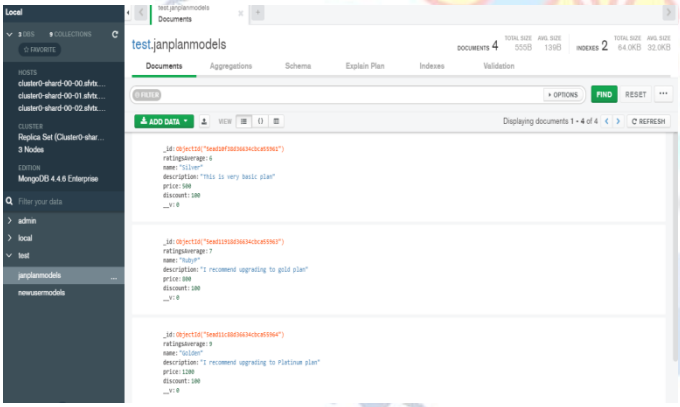


- MongoDB stores data in flexible, JSON-like documents, fields can vary from document to document and data structure can be changed over time.
- The document model maps to the objects in application code, making data easy to work with
- Ad hoc queries, indexing, and real time aggregation provide powerful ways to access and analyze data
- MongoDB is a distributed database at its core, so high availability, horizontal scaling, and geographic distribution are built in and easy to use


```
{
  "_id": "5cf0029caff5056591b0ce7d",
  "firstname": "Jane",
  "lastname": "Wu",
  "address": {
    "street": "1 Circle Rd",
    "city": "Los Angeles",
    "state": "CA",
    "zip": "90404"
  },
  "hobbies": ["surfing", "coding"]
}
```


Personal Database used snippet





Code Used –

Used vs code to write the full development code which consists the authentication code i.e. Login, logout, signup, forget password and reset password, Is user logged In, Is admin and Is authorised code.

Authentication code

authController.js

```
controller > authController.js > signup
1   const { JWT_SECRET } = process.env.JWT_SECRET || require('../configs/config');
2   const userModel = require('../model/userModel');
3   const Email = require('../utility/email');
4   const jwt = require('jsonwebtoken');
5
6   async function signup(req, res) {...
27  }
28
29  async function login(req, res) {...
60  }
61
62  async function logout(req, res) {...
67  }
68
69  async function protectRoute(req, res, next) {...
112 }
113
114 async function isUserLoggedIn(req, res, next) {...
141 }
142 // authorization
143 async function isAdmin(req, res, next) {...
158 }
159
160 function isAuthorized(roles) {...
180 }
181
```

Signup Code

```
async function signup(req, res) {
  try {
    const user = await userModel.create({
      name: req.body.name,
      email: req.body.email,
      password: req.body.password,
      passwordConfirm: req.body.passwordConfirm,
    });
    // const emailOptions = {};
    // emailOptions.html = `<h1>Welcome New User </h1> `;
    // emailOptions.to = email;
    // emailOptions.from = "customersupport@everyone.com";
    // emailOptions.subject = "Welcome" ;
    // await Email(emailOptions);
    res.status(201).json({
      status: 'user signed up',
      user,
    });
  } catch (err) {
    res.status(400).json({ err: err.message });
  }
}
```

Login Code


```
JS authController.js   JS login.js U ●
controller > JS login.js > login > token
async function login(req, res) {
  try {
    const { email, password } = req.body;
    const user = await userModel.findOne({ email }).select('+password');
    if (user) {
      if (password == user.password) {
        const { _id } = user;
        const token = jwt.sign({ id: _id }, JWT_SECRET, {
          expiresIn: Date.now() + 1000 * 60 * 30,
        });
        res.cookie('jwt', token, { httpOnly: true });
        res.status(200).json({
          status: 'successfull',
          user,token
        });
      } else {
        throw new Error("user or password didn't match");
      }
    } else {
      throw new Error("user or password didn't match ");
    }
  } catch (err) {
    console.log(err);
    res.json({
      err: err.message,
```

```
async function isUserLoggedIn(req, res, next) {
  //  1. token verify   2. if(token) then req.userName add kar dega
  try {
    let token;
    if (req.cookies.jwt) {
      // this came from login
      token = req.cookies.jwt;
      //console.log(token);
    }
    if (token) {
      const payload = jwt.verify(token, JWT_SECRET);
      if (payload) {
        const user = await userModel.findById(payload.id);
        req.role = user.role;
        req.id = payload.id; // user id jo mongo providde kar raha hai
        req.userName = user.name;
        next();
      } else {
        next();
      }
    } else {
      next(); // if token hai => next() call karunga + add users name
    }
  } catch (err) {
    console.log(err);
    next();
  }
}
```

Logout

```
JS authController.js   JS logout.js U ●
controller > JS logout.js > logout
async function logout(req, res) {
  res.cookie('jwt', 'bgfdgcgf', { expires: new Date(Date.now() + 100) });
  res.json({
    status: 'logged Out',
  });
}
```

Protect Route

```
JS authController.js   JS protectRoute.js U ●
controller > JS protectRoute.js > protectRoute
async function protectRoute(req, res, next) {
  try {
    let token;
    if (req.headers && req.headers.authorization) {
      token = req.headers.authorization.split(' ').pop();
    } else if (req.cookies && req.cookies.jwt) {
      token = req.cookies.jwt;
    } else {
      throw new Error('Please provide a token');
    }
    if (token) {
      const decryptedData = jwt.verify(token, JWT_SECRET);
      if (decryptedData) {
        const id = decryptedData.id;
        console.log(id);
        req.id = id;
        next();
      } else {
        throw new Error('Invalid Token');
      }
    } else {
      throw new Error('Please login again to access this route ');
    }
  } catch (err) {
    console.log(err);
    let clientType = req.get('User-Agent');
    if (clientType.includes('Mozilla') == true) {
      return res.redirect('/login'); // return ends the startement here only (USING EXPRESS)
    } else {
      res.status(200).json({
        status: 'unsuccessfull',
        err: err.message,
```

isUserLoggedIn

isAdmin –

```
controller > JS protectRoute.js > isAdmin
async function isAdmin(req, res, next) {
  try {
    const user = await userModel.findById(req.id);
    if (user) {
      if (user.role == 'admin') {
        next();
      } else {
        throw new Error('User not authorized');
      }
    } else {
      throw new Error('User not found');
    }
  } catch (err) {
    res.status(400).json({ err: err });
  }
}
```

isAuthorised Code-

```
controller > JS protectRoute.js > isAuthorized
function isAuthorized(roles) {

  return async function (req, res, next) {

    try {
      const { id } = req; // req se "id" aa jayegi
      const user = await userModel.findById(id);
      console.log(user);
      const { role } = user; // taking out "roles"
      if (roles.includes(role) == true) {

        next();
      } else {
        throw new Error('You are not authorized ');
      }
    } catch (err) {
      console.log(err);
      res.status(403).json({ err: err.message });
    }
  };
}
```

Forget Password -

```
controller > JS protectRoute.js > forgetPassword
async function forgetPassword(req, res) {
  try {
    const { email } = req.body;
    const user = await userModel.findOne({ email: email });

    if (user) {
      // console.log(user);
      const token = user.createToken();
      await user.save({ validateBeforeSave: false });
      // email
      const resetPasswordLink = `http://localhost:3000/resetPassword/${token}`;
      const emailOptions = {};
      emailOptions.html = `<h1>Please click on the link to reset your password </h1>
      <p>${resetPasswordLink}</p>`;
      emailOptions.to = email;
      emailOptions.from = 'customersupport@everyone.com';
      emailOptions.subject = 'Reset Password Link';
      await Email(emailOptions);

      res.status(200).json({
        resetPasswordLink,
        status: `Check Gmail ${email}`,
      });
    } else {
      throw new Error('You does not exist');
    }
  } catch (err) {
    console.log(err);
    res.status(400).json({
      err: err.message,
    });
  }
```

Handle reset request

```
controller > JS protectRoute.js > handleResetRequest
async function handleResetRequest(req, res, next) {
  try {
    const { token } = req.params; // params se aayega token
    console.log(token);
    let user = await userModel.findOne({ resetToken: token });

    if (user) {
      // if(user) then verify token
      req.token = token;
      //console.log( req.token)
      next();
    } else {
      res.redirect('/somethingWentWrong');
    }
  } catch (err) {
    res.redirect('/somethingWentWrong');
  }
}
```

Reset Password code –

```
controller > JS protectRoute.js > resetPassword
async function resetPassword(req, res) {
  try {
    const token = req.params.token;
    const user = await userModel.findOne({ resetToken: token });
    if (user) {
      if (Date.now() < user.expiresIn) {
        const { password, confirmPassword } = req.body;
        user.handleResetRequest(password, confirmPassword);
        await user.save();
        res.status(200).json({
          status: 'user password updated login with new password',
        });
      } else {
        throw new Error('token has expired');
      }
    } else {
      throw new Error('user not found');
    }
  } catch (err) {
    console.log(err);
    res.status(400).json({ err: err.message });
  }
}
```

Stripe API and Booking Controller Code –

```
controller > JS protectRoute.js > createSession > session > line_items
let SK = process.env.SK || require('../configs/config').SK;
const stripe = require('stripe')(SK);
const planModel = require('../model/planModel');
const userModel = require('../model/userModel');

async function createSession(req, res) {
  try {
    let id = req.id;  let userId = id;  let { planId } = req.body;

    const user = await userModel.findById(userId);
    const plan = await planModel.findById(planId);

    const session = await stripe.checkout.sessions.create({
      payment_method_types: ['card'],
      customer_email: user.email,
      client_refernce_id: req.planId,
      line_items: [
        {
          name: plan.name, description: plan.description,
          amount: plan.price * 100,     currency: 'usd',
          quantity: 1,
        },
      ],
      success_url: `${req.protocol}://${req.get('host')}/profile`,
      cancel_url: `${req.protocol}://${req.get('host')}/profile`,
    });
    res.status(200).json({
      status: 'success',
      session,
    });
  } catch (err) {
    res.status(200).json({ err: err.message });
```

DockerCommandsinthecodeeditor

FROM, COPY, RUN, CMD are the instructions tellingthe docker server what to do and adjacent to it aretheargumentstotheinstruction.

E. Copycommand

COPY	./	./

"COPY . /" - path to folder to copy from on yourmachinerelativetobuildcontext.

"COPY ./ ./" – place to copy stuff to inside theconatiner

F. PortMapping:DockerRunwithPortMapping

docker	run	-p	5000	:	6000	<image name>

5000:
itistheportwhichroutetheincomingrequeststothisportont helocalhostto.

6000:thisistheportinsidethecontainer

WORKDIR–Itstandsforworkdirectory

/usr/app – Any following command will be executedrelativetothis pathinthecontainer.

WORKDIR	/usr/app

G. WhatisDockercompose?

Insimplestepsdocker-composecanbeunderstoodasbelow inthesamesteps

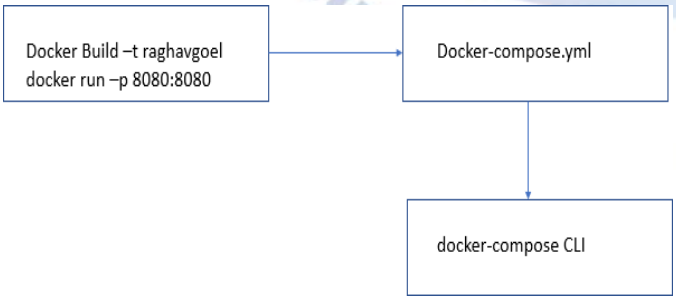
Separate CLI that gets installed along with Docker

Used to start up multiple Docker containers at the same time

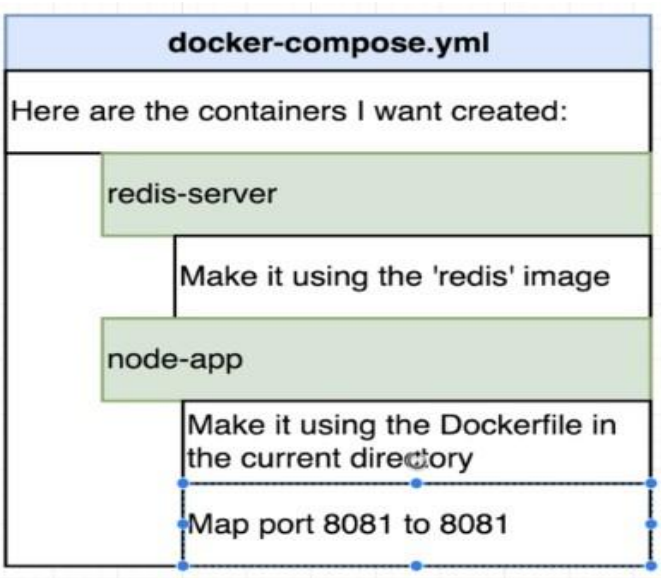
Automates some of the long-winded arguments we were passing to 'docker run'

Docker compose is the separate CLI that getsinstalled along with docker which is used to startupmultiple docker containers at the same time. It alsoautomatessomeifthelongwindedargumentswhichw epassedtodocker-run.

Thewholepurposeofdocker-composeistomakeexecutin gdockerruneasier.



H. Blueprintofdocker-compose.ymlfile



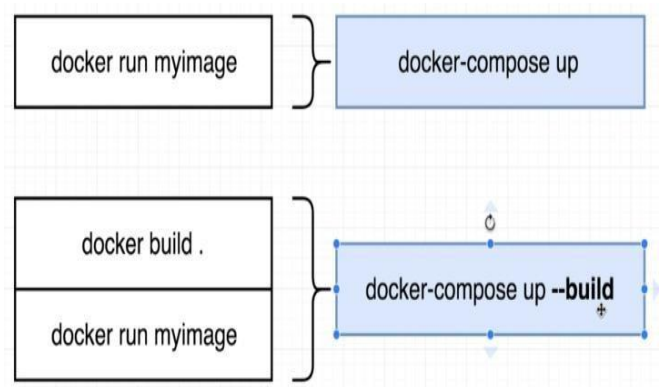
I. Results

```yaml
version: '3'
services:
  redis-server:
    image: 'redis'
  web:
    build:
      context: .
      dockerfile: Dockerfile.dev
    ports:
      - '3000:3000'
    volumes:
      - /app/node_modules
  tests:
    build:
      context: .
      dockerfile: Dockerfile.dev
    volumes:
      - /app/node_modules
      - .:/app
    command: ['npm', 'run', 'test']
```

DISCUSSION

Practicaluseofdocker-composecommands-

Suppose myimage is some any image which we want to runandbuild.



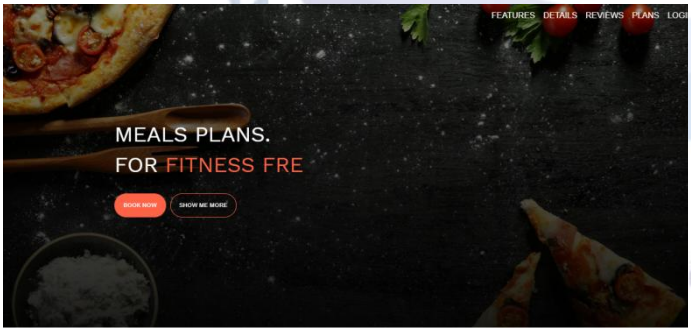
"build."assumesthatwe'ddockerfile insideof current working directory.

I. LaunchandStopDockercontainersusingcomposecommands

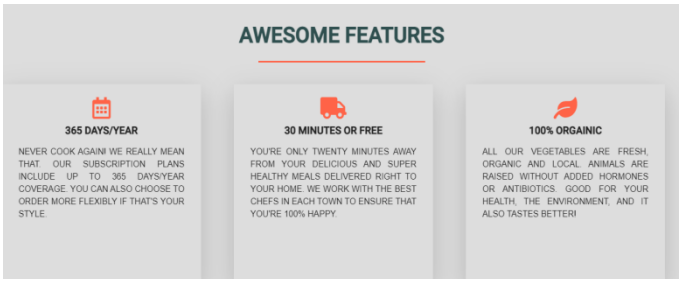
Command:docker-composeup-d

To stop the docker container.

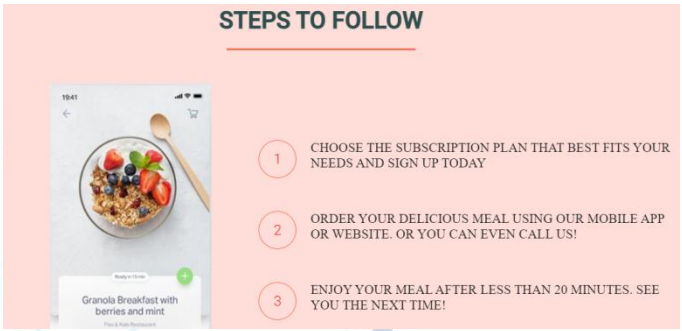
ScreenShots of Project –



Project Snippet 1

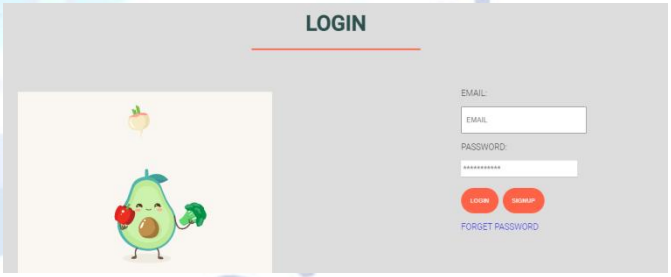


Project Snippet 2



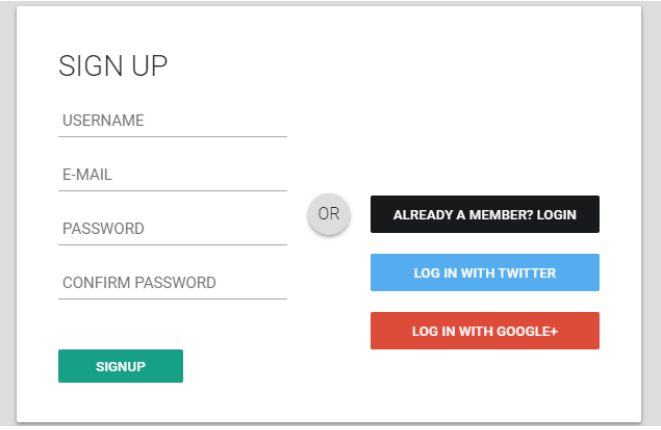
Project Snippet 3

Project Snippet 4



Project Snippet 5

Project Snippet 6



Project Snippet 7

REFERENCES

1. Kubernetes https://docs.docker.com/reference/
2. Docker and Conatiner
3. https://www.docker.com/resources/what-container
4. SOA https://en.wikipedia.org/wiki/Service-oriented_architecture
5. MongoDB https://docs.mongodb.com/
6. Vs code https://code.visualstudio.com/docs
7. Javascript https://developer.mozilla.org/en-US/docs/Web/JavaScript
8. Raghav Goel, T Dr Bhoomi Gupta (2020) Introduction to Containerization 6: 12. 147- 151 12.
9. W3Schools Javascript Reference https://www.w3schools.com/jsref/default.asp
10. Npm Packages https://www.npmjs.com/
11. Github Repository https://github.com/raaghaav/Food-App-complete
12. Live Application https://food-application.herokuapp.com/