DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CSC4121 ENTERPRISE APPLICATION DEVELOPMENT AND DEPLOYMENT USING IBM CLOUD

B.TECH CSE

SEPTEMBER 2021

Name: ABIRAMI.G

Register No: 180071601001

Semester & Section: VII Semester CSE-A

B.S. Abdur Rahman Crescent Institute of Science & Technology
Deemed to be University u/s 3 of the UGC Act, 1956

BONAFIDE CERTIFICATE

Certified that this is the bonafide record of the work done by

……**ABIRAMI.G**……

Register No. of **180071601001** **VII** Semester **B.TECH COMPUTER SCIENCE AND ENGINEERING** in the **CSC4121 ENTERPRISE APPLICATION DEVELOPMENT AND DEPLOYMENT USING IBM CLOUD** during the year **2021.**

Course Faculty Head of the Department

Date :

Submitted for the Practical Examination held on ……

Register No.

Examiner

B.S. Abdur Rahman Crescent Institute of Science & Technology
Deemed to be University u/s 3 of the UGC Act, 1956

Name of the Student : ABIRAMI.G

RRN : 180071601001

Department of the Student : Computer Science & Engineering

Semester from : August 2021 – September 2021

Class &Section : IV year – VII Semester- A

Course Code & Name : CSC4121 Enterprise Application Development And Deployment Using IBM Cloud

S.No	Date	Name of the Experiment	Marks (100)
1		Node Js – Simple Program	
2		Node Js – Creating Modules	
3		Node Js – Query String (seasons)	
4		Node Js – Query String (date and month)	
5		Node Js – NPM	
6		Node Js – File System Module	
7		Node Js in IBM Cloud	
8		DevOps in IBM Cloud	
9		Dockers in Ubuntu	
10		Node Js – Callbacks	
11		Express Js	

Average marks

Course faculty Signature

EX.NO:1

NODE JS – SIMPLE PROGRAM

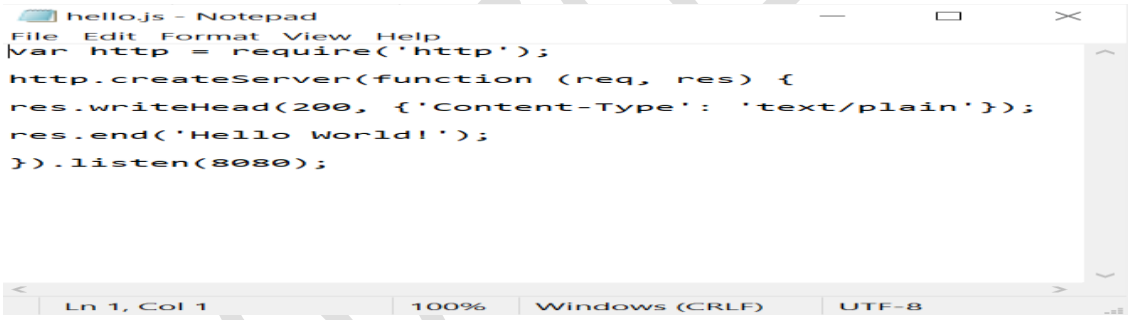
AIM:

To display "Hello World" using Node JS.

PROCEDURE:

STEP – 1:

- Install node js.
- Open notepad and type the following code and save it as 'hello.js'.
- Here the file is saved in "Desktop" location.



STEP – 2:

- Now open command Prompt and navigate to the file location.
- Then run the command "node filemname.js".
- This command will make the pc act like a server. And if anyone tries to access port 8080 they will receive message "hello world".

STEP – 3:

- Now open any browser and type “localhost:8080”.
- This will display “hello world” message as shown

180716012

RESULT:

Thus, simple Node Js program has been successfully implemented.

EX.NO: 2

NODE JS – CREATING MODULES

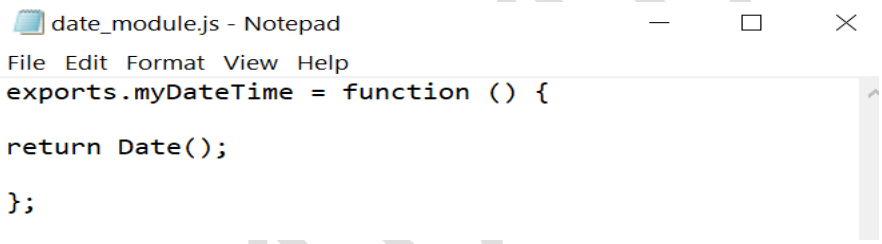
AIM:

To create modules and display current date and time using Node JS.

PROCEDURE:

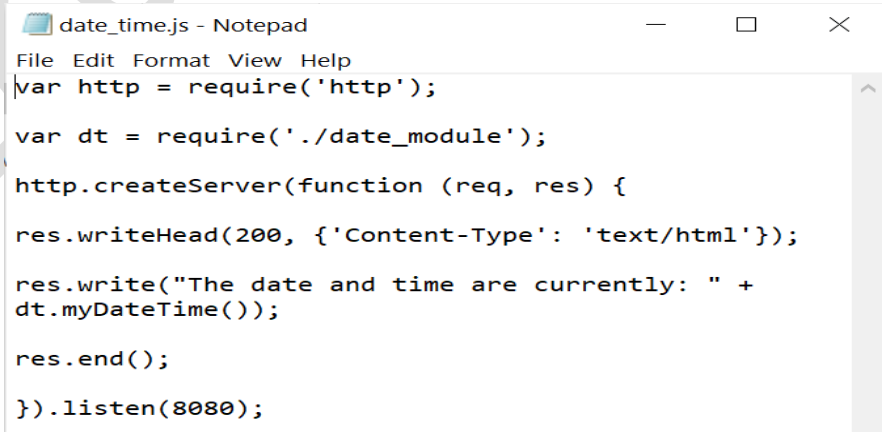
STEP – 1:

- Open notepad.
- Create a date module as shown and save it as “date_module.js” (location Desktop)



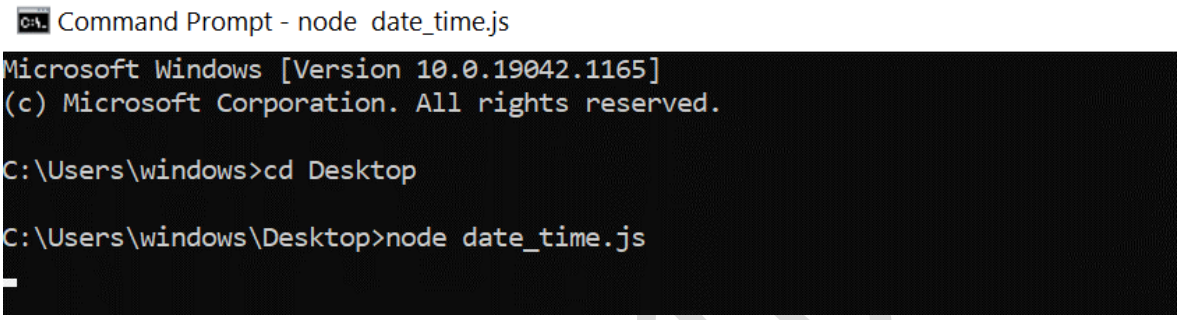
STEP – 2:

- Open notepad and create a new file named “date_time.js”.
- Type the code as shown and save it in Desktop.
- This returns the current date and time if someone tries to access 8080 port.



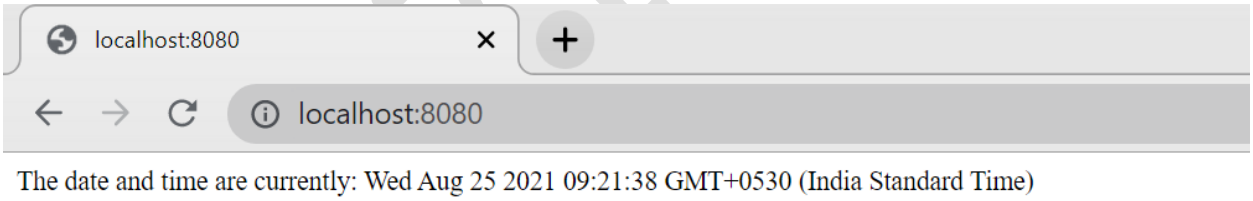
STEP – 3:

- Now open command Prompt and navigate to the file location.
- Then run the command “node filemname.js”.
- This command will make the pc act like a server. And if anyone tries to access port 8080 they will receive the current date and time



STEP – 4:

- Now open any browser and type “localhost:8080”.
- This will display current date and time as shown.



RESULT:

Thus, modules have been created to display date and time successfully and implemented.

EX.NO:3

NODE JS – QUERY STRING (SEASONS)

AIM:

To implement Query Strings for different seasons using Node JS.

PROCEDURE:

STEP – 1:

- Install node js.
- Open notepad and type the following code and save it as 'season.js'.
- Here the file is saved in any preferred location.

season.js - Notepad

File Edit Format View Help

```
var http = require('http');

http.createServer(function (req, res) {

res.writeHead(200, {'Content-Type': 'text/html'});

res.write(req.url);

res.end();

}).listen(8080);
```

STEP – 2:

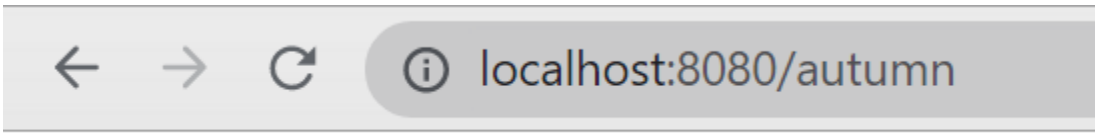
- Now open command Prompt and navigate to the file location.
- Then run the command "node filemname.js".

Command Prompt - node season.js

```
Microsoft Windows [Version 10.0.19042.1165]
(c) Microsoft Corporation. All rights reserved.

C:\Users\windows>cd Desktop

C:\Users\windows\Desktop>cd college

C:\Users\windows\Desktop\college>cd "semester 7"

C:\Users\windows\Desktop\college\semester 7>cd "IBM cloud"

C:\Users\windows\Desktop\college\semester 7\IBM cloud>node season.js
```

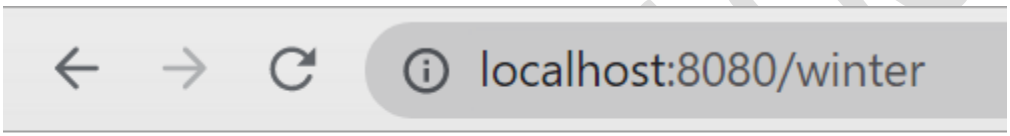
STEP – 3:

- The address: http://localhost:8080/autumn will produce the date and month "/autumn".



/autumn

- The address: http://localhost:8080/winter will produce the date and month "/winter".



/winter

RESULT:

Thus, Query strings for displaying seasons has been successfully implemented.

EX.NO: 4

NODE JS – QUERY STRING (DATE AND MONTH)

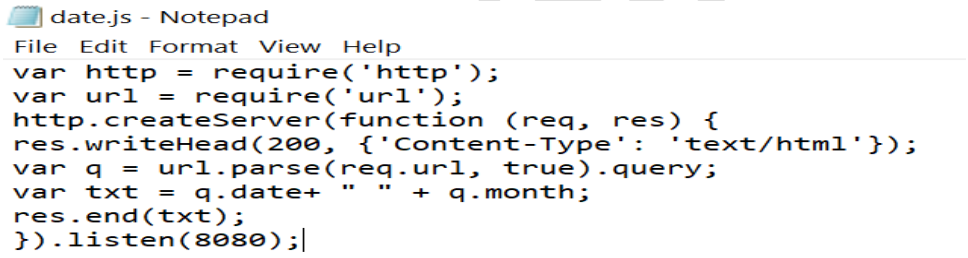
AIM:

To implement Query Strings for date and month using Node JS.

PROCEDURE:

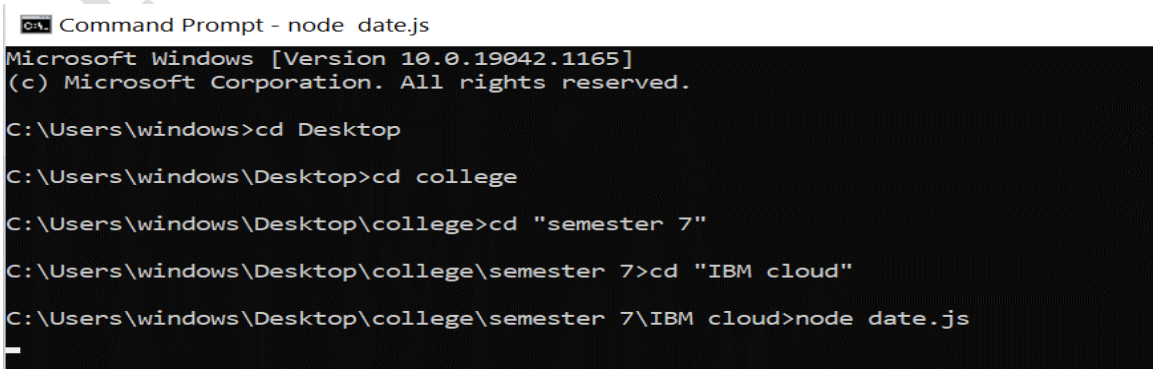
STEP – 1:

- Install node js.
- Open notepad and type the following code and save it as ‘date.js’.
- Here the file is saved in any preferred location.



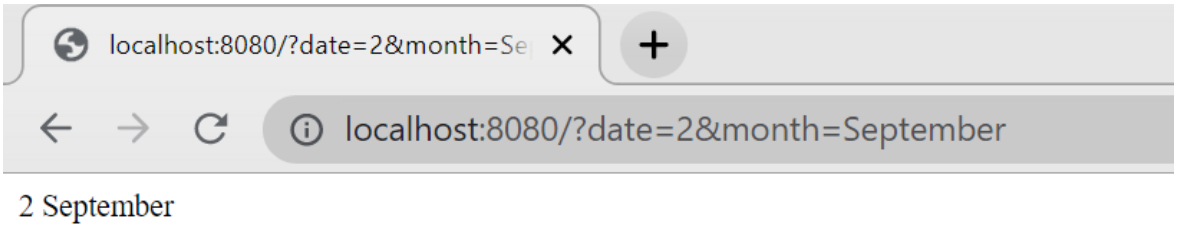
STEP – 2:

- Now open command Prompt and navigate to the file location.
- Then run the command “node filemname.js”.



STEP – 3:

- The address: http://localhost:8080/?date=2&month=September will produce the date and month "2 September".



RESULT:

Thus, Query strings for displaying date and month has been successfully implemented.

1800716010

EX.NO: 5

NODE JS -NPM

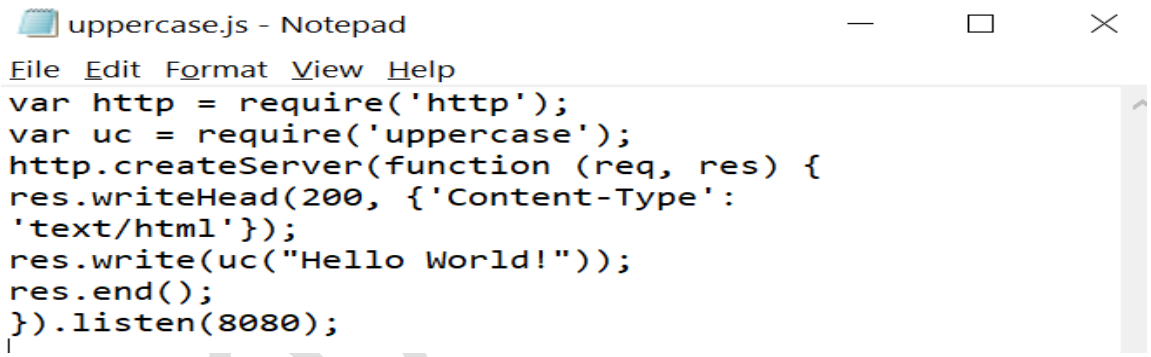
AIM:

To use Node JS - NPM and convert small letters to capital letters.

PROCEDURE:

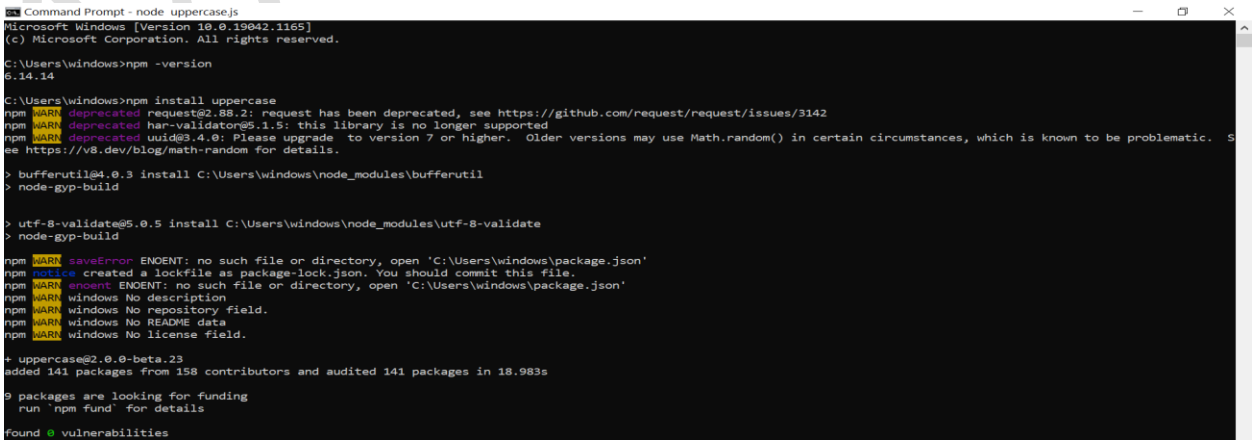
STEP – 1:

- Install node js.
- Open notepad and type the following code and save it as 'uppercase.js'.
- Here the file is saved in any preferred location.



STEP – 2:

- Now open command Prompt and install the package uppercase.
- Command : npm install uppercase



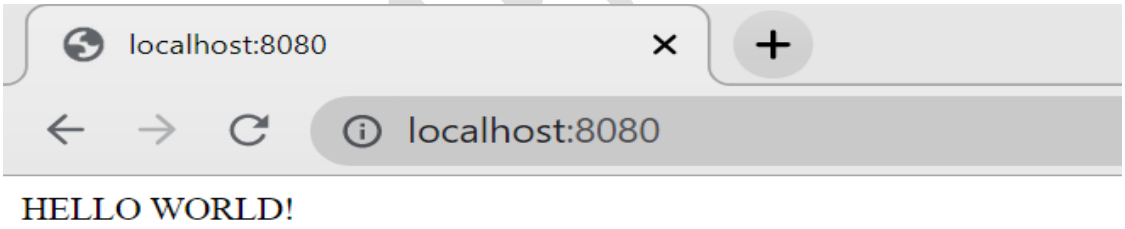
STEP – 3:

- navigate to the file location.
- Then run the command “node filemname.js”.

```
C:\Users\windows>cd Desktop

C:\Users\windows\Desktop>cd college

C:\Users\windows\Desktop\college>cd "Semester 6"

C:\Users\windows\Desktop\college\Semester 6>cd..

C:\Users\windows\Desktop\college>cd "semester 7"

C:\Users\windows\Desktop\college\semester 7>cd "IBM cloud"

C:\Users\windows\Desktop\college\semester 7\IBM cloud>node uppercase.js
```

STEP – 4:

- Now open any browser and type “localhost:8080”.
- This will display “HELLO WORLD” message as shown in uppercase.



RESULT:

Thus, NPM has been used to convert small letters to capital letters successfully.

EX.NO: 6

NODE JS - FILE SYSTEM MODULE

AIM:

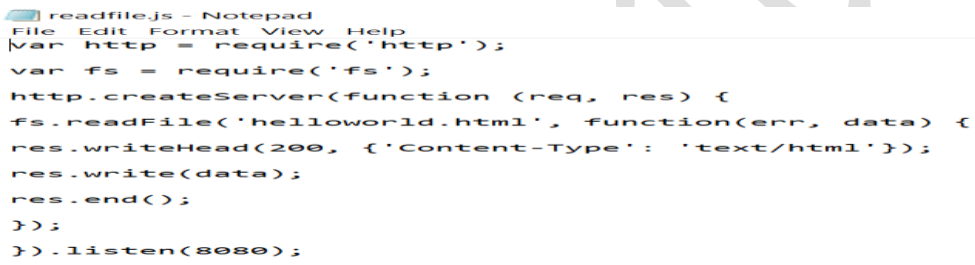
To implement File System Module using Node JS.

PROCEDURE:

READ FILES:

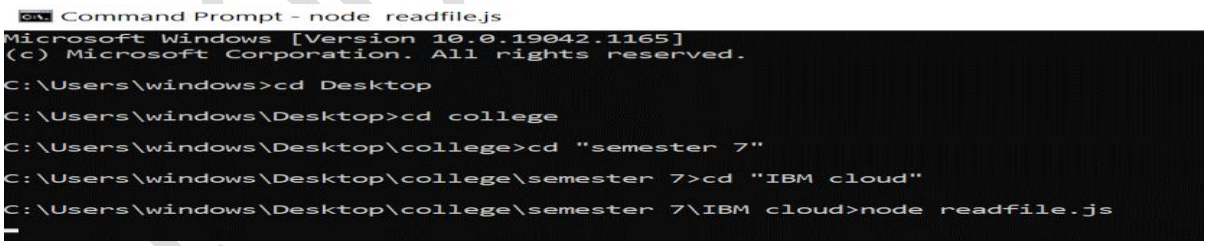
STEP - 1:

- Create a file named “readfile.js” and type the code shown.



STEP - 2:

- Go to the command line and navigate to the file location. Then type “node readfile.js” to read the html file.



STEP - 3:

- Open browser and type https://localhost:8080/ to view the output html file.

APPEND FILES:

STEP - 1:

- Create a file named “append.js” and type the code shown.

append.js - Notepad

File Edit Format View Help

```
var fs = require('fs');
fs.appendFile('newfile1.txt', 'Hello content!', function
(err) {
if (err) throw err;
console.log('Saved!');
});
```

STEP - 2:

- Go to the command line and navigate to the file location. Then type “node append.js” to append the “newfile1.txt” file.

Command Prompt

```
Microsoft Windows [Version 10.0.19042.1165]
(c) Microsoft Corporation. All rights reserved.

C:\Users\windows>cd Desktop\college\semester 7\IBM cloud

C:\Users\windows\Desktop\college\semester 7\IBM cloud>node append.js
Saved!
```

STEP - 3:

- The “newfile1.txt” now has the appended text.

newfile1.txt - Notepad

File Edit Format View Help

```
Hello content!
```

OPEN FILE:

STEP - 1:

- Create a file named “open.js” and type the code shown.

open.js - Notepad

File Edit Format View Help

```
var fs = require('fs');
fs.open('newfile2.txt', 'w', function (err, file) {
if (err) throw err;
console.log('Saved!');
});
```

STEP - 2:

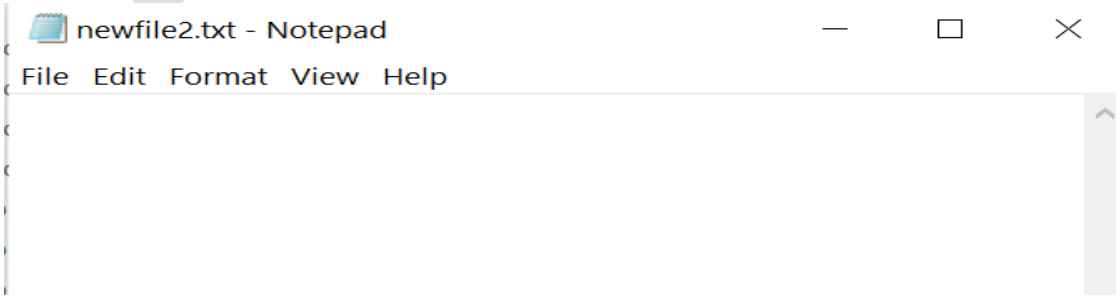
- Go to the command line and navigate to the file location. Then type “node append.js” to open a new file “newfile2.txt”.

Command Prompt

```
Microsoft Windows [Version 10.0.19042.1165]
(c) Microsoft Corporation. All rights reserved.

C:\Users\windows>cd Desktop\college\semester 7\IBM cloud

C:\Users\windows\Desktop\college\semester 7\IBM cloud>node open.js
Saved!
```

STEP - 3:

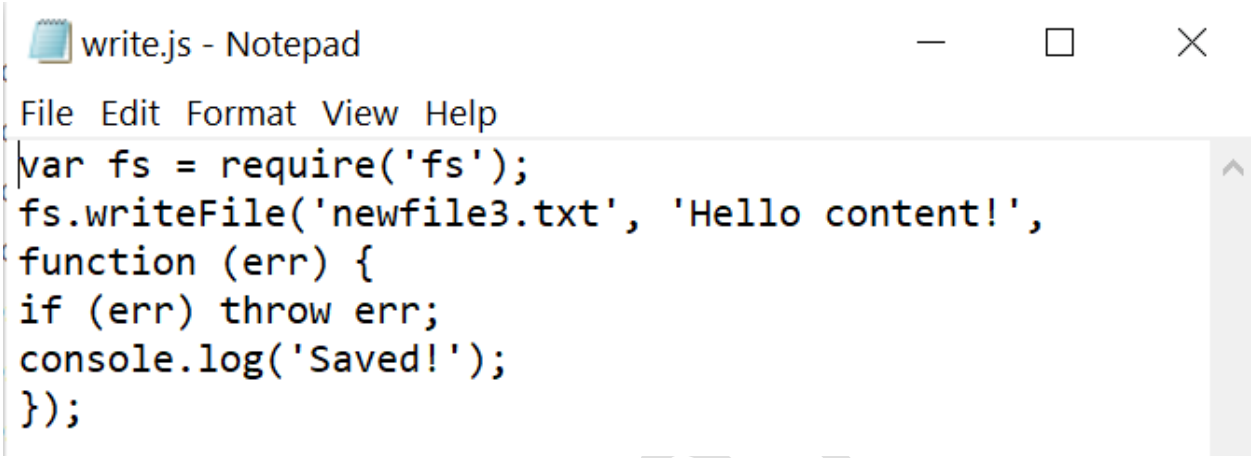
- The new file “newfile2.txt” has been created.



WRITE FILE:

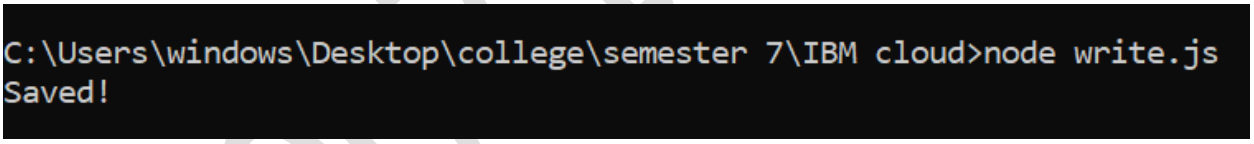
STEP - 1:

- Create a file named “write.js” and type the code shown.



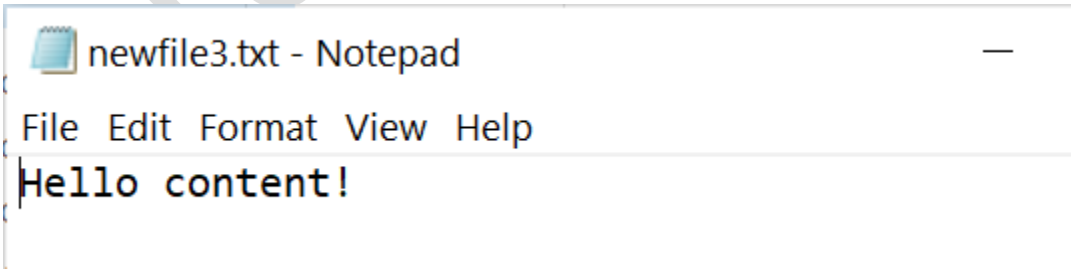
STEP - 2:

- Go to the command line and navigate to the file location. Then type “node append.js” to write into the file “newfile3.txt”.



STEP - 3:

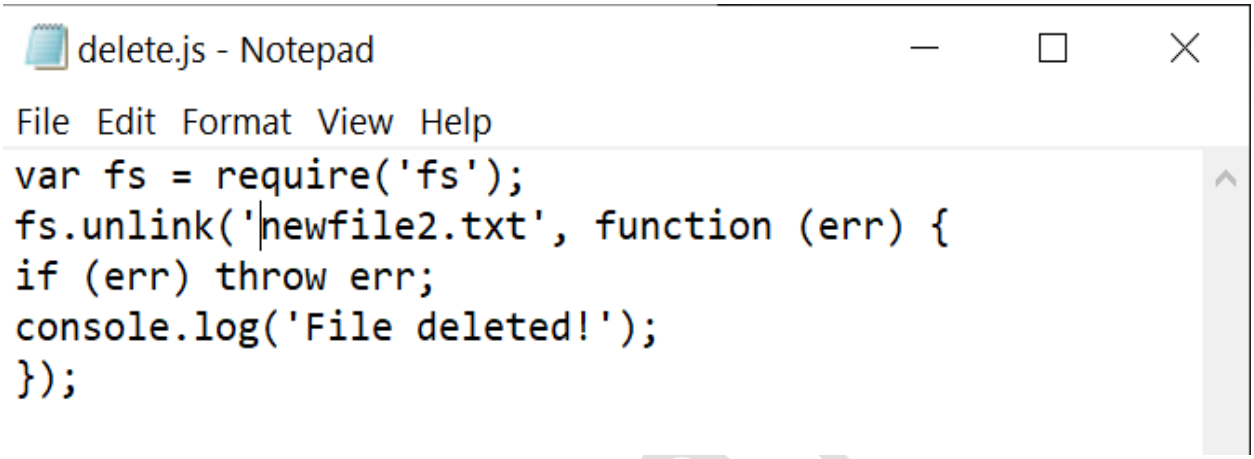
- “newfile3.txt” has been written with the given text.



DELETE FILE:

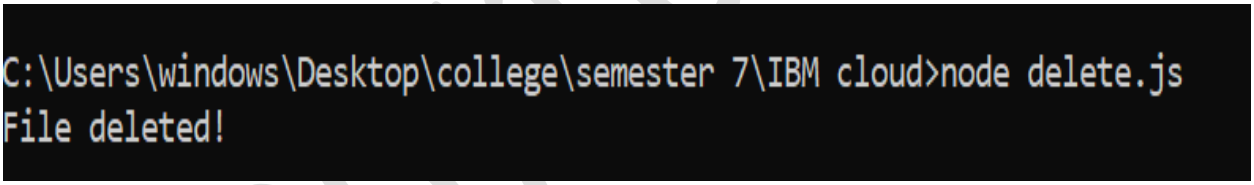
STEP - 1:

- Create a file named “delete.js” and type the code shown.



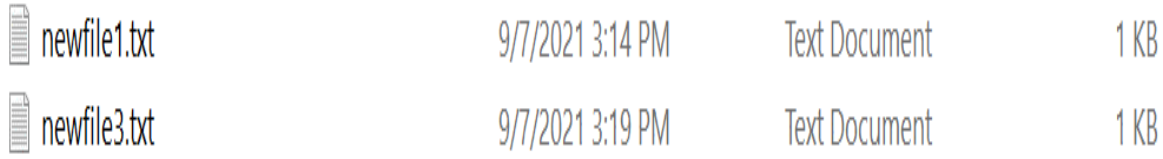
STEP – 2:

- Go to the command line and navigate to the file location. Then type “node append.js” to delete the file “newfile2.txt”.



STEP - 3:

- “newfile2.txt” has been deleted successfully.



RENAME FILE:

STEP - 1:

- Create a file named “rename.js” and type the code shown.

rename.js - Notepad

File Edit Format View Help

```
var fs = require('fs');
fs.rename('newfile1.txt', 'myrenamedfile.txt', function (err) {
if (err) throw err;
console.log('File Renamed!');
});
```

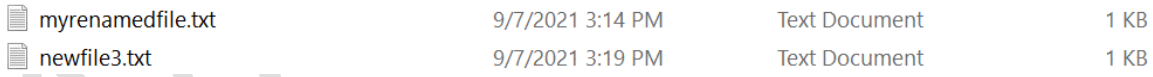
STEP - 2:

- Go to the command line and navigate to the file location. Then type “node append.js” to rename the file “newfile1.txt” to “myrenamedfile.txt”.

```
C:\Users\windows\Desktop\college\semester 7\IBM cloud>node rename.js
File Renamed!
```

STEP - 3:

- “newfile1.txt” has been renamed to “myrenamedfile.txt”.



RESULT:

Thus, File System module has been successfully implemented using Node JS.

EX.NO: 7

NODE JS IN IBM CLOUD

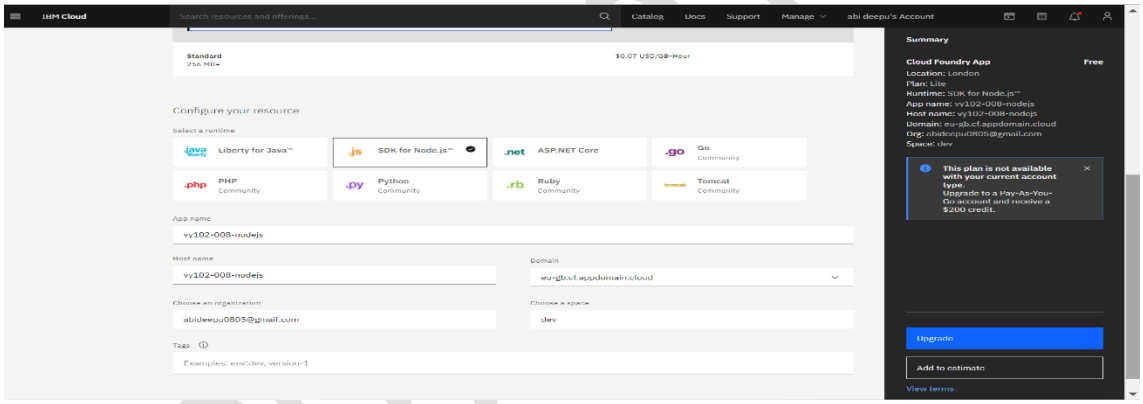
AIM:

To print hello world in Node JS using IBM cloud.

PROCEDURE:

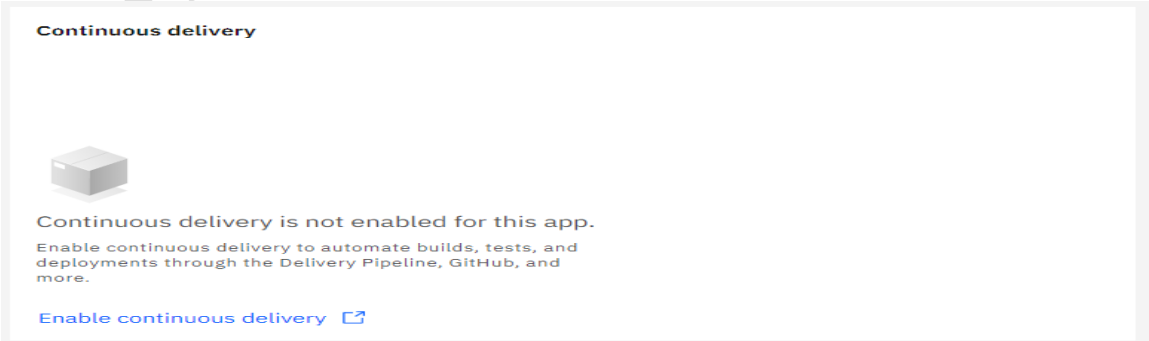
STEP – 1:

- Login to IBM cloud and create a new cloud foundry app for SDK for Node JS.
- Name the app and click create.



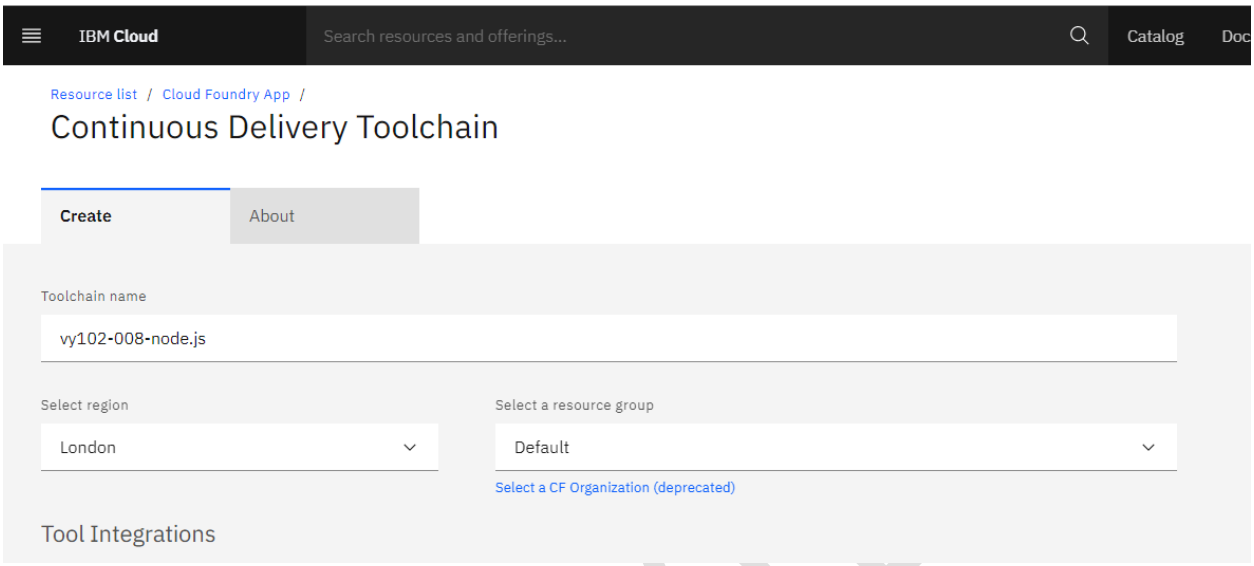
STEP – 2:

- In the overview tab inside the created app, scroll down to continuous delivery and click "enable continuous delivery".

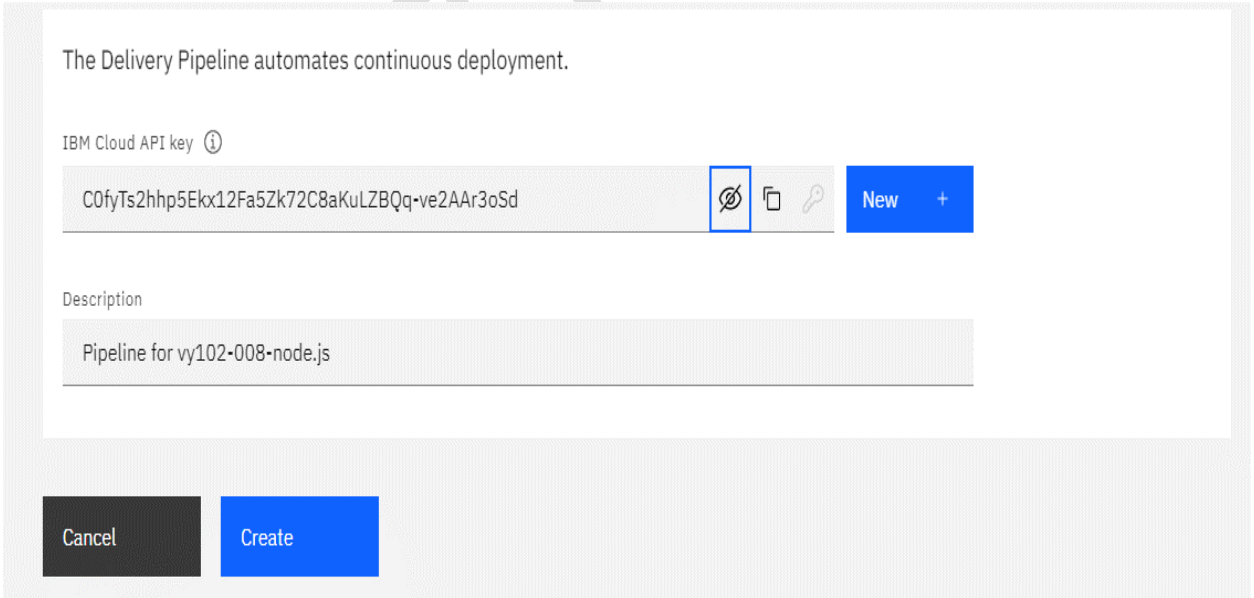


STEP – 3:

- Select region as “London” and set resource group as “default”.

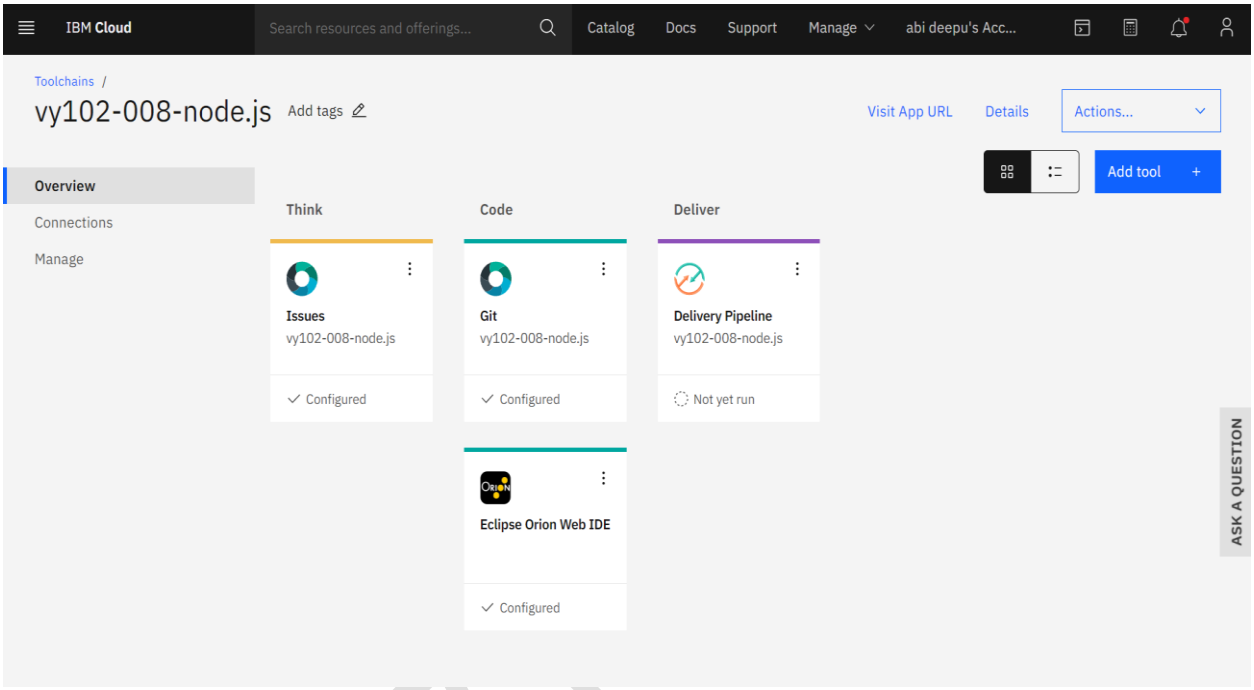


- Scroll down in the same page and click on “New +” to create a new API key.
- Then click “Create”.



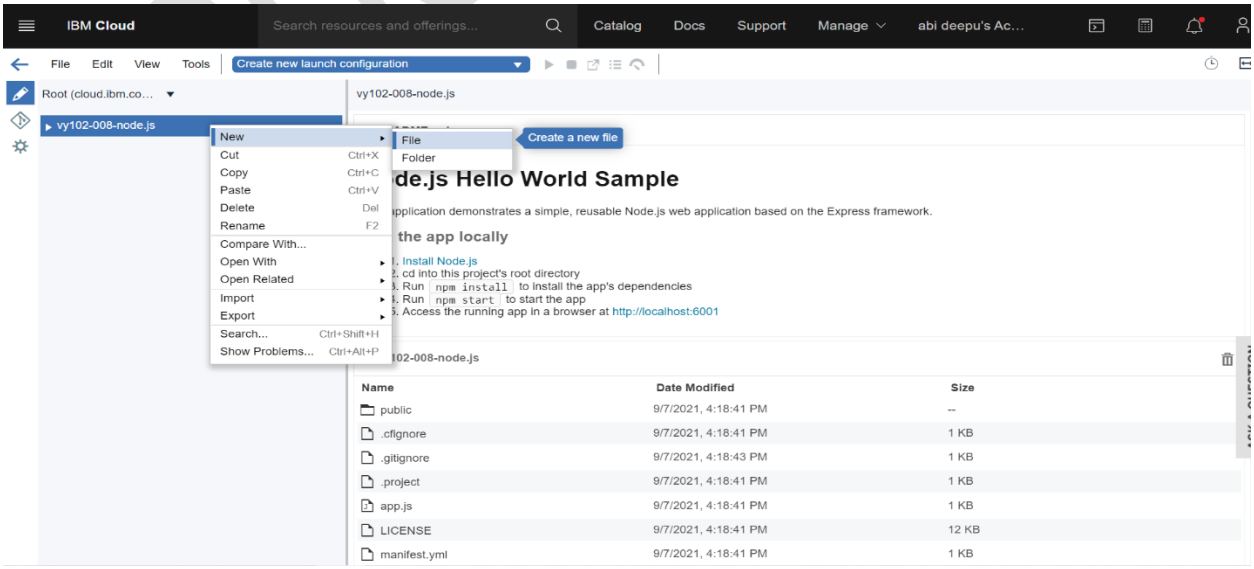
STEP – 4:

- Now click on “Eclipse Orion Web IDE”.
- This will create a new working space.



STEP – 5:

- Now click on File -> New and name the file as manifest.yml.



STEP – 6:

- Type the following code in “manifest.yml”.

manifest.yml

```
applications:
  - path: .
    name: vy102-008-node.js
    environment_json: {}
    memory: 256M
    instances: 1
    disk_quota: 1024M
    services: []
```

STEP – 7:

- Similarly create a new file named “package.json” and type the following code.

package.json

```
{
    "name": "NodejsStarterApp",
    "version": "0.0.1",
    "description":"A hello world nodejs sample :)",
    "private": true,
    "scripts": {
        "start": "node app.js"
    },
    "dependencies": {
        "express": "4.15.x",
        "cfenv": "1.0.x"
    },
    "repository": {}
}
```

STEP – 8:

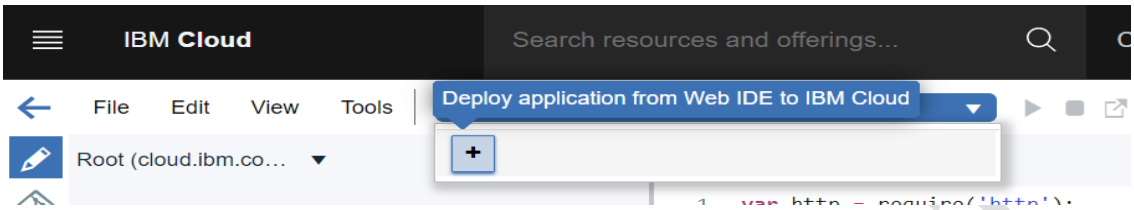
- Create a new file named “app.js” and type the following code.

app.js

```
var http = require('http');

var port = process.env.VCAP_APP_PORT || 8080;

http.createServer(function (request,response) {

    response.writeHead(200, {'Content-Type': 'text/plain'});
    response.end('Hello World :)')

}).listen(port);
```

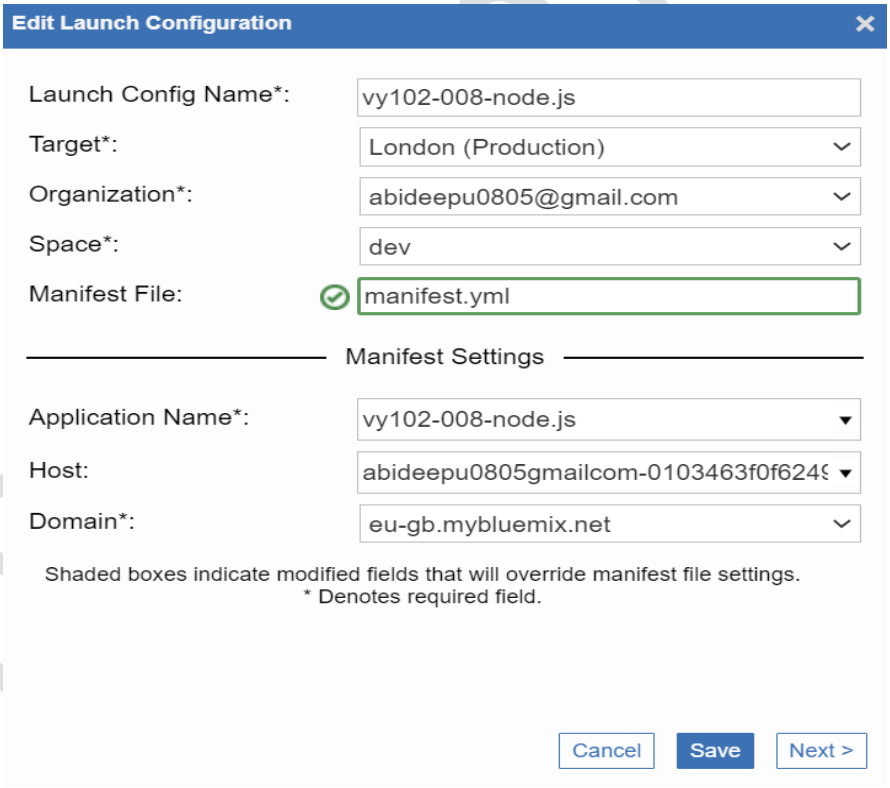
STEP – 9:

- Now click on the "+" to deploy application from web IDE to IBM cloud.



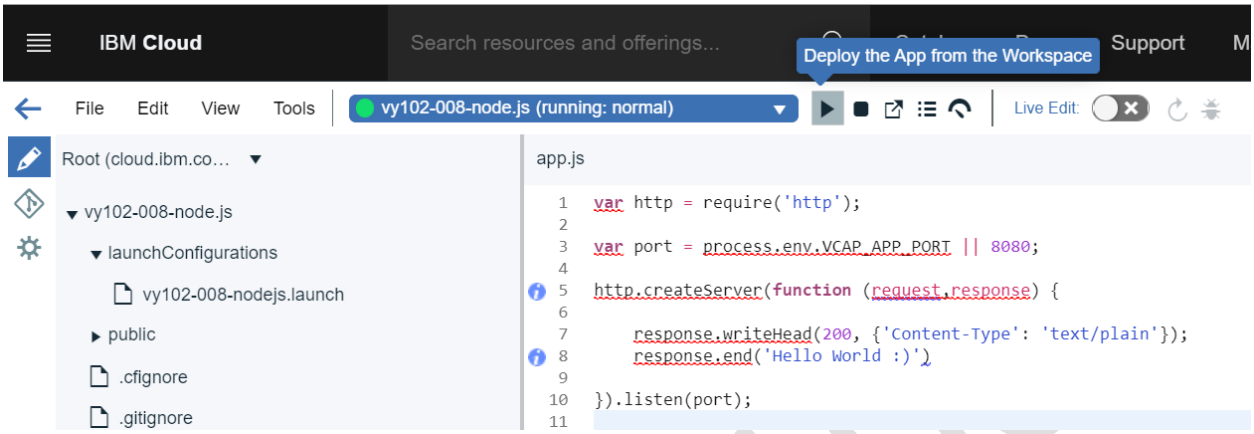
STEP – 10:

- In the launch configuration, set organization as your email id, Space as "dev" and click "save".



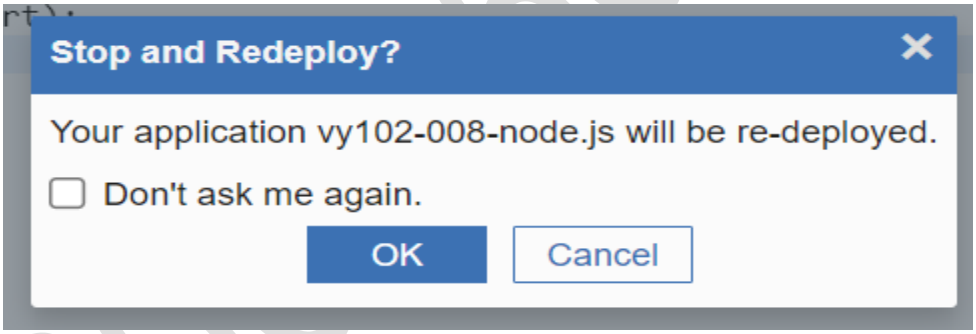
STEP – 11:

- Click on the play button to deploy app from workspace.



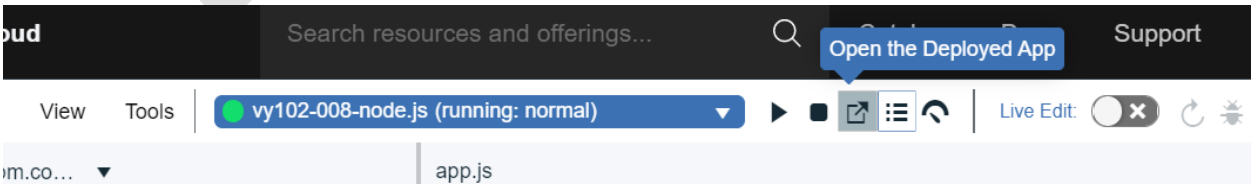
STEP – 12:

- Click on "ok" for app to be re-deployed.



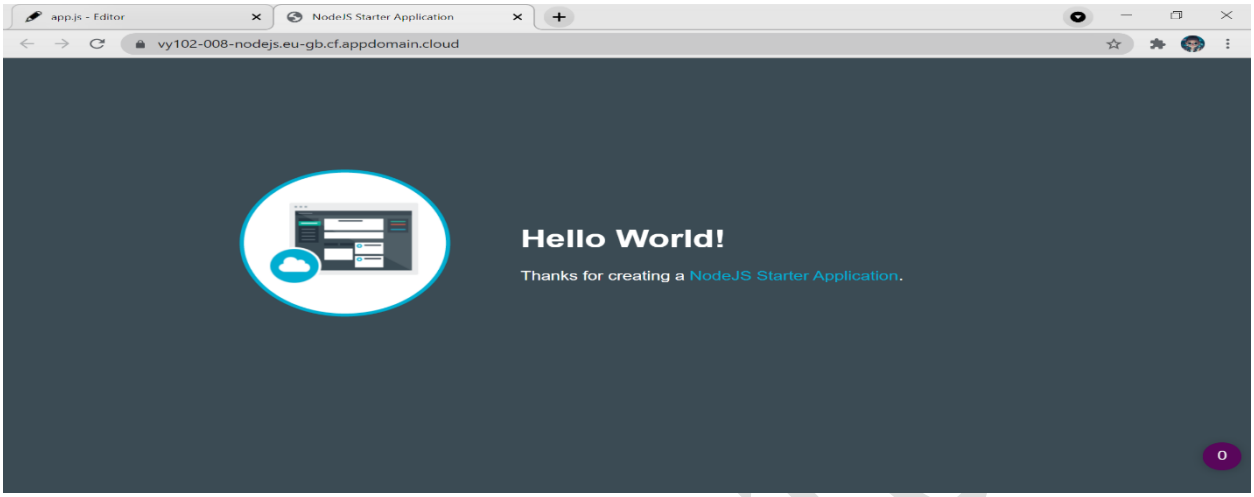
STEP – 13:

- Once the workspace is reloaded, click on "open the deployed app" icon as shown.



STEP – 14:

- The node JS application has been successfully deployed.



RESULT:

Thus, Query Node Js in IBM Cloud has been successfully implemented.

EX.NO: 8

DEVOPS USING IBM CLOUD

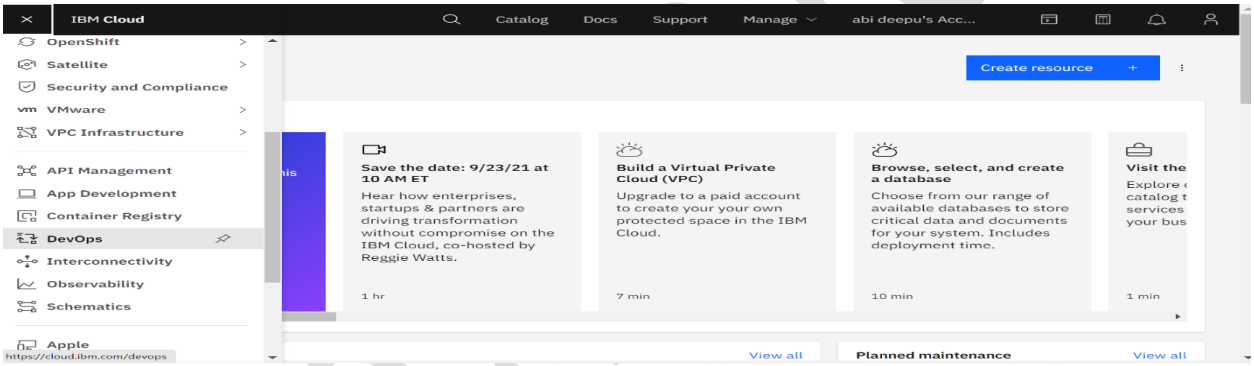
AIM:

To implement DevOps using IBM cloud.

PROCEDURE:

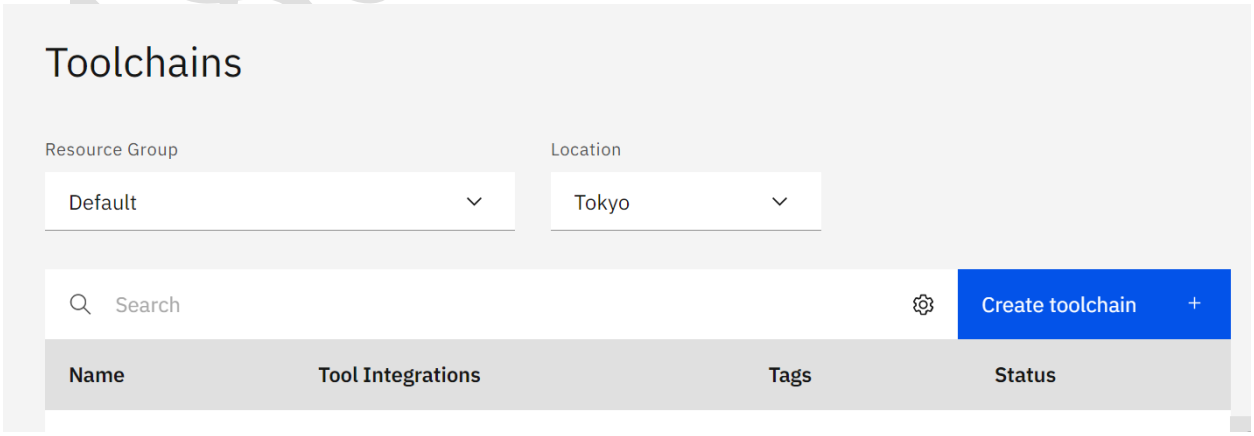
STEP – 1:

- Login to IBM cloud account.
- Open the left Navigation bar and go to DevOps category.



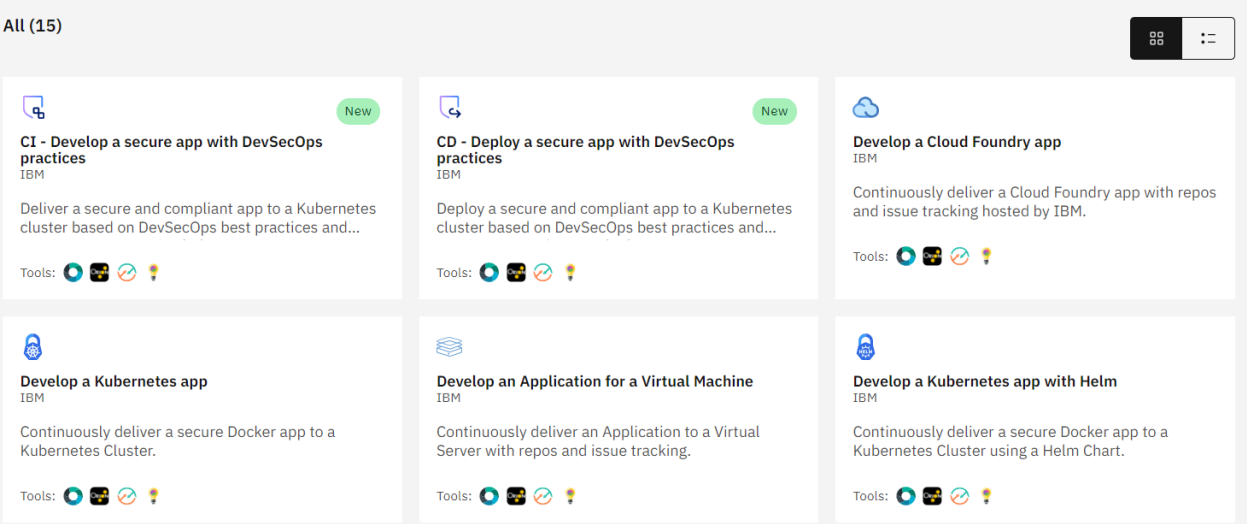
STEP – 2:

- Ensure the LOCATION is “Tokyo” and click on Create Toolchain Button.



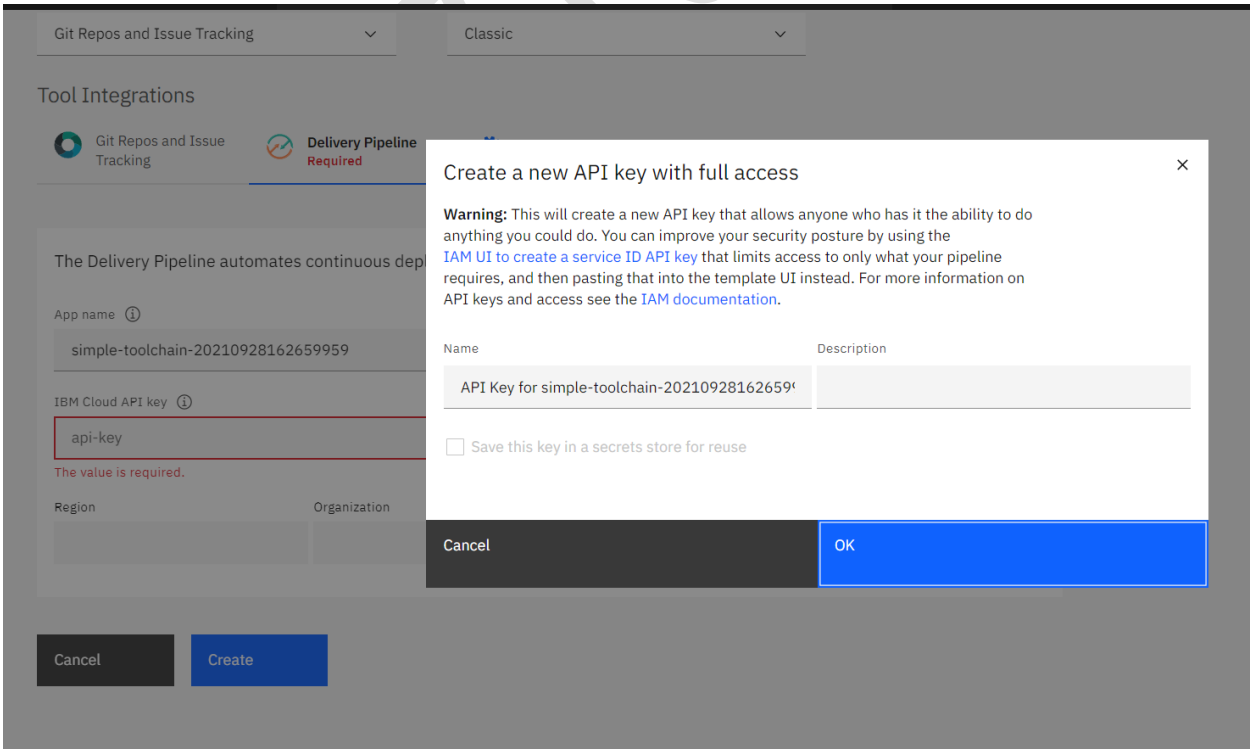
STEP – 3:

- Click on Develop a cloud foundry app with DevOps insights option.



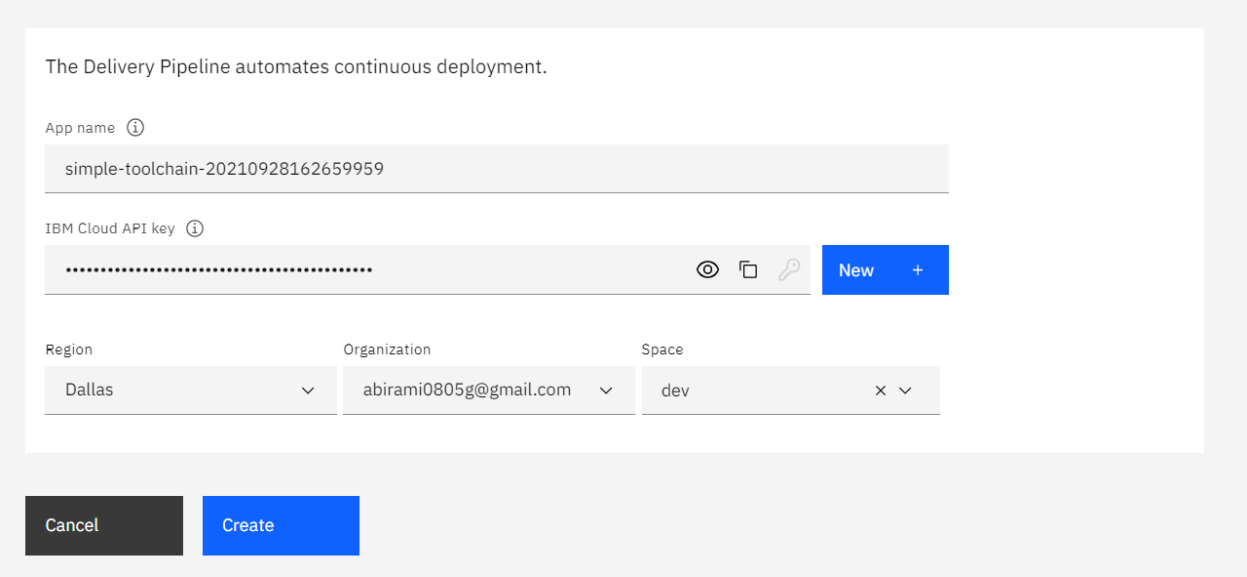
STEP – 4:

- In the delivery pipeline create new api key.



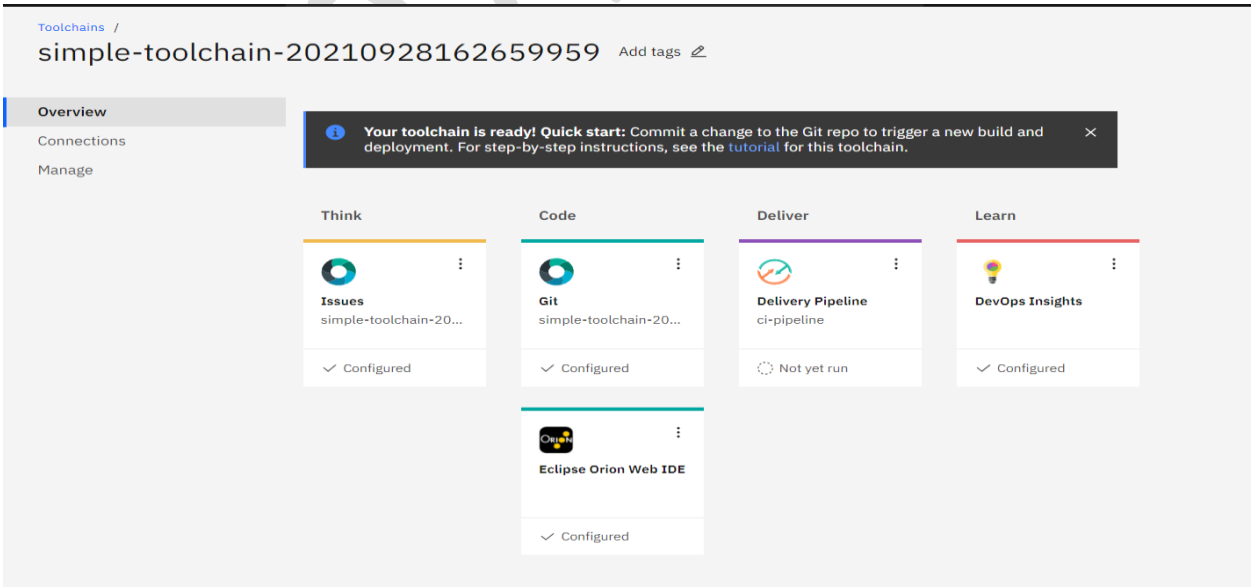
STEP – 5:

- The Space is automatically loaded.
- Now Click create.



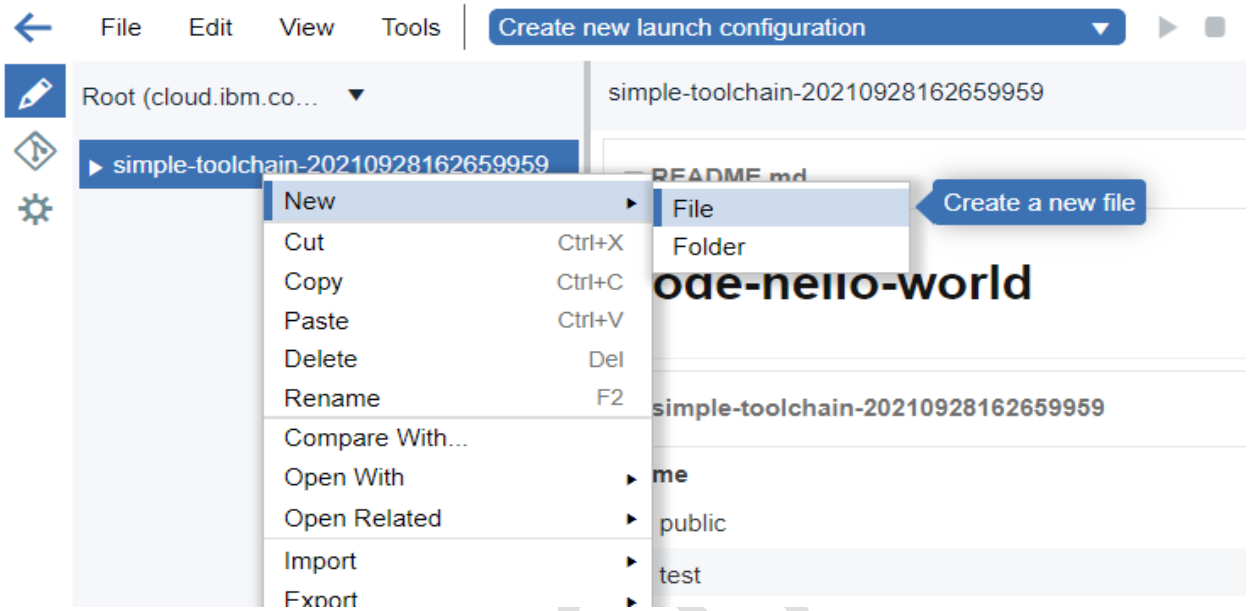
STEP – 6:

- Once the toolchain is configured, Open Eclipse Orion Web IDE.



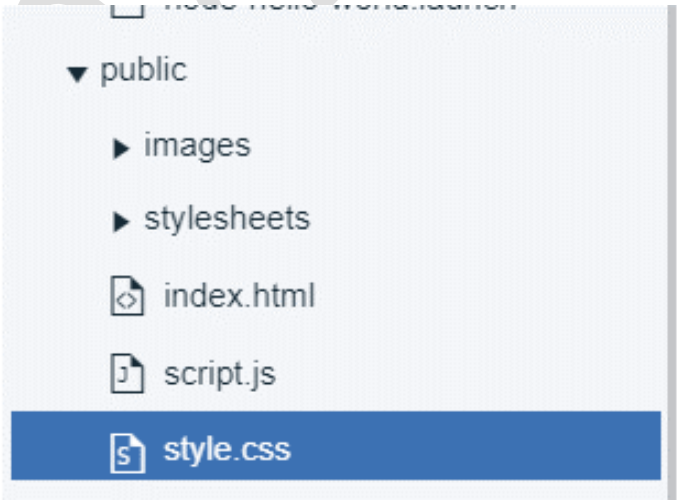
STEP – 7:

- Create 3 new files as shown.



STEP – 8:

- The files should be under the public folder.



STEP – 9:

- Create file named index.html and type the code below.

CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="Description" content="My Weather App">
   <title>Weather App</title>
   <link rel="stylesheet" href="style.css">
   <script src="https://kit.fontawesome.com/6392c1211e.js" crossorigin="anonymous"><
/script>
   <link rel="preconnect" href="https://fonts.googleapis.com">
   <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>

   <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;700;900&di
splay=swap"
rel="stylesheet">
   <script src="script.js" defer></script>
</head>
<body>
   <div class="Search">
      <input type="text" class="Searchbar" placeholder="Search">
      <button><i class="fas fa-search-location"></i></button>
   </div>
   <div class="Card Loading">
      <div class="Card-1">
         <div class="part1">
            <h2 class="Day">Tuesday</h2>
            <div class="Date">15 Jan 2019</div>
            <div class="Location">
               <i class="fas fa-map-marker-alt"></i>
               <span class="City">Paris, FR</span>
            </div>
         </div>
         <div class="part2">
```

180716010 01

```
            <img src="https://openweathermap.org/img/wn/04n.png" alt="" class="icon" h
eight="90px"
width="90px" />
            <h1 class="Temp">25°C</h1>
            <div class="Description">Sunny</div>
          </div>
        </div>
<div class="Card-2">
          <table cellpadding="4">
            <tr>
              <td>PRECIPITATION</td>
              <td>0 %</td>
            </tr>
            <tr>
              <td>HUMIDITY</td>
              <td>34 %</td>
            </tr>
            <tr>
              <td>WIND</td>
              <td>0 km/h</td>
            </tr>
          </table>
 <div class="Weekdays">
            <div class="Days One">
              <img src="https://openweathermap.org/img/wn/04n.png" alt="" class="icon"
/>
              <p id="d1">Tue</p>
              <p id="t1">29°C</p>
            </div>
            <div class="Days Two">
              <img src="https://openweathermap.org/img/wn/04n.png" alt="" class="icon"
/>
              <p id="d2">Wed</p>
              <p id="t2">21°C</p>
            </div>
            <div class="Days Three">
              <img src="https://openweathermap.org/img/wn/04n.png" alt="" class="icon"
/>
              <p id="d3">Thur</p>
```

1807160101

```
                <p id="t3">8°C</p>
            </div>
            <div class="Days Four">
                <img src="https://openweathermap.org/img/wn/04n.png" alt="" class="icon"
    />
                <p id="d4">Fri</p>
                <p id="t4">19°C</p>
            </div>
        </div>
    </div>
</body>
</html>
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="Description" content="My Weather App">
    <title>Weather App</title>
    <link rel="stylesheet" href="style.css">
    <script src="https://kit.fontawesome.com/6392c1211e.js" crossorigin="anonymous"></script>
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
      <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;700;900&display=swap"
rel="stylesheet">
    <script src="script.js" defer></script>
</head>
<body>
    <div class="Search">
        <input type="text" class="Searchbar" placeholder="Search">
        <button><i class="fas fa-search-location"></i></button>
    </div>
    <div class="Card Loading">
        <div class="Card-1">
            <div class="part1">
                <h2 class="Day">Tuesday</h2>
                <div class="Date">15 Jan 2019</div>
                <div class="location">
                    <i class="fas fa-map-marker-alt"></i>
                    <span class="City">Paris, FR</span>
                </div>
```

STEP – 10:

- Create file named style.css and type the code below.

CODE:

```
html {
    font-family: 'Roboto', sans-serif;
    color: white;
    font-size: 20px;
    width: 100%;
    height: 100%;
}
.Search {
```

```
    display: flex;
    justify-content: center;
    align-items: center;
    margin: 12px auto;
}
button {
    margin: 0.3rem;
    width: 1.7rem;
    height: 1.7rem;
    border-radius: 50%;
    background: #3b3d54;
    color: white;
    border: none;
    outline: none;
    transition: background 0.1s ease-in-out;
}
.Searchbar:hover {
    box-shadow: 2px 2px 5px rgba(0,0,0,0.7);
}
button:hover {
    background: #4c4f69;
    cursor: pointer;
    box-shadow: 2px 2px 5px rgba(0,0,0,0.7);
}
input {
    border: none;
    outline: none;
    padding: 0.2rem 1rem;
    border-radius: 15px;
    background: #3b3d54;
    color: white;
    font-family: inherit;
    font-size: 105%;
}
.Card {
    display: flex;
    margin: 12px 20rem;
    width: auto;
    height: auto;
}
```

1907160101

```
.Card-1 {
    display: flex;
    flex-direction: column;
    justify-content: space-between;
    border-radius: 25px;
    width: 24rem;
    height: 26rem;
    flex-grow: 1;
    padding: 0.5rem;
    position: relative;
    box-shadow: 3px 3px 20px rgba(0,0,0,0.7);
    text-shadow: #000 1px 0 25px;
    transition: transform 500ms ease;
}
.Card-1::after {
    content: "";
    border-radius: 25px;
    background: var(--background);
    background-position: center;
    background-repeat: no-repeat;
    filter: brightness(90%);
    opacity: 0.92;
    top: 0;
    left: 0;
    bottom: 0;
    right: 0;
    position: absolute;
      z-index: -1;
}
.Card-2 {
    border-radius: 25px;
    width: 24rem;
    height: 26rem;
    flex-grow: 1;
    padding: 0.5rem;
    transition: width 0.2s, height 0.2s;
    display: flex;
    flex-direction: column;
    justify-content: space-between;
    position: relative;
```

1807160101

```
    box-shadow: 3px 3px 20px rgba(0,0,0,0.7);
    transition: transform 500ms ease;
}
.Card-2::after {
    content:"";
    border-radius: 25px;
    background-color: #252836;
    opacity: 0.92;
    top: 0;
    left: 0;
    bottom: 0;
    right: 0;
    position: absolute;
    z-index: -1;
}
.Card-1:hover, .Card-2:hover {
    transform: scale(1.05);
   }
.Day {
    margin-bottom: 0px;
    padding-bottom: 0px;
    font-size: 2rem;
    margin-left: 15px;
}
.Date {
    margin-top: none;
      padding-top: none;
    margin-left: 15px;
   }
.Location {
    margin-top: 15px;
    margin-left: 5px;
    font-size: 22px;
}
.Card-1 i, .icon {
    margin-left: 15px;
    margin-bottom: 0px;
}
.Temp {
    margin-left: 15px;
```

180716010 01

```
    margin-top: 0px;
    padding-top: 0px;
    margin-bottom: 0px;
    font-size: 4rem;
}
.Description {
    font-size: 1.4rem;
    font-weight: bold;
    margin-left: 20px;
    margin-bottom: 10px;
}
table {
    margin: 1.2rem;
    margin-top: 2rem;
    margin-bottom: 2rem;
}
tr td:nth-child(1) {
    font-weight: bold;
    text-align: left;
}
tr td:nth-child(2) {
    text-align: right;
    width: 90%;
}
.Weekdays {
    display: flex;
    flex-direction: row;
    justify-content: space-evenly;
    border-radius: 1rem;
    background-color: #303445;
    width: auto;
    height: auto;
    margin: auto;
}
.Days {
    padding: 0.2rem;
    border-radius: 1rem;
    padding-top: 1rem;
    flex-direction: column;
    text-align: center;
```

190716010 01

```
    background-color: #303445;
    transition: background-color 300ms ease;
}
.Days:hover {
    background-color: whitesmoke;
    color: #303445;
    border-radius: 1rem;
    cursor: pointer;
}
.Days:hover img {
    filter: invert(1);
}
.Card.Loading {
    visibility: hidden;
    max-height: 20px;
    position: relative;
}
.Card.Loading::after {
    visibility: visible;
    content: "Loading...";
    color: white;
    position: absolute;
    top: 0;
}
```

style.css

```
1  html {
2      font-family: 'Roboto', sans-serif;
3      color: white;
4      font-size: 20px;
5      width: 100%;
6      height: 100%;
7  }
8  .Search {
9      display: flex;
10     justify-content: center;
11     align-items: center;
12     margin: 12px auto;
13 }
14 button {
15     margin: 0.3rem;
16     width: 1.7rem;
17     height: 1.7rem;
18     border-radius: 50%;
19     background: #3b3d54;
20     color: white;
21     border: none;
22     outline: none;
23     transition: background 0.1s ease-in-out;
```

2160101

STEP – 11:

- Create file named script.js and type the code below.

CODE:

```
let D = null;
let weather = {
   "apiKey": "dea886ab571fa9ba5defd3fe2cb73358",
   fetchWeather: function(city) {
        fetch("https://api.openweathermap.org/data/2.5/forecast?q="
        + city
        + "&units=metric&appid="
      + this.apiKey
      )
          .then((response) => response.json())
          .then((data) => {D = data; this.displayWeather(data, 1);});
   },
   displayWeather: function(data, DayNumber) {
      const cityname = data.city.name;
      const country = data.city.country;
      // DAY 1
      let date1 = data.list[0].dt_txt;
      const icon1 = data.list[0].weather[0].icon;
      const description1 = data.list[0].weather[0].description;
      const temp1 = data.list[0].main.temp;
      const humidity1 = data.list[0].main.humidity;
      const speed1 = Math.round((Number(data.list[0].wind.speed) * 3.6) * 100) / 100;
      const pop1 = data.list[0].pop;
      const main1 = data.list[0].weather[0].main;
      // DAY 2
      let date2 = data.list[7].dt_txt;
      const icon2 = data.list[7].weather[0].icon;
      const description2 = data.list[7].weather[0].description;
      const temp2 = data.list[7].main.temp;
      const humidity2 = data.list[7].main.humidity;
      const speed2 = Math.round((Number(data.list[7].wind.speed) * 3.6) * 100) / 100;
      const pop2 = data.list[7].pop;
      const main2 = data.list[7].weather[0].main;
      // DAY 3
      let date3 = data.list[15].dt_txt;
```

1807160101

```
        const icon3 = data.list[15].weather[0].icon;
        const description3 = data.list[15].weather[0].description;
        const temp3 = data.list[15].main.temp;
        const humidity3 = data.list[15].main.humidity;
        const speed3 = Math.round((Number(data.list[15].wind.speed) * 3.6) * 100) / 100;
        const pop3 = data.list[15].pop;
        const main3 = data.list[15].weather[0].main;
        // DAY 4
        let date4 = data.list[23].dt_txt;
        const icon4 = data.list[23].weather[0].icon;
        const description4 = data.list[23].weather[0].description;
        const temp4 = data.list[23].main.temp;
        const humidity4 = data.list[23].main.humidity;
        const speed4 = Math.round((Number(data.list[23].wind.speed) * 3.6) * 100) / 100;
        const pop4 = data.list[23].pop;
        const main4 = data.list[23].weather[0].main;

        let months  = ['', 'January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', '
September',
'October', 'November', 'December'];
        function dateList(date) {
            let myarr1 = date.split(" ");
            date = myarr1[0];
            let myArr2 = date.split("-");
            let year = myArr2[0]
            let month = months[Number(myArr2[1])];
            let d = myArr2[2];
            return [d, month, year];
        }
        let arr1 = dateList(date1);
        let d = new Date(`${arr1[0]} ${arr1[1]}, ${arr1[2]}`);
        let arr2 = dateList(date2);
        let dx = new Date(`${arr2[0]} ${arr2[1]}, ${arr2[2]}`);
        let arr3 = dateList(date3);
        let dy = new Date(`${arr3[0]} ${arr3[1]}, ${arr3[2]}`);
        let arr4 = dateList(date4);
        let dz = new Date(`${arr4[0]} ${arr4[1]}, ${arr4[2]}`);
        let weekday = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Sat
urday'];
        let day1 = weekday[d.getDay()];
```

180716010 01

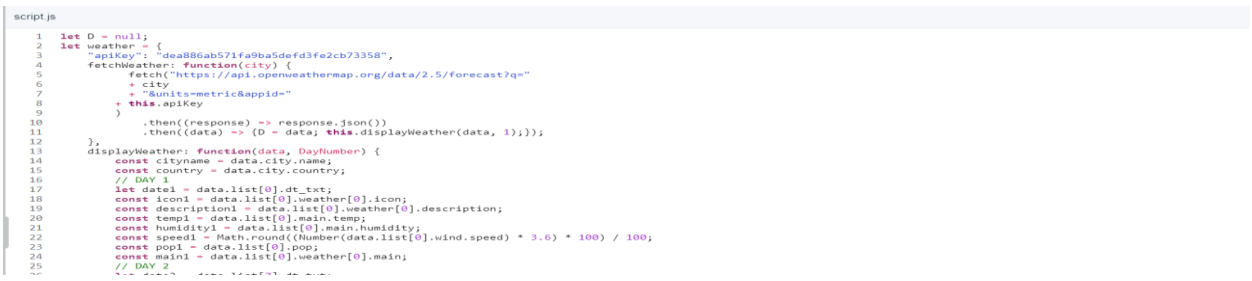
```
        let day2 = weekday[dx.getDay()];
        let day3 = weekday[dy.getDay()];
        let day4 = weekday[dz.getDay()];
        function modify(day, arr, icon, temp, desc, pop, humidity, speed) {
            document.querySelector(".Day").innerText = day;
                document.querySelector(".Date").innerText = `${arr[0]} ${arr[1]} ${arr[2]}`;
             document.querySelector(".Card-
1 .part2 img").src = "https://openweathermap.org/img/wn/" +
icon + ".png";
            document.querySelector(".Temp").innerText = Math.round(Number(temp)) + "°C
";
                document.querySelector(".Description").innerText = desc;
            document.querySelector("table tr:nth-child(1) td:nth-
child(2)").innerText = pop + " %";
            document.querySelector("table tr:nth-child(2) td:nth-
child(2)").innerText = humidity + " %";
            document.querySelector("table tr:nth-child(3) td:nth-
child(2)").innerText = speed + " km/h";
        }
        function CardBackground(main) {
            let style = document.querySelector('.Card-1').style;
            if (main == 'Rain') {
                style.setProperty('--
background', "url('https://source.unsplash.com/1600x900/?Rain')");
              } else if (main == 'Clouds') {
                style.setProperty('--
background', "url('https://source.unsplash.com/1600x900/?Clouds')");
              } else if (main == 'Clear') {
                style.setProperty('--
background', "url('https://source.unsplash.com/1600x900/?Sun')");
            } else {
                style.setProperty('--
background', "url('https://source.unsplash.com/1600x900/?" + main +"')");
                        }
        }
        switch(DayNumber) {
            case 1:
                modify(day1, arr1, icon1, temp1, description1, pop1, humidity1, speed1);
                CardBackground(main1);
                break;
```

1807160101

```
            case 2:
                modify(day2, arr2, icon2, temp2, description2, pop2, humidity2, speed2);
                CardBackground(main2);
                break;
            case 3:
                modify(day3, arr3, icon3, temp3, description3, pop3, humidity3, speed3);
                CardBackground(main3);
                break;
            case 4:
                modify(day4, arr4, icon4, temp4, description4, pop4, humidity4, speed4);
                CardBackground(main4);
                break;
        }
        // modifications common to all 4 days
        document.querySelector(".City").innerText = cityname + ", " + country;

        document.querySelector(".One img").src = "https://openweathermap.org/img/wn/" +
 icon1 +
".png";
        document.querySelector("#d1").innerText = day1.substr(0, 3);
        document.querySelector("#t1").innerText = Math.round(Number(temp1)) + "°C";

        document.querySelector(".Two img").src = "https://openweathermap.org/img/wn/"
+ icon2 +
".png";
        document.querySelector("#d2").innerText = day2.substr(0, 3);
        document.querySelector("#t2").innerText = Math.round(Number(temp2)) + "°C";

        document.querySelector(".Three img").src = "https://openweathermap.org/img/wn/"
+ icon3 +
".png";
        document.querySelector("#d3").innerText = day3.substr(0, 3);
        document.querySelector("#t3").innerText = Math.round(Number(temp3)) + "°C";

        document.querySelector(".Four img").src = "https://openweathermap.org/img/wn/"
+ icon4 +
".png";
        document.querySelector("#d4").innerText = day4.substr(0, 3);
        document.querySelector("#t4").innerText = Math.round(Number(temp4)) + "°C";
        document.querySelector(".Card").classList.remove("Loading");
```

1807160101

```
        document.body.style.backgroundImage = "url('https://source.unsplash.com/1600x900/?" +
cityname +"')"
    },
    search: function() {
        this.fetchWeather(document.querySelector(".Searchbar").value);
    },
};
document.querySelector(".Search button").addEventListener("click", function() { weather.search(); });
document.querySelector(".Searchbar").addEventListener("keyup", function(event) {
    if (event.key == "Enter") {
        weather.search();
    }
});

document.querySelector(".One").addEventListener("click", function() { weather.displayWeather(D , 1)
});

document.querySelector(".Two").addEventListener("click", function() { weather.displayWeather(D , 2)
});

document.querySelector(".Three").addEventListener("click", function() { weather.displayWeather(D , 3)
});

document.querySelector(".Four").addEventListener("click", function() { weather.displayWeather(D , 4)
});
weather.fetchWeather("delhi");
```

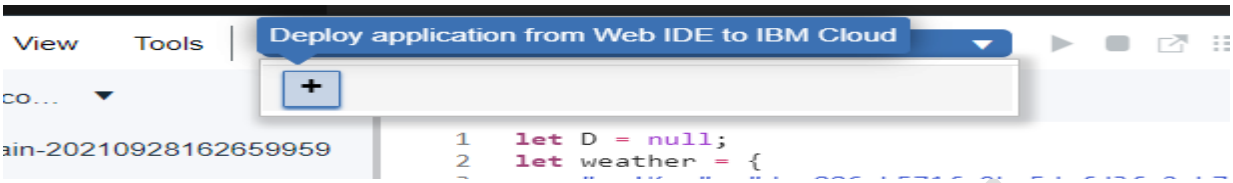


```
script.js
let D = null;
let weather = {
    "apiKey": "dea886ab571fa9ba5defd3fe2cb73358",
    fetchWeather: function(city) {
          fetch("https://api.openweathermap.org/data/2.5/forecast?q="
          + city
          + "&units=metric&appid="
        + this.apiKey
        )
            .then((response) => response.json())
            .then((data) => {D = data; this.displayWeather(data, 1);});
    },
    displayWeather: function(data, DayNumber) {
        const cityname = data.city.name;
        const country = data.city.country;
        // DAY 1
        let date1 = data.list[0].dt_txt;
        const icon1 = data.list[0].weather[0].icon;
        const description1 = data.list[0].weather[0].description;
        const temp1 = data.list[0].main.temp;
        const humidity1 = data.list[0].main.humidity;
        const speed1 = Math.round((Number(data.list[0].wind.speed) * 3.6) * 100) / 100;
        const pop1 = data.list[0].pop;
        const main1 = data.list[0].weather[0].main;
        // DAY 2
```

1807160101

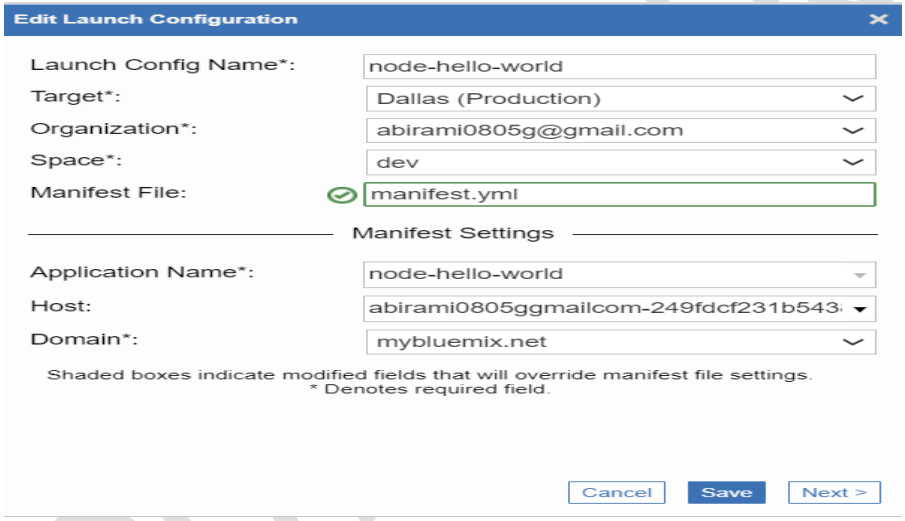
STEP – 12:

- Now click on the plus button to deploy application from web IDE to IBM cloud.



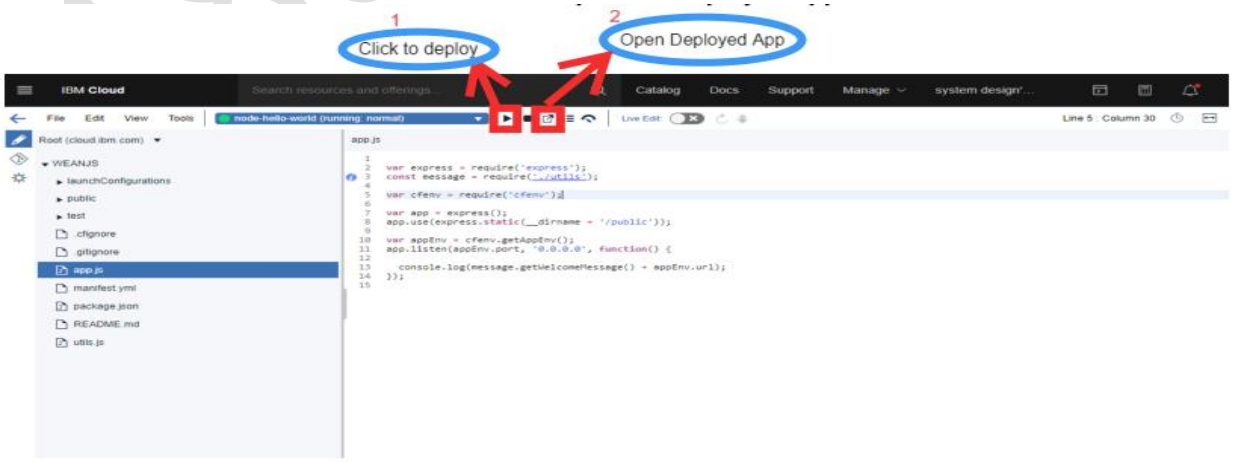
STEP – 13:

- Edit the launch configurations as shown and click save.



STEP – 14:

- First click to deploy the app then click the open deploy app option as shown.



STEP – 15:

- This will open the app in a new window and display the output.

RESULT:

Thus, DevOps using IBM cloud has been successfully implemented.

EX.NO: 9

DOCKERS USING UBUNTU

AIM:

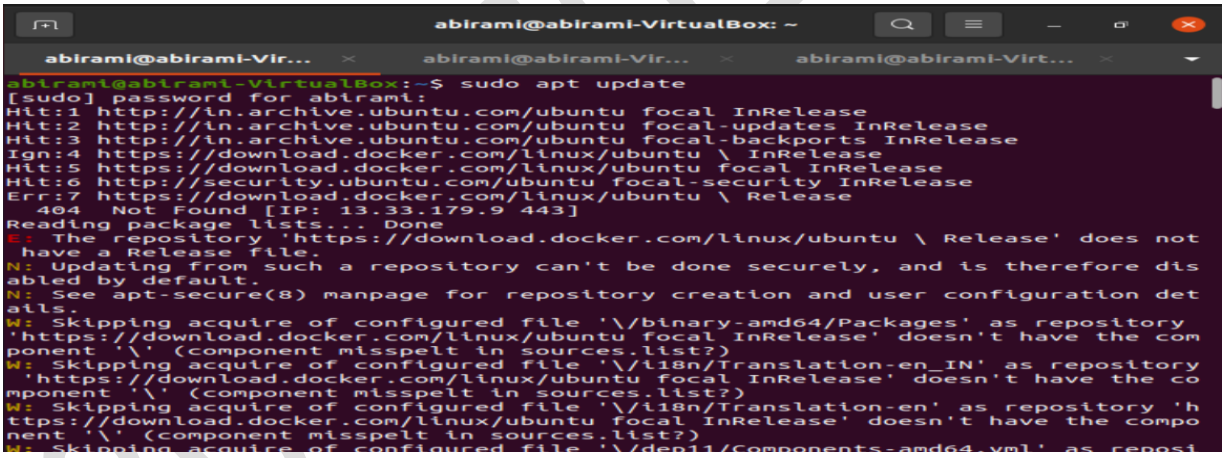
To implement Dockers in Ubuntu.

PROCEDURE:

STEP – 1:

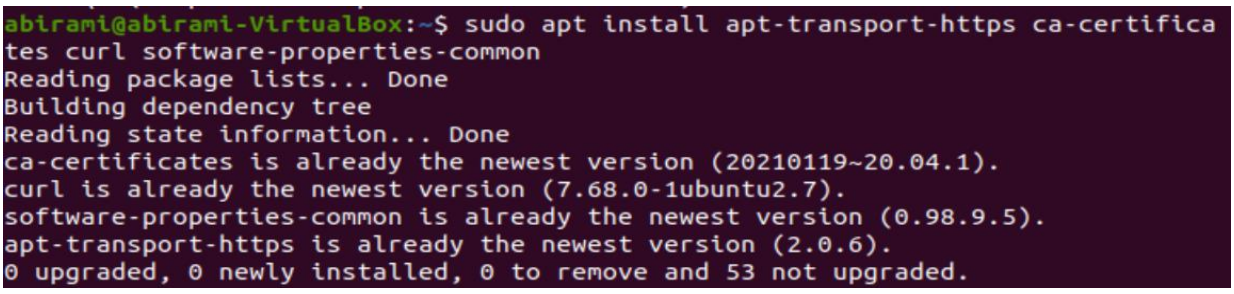
INSTALLING DOCKERS ON UBUNTU:

- Update the default packages and install the dependencies needed to install and run Docker.
- $ sudo apt update



STEP – 2:

- $ sudo apt install apt-transport-https ca-certificates curl software-properties-common



STEP – 3:

- Next add the GPG key for the docker repository.
- $ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add –

```
abirami@abirami-VirtualBox:~$ curl -fsSL https://download.docker.com/linux/ubun
tu/gpg | sudo apt-key add -
OK
```

STEP – 4:

- Now add the stable docker APT repository to your system's software repository list.
- $ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"

```
abirami@abirami-VirtualBox:~$ sudo add-apt-repository "deb [arch=amd64] https:/
/download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
Hit:1 http://in.archive.ubuntu.com/ubuntu focal InRelease
Hit:2 http://in.archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:3 http://in.archive.ubuntu.com/ubuntu focal-backports InRelease
Ign:4 https://download.docker.com/linux/ubuntu \ InRelease
Hit:5 https://download.docker.com/linux/ubuntu focal InRelease
Hit:6 http://security.ubuntu.com/ubuntu focal-security InRelease
Err:7 https://download.docker.com/linux/ubuntu \ Release
  404  Not Found [IP: 13.33.179.80 443]
Reading package lists... Done
E: The repository 'https://download.docker.com/linux/ubuntu \ Release' does not
 have a Release file.
N: Updating from such a repository can't be done securely, and is therefore dis
abled by default.
N: See apt-secure(8) manpage for repository creation and user configuration det
ails.
W: Skipping acquire of configured file '\/binary-amd64/Packages' as repository
'https://download.docker.com/linux/ubuntu focal InRelease' doesn't have the com
ponent '\' (component misspelt in sources.list?)
W: Skipping acquire of configured file '\/i18n/Translation-en_IN' as repository
 'https://download.docker.com/linux/ubuntu focal InRelease' doesn't have the co
mponent '\' (component misspelt in sources.list?)
```

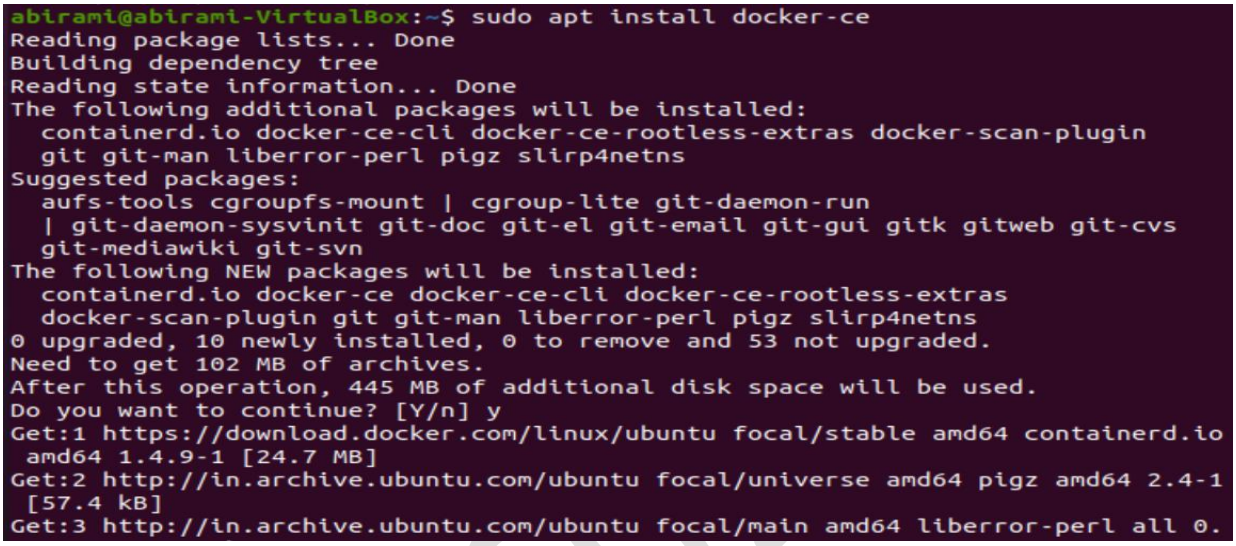
STEP – 5:

- The above step enabled the docker repository, you need to update your default apt packages.
- $ sudo apt update

```
abirami@abirami-VirtualBox:~$ sudo apt update
Hit:1 http://in.archive.ubuntu.com/ubuntu focal InRelease
Hit:2 http://in.archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:3 http://in.archive.ubuntu.com/ubuntu focal-backports InRelease
Ign:4 https://download.docker.com/linux/ubuntu \ InRelease
Hit:5 https://download.docker.com/linux/ubuntu focal InRelease
Hit:6 http://security.ubuntu.com/ubuntu focal-security InRelease
Err:7 https://download.docker.com/linux/ubuntu \ Release
  404  Not Found [IP: 13.33.179.80 443]
Reading package lists... Done
E: The repository 'https://download.docker.com/linux/ubuntu \ Release' does not
 have a Release file.
N: Updating from such a repository can't be done securely, and is therefore dis
abled by default.
N: See apt-secure(8) manpage for repository creation and user configuration det
ails.
W: Skipping acquire of configured file '\/binary-amd64/Packages' as repository
'https://download.docker.com/linux/ubuntu focal InRelease' doesn't have the com
ponent '\' (component misspelt in sources.list?)
W: Skipping acquire of configured file '\/i18n/Translation-en' as repository 'h
```

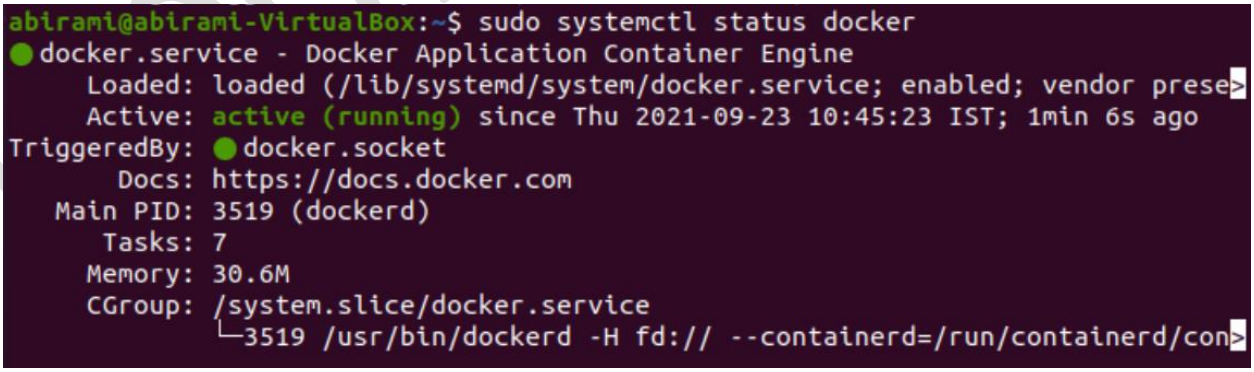
STEP – 6:

- Now install the docker community edition.
- $ sudo apt install docker-ce



STEP – 7:

- After the installation is complete, docker service will start automatically. We can verify that by checking the status of our docker service.
- $ sudo systemctl status docker



- Docker is installed and running.

STEP – 8:

CREATE DOCKER IMAGE:

- To create a docker image, create a directory and place the executable jar created by maven in that directory.

```
abirami@abirami-VirtualBox:~/docker-exercise$ ls -l
total 480
-rw-rw-r-- 1 abirami abirami 485165 Sep 23 18:10 calcProject-0.2-jar-with-depend
encies.jar
-rwxrwxrwx 1 abirami abirami    279 Sep 23 13:54 Dockerfile
```

STEP – 9:

- Now, create a file named “Dockerfile” in the same directory and write the code as shown.

```
abirami@abirami-VirtualBox:~/docker-exercise$ cat Dockerfile
#Dockerfile for running java application
FROM anapsix/alpine-java
MAINTAINER devopsuser@gmail.com
COPY calcProject-0.2-jar-with-dependencies.jar /home/calcProject-0.2-jar-with-de
pendencies.jar
CMD ["java","-jar","/home/calcProject-0.2-jar-with-dependencies.jar","2","3","1"
]
```

STEP – 10:

- Now, run the below step to build the docker image.
- $ sudo docker build -t devops_calc:1.0 .

```
abirami@abirami-VirtualBox:~/docker-exercise$ sudo docker build -t devops_calc:1
.0 .
Sending build context to Docker daemon  487.9kB
Step 1/4 : FROM anapsix/alpine-java
 ---> c45785c254c5
Step 2/4 : MAINTAINER devopsuser@gmail.com
 ---> Using cache
 ---> 9c3b06daff03
Step 3/4 : COPY calcProject-0.2-jar-with-dependencies.jar /home/calcProject-0.2-
jar-with-dependencies.jar
 ---> 8369bce78fb6
Step 4/4 : CMD ["java","-jar","/home/calcProject-0.2-jar-with-dependencies.jar",
"2","3","1"]
 ---> Running in 4a9d90d213e8
Removing intermediate container 4a9d90d213e8
 ---> a233507a8eec
Successfully built a233507a8eec
Successfully tagged devops_calc:1.0
```

STEP – 11:

- You can check that your image is created using the below command.
- $ sudo docker images

```
abirami@abirami-VirtualBox:~/docker-exercise$ sudo docker images
REPOSITORY            TAG       IMAGE ID       CREATED          SIZE
devops_calc           1.0       a233507a8eec   31 seconds ago   126MB
anapsix/alpine-java   latest    c45785c254c5   2 years ago      126MB
```

STEP – 12:

- Now, run the build image by issuing the below command.
- $ sudo docker run devops_calc:1.0

```
abirami@abirami-VirtualBox:~/docker-exercise$ sudo docker run devops_calc:1.0
0 [main] INFO com.devops.calcProject.CalcMain  - First number: 2
1 [main] INFO com.devops.calcProject.CalcMain  - Second number: 3
1 [main] INFO com.devops.calcProject.CalcMain  -
1: Addition.
2: Subtraction.
3: Multiplication.
4:  Divide.
5: Remainder.
6: Exit.
2 [main] INFO com.devops.calcProject.CalcMain  -
Your choice: 1
7 [main] INFO com.devops.calcProject.CalcMain  - Result is: 5.0
```

- Dockerized application has executed successfully.

RESULT:

Thus, Dockers has been successfully implemented in Ubuntu.

EX.NO: 10

CALLBACKS

AIM:

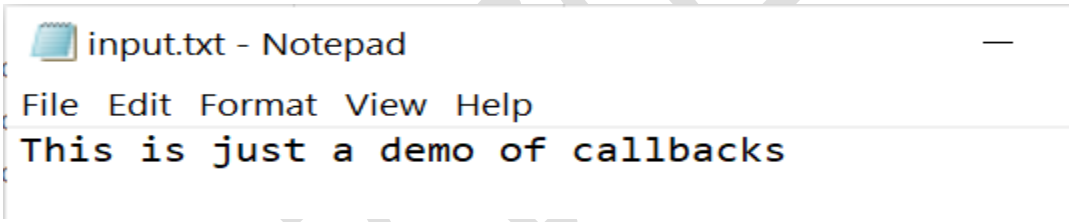
To implement callbacks (blocking and non - blocking calls) using Node JS.

PROCEDURE:

BLOCKING CALL:

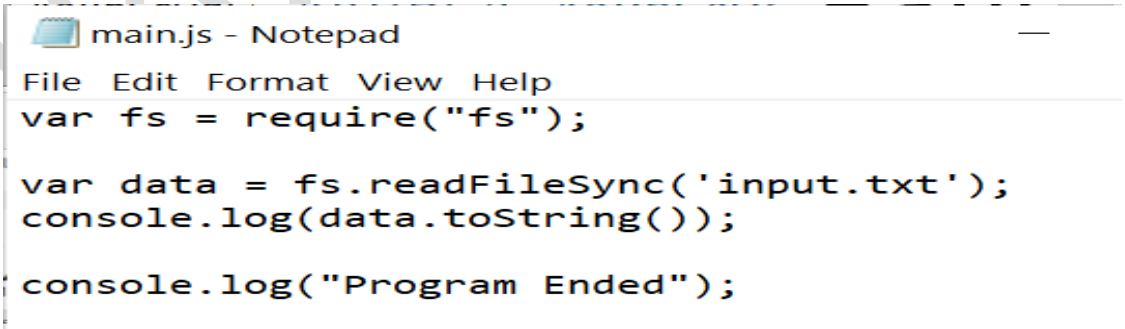
STEP – 1:

- Install node js.
- Open notepad and type the following code and save it as ‘input.txt’.
- Here the file is saved in any preferred location.



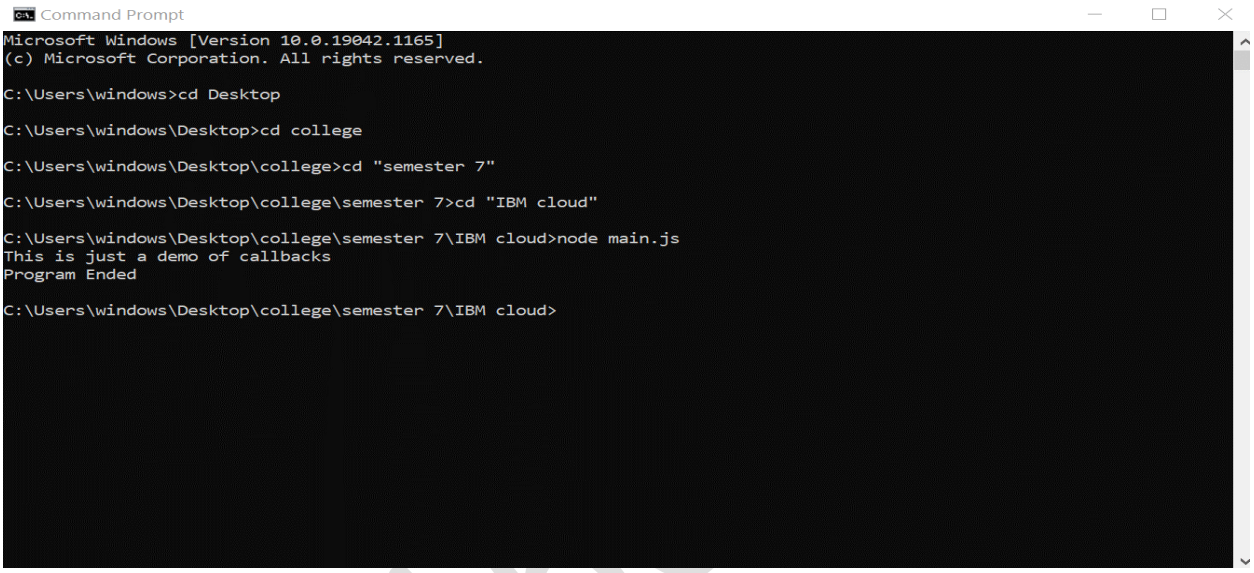
STEP – 2:

- Open another file and type the following code and save it as ‘main.js’.



STEP – 3:

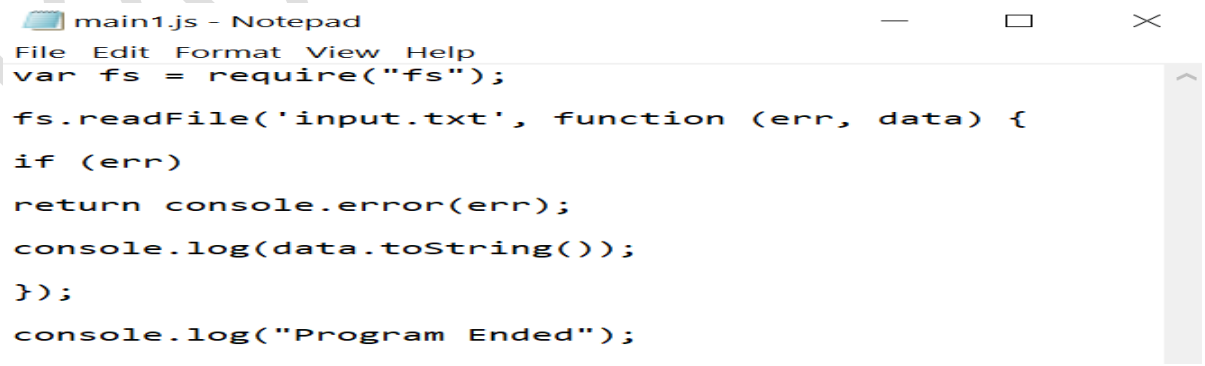
- Now open command Prompt and navigate to the file location.
- Then run the command “node filemname.js”.
- program blocks until it reads the file and then only it proceeds to end the program.



NON - BLOCKING CALL:

STEP – 1:

- Open notepad and type the following code and save it as ‘main1.js’.
- Here the file is saved in any preferred location.



STEP – 2:

- Now open command Prompt and navigate to the file location.
- Then run the command “node filemname.js”.
- program does not wait for file reading and proceeds to print "Program Ended" and at the same time, the program without blocking continues reading the file.

```
C:\Users\windows\Desktop\college\semester 7\IBM cloud>node main1.js
Program Ended
This is just a demo of callbacks
```

18007160100

RESULT:

Thus, Callbacks for blocking and non-blocking calls has been successfully implemented.

EX.NO: 11

EXPRESS JS

AIM:

To implement Routing /URL Building /Middleware/Templating/Serving static files/Form data using Express Js.

PROCEDURE:

STEP – 1:

INSTALL EXPRESS JS:

- Create a new directory called hello-world in Desktop and navigate to hello-world directory.

```
C:\Users\windows>cd Desktop

C:\Users\windows\Desktop>mkdir hello-world

C:\Users\windows\Desktop>cd hello-world
```

STEP – 2:

- Create package.json using “npm.init” code in terminal.
- Keep pressing enter key. Enter your name in the author filed and type “yes”.

```
C:\Users\windows\Desktop\hello-world>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (hello-world)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author: abirami
license: (ISC)
About to write to C:\Users\windows\Desktop\hello-world\package.json:

{
  "name": "hello-world",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "abirami",
  "license": "ISC"
}


Is this OK? (yes) yes
```

STEP – 3:

- Now install express and check if its install correctly as shown.

```
C:\Users\windows\Desktop\hello-world>npm install express --save
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN hello-world@1.0.0 No description
npm WARN hello-world@1.0.0 No repository field.

+ express@4.17.1
added 50 packages from 37 contributors and audited 50 packages in 10.612s
found 0 vulnerabilities


C:\Users\windows\Desktop\hello-world>dir node_modules
 Volume in drive C is OS
 Volume Serial Number is ECC8-C6D6
```

STEP – 4:

- Install "nodemon" tool.

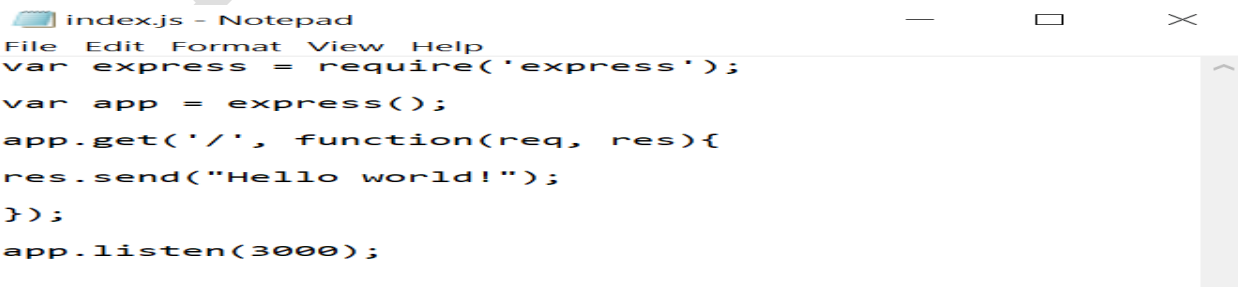
```
C:\Users\windows\Desktop\hello-world>npm install -g nodemon
C:\Users\windows\AppData\Roaming\npm\nodemon -> C:\Users\windows\AppData\Roaming\npm\node_modules\nodemon\bin\nodemon.js

> nodemon@2.0.13 postinstall C:\Users\windows\AppData\Roaming\npm\node_modules\nodemon
> node bin/postinstall || exit 0

Love nodemon? You can now support the project via the open collective:
 > https://opencollective.com/nodemon/donate

npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@~2.3.2 (node_modules\nodemon\node_modules\chokidar\node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.3.2: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})

+ nodemon@2.0.13
added 125 packages from 53 contributors in 22.597s
```

WORK IN EXPRESS JS:

HELLO WORLD:

STEP – 5:

- Create a file called "index.js" inside the "hello-world" directory and type the following code.



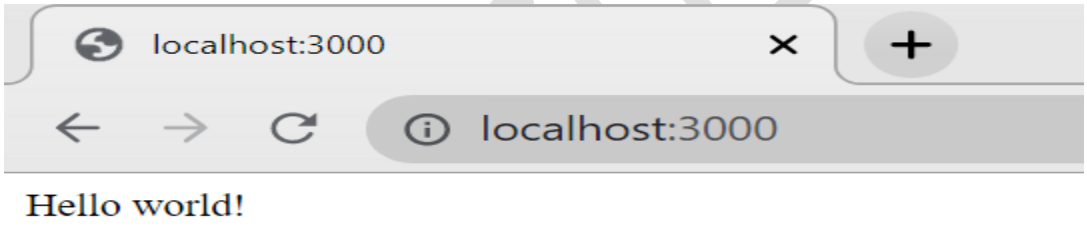
STEP – 6:

- Go to the terminal and type “nodemon index.js” to start the server.

```
C:\Users\windows\Desktop\hello-world>nodemon index.js
[nodemon] 2.0.13
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index.js`
```

STEP – 7:

- Now open a browser window and type “localhost:3000/”.
- This will display “hello world” message as shown.



ROUTING:

STEP – 8:

- Create a new file called “routing.js” in the “hello-world” directory and type the code below.

routing.js - Notepad

File Edit Format View Help

```
var express = require('express');

var app = express();

app.get('/hello', function(req, res){

res.send("Hello World!");

});
app.listen(3000);
```

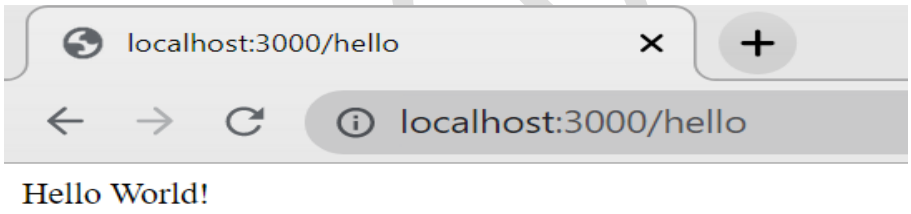
STEP – 9:

- Go to the terminal and type “nodemon routing.js” to start the server.

```
C:\Users\windows\Desktop\hello-world>nodemon routing.js
[nodemon] 2.0.13
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node routing.js`
```

STEP – 10:

- Now open a browser window and type “http://localhost:3000/”.
- This will display “hello world” message as shown.



CALLING POST METHOD:

STEP – 11:

- Create a new file called “routing1.js” in the “hello-world” directory and type the code below.

routing1.js - Notepad

File Edit Format View Help

```
var express = require('express');
var app = express();
app.get('/hello', function(req, res){
res.send("Hello World!");
});
app.post('/hello', function(req, res){
res.send("You just called the post method at '/hello'
});
app.listen(3000);
```

STEP – 12:

- Go to the terminal and type “nodemon routing1.js” to start the server.

```
C:\Users\windows\Desktop\hello-world>nodemon routing1.js
[nodemon] 2.0.13
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node routing1.js`
```

STEP – 13:

- Using the curl command in a different terminal, the post method is being called.

```
C:\Users\windows>curl -X POST "http://localhost:3000/hello"
You just called the post method at '/hello'!
```

ROUTERS:

STEP – 14:

- Type the following code in “things.js” inside “hello-world” directory.

things.js - Notepad

File Edit Format View Help

```
var express = require('express');

var router = express.Router();

router.get('/', function(req, res){

res.send('GET route on things.');

});

router.post('/', function(req, res){

res.send('POST route on things.');

});

//export this router to use in our index.js

module.exports = router;
```

STEP – 15:

- Create “routers.js” inside “hello-world” directory and type the following code.

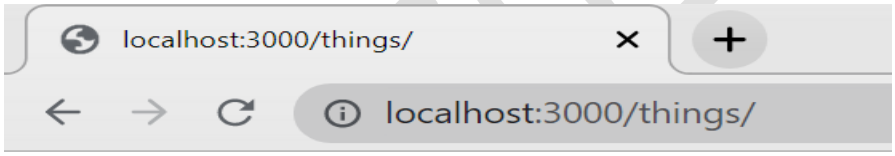
routers.js - Notepad

File Edit Format View Help

```
var express = require('Express');

var app = express();

var things = require('./things.js');

//both index.js and things.js should be in same direc

app.use('/things', things);

app.listen(3000);
```

STEP – 16:

- Open terminal and type “nodemon routers.js” to start the server.
- Now open a browser window and type “http://localhost:3000/things/”.



GET route on things.

URL BUILDING:

STEP – 17:

- Create “url_building.js” inside “hello-world” directory and type the following code.

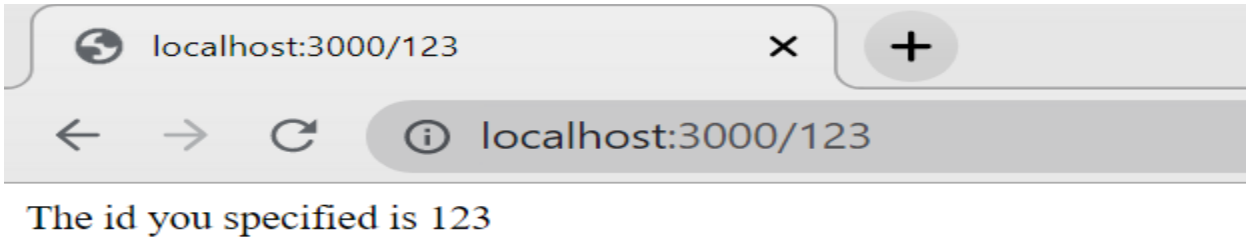
url_building.js - Notepad

File Edit Format View Help

```
var express = require('express');

var app = express();

app.get('/:id', function(req, res){

res.send('The id you specified is ' + req.params.id);

});

app.listen(3000);
```

STEP – 18:

- Open terminal and type “nodemon url_building.js” to start the server.
- Now open a browser window and type “http://localhost:3000/123”



COMPLEX METHOD:

STEP – 19:

- Create “url_building.js” inside “hello-world” directory and type the following code.

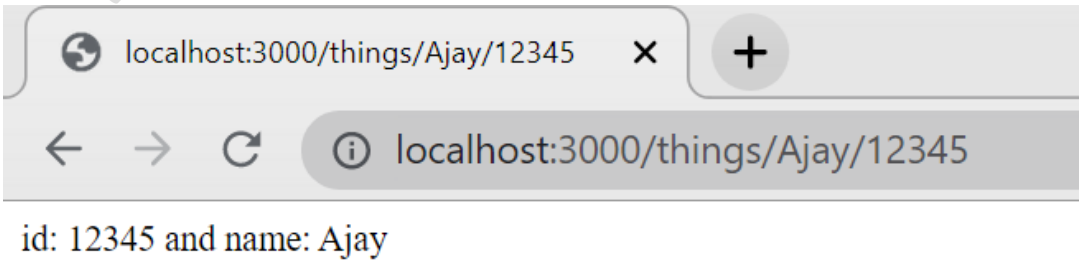
url_building.js - Notepad

File Edit Format View Help

```
var express = require('express');

var app = express();

app.get('/things/:name/:id', function(req, res) {

res.send('id: ' + req.params.id + ' and name: ' + req.params.name);

});

app.listen(3000);
```

STEP – 20:

- Open terminal and type “nodemon url_building.js” to start the server.
- Now open a browser window and type “http://localhost:3000/things/Ajay/12345”



MIDDLEWARE:

STEP – 21:

- Create “middleware.js” inside “hello-world” directory and type the following code.

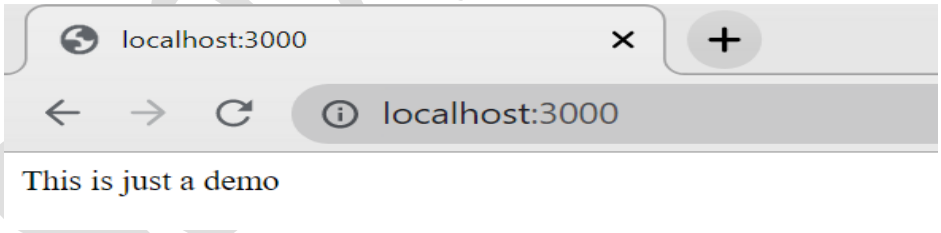
middleware.js - Notepad

File Edit Format View Help

```
var express = require('express');

var app = express();

app.use(function(req, res, next){

console.log("A new request received at " + Date.now());

next();

});

app.get('/', function(req, res){

res.send('This is just a demo');

});

app.listen(3000);
```

STEP – 22:

- Open terminal and type “nodemon middleware.js” to start the server.
- Now open a browser window and type “http://localhost:3000”



STEP – 23:

- The above middleware is called for every request on the server. So after every request, we will get the following message in the console.

```
[nodemon] starting `node middleware.js`
A new request received at 1632677255525
```

STEP – 24:

- To restrict it to a specific route (and all its subroutes), provide that route as the first argument of app.use().
- Create “middleware.js” inside “hello-world” directory and type the following code.

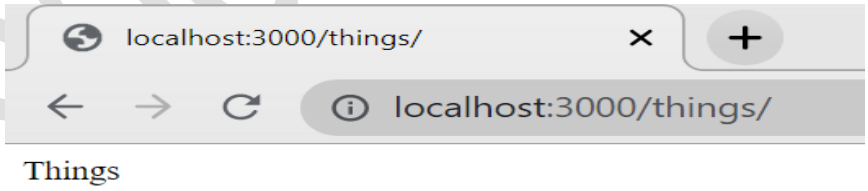
middleware.js - Notepad

File Edit Format View Help

```
var express = require('express');

var app = express();

//Middleware function to log request protocol

app.use('/things', function(req, res, next){

console.log("A request for things received at " + Date.now());

next();

});

// Route handler that sends the response

app.get('/things', function(req, res){

res.send('Things');

});

app.listen(3000);
```

STEP – 25:

- Open terminal and type “nodemon middleware.js” to start the server.
- Now open a browser window and type “http://localhost:3000”



STEP – 26:

- Now whenever you request any subroute of '/things', only then it will log the time.

```
[nodemon] starting `node middleware.js`
A new request received at 1632677255525
[nodemon] restarting due to changes...
[nodemon] starting `node middleware.js`
A request for things received at 1632677330905
```

ORDER OF MIDDLEWARE CALLS:

STEP – 27:

- Create a file named "middleware.js" inside "hello-world" directory and type the code shown.

middlewarejs - Notepad
File Edit Format View Help

```
var express = require('express');
var app = express();
//First middleware before response is sent
app.use(function(req, res, next){
console.log("Start");
next();
});
//Route handler
app.get('/', function(req, res, next){
res.send("Middle");
next();
});
app.use('/', function(req, res){
console.log('End');
});
app.listen(3000);
```

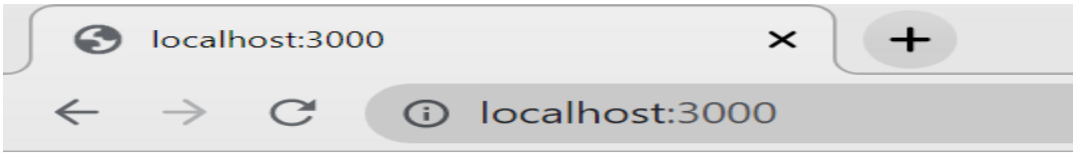
STEP – 28:

- Go to terminal and type "nodemon middleware.js" to start the server.

```
C:\Users\windows\Desktop\hello-world>nodemon middleware.js
[nodemon] 2.0.13
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node middleware.js`
Start
End
```

STEP – 29:

- When we visit 'https://localhost:3000/' after running this code, we receive the response as Middle and on our console
  - Start
  - End



Middle

THIRD PARTY MIDDLEWARE:

STEP – 30:

- Install body parser using the command shown below.

```
C:\Users\windows\Desktop\hello-world>npm install body-parser --save
npm WARN hello-world@1.0.0 No description
npm WARN hello-world@1.0.0 No repository field.

+ body-parser@1.19.0
updated 1 package and audited 51 packages in 2.398s
found 0 vulnerabilities
```

STEP – 31:

- Now create 'index.js' file inside 'hello-world' directory and type the following code.

index.js - Notepad

File Edit Format View Help

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

const bodyParser = require('body-parser');

const app = express();

app.use(bodyParser.urlencoded({ extended: true }));

app.post('/post-test', (req, res) => {

console.log('Got body:', req.body);

res.sendStatus(200);

});

app.listen(8080, () => console.log(`Started server at
http://localhost:8080!`));
```

STEP – 32:

- Now go to the terminal and type "nodemon index" to start the server.

```
C:\Users\windows\Desktop\hello-world>nodemon index
[nodemon] 2.0.13
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index index.js`
Started server at
http://localhost:8080!
```

STEP – 33:

- In another terminal, go to 'hello-world' location and type :
- curl -d "username=scott&password=secret&website=stackabuse.com" -X POST http://localhost:8080/post-test

```
C:\Users\windows\Desktop\hello-world>curl -d "username=scott&password=secret&website=stackabuse.com" -X POST http://localhost:8080/post-test
OK
```

STEP – 34:

- open the previous command line to see the post request output as shown.

```
C:\Users\windows\Desktop\hello-world>nodemon index
[nodemon] 2.0.13
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index index.js`
Started server at
http://localhost:8080!
Got body: { username: 'scott', password: 'secret', website: 'stackabuse.com' }
```

TEMPLATING:

STEP – 35:

- Install pug using ": npm - -save pug" command.
- Create "index.js" file in the "hello-world" directory and type the code below.

index.js - Notepad

File Edit Format View Help

```
var express = require('express');

var app = express();

app.set('view engine', 'pug');

app.set('views','./views');

app.get('/first_template', function(req, res){

res.render('first_view');

});

app.listen(3000);
```

STEP – 36:

- Create a new folder named “views” inside ‘hello-world’ directory.
- Create a new file “first_view.pug” inside the “views” directory and type the code below.

first_view.pug

```
doctype html

html

head

title = "Hello Pug"

body

p.greetings#people Hello World!

```

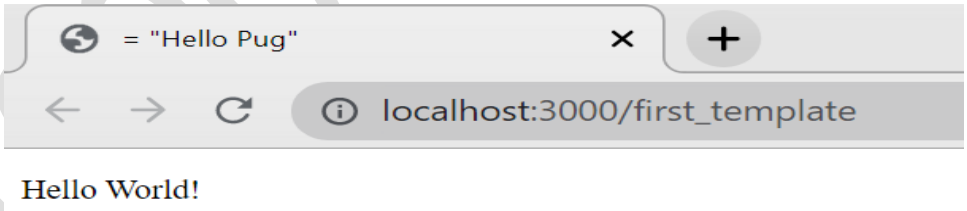
STEP – 37:

- Go to terminal and type “nodemon index” to start the server.

```
C:\Users\windows\Desktop\hello-world>nodemon index
[nodemon] 2.0.13
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index index.js`
```

STEP – 38:

- Go to the browser and type https://localhost:3000/first_template to view the output.



PASSING VALUES TO TEMPLATES:

STEP – 39:

- Create “index.js” file in the “hello-world” directory and type the code below.

index.js - Notepad

File Edit Format View Help

```
app.set('view engine', 'pug');
app.set('views','./views');
app.get('/dynamic_view', function(req, res){
res.render('dynamic', {
name: "Demo",
url:"http://www.demo.com"
});
});
app.listen(3000);
```

STEP – 40:

- Create a new file “dynamic.pug” inside the “views” directory and type the code below.

dynamic.pug - Notepad

File Edit Format View Help

```
html
head
title=name
body
h1=name
a(href = url) URL
```

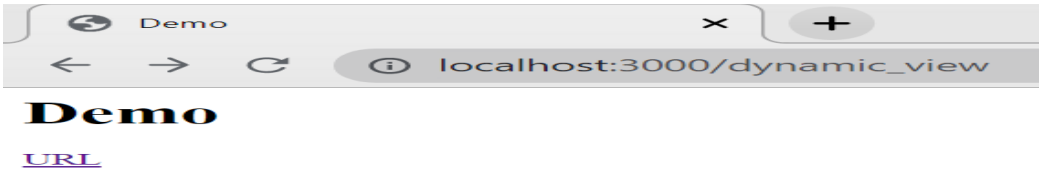
STEP – 41:

- Go to terminal and type “nodemon index” to start the server.

```
C:\Users\windows\Desktop\hello-world>nodemon index
[nodemon] 2.0.13
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index index.js`
```

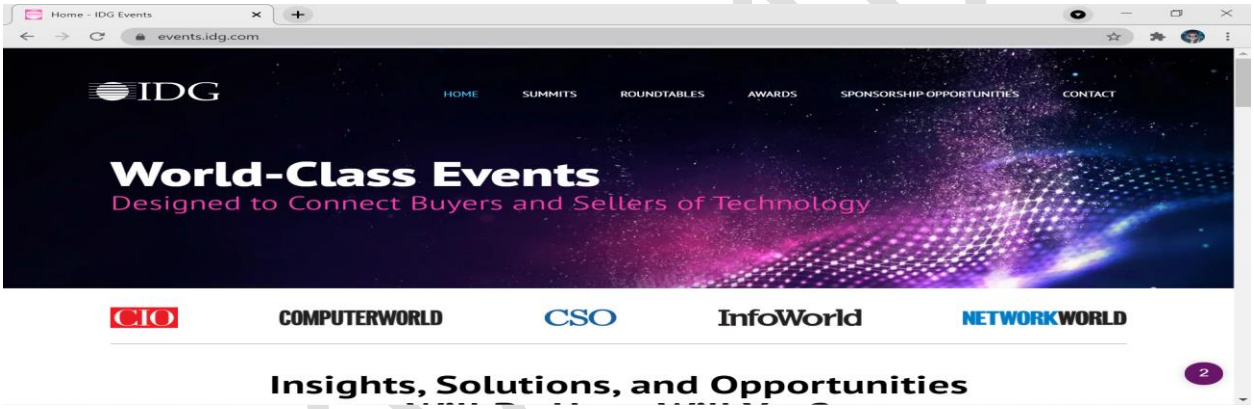
STEP – 42:

- Go to the browser and type https://localhost:3000/dynamic_view to view the output.
- Click on the URL.



STEP – 43:

- This will redirect to another webpage as shown.



STEP – 44:

- Another way to pass values to templates is shown below.
- Create "index.js" file in the "hello-world" directory and type the code below.

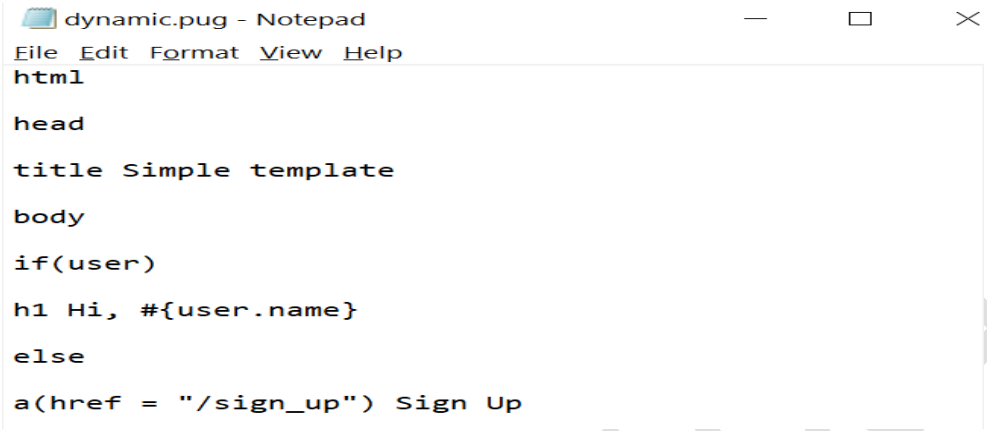
index.js - Notepad

File Edit Format View Help

```
var app = express();

app.set('view engine', 'pug');

app.set('views','./views');

app.get('/dynamic_view', function(req, res){

res.render('dynamic', {

user: {name: "Ayush", age: "20"}

});

});

app.listen(3000);
```

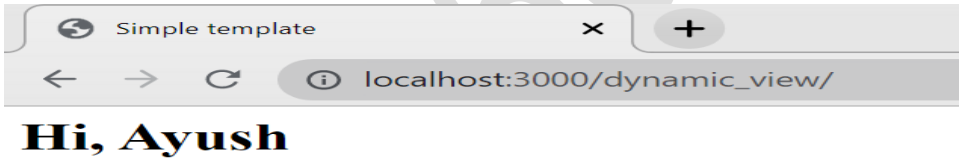
STEP – 45:

- Create a new file “dynamic.pug” inside the “views” directory and type the code below.



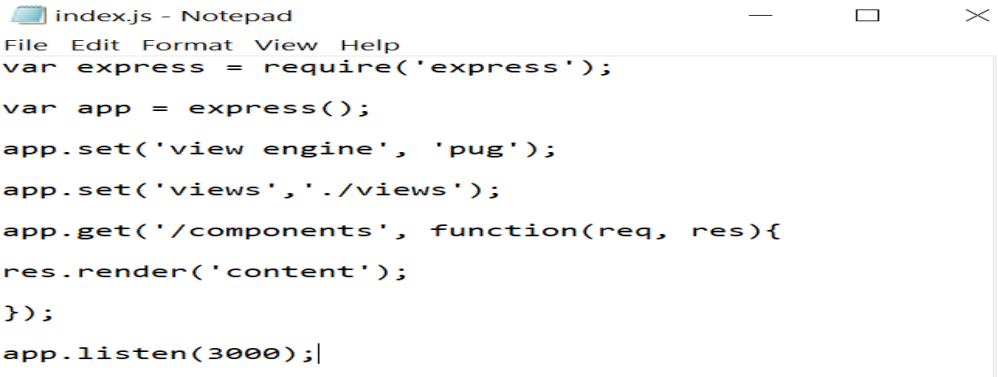
STEP – 46:

- Go to the browser and type https://localhost:3000/dynamic_view to view the output.



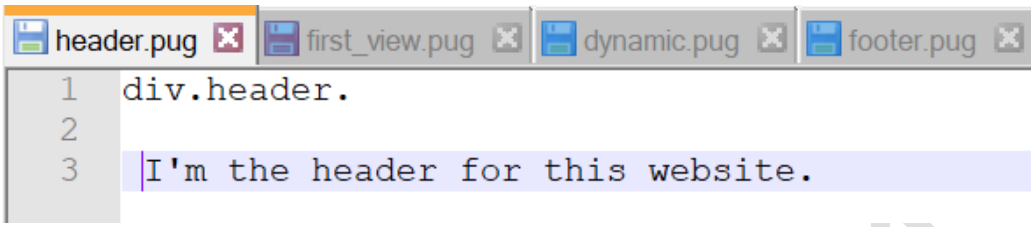
STEP – 47:

- One more way to pass values to templates is shown below.
- Create “index.js” file in the “hello-world” directory and type the code below.



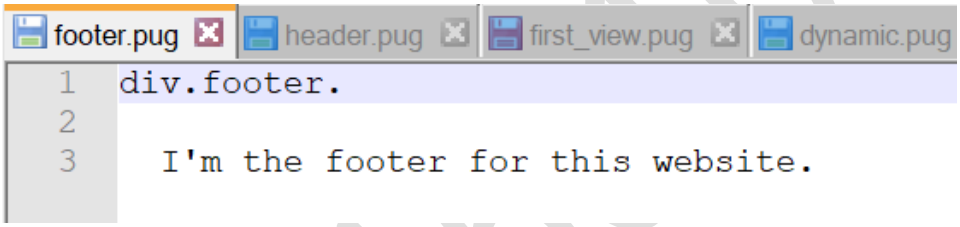
STEP – 48:

- Create a new file “header.pug” inside the “views” directory and type the code below.



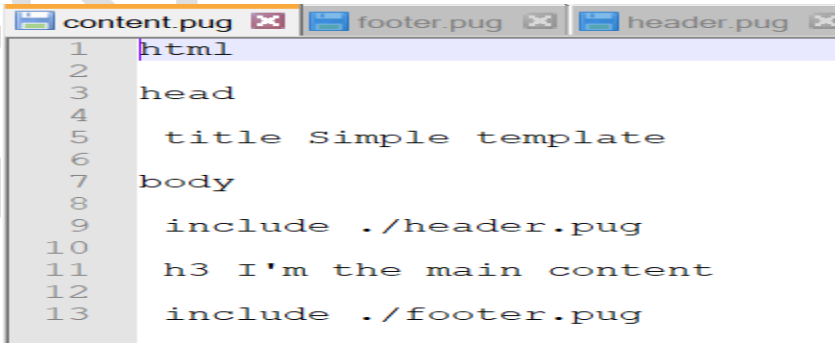
STEP – 49:

- Create a new file “footer.pug” inside the “views” directory and type the code below.



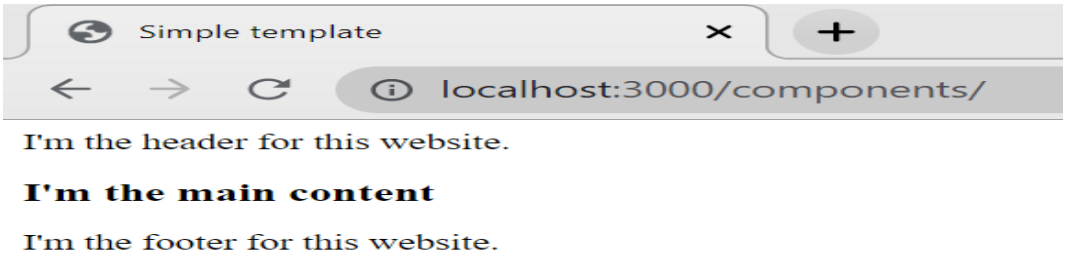
STEP – 50:

- Create a new file “content.pug” inside the “views” directory and type the code below.



STEP – 51:

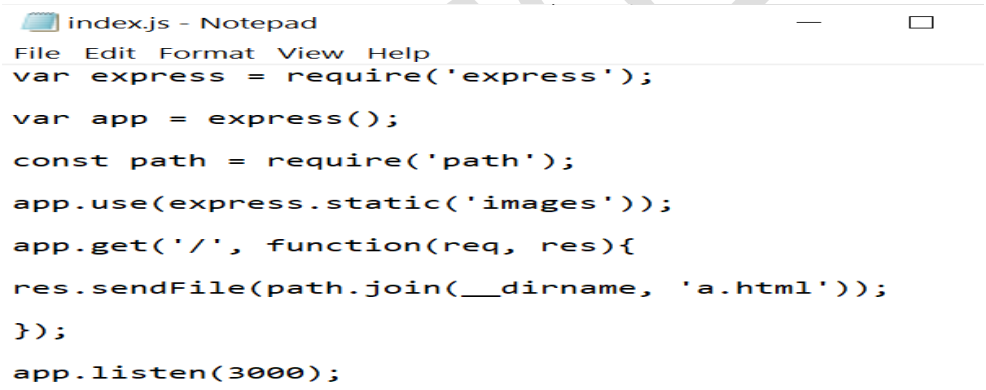
- Go to the browser and type https://localhost:3000/components/ to view the output.



SERVING STATIC FILES:

STEP – 52:

- Create "index.js" file in the "hello-world" directory and type the code below.



STEP – 53:

- Create a new directory in "hello-world" called "images" and paste any image named "test.png" inside it.

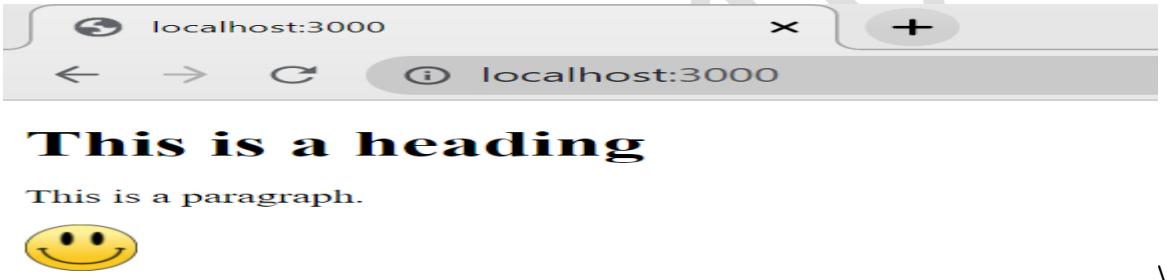
STEP – 54:

- Create “a.html” file in the “hello-world” directory and type the code below.

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
<img src="test.png" alt="Smiley face" height="42" width="42">
</body>
</html>
```

STEP – 55:

- Go to the browser and type https://localhost:3000/ to view the output.



FORM DATA:

STEP – 56:

- Create a new directory in “hello-world” called “public” .
- Create “login.html” file in the “public” directory and type the code below.

login.html - Notepad

File Edit Format View Help

```
<form method="post" action="/handler">
<input type="text" name="username" /><br />
<input type="password" name="password" /><br />
<input type="submit" />
</form>
```

STEP – 57:

- Create “index.js” file in the “hello-world” directory and type the code below.

index.js - Notepad
File Edit Format View Help

```
var path = require('path');
var express = require('express');
var app = express();
app.use(express.urlencoded({ extended: true }));
app.use(express.static(path.join(__dirname, 'public')));
app.post('/handler', function (req, res) {
console.log(req.body);
res.write(req.body.username);
res.write(req.body.password);
res.send(); // or res.end
});
app.listen(3000);
```

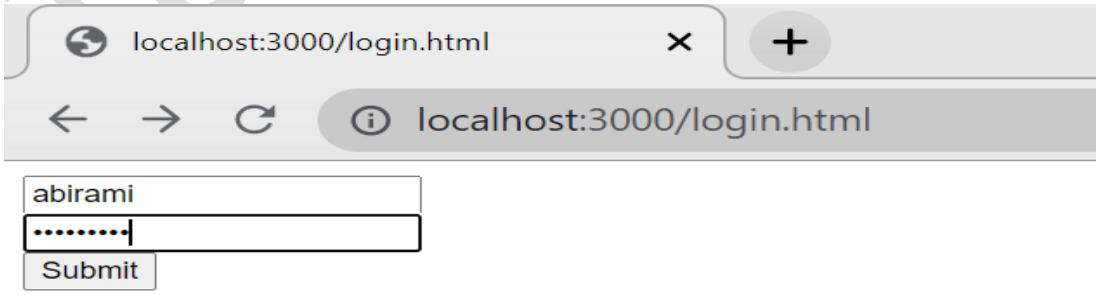
STEP – 58:

- Go to terminal and type “nodemon index” to start the server.

```
C:\Users\windows\Desktop\hello-world>nodemon index
[nodemon] 2.0.13
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index index.js`
```

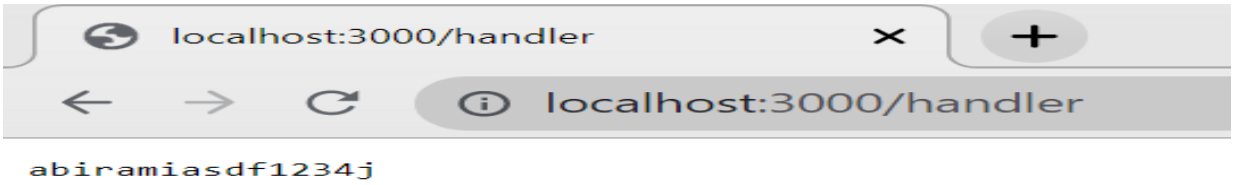
STEP – 59:

- Go to the browser and type https://localhost:3000/login.html to view the login form.



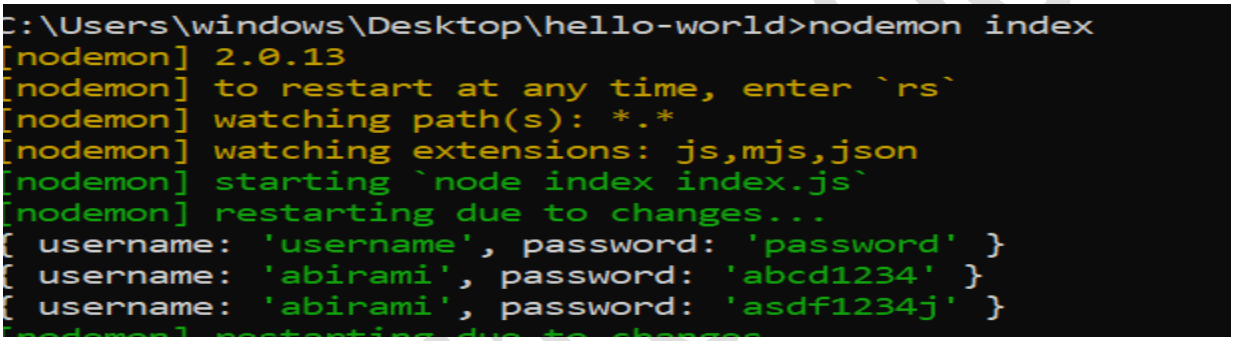
STEP – 60:

- Once you hit submit, the username and password gets displayed as shown.



STEP – 61:

- In the terminal the entered username and password is displayed.



RESULT:

Thus, Routing /URL Building /Middleware/Templating/Serving static files/Form data has been successfully executes using Express Js.