Node JS:

- Install Node js on local VM.
- Create 2 API's running on ports 6080 and 7080 with messages "Hello Node JS" and "Node JS installed successfully" respectively.
- Install pm2 tool and create 4 clusters of bothe Node's.
- Delete all 4 clusters one-by-one

⇒ Node js and node package manager (npm) are installed on local VM with commands:

Sudo apt install nodejs

Sudo apt install npm

```
Setting up nodejs-doc (10.19.0~dfsg-3ubuntu1) ...
Setting up nodejs (10.19.0~dfsg-3ubuntu1) ...
update-alternatives: using /usr/bin/nodejs to provide /usr/bin/js (js) in auto mode
Processing triggers for libc-bin (2.31-0ubuntu9.2) ...
Processing triggers for man-db (2.9.1-1) ...
bijay@batman:/usr/local$ node -v
v10.19.0
bijay@batman:/usr/local$ npm

Command 'npm' not found, but can be installed with:

sudo apt install npm

bijay@batman:/usr/local$ sudo apt install npm
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
```

Then we created a directory named Reactnode for this assignment,

Mkdir Reactnode

Cd Reactnode

Inside the directory, a file, index.js is created with sample code,

Nano index.js

By using the command '***cat index.js***', we can view the content of the index.js file.

```
root@batman:/home/bijay/Reactnode# cat index.js
const http = require('http');
const hostname = '192.168.1.67';
const port = 6080;
const server = http.createServer((req, res) =>
        {
                res.statusCode = 200;
                res.setHeader('Content-Type', 'text/plain');
                res.end('Hello Node JS \n');
        });

server.listen(port, hostname, () => {
        console.log(`Server running at http://${hostname}:${port}/`);
        });
```

When running the index.js file with node command:

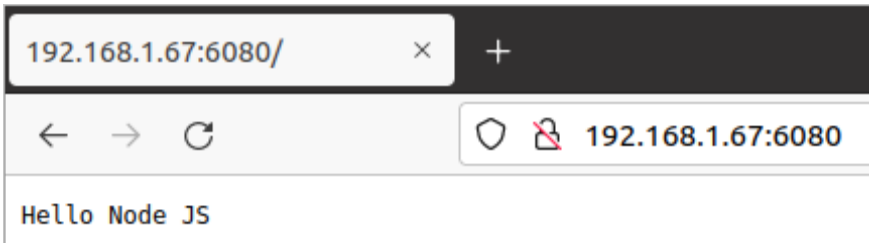
Node index.js

```
root@batman:/home/bijay/Reactnode# nano index.js
root@batman:/home/bijay/Reactnode# node index.js
Server running at http://192.168.1.67:6080/
```

Server starts at the specified port and ip.

At the browser, when we enter the ip and port number: 192.168.1.67:6080,

We see the following result:



Again for next API, we copy the index.js file and perform necessary editing,

Cp index.js secondindex.js

We specify the port number to be 7080 as shown, and msg to be shown is edited too.

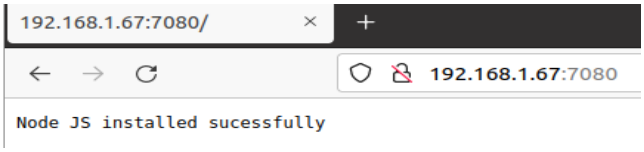
```
root@batman:/home/bijay/Reactnode# cat secondindex.js
const http = require('http');
const hostname = '192.168.1.67';
const port = 7080;
const server = http.createServer((req, res) =>
        {
                res.statusCode = 200;
                res.setHeader('Content-Type', 'text/plain');
                res.end('Node JS installed sucessfully\n');
        });

server.listen(port, hostname, () => {
        console.log(`Server running at http://${hostname}:${port}/`);
        });


root@batman:/home/bijay/Reactnode#
```

With command, ***Node secondindex.js***

We can host the app, and on browser, when we enter 192.168.1.67:7080, we get this result:



Now, pm2 was installed in our local vm,

Npm install pm2@latest -g

```
+ pm2@5.1.2
added 180 packages from 200 contributors in 49.295s
root@batman:/home/bijay/Reactnode#
```

Now, we can start index.js with PM2 as:

pm2 start index.js

And again, for next file,

pm2 start secondindex.js

```
root@batman:/home/bijay/Reactnode# pm2 start index.js
[PM2] Starting /home/bijay/Reactnode/index.js in fork_mode (1 instance)
[PM2] Done.
```

id	name	mode	↺	status	cpu	memory
0	index	fork	0	online	0%	20.6mb

```
root@batman:/home/bijay/Reactnode# pm2 start secondindex.js
[PM2] Starting /home/bijay/Reactnode/secondindex.js in fork_mode (1 instance)
[PM2] Done.
```

id	name	mode	↺	status	cpu	memory
0	index	fork	0	online	0%	41.9mb
1	secondindex	fork	0	online	0%	8.4mb

We can concurrently run these processes with pm2.

With command ***pm2 plus,*** we can get an extended view of all apps and databases in one single place, at real-time or through history.

In order to start 4 instances of both the APIs, these commands were used:

Pm2 start index.js -i 4

Pm2 start secondindex.js -i 4

```
root@batman:/home/bijay/Reactnode# pm2 start index.js -i 4
[PM2] Starting /home/bijay/Reactnode/index.js in cluster_mode (4 instances)
[PM2] Done.
PM2+ activated | Instance Name: batman-67c0 | Dash: https://app.pm2.io/#/r/zpc3f8b8bx7apgu
```

id	name	mode	↺	status	cpu	memory
0	index	cluster	0	online	1.1%	41.2mb
1	index	cluster	0	online	0%	41.9mb
2	index	cluster	0	online	0%	41.5mb
3	index	cluster	0	online	0%	33.9mb

```
root@batman:/home/bijay/Reactnode# pm2 start secondindex.js -i 4
[PM2] Starting /home/bijay/Reactnode/secondindex.js in cluster_mode (4 instances)
[PM2] Done.
PM2+ activated | Instance Name: batman-67c0 | Dash: https://app.pm2.io/#/r/zpc3f8b8bx7apgu
```

id	name	mode	↺	status	cpu	memory
0	index	cluster	0	online	0%	41.4mb
1	index	cluster	0	online	0%	42.2mb
2	index	cluster	0	online	0%	41.9mb
3	index	cluster	0	online	0%	41.9mb
4	secondindex	cluster	0	online	0%	41.8mb
5	secondindex	cluster	0	online	0%	42.0mb
6	secondindex	cluster	0	online	0%	41.5mb
7	secondindex	cluster	0	online	0%	33.6mb

```
root@batman:/home/bijay/Reactnode#
```

To delete the clusters one by one, the id of each cluster was mentioned with stop command:

Pm2 stop 0

Pm2 stop 1

Pm2 stop 2

Pm2 stop 3

Pm2 stop 4

Pm2 stop 5

Pm2 stop 6

Pm2 stop 7

root@batman: /home/bijay/Reactnode

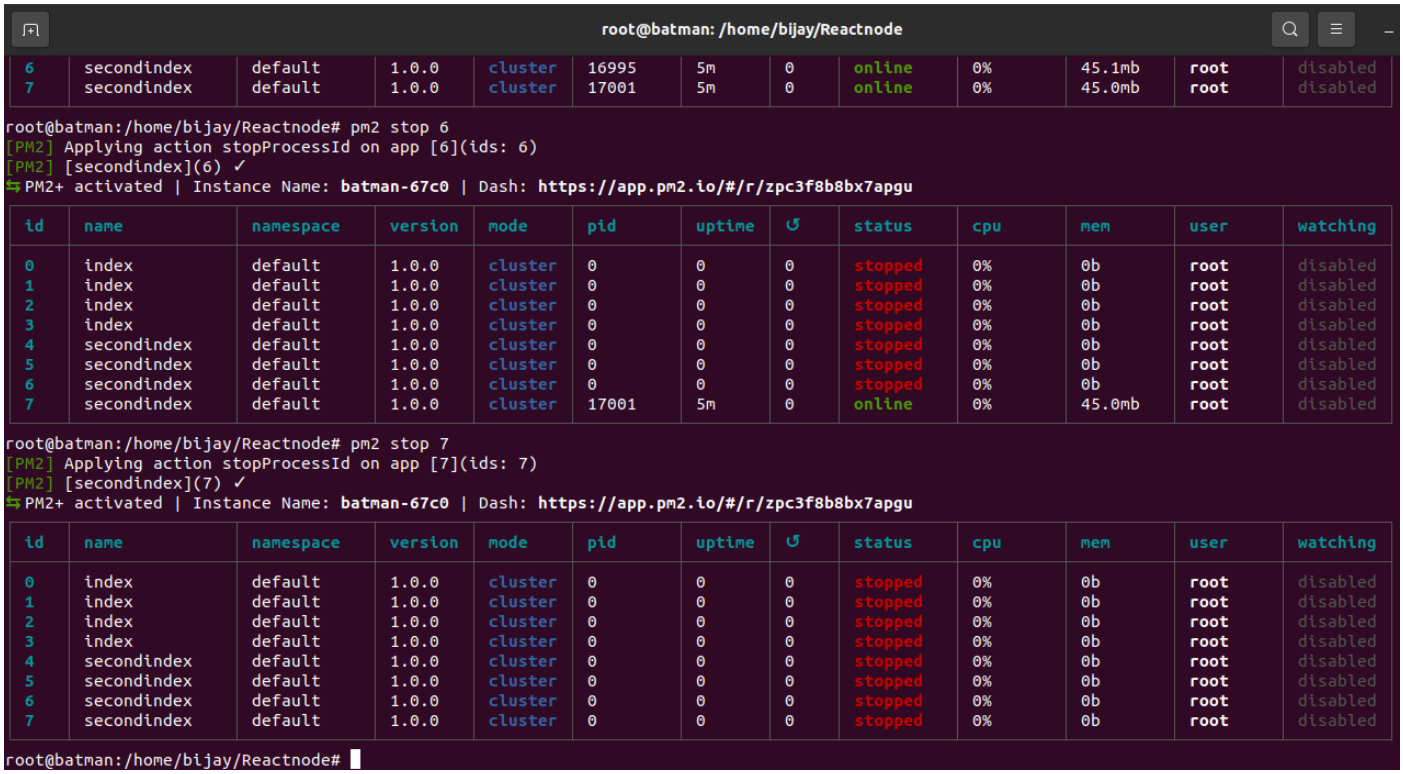
```
root@batman:/home/bijay/Reactnode# pm2 stop 0
[PM2] Applying action stopProcessId on app [0](ids: 0)
[PM2] [index](0) ✓
PM2+ activated | Instance Name: batman-67c0 | Dash: https://app.pm2.io/#/r/zpc3f8b8bx7apgu
```

id	name	namespace	version	mode	pid	uptime	↺	status	cpu	mem	user	watching
0	index	default	1.0.0	cluster	0	0	0	stopped	0%	0b	root	disabled
1	index	default	1.0.0	cluster	16927	4m	0	online	0%	45.1mb	root	disabled
2	index	default	1.0.0	cluster	16940	4m	0	online	0%	45.8mb	root	disabled
3	index	default	1.0.0	cluster	16941	4m	0	online	0%	45.3mb	root	disabled
4	secondindex	default	1.0.0	cluster	16981	4m	0	online	0%	44.5mb	root	disabled
5	secondindex	default	1.0.0	cluster	16982	4m	0	online	0%	45.1mb	root	disabled
6	secondindex	default	1.0.0	cluster	16995	4m	0	online	0%	45.1mb	root	disabled
7	secondindex	default	1.0.0	cluster	17001	4m	0	online	0%	45.0mb	root	disabled

```
root@batman:/home/bijay/Reactnode# pm2 stop 1
[PM2] Applying action stopProcessId on app [1](ids: 1)
[PM2] [index](1) ✓
PM2+ activated | Instance Name: batman-67c0 | Dash: https://app.pm2.io/#/r/zpc3f8b8bx7apgu
```

id	name	namespace	version	mode	pid	uptime	↺	status	cpu	mem	user	watching
0	index	default	1.0.0	cluster	0	0	0	stopped	0%	0b	root	disabled
1	index	default	1.0.0	cluster	0	0	0	stopped	0%	0b	root	disabled
2	index	default	1.0.0	cluster	16940	4m	0	online	0%	45.8mb	root	disabled
3	index	default	1.0.0	cluster	16941	4m	0	online	0%	45.3mb	root	disabled
4	secondindex	default	1.0.0	cluster	16981	4m	0	online	0%	44.5mb	root	disabled
5	secondindex	default	1.0.0	cluster	16982	4m	0	online	0%	45.1mb	root	disabled
6	secondindex	default	1.0.0	cluster	16995	4m	0	online	0%	45.1mb	root	disabled
7	secondindex	default	1.0.0	cluster	17001	4m	0	online	0%	45.0mb	root	disabled

```
root@batman:/home/bijay/Reactnode# pm2 stop 2
[PM2] Applying action stopProcessId on app [2](ids: 2)
[PM2] [index](2) ✓
PM2+ activated | Instance Name: batman-67c0 | Dash: https://app.pm2.io/#/r/zpc3f8b8bx7apgu
```



In this way, nodejs can be used to create APIs and pm2 can be used to manage the processes.

React JS:

- Install React.js.
- Creating a React Application and print message "Hello React.js"
- Change the default port 3000 to 3001

React is installed in system with create-react-app, which creates a starter package of react app, with provided name:

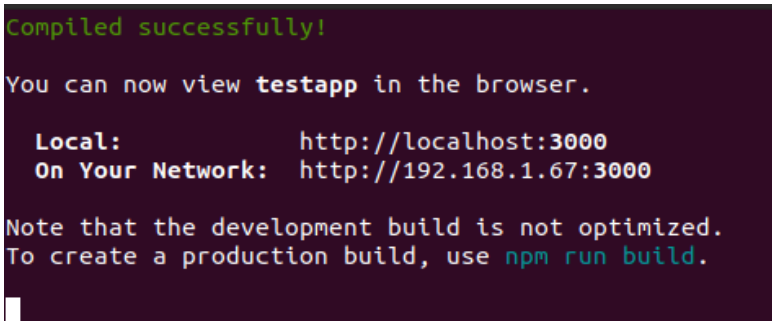
Npx create-react-app testapp

This command will create a sample react app along with installing required dependencies like react, react-dom, react-scripts, cra-template. (We can also do ***npm install react*** to install react)

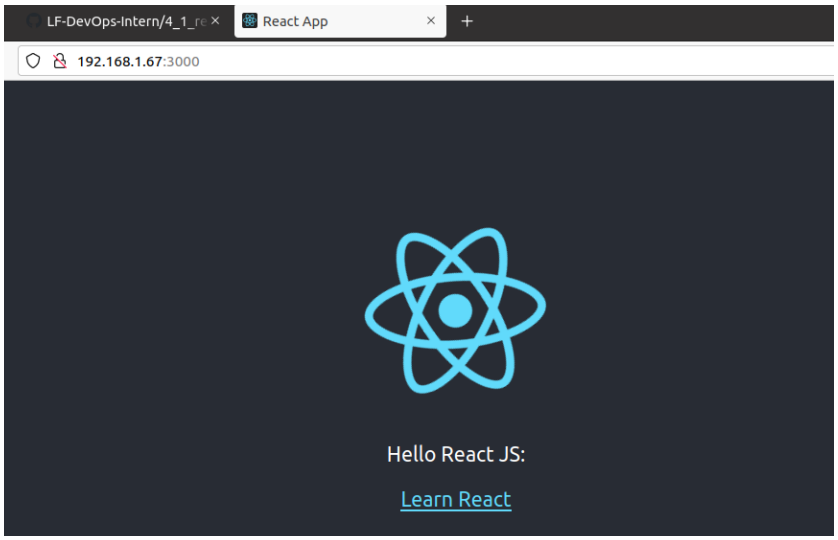
Then ***cd testapp,*** to go to the project directory.

To start the development server at default port 3000,

Npm start

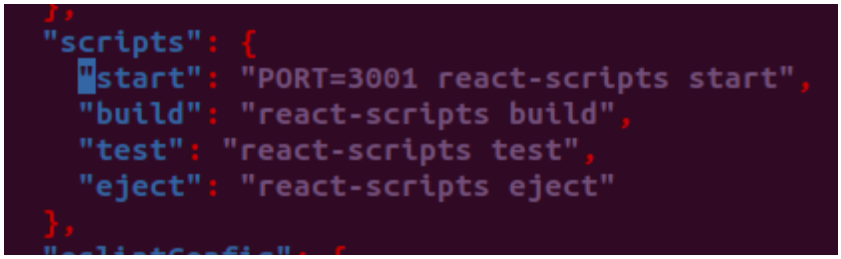


Here is the screenshot of the running server, at port 3000.



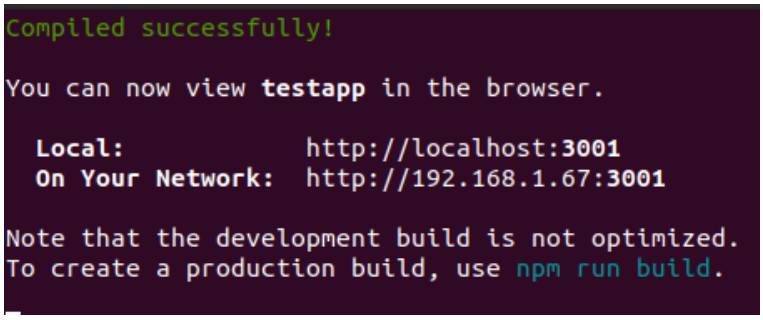
Inside the project directory, the package.json file is edited for port configuration.

Nano package.json

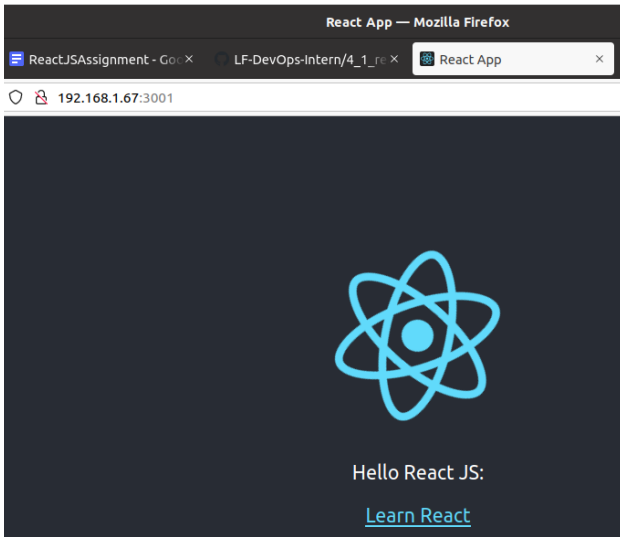


PORT=3001 is added with *start* under *scripts* and saved and exited.

Finally, ***npm start*** to start the server again and the port is changed to 3001.



And we can verify the running server from the browser, which is running at port 3001.



In this way, we can create a react application and change its port configurations.