**Step 2: Deploying on an IIS Server on AWS**

**A. Set Up an AWS Windows Server with IIS**

1. **Launch a Windows Server EC2 instance on AWS**:
   * Go to the AWS EC2 dashboard and select "Launch Instance."
   * Choose a Windows Server AMI (e.g., Windows Server 2019).
   * Configure the instance (minimum t2.micro for testing).
   * Make sure the security group allows inbound traffic on HTTP (port 80) and any custom port you might use for Node (e.g., port 5000).
2. **Connect to your Windows instance**:
   * Connect to the instance using RDP.
   * Use the administrator credentials provided by AWS.
3. **Install IIS on Windows Server**:
   * Open **Server Manager** > **Manage** > **Add Roles and Features**.
   * Select **Web Server (IIS)** and follow the prompts to install it.

**B. Configure IIS to Serve the React and Node Application**

1. **Install Node.js** on the server:
   * Download the Windows installer for Node.js from the [Node.js website](https://nodejs.org).
   * Install Node.js, ensuring you select "Add to PATH" during the installation.

**Create a Simple React and Node Application**

1. **Set up the React frontend**:

npx create-react-app my-app

cd my-app

npm start

 This will create a basic React app with an initial setup.

 **Set up the Node.js backend**:

* Create a folder named server outside of the my-app directory.
* Inside the server folder, initialize a Node project

npm init -y

npm install express cors

Create an index.js file for the Express server

// server/index.js

const express = require("express");

const cors = require("cors");

const app = express();

const PORT = process.env.PORT || 5000;

app.use(cors());

app.get("/api", (req, res) => {

res.json({ message: "Hello from the backend!" });

});

app.listen(PORT, () => {

console.log(`Server is running on http://localhost:${PORT}`);

});

Start the Node server:

node index.js

**Connect React with Node**:

* In my-app, create a new proxy in package.json

"proxy": <http://localhost:5000>

In a React component, fetch data from the backend

// my-app/src/App.js

import React, { useEffect, useState } from "react";

function App() {

const [message, setMessage] = useState("");

useEffect(() => {

fetch("/api")

.then((response) => response.json())

.then((data) => setMessage(data.message));

}, []);

return (

<div className="App">

<h1>{message}</h1>

</div>

);

}

export default App;

**Build the React app**:

cd my-app

npm run build

This will generate a build folder that contains your React app, ready for deployment

 **Serve the React build**:

* In IIS, navigate to the **Sites** section, and create a new site for the React app:
  + **Site name**: ReactApp
  + **Physical path**: Select the path to the my-app/build directory.
  + **Binding**: Set to HTTP, port 80, or a custom port if necessary.
* Open the site in your browser by visiting http://<your\_server\_ip>.

 **Run the Node server**:

* Open PowerShell or Command Prompt on the server.

Navigate to the server directory

cd path\to\server

node index.js

To keep the server running, you may use a process manager like pm2

npm install -g pm2

pm2 start index.js --name "NodeServer"

pm2 startup

**Check Firewall Rules**

**Ensure the Windows firewall and AWS Security Group allow inbound traffic on port 5000.**

**Windows Firewall:**

1. **Open Windows Defender Firewall.**
2. **Go to Advanced Settings.**
3. **Add an inbound rule to allow TCP traffic on port 5000.**

**AWS Security Group:**

1. **In the AWS Management Console, locate your EC2 instance.**
2. **Under Security Groups, edit the inbound rules:**
   * **Add a rule for:**
     + **Type: Custom TCP Rule**
     + **Protocol: TCP**
     + **Port Range: 5000**
     + **Source: 0.0.0.0/0 (or restrict to your IP for security).**

Test Node.js Server

<http://44.211.76.120:5000/api>

Restart Your Node Server Using PM2

pm2 start index.js --name "NodeServer"

**Check the Status of Your Application**  
To verify your app is running:

pm2 list

pm2 stop NodeServer

pm2 delete NodeServer

pm2 logs NodeServer

**Update API Call in App.js**

Replace the relative /api with the full URL, including the backend port:

useEffect(() => {

fetch("http://44.211.76.120:5000/api") // Explicitly mention the backend address

.then((response) => response.json())

.then((data) => setMessage(data.message))

.catch((error) => console.error("Error fetching API:", error));

}, []);

Enable cors

const cors = require('cors'); // Import the cors package

app.use(cors());