The **3-Tier Web App with CI/CD to AWS Elastic Beanstalk** is a **powerful and real-world DevOps project** that demonstrates:

Full-stack deployment  
Infrastructure management  
Automated delivery pipeline  
AWS service integration

# Project Name: CI/CD Pipeline for 3-Tier Web Application on AWS Elastic Beanstalk

**Project Overview:**

A 3-tier architecture includes:

1. **Frontend** – React or static HTML (served via S3 or part of backend)
2. **Backend** – Node.js or Python Flask REST API
3. **Database** – AWS RDS (PostgreSQL or MySQL)

The application is deployed using **Elastic Beanstalk**, and a **CI/CD pipeline** is built using **GitHub + AWS CodePipeline + CodeBuild**.

[ GitHub ]

↓ (Push Code)

[ CodePipeline ]

↓

[ CodeBuild ]

↓

[ Elastic Beanstalk ]

↓

[ Backend App (EC2) ]

↓

[ RDS Database ]

**Tools Used:**

* **Frontend**: React.js / HTML/CSS
* **Backend**: Node.js or Python Flask
* **Database**: AWS RDS (MySQL/PostgreSQL)
* **CI/CD**: GitHub, CodePipeline, CodeBuild
* **Hosting**: Elastic Beanstalk (managed EC2, Load Balancer)
* **Monitoring**: CloudWatch

**1. Build the Application**

* **Frontend**:
  + Create a simple React.js app (or static HTML page)
  + Include a contact form or dynamic component

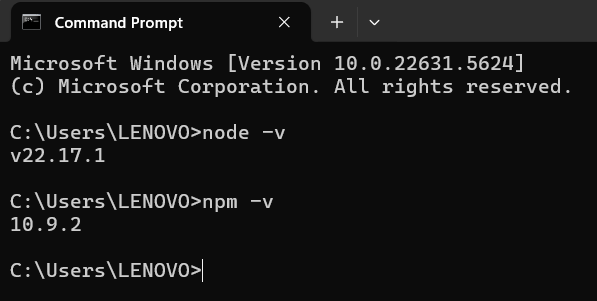
### Step-by-Step: Create a Simple React App (Frontend Tier of 3-Tier Web App)

**Step 0: Prerequisites — Install Tools on Your Laptop**

Before starting, install the following (only once):

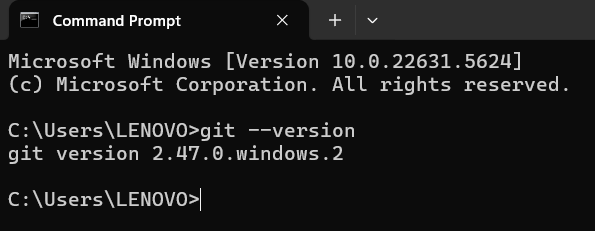
**1. Install Node.js**

* Go to: https://nodejs.org/
* Download the **LTS version** for your OS (Windows/Mac)
* Install → Then verify:
* node -v
* npm -v



**2. Install Git**

* Go to: <https://git-scm.com/downloads>
* Download for your OS
* Install → Then verify:
* Git –version



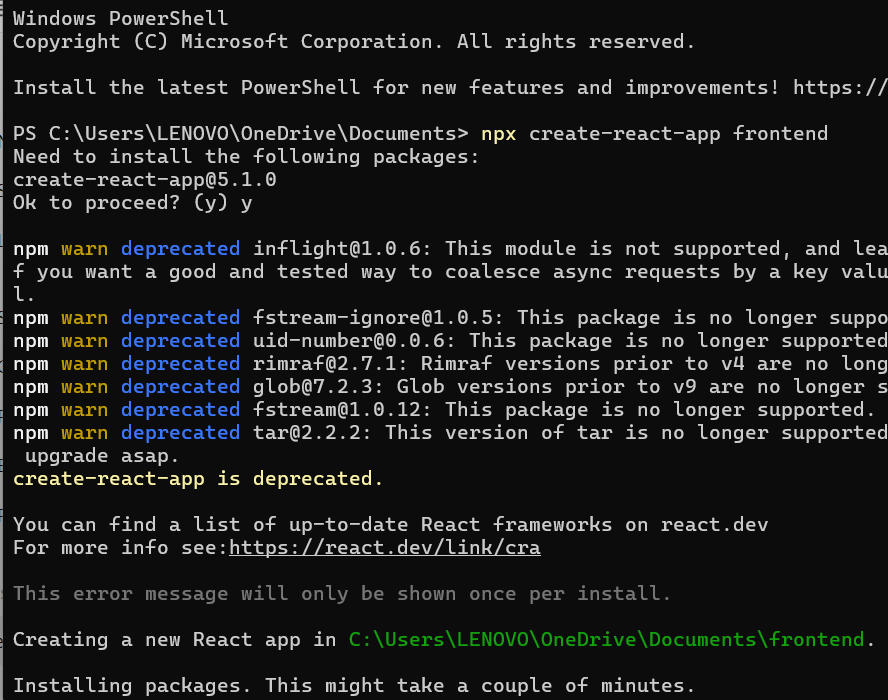
**3. Install Visual Studio Code (VS Code)**

* Go to: https://code.visualstudio.com/
* Install the editor for writing code easily

**Step 1: Create Your Project Folder**

1. Open any folder on your computer (e.g., Documents)
2. Right-click → Open in Terminal (or Command Prompt)
3. Run this command:

npx create-react-app frontend



It will take a few minutes. This will automatically create a folder frontend with a sample React app.

Step 2: Open the React App in VS Code

cd frontend

code .

This opens your project in VS Code.

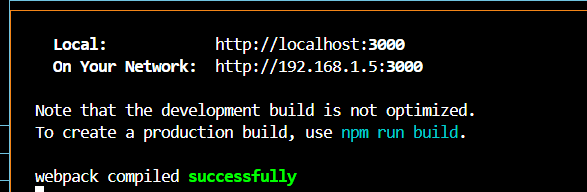
**Step 3: Run the React App**

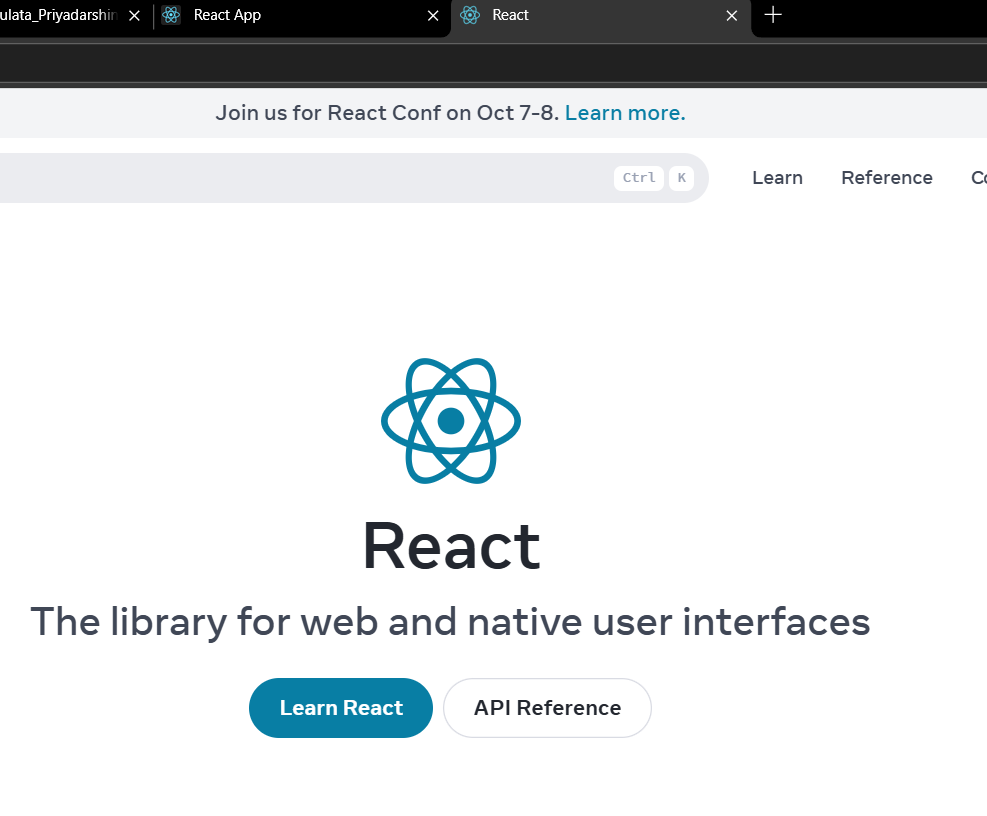
In the terminal (inside VS Code or Command Prompt):

npm start

* This will open: <http://localhost:3000> in your browser
* You’ll see the default **React welcome page**

**Congratulations! You just created and launched your first frontend React App.**





* **Backend**:
  + REST API using Node.js or Flask
  + Handles CRUD (Create, Read, Update, Delete) with a database
* **Database**:
  + Provision **RDS MySQL/PostgreSQL** instance via AWS console
  + Store sample data (users, tasks, invoices, etc.)

### **Next Steps**

Now that your app is running, you can:

* Add a **contact form component**

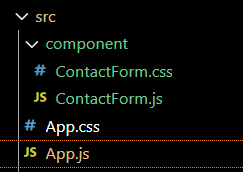
Step-by-Step: Add Contact Form in Your React App

**Step 1: Create a New Component for the Form**

**1. In VS Code:**

* In the src folder, create a new folder called components
* Inside components, create a file called **ContactForm.js**

**Your folder should now look like this:**



**Step 2: Add Code to ContactForm.js**

Paste the below code inside ContactForm.js:

import React, { useState } from 'react';

import './ContactForm.css';

const ContactForm = () => {

const [formData, setFormData] = useState({ name: '', email: '', message: '' });

const handleChange = (e) => {

setFormData({ ...formData, [e.target.name]: e.target.value });

};

const handleSubmit = (e) => {

e.preventDefault();

alert("Thank you, " + formData.name + "! Your message has been received.");

// Here, you can later send data to backend

};

return (

<div className="contact-form">

<h2>Contact Us</h2>

<form onSubmit={handleSubmit}>

<input type="text" name="name" placeholder="Your Name" required onChange={handleChange} />

<input type="email" name="email" placeholder="Your Email" required onChange={handleChange} />

<textarea name="message" placeholder="Your Message" rows="4" required onChange={handleChange}></textarea>

<button type="submit">Submit</button>

</form>

</div>

);

};

export default ContactForm;

**Step 3: Create CSS File for Styling**

In the same components folder:

* Create a file called **ContactForm.css**
* Paste the below code:

.contact-form {

max-width: 500px;

margin: 30px auto;

padding: 20px;

background-color: #f0f8ff;

border: 1px solid #ccc;

border-radius: 8px;

}

.contact-form h2 {

text-align: center;

}

.contact-form input,

.contact-form textarea {

width: 100%;

padding: 10px;

margin: 10px 0;

border: 1px solid #999;

border-radius: 5px;

}

.contact-form button {

width: 100%;

padding: 12px;

background-color: #0077cc;

color: white;

border: none;

border-radius: 5px;

font-size: 16px;

}

**Step 4: Use ContactForm in App.js**

Open src/App.js and **replace the default content** with this:

import React from 'react';

import './App.css';

import ContactForm from './component/ContactForm';

function App() {

return (

<div className="App">

<h1>Welcome to My 3-Tier Web App</h1>

<ContactForm />

</div>

);

}

export default App;

**Step 5: Restart the App (If Needed)**

If the app isn’t already running, restart it:

npm start

**Final Result**

Go to: http://localhost:3000  
You will now see:

* A **Welcome message**
* A clean **Contact Us form**
* Submitting the form shows a **thank you alert**

