Step 1: Install Node.js and npm

1. Download Node.js:

- Visit the official Node.js website.
- Download the LTS (Long Term Support) version suitable for your operating system.

2. Install Node.js:

- Run the installer and follow the setup instructions, accepting the default options.
- Node.js includes npm (Node Package Manager), which will be installed automatically.

3. Verify Installation:

- Open your terminal or command prompt.

Run the following commands to ensure Node.js and npm are installed:

```
node -v
npm -v
```

- You should see version numbers for both Node.js and npm.

Step 2: Set Up Your Project Directory

1. Create a Project Folder:

Create a new directory for your Express project. You can do this via the terminal:

```
mkdir my-express-app
cd my-express-app
```

2. Initialize a New Node.js Project:

In your project directory, initialize a new Node.js project with the following command:
`npm init -y`

- This will create a `package.json` file with default settings.

Step 3: Install Express

1. Install Express:

Use npm to install Express in your project:

```
npm install express --save
```

- This command installs Express and adds it as a dependency in your `package.json`.

Step 4: Create the Express Server

1. Create an Entry File:

In your project directory, create a file named `index.js`:

- This will be the entry point for your Express application.

2. Set Up the Basic Server:

Open `index.js` in your code editor and add the following code:

```
const express = require('express');
const app = express();
const port = 3000;

app.get('/', (req, res) => {
  res.send('Hello World!');
});

app.listen(port, () => {
  console.log(`Example app listening at http://localhost:${port}`);
});
```

3. Run the Server:

In the terminal, start your Express server by running:

```
node index.js
```

- Open your browser and go to `http://localhost:3000`. You should see "Hello World!" displayed.

Step 5: Set Up Development Tools

1. Install Nodemon for Development:

Nodemon automatically restarts your server when you make changes to your code. Install it globally with:

```
npm install -g nodemon
```

Alternatively, you can install it as a dev dependency in your project:

```
npm install nodemon --save-dev
```

2. Use Nodemon to Start the Server:

Update the `package.json` to use Nodemon for development. Add the following under the "scripts" section:

```
"scripts": {
  "start": "node index.js",
  "dev": "nodemon index.js"
}
```

Now, start the server in development mode using:

```
npm run dev
```

Step 6: Set Up Middleware (Optional)

1. Install Middleware:

You can add middleware like `body-parser` for parsing JSON, `morgan` for logging, etc. For example, to install `body-parser`:

```
npm install body-parser
```

To use it in your `index.js`:

```
const bodyParser = require('body-parser');
app.use(bodyParser.json());
```

Step 7: Organize Your Project Structure (Optional)

1. Create Folders for Routes, Controllers, and Models:

- As your project grows, you may want to organize your files:
  - **routes/**: For route definitions.
  - **controllers/**: For business logic.
  - **models/**: For database models.

2. Example Structure:

```
my-express-app/
├── index.js
├── package.json
├── routes/
│   └── index.js
├── controllers/
└── models/
```

Step 8: Deploying Your Express App (Optional)

1. Prepare for Deployment:

- Ensure your app runs without Nodemon by using `npm start`.
- Consider setting up environment variables using a package like `dotenv`.

2. Deploy to a Cloud Service:

- Deploy your app to a platform like Heroku, AWS, or Vercel following their specific deployment guides.