update with **Node.js 14**

If you want to update the package.json to align with **Node.js 14** (instead of Node.js 20), here’s the updated version. This ensures compatibility with the node:14 base image in your Dockerfile.

Shape

**Updated package.json for Node.js 14**

json

Copy

{

"name": "node-js-sample",

"version": "0.2.0",

"description": "A sample Node.js app using Express 4",

"main": "app.js",

"scripts": {

"start": "node app.js"

},

"dependencies": {

"express": "^4.17.1"

},

"engines": {

"node": "14.x"

},

"repository": {

"type": "git",

"url": "https://github.com/heroku/node-js-sample"

},

"keywords": [

"node",

"heroku",

"express"

],

"author": "Mark Pundsack",

"contributors": [

"Zeke Sikelianos <zeke@sikelianos.com> (http://zeke.sikelianos.com)"

],

"license": "MIT"

}

Shape

**Key Changes**

1. **Node.js Version in engines**:

* Updated to "node": "14.x" to match the node:14 base image in the Dockerfile.

1. **Express Version**:

* Updated to "express": "^4.17.1", which is a stable version compatible with Node.js 14.

1. **Entry Point**:

* The main and scripts fields are set to use app.js as the entry point. If your entry point is index.js, update these fields accordingly.

Shape

**Why These Changes?**

1. **Node.js 14 Compatibility**:

* Node.js 14 is an LTS (Long-Term Support) version, and the engines field ensures that the application runs on Node.js 14.x.

1. **Express Compatibility**:

* Express 4.17.1 is a stable version that works well with Node.js 14.

1. **Consistency with Dockerfile**:

* If your Dockerfile uses node:14, the package.json should specify Node.js 14 in the engines field to avoid compatibility issues.

Shape

**Steps to Use the Updated package.json**

1. Save the updated package.json in your project directory.
2. Install dependencies:

bash

Copy

npm install

1. Test the application:

bash

Copy

npm start

* Access the application at http://localhost:5000 to confirm it works.

Shape

**Example app.js for Reference**

Ensure your app.js file is correctly configured. Here’s an example:

javascript

Copy

const express = require('express');

const app = express();

const port = 5000;

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

res.send('Hello World!');

});

app.listen(port, () => {

console.log(`App listening at http://localhost:${port}`);

});

Shape

**Final Checks**

1. Verify the package.json file is valid:

bash

Copy

npm install

* If there are no errors, the file is valid.

1. Test the application:

bash

Copy

npm start

* Access the application at http://localhost:5000 to confirm it works.

Shape

**Dockerfile for Node.js 14**

If you’re using Node.js 14, ensure your Dockerfile is also updated to use node:14:

Dockerfile

Copy

# Use the official Node.js 14 image as the base image

FROM node:14

# Set the working directory inside the container

WORKDIR /usr/src/app

# Copy package.json and package-lock.json to the working directory

COPY package\*.json ./

# Install dependencies

RUN npm install

# Copy the rest of the application code to the working directory

COPY . .

# Create a non-root user and switch to it

RUN useradd -m appuser

USER appuser

# Expose port 5000 for the application

EXPOSE 5000

# Command to start the application

CMD ["npm", "start"]

Shape

This updated package.json and Dockerfile are now fully aligned with Node.js 14 and should work seamlessly together.