Node.js Örnek

Web ekranında "Merhaba Docker" yazısını gösterecek bir Node.js uygulamasını Dockerize etmek için aşağıdaki adımları takip edebilirsiniz. İlk olarak, gerekli Node.js uygulamasını yazıp, ardından Dockerfile'ı oluşturacağız.

/proje-dizini
├── Dockerfile
├── package.json
└── index.js

1. Node.js Uygulamasını Yazma

Öncelikle, Node.js uygulamasının kodunu yazmalısınız. Bu uygulama, basit bir web sunucusu oluşturacak ve "Merhaba Docker" mesajını gösterecektir.

index.json

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

app.get('/', (req, res) => {
  res.send('Merhaba Docker');
});

const PORT = 3070;
app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}`);
});
```

`index.js` dosyasında, Express.js kullanarak basit bir HTTP sunucusu oluşturup "Merhaba Docker" mesajını göstereceğiz.

2. `package.json` Dosyasını Oluşturma

Uygulamanız için gerekli bağımlılıkları tanımlayan bir `package.json` dosyası oluşturun.

package.json

```
{
  "name": "dockerexample",
  "version": "1.0.0",
  "description": "A simple Docker example with Node.js",
  "main": "index.js",
  "scripts": {
    "start": "node index.js"
```

```
  },
  "author": "Your Name",
  "license": "ISC",
  "dependencies": {
    "express": "^4.18.2"
  }
}
```

`package.json` dosyası, Node.js projesi için bağımlılıkları belirtir.

3. Dockerfile Oluşturma

Dockerfile, uygulamanızı Docker imajına dönüştürmek için kullanılır. İşte "Merhaba Docker" yazan uygulamanızı Dockerize etmek için gereken Dockerfile:

Dockerfile

```
# Node.js base image (latest LTS version)
FROM node:20-slim

# Label for metadata
LABEL "about"="This file demonstrates a basic Node.js app with Docker."

# Set environment variable for the working directory
ENV dizin=/home/ibb-tech
# Create and set permissions for the working directory
RUN mkdir -p $dizin && chown node:node $dizin

# Switch to non-root user
USER node

# Set the working directory
WORKDIR $dizin

# Copy package.json and install dependencies
COPY --chown=node:node package.json ./

RUN npm install

# Copy the application files
COPY --chown=node:node index.js ./

# Expose port 3070
EXPOSE 3070
```

```
# Default command to run the application
ENTRYPOINT ["node", "index.js"]
```

4. Docker İmajını Oluşturma ve Çalıştırma

Docker İmajını Oluşturma

Bu dosyaları aynı dizine koyduktan sonra, Docker imajını şu komutla oluşturabilirsiniz:

```
docker build -t node-ornek .
```

Konteyneri Başlatma

Docker konteynerini başlatmak için şu komutu kullanabilirsiniz:

```
docker run -d -p 3070:3070 node-ornek
```

5. Web Ekranını Test Etme

Tarayıcınızı açın ve `http://localhost:3070` adresine gidin. Web sayfasında "Merhaba Docker" mesajını görmelisiniz.

Dockerfile Açıklaması

Bu Dockerfile, Node.js tabanlı bir uygulamayı Docker imajı olarak oluşturmak için gerekli adımları içeriyor. Aşağıda, her bir komutun adım adım açıklamasını bulabilirsiniz:

1. Base Image Seçimi

```
FROM node:20-slim
```

- **FROM**: Bu komut, Docker imajının temelini belirler. `node:20-slim`, Node.js'in en son LTS (Long-Term Support) sürümünün daha hafif bir sürümünü kullanır. "Slim" sürümü, daha küçük boyutlu bir imajdır ve yalnızca gerekli olan bileşenleri içerir.

2. Metadata Ekleme

```
LABEL "about"="This file demonstrates a basic Node.js app with Docker."
```

- **LABEL**: İmajın içine metadata (bilgi) eklemek için kullanılır. Bu örnekte, imajın ne amaçla oluşturulduğunu belirten bir açıklama eklenmiştir. Bu bilgi, Docker imajını tanımlamada yardımcı olabilir.

3. Ortam Değişkeni Tanımlama

```
ENV dizin=/home/ibb-tech
```

- **ENV**: Ortam değişkenleri tanımlamak için kullanılır. Bu örnekte, `WORKDIR` adında bir değişken tanımlanmış ve değeri `/home/ibb-tech` olarak belirlenmiştir. Bu, daha sonra çalışma dizinini belirlemede kullanılacak.

4. Çalışma Dizini Oluşturma ve İzin Ayarları

```
RUN mkdir -p $dizin && chown node:node $WORKDIR
```

- **RUN**: Docker konteyneri içinde bir komut çalıştırmak için kullanılır. Bu komutla, `$dizin` (yani `/home/ibb-tech`) dizini oluşturulur ve bu dizinin sahipliği `node` kullanıcısına atanır.

5. Kullanıcı Değiştirme

```
USER node
```

- **USER**: İmaj içinde çalıştırılacak komutları belirli bir kullanıcı altında çalıştırmak için kullanılır. Bu örnekte, kök (root) kullanıcı yerine `node` kullanıcısı altında komutlar çalıştırılır. Bu, güvenlik açısından daha iyi bir uygulamadır.

6. Çalışma Dizini Belirleme

```
WORKDIR $dizin
```

- **WORKDIR**: Konteyner içinde komutların çalıştırılacağı çalışma dizinini belirler. Bu komutla, `$dizin` (yani `/home/ibb-tech`) dizini çalışma dizini olarak ayarlanır. Tüm komutlar bu dizin içinde çalıştırılır.

7. `package.json` Dosyasını Kopyalama

```
COPY --chown=node:node package.json ./
```

- **COPY**: Bu komut, yerel makinedeki dosyaları Docker imajına kopyalar. Bu örnekte, `package.json` dosyası konteynerin çalışma dizinine (`./`) kopyalanır. Ayrıca, `--chown=node:node` seçeneği, dosyanın sahipliğini `node` kullanıcısı ve grubuna atar.

8. Node.js Bağımlılıklarını Yükleme

```
RUN npm install
```

- **RUN**: Node.js'in `npm install` komutunu çalıştırarak, `package.json` dosyasında tanımlanan bağımlılıkları yükler. Bu, uygulamanın çalışması için gereken kütüphaneleri indirir ve kurar.

9. Uygulama Dosyalarını Kopyalama

```
COPY --chown=node:node index.js ./
```

- **COPY**: `index.js` dosyasını yerel makineden Docker imajına kopyalar. Bu dosya, uygulamanın çalıştırılmasından sorumlu olan ana dosyadır. `--chown=node:node` seçeneğiyle, dosyanın sahipliği `node` kullanıcısına atanır.

10. Port Açma

```
EXPOSE 3070
```

- **EXPOSE**: Konteynerin hangi portları dinleyeceğini belirtir. Bu örnekte, uygulama 3070 portunda çalışacağı için bu port açılır. Ancak, bu komut sadece belgeleme amacı taşır; konteyner dışına bu portu açmak için `docker run -p 3070:3070` komutu kullanılır.

11. Varsayılan Komutu Belirleme

```
ENTRYPOINT ["node", "index.js"]
```

- **ENTRYPOINT**: Konteyner başlatıldığında çalıştırılacak varsayılan komutu belirtir. Bu örnekte, `node` komutuyla `index.js` dosyası çalıştırılır. Bu, uygulamanın başlatılmasını sağlar.

Sonuç

Bu Dockerfile, Node.js ile basit bir web uygulamasını Docker imajı olarak oluşturur. İmaj oluşturulduktan sonra, konteyner çalıştırıldığında `index.js` dosyası çalışır ve tarayıcıda "Merhaba Docker" mesajı gösterilir.