[**https://github.com/gkandemi/docker**](https://github.com/gkandemi/docker)

**DOCKER**

* **docker images** : Oluşturulan images leri listeler
* **docker system prune** : kapanmış olan bütün container dosyalarını temizler
* **docker rmi ImageId** : image i kaldırır
* **docker build –tag ProjeAdi .** : docker projesini kurar
* **docker fs** : çalışan container ları listeler
* **docker run ProjeAdi** : docker projesini çalıştırır. Terminal kapandığında docker durur.
* **docker run -p 3000:3000 ProjeAdi** : docker’daki proje portunu bilgisayarın portuna bağlayıp projeyi çalıştırır.
* **docker kill ImageId** : çalışan docker projesini durdurur
* **docker run -it --name bash\_ubuntu ubuntu** : ubuntu image ne isim verip çalıştırır
* **docker start ImageName** : image i arka planda çalıştırır.
* **docker stop ImageName** : çalışan image i durdurur.
* **docker container rm $(docker container ls -aq)** : bütün image leri kaldırır
* **docker run -d ImageId** : redis i arka planda çalıştırır.
* **docker attach ImageId** : arka planda çalışan image in terminalde loglarını gösterir. Anlık log ları gösterir.
* **docker container logs ImageId** : arka planda çalışan image in log larını getirir. Bütün log ları gösterir.

**Valume Mapping**

* **docker run -v /opt/data:/data/db -p 27017:27017 mongo** : -v (valume) ile image içinde veri saklanması sağlanıyor. veriler /opt/data içinde saklanıyor.
* **docker inspect ImageId** : image veya container bilgilerini gösterir

**Link**

çalışan iki container ı birbirine bağlama yöntemidir.

ilk olarak bağlanılacak container çalıştırılır. Daha sonra bağlanan container link lenerek çalıştırılır.

docker run --name mysql-server -p 3306:3306 -e MYSQL\_ROOT\_PASSWORD=test123 -d mysql

docker run --name pmyadmin -p 8000:80 --link mysql-server:db -d phpmyadmin/phpmyadmin

(**“db” mysql ortamında “localhost” anlamına gelmektedir)**

* Yukarıda mysql ile phpmyadmin birbirine bağlanmıştır. Bu sayede phpmyadmin üzerinden mysql işlemleri yapılmaktadır.
* localhost:8000 üzerinden phpmyadmin e giriş yapılabilmektedir.

**ÖRNEK**

* Veritabanımızda verilerin saklaması için valume işlemi ekleniyor.

docker run --name mysql-server -p 3306:3306 -v /opt/data:/etc/mysql/conf.d -e MYSQL\_ROOT\_PASSWORD=test123 -d mysql

docker run --name pmyadmin -p 8000:80 --link mysql-server:db -d phpmyadmin/phpmyadmin

* kurmuş olduğumuz container ları daha sonra yeniden başlamak için

**docker start mysql-server**

**docker start pmyadmin**

komutları kullanılarak çalıştırabiliriz.

**DOCKER NETWORK TÜRLERİ**

**docker network ls :** tanımlı network listesini gösterir.

* bridge
* docker run monge

docker içinde ip ile birbirlerine bağlanmıştır.

* none
* docker run monge --network=none

içeriden veya dışarıdan herhangi bir erişim yoktur

* host
* docker run mongo --network=host

docker üzerinden port ile birbirine bağlanmıştır

* kullanıcı tanımlı network.

default olarak gelen bridge network ünde 172 li ip yerine yeni bir ip bloğu tanımlanmasıdır.

* docker network create --driver bridge --subnet 182.18.0.0/16 --gateway 182.18.0.1 todo-app-network

**örnek**

* docker network create --driver bridge --subnet 182.18.0.1/24 --gateway 182.18.0.1 custom-network

kendi network bağlantımızı oluşturuyoruz.

* docker run --name mongo-server --net custom-network -d mongo

mongo kurulumu yapılıyor. ismi mongo-server, custom-network seçiliyor ve arka planda çalıştırılıyor.

* docker run --net custom-network -p 3000:3000 gkandemir/todo-app

todo-app kurulumu yapılıyor. custom-network seçiliyor, 3000 port seçiliyor.

**KENDİ DOCKER CONTAİNER IMIZI OLUŞTURALIM**

**Örnek 1: Manuel Kurulum**

* İlk olarak ubuntu 18.04 sürümü olan image dosyasını indiriyoruz ve çalıştırıyoruz.

docker run -it ubuntu:18.04

* ubuntu yu güncelliyoruz ve nodejs 10 sürümünü kuruyoruz. /opt/node-app klasörünü oluştur. klasör içine örnek index.js dosyası oluştur.

1 apt-get update

2 apt-get install curl -y

3 curl -sL https://deb.nodesource.com/setup\_10.x | bash

4 apt-get install nodejs -y

5 node

6 ls

7 cd opt

8 mkdir node-app

9 cd node-app/

10 vi index.js

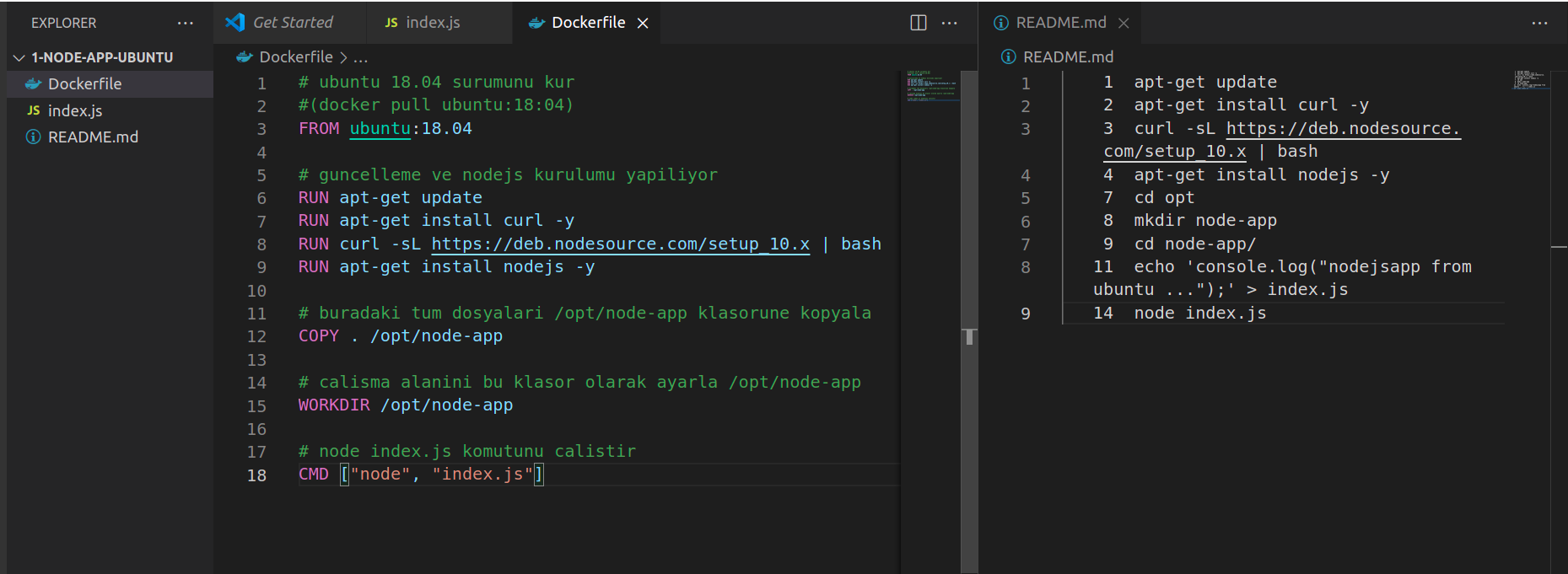
11 echo 'console.log("nodejsapp from ubuntu ...");' > index.js

12 ls

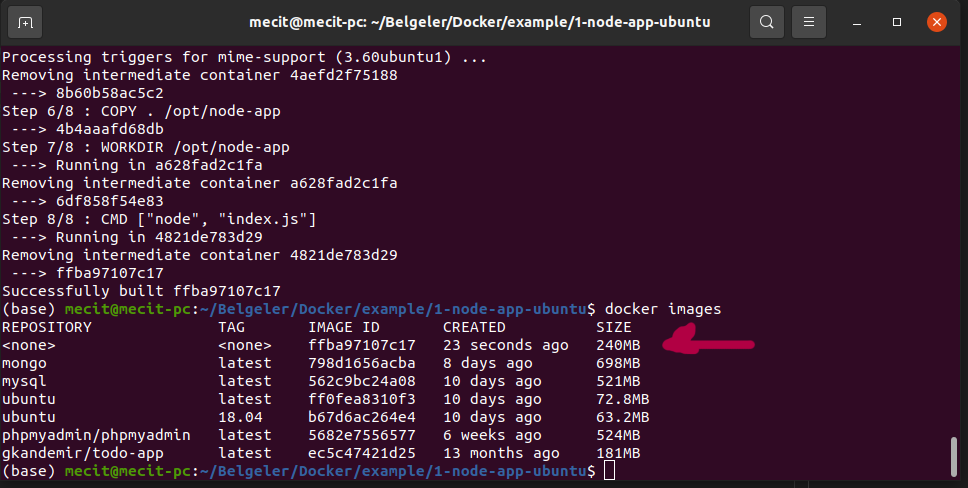
13 cat index.js

14 node index.js

* yukarıda manuel olarak yaptığımız kurulumları projemize “Dockerfile” dosyası ekleyerek otomatik hale getirelim.

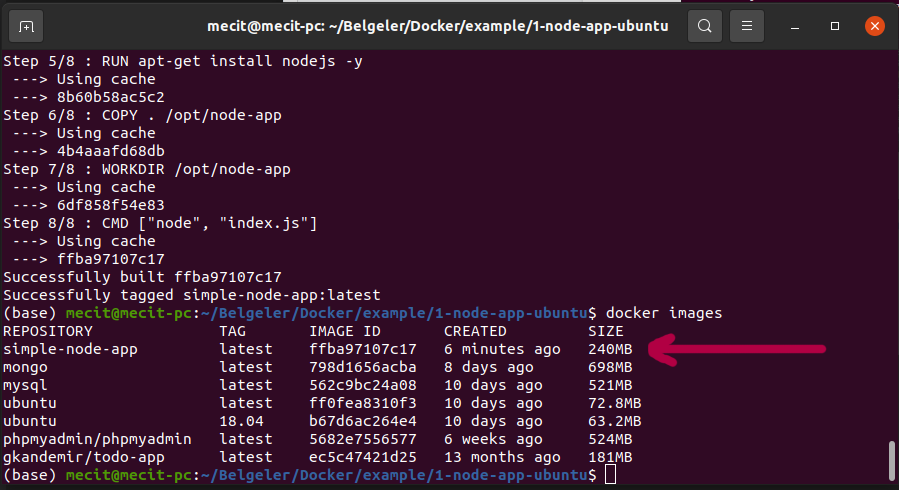


* terminalde projemizin bulunduğu dizini açıyoruz
* **docker build .**  komutu ile container kurulumu yapılıyor



* container isimsiz olarak görünüyor. isim vermek için bu komut çalıştırılır.

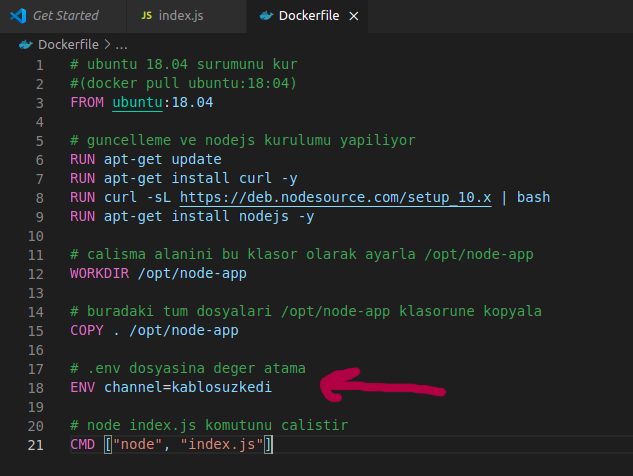
**docker build . -t simple-node-app**



* **docker run simple-node-app** komutu ile container çalıştırılır.

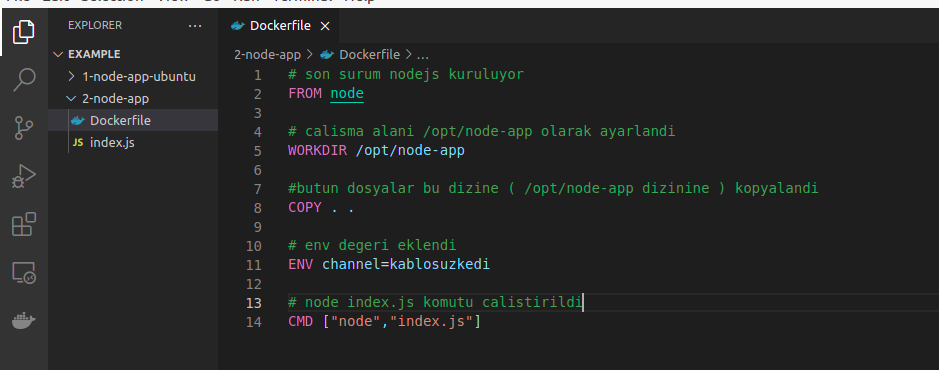
**.env değeri ekleme**

* “Dockerfile” dosyasına “ENV channel=kablosuzkedi” satır ile .env değeri atanır.



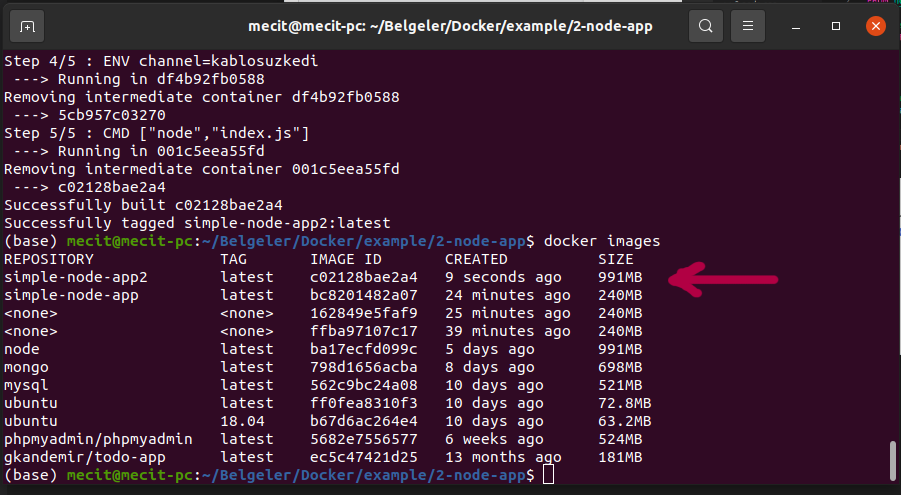
**Örnek 2: Node Image ile Kurulum**

* ilk başlıkta projemize gerekli şeyleri manuel olarak kurduk. şimdi resmi “node” image ile kurulum yapacağız.
* 2-node-app klasörüne index.js dosyası kopyalandı. yeni bir “Dockerfile” dosyası oluşturuldu. resmi “node” image kuruluyor.



* proje dizine gidilerek bu komut çalıştırılır

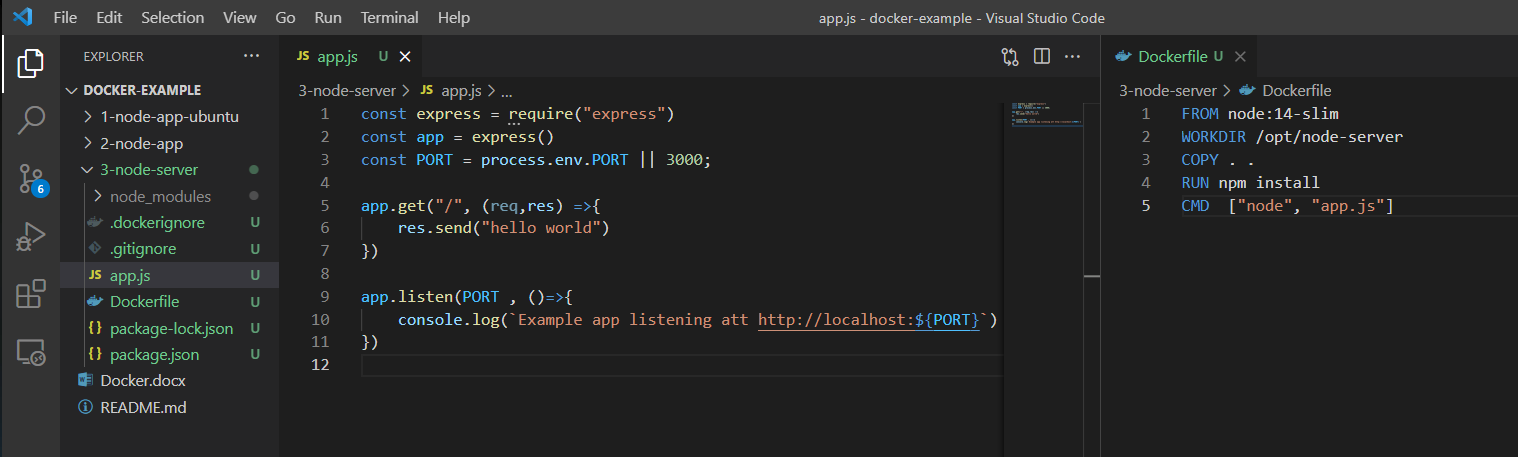
**docker build . -t simple-node-app2**



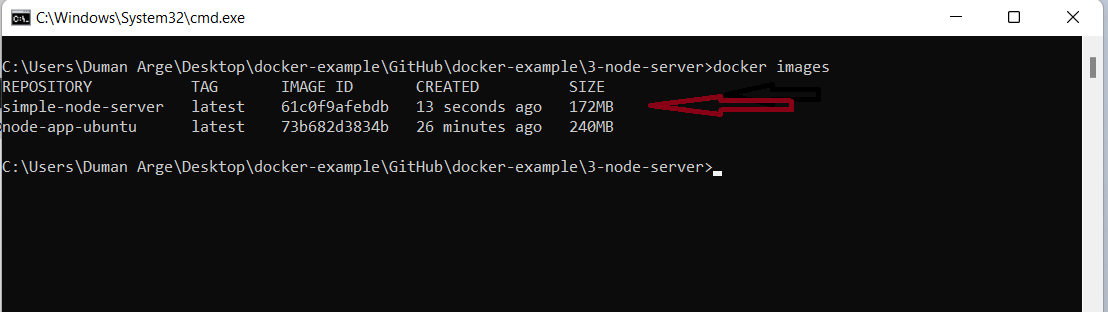
**Örnek 3: Node Image Node Server**

node image üzerinde node-server express uygulaması yapıyoruz.

* “app.js” dosyasında express örneği bulunuyor.
* “.dockerignore” dosyası docker a kopyalamasını istemediğimiz dosyaları içeriyor.



* **docker build . -t simple-node-server** komutu ile docker container kurulumu yapılıyor

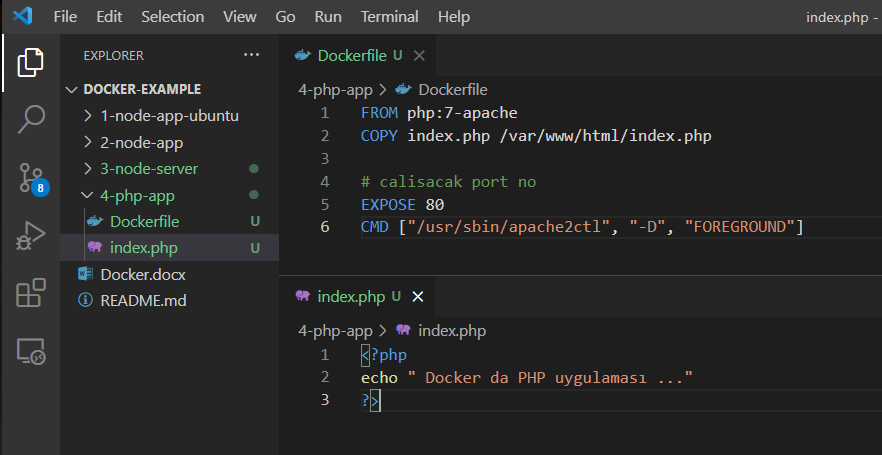


* **docker run -p 3000:3000 simple-node-server** komutu ile 3000 portu üzerinde proje çalıştırılıyor.

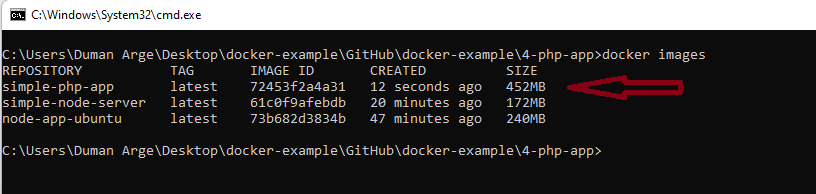
**Örnek 4: PHP-apache Image ile PHP Uygulaması**

php-apache image container kullanarak php projesi çalıştırıyoruz.

* php-app projesinde “Dockerfile” ve “index.php” dosyası oluşturuyoruz.



* **docker build . -t simple-php-app** komutu ile container kurulumu yapılıyor.

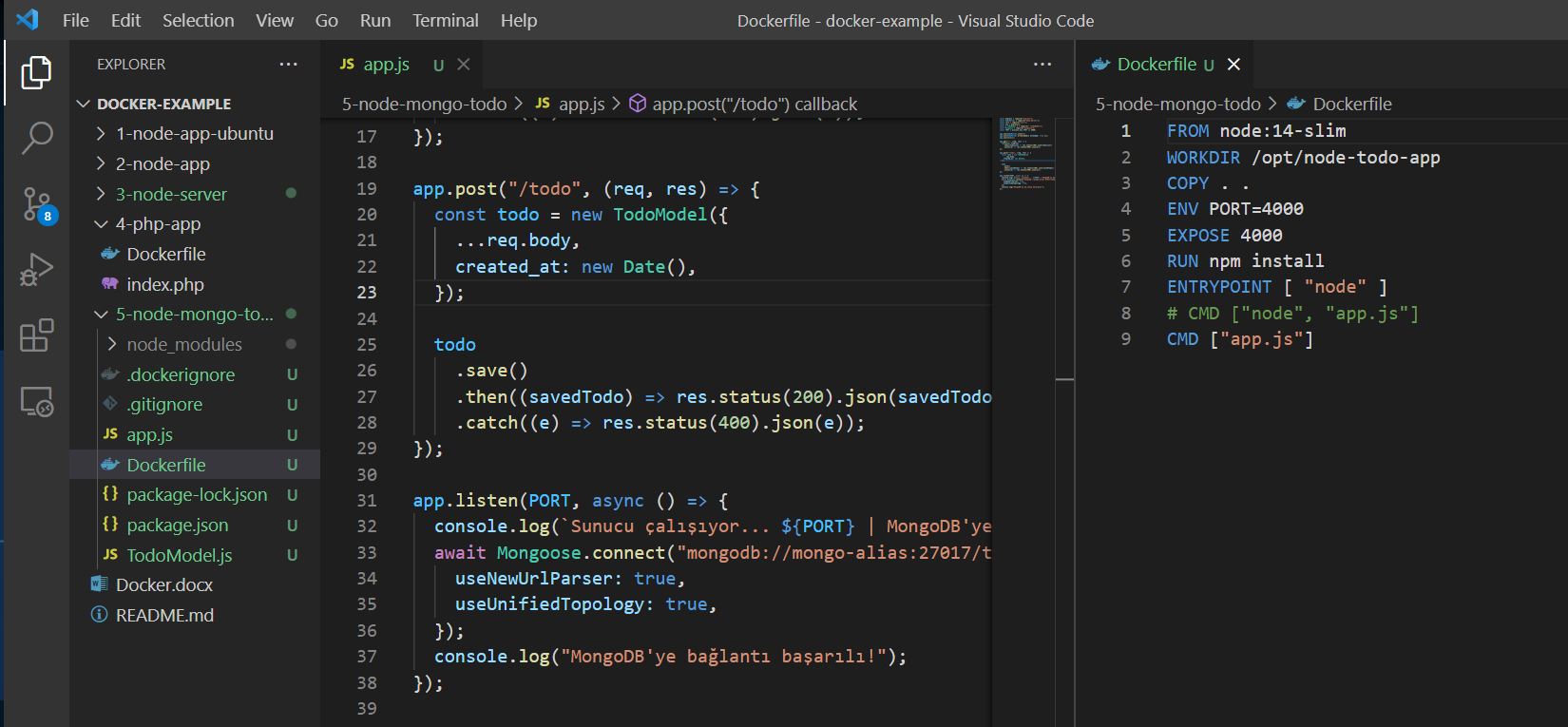


* **docker run -p 8080:80 simple-php-app** komutu ile container başlatılıyor.

**Örnek 5: Node Mongo Todo Uygulaması**

“node” ve “mongo” olarak 2 adet container kurulacak. 2 ayrı container link yöntemi ile bağlanacak.

Projede node image container kurulumu için dockerfile komutları tanımlandı.



* **docker build . -t todo-app** komutu ile container kurulumu yapılıyor.
* **docker run --name mongo-server -p 27017:27017 -d** **mongo** komutu ile mongo container kurulumu yapılıyor.
* **docker run --link mongo-server:mongo-alias -p 4000:4000 todo-app** komutu node-server projesini çalıştırıyor. 4000 portunu kullanıyor. link yöntemi ile mongo-server container ınına bağlanıyor. (mongo-alias mongo için default tanımlamadır)

