بخش اول

کسرا نوربخش 810100230

هومان مشیری امین 810100210

# مقدمه

در این بخش از تمرین به کار با Docker پرداختیم.

**1. تولد بیبی !**

**داکرفایل:**

ابتدا برای پروژه go داکر فایلی را نوشتیم که اندازه کمی در image خروجی داشته باشد، سپس با استفاده از دستور go mod init، یک go module برای آن ساختیم و با استفاده از دستور docker build -t go-webpage . آن را بیلد کردیم و با استفاده از docker run -d -p 8080:80 --name go-web go-webpage کانتینر را اجرا کردیم(همچنین نیاز بود تا از نسخه go ای استفاده بکنیم که بتواند با module اجرا شود که 1.24 alpine را انتخاب کردیم)، اجرای صحیح سایت:

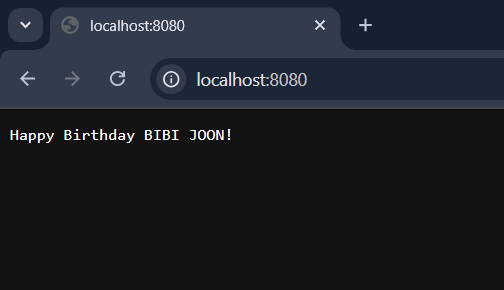
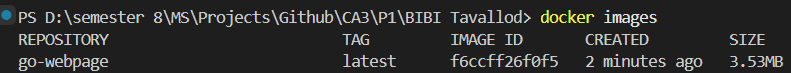
****

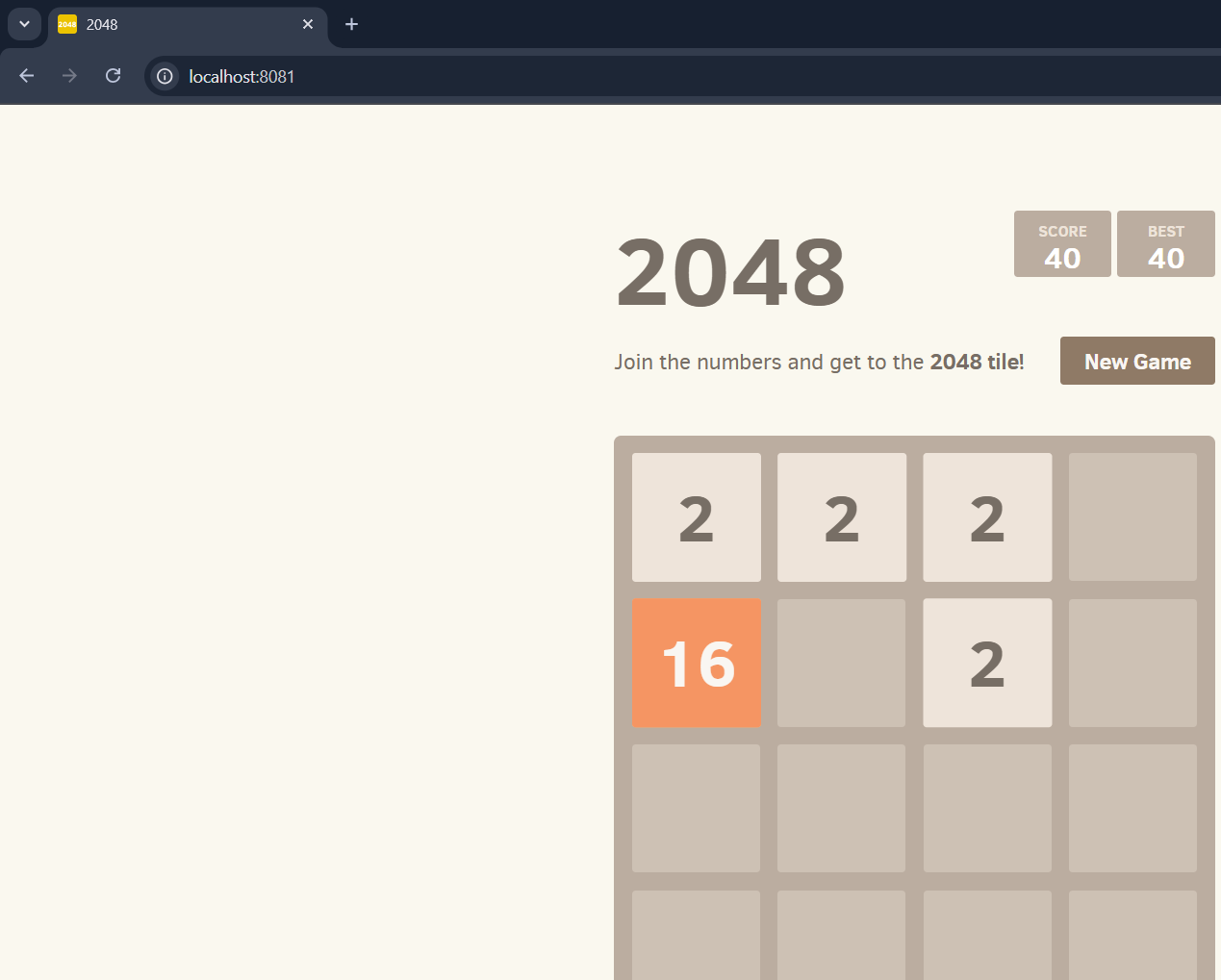
Image اولیه ساخته شده حجمش برابر با 7.88MB بود که به سراغ کمتر کردن آن رفتیم و به نتیجه زیر رسیدیم:

****

وقتی توانستیم که حجم را کم بکنیم که از upx استفاده کردیم که در واقع یک Ultimate Packer برای کد های اجرایی می باشد. با استفاده از -ldflags="-s -w" بعد از دستور go build اطلاعات مربوط به دیباگ را هم حذف کردیم تا سایز را کاهش دهیم، در نهایت با استفاده از upx --best --lzma -q که فلگ های بهترین متد فشرده سازی و الگوریتم lzma برای فشرده سازی و در مد quiet به ترتیب می باشد، باز هم حجم را کاهش دادیم. داکرفایل multistage می باشد و در stage بعدی ما از scratch که یک base image خالی می باشد شروع می کنیم و برنامه تولید شده از stage قبل را در آن قرار می دهیم.

برای بازی 2048 هم کد آن را clone کردیم و dockerfile بسیار ساده ای را نوشتیم: FROM nginx:alpine

COPY . /usr/share/nginx/html چون کد بازی فقط html/css بود. اجرای کانتینر:



**داکرکامپوز:**

بخش سبتن

**2. سوآرم!**

بخش سبتن

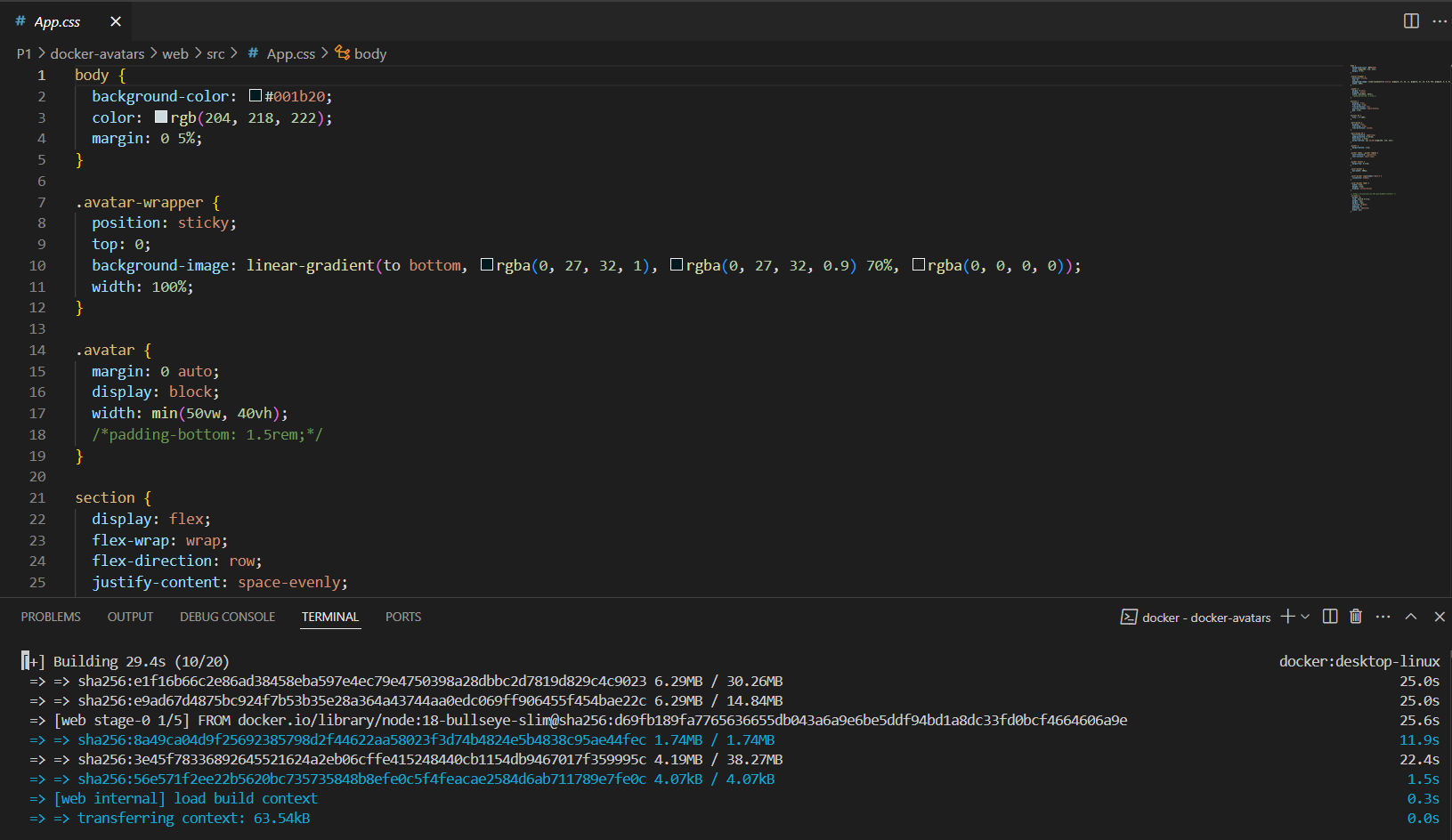
**امتیازی**

بخش سبتن

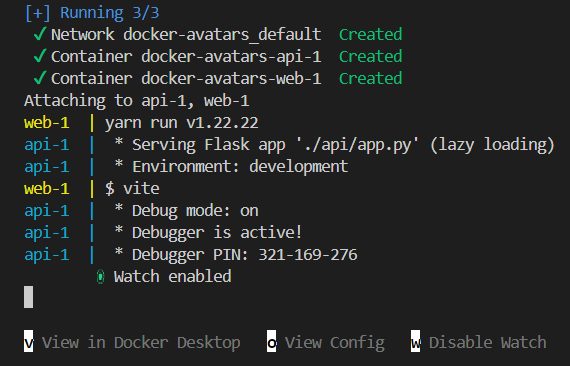
**سوالات تشریحی**

1. دستور docker compose watch برای این آمد تا فرآیند توسعه کد را روان تر کند و به نوعی قابلیت hot-reload را اضافه می کند. این دستور به صورت هوشمند به فایل های تغییر داده شده نگاه می کند و آن ها را با کانتینر در حال اجرا sync می کند و حتی در صورت نیاز می تواند که restart هم بکند. این قابلیت به خصوص در هنگام develop کاربردی است(خصوصاً سمت front) چون فایل ها مدام عوض می شوند و می توانیم تغییر را ببینیم. برای دیدن این موضوع در عمل پروژه را با استفاده از git clone <https://github.com/dockersamples/avatars> کلون کردیم. با استفاده از دستور:

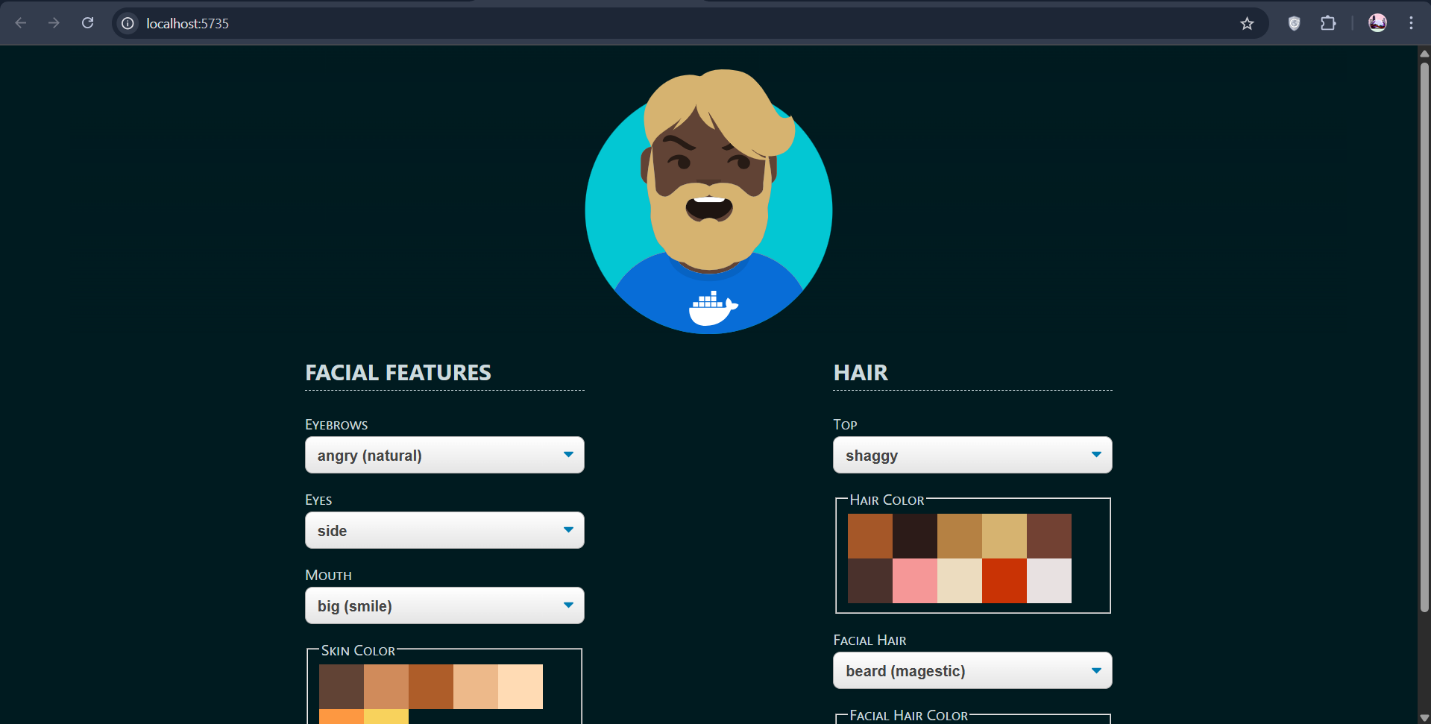
docker compose up –build پروژه را بیلد کردیم و بالا آوردیم، در فایل compose.yaml دیدیم که مسیر web واچ شده است پس به داخل مسیر web/src/ رفتیم تا css موجود در App.css را تغییر دهیم. در عکس زیر می توانیم فایلی که می خواهیم در آن تغییر دهیم و همچنین مراحل build شدن را مشاهده کنیم:



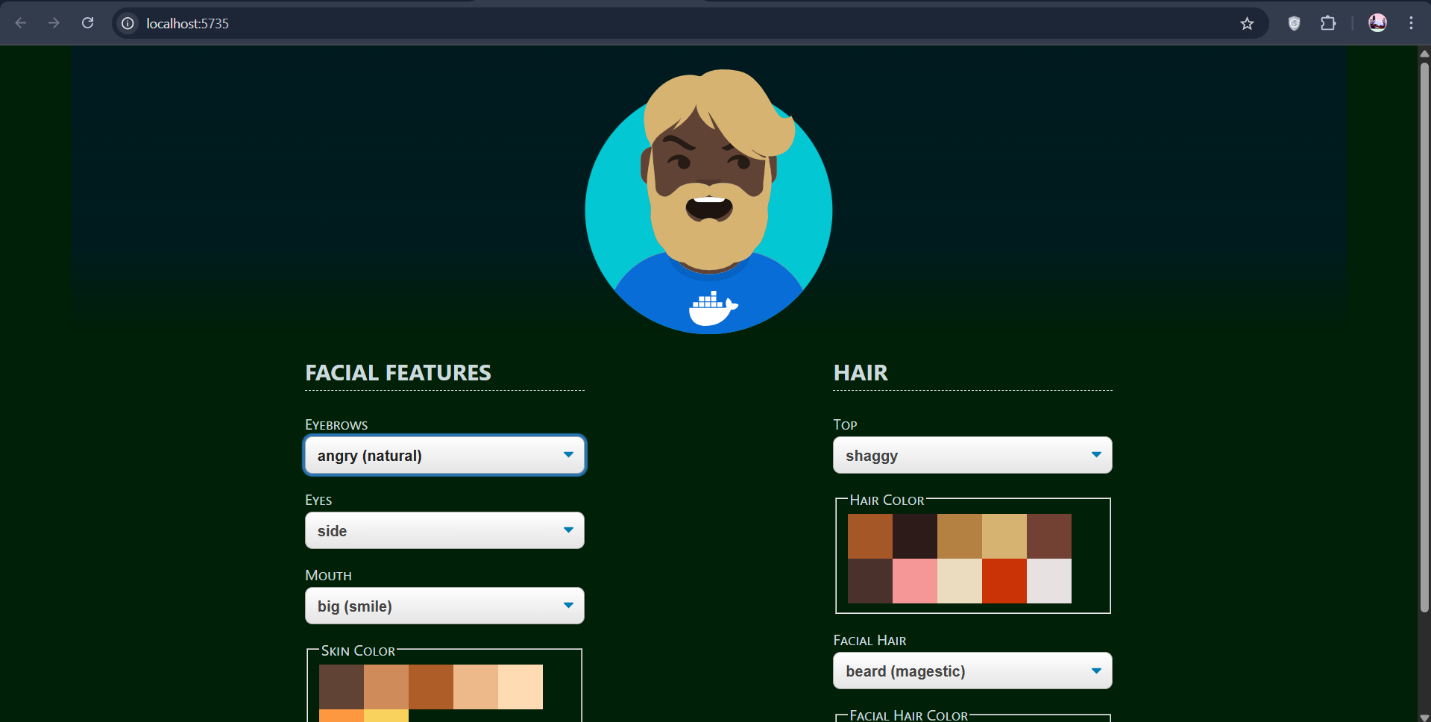
بعد از بیلد موفقیت آمیز با زدن w توانستیم watch را فعال کنیم:



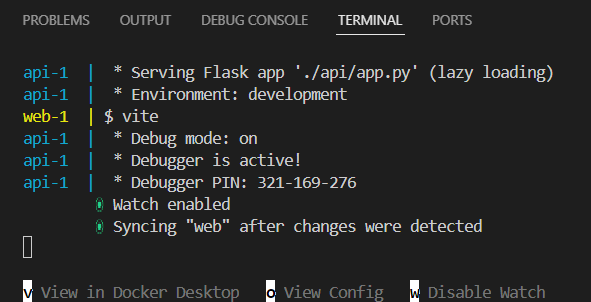
سایتی که بعد از اجرای پروژه بالا آمد:



سپس با استفاده از تغییر background color، آن را به سبز تغییر دادیم:



که دیگر نیاز به rebuild نبود و خودش اتومات sync شد:



2. docker bake در واقع یک ابزار سطح بالا برای ارکستراسیون مابین docker image ها می باشد. با استفاده از آن می توانیم چندین image را به صورت موازی با استفاده از یک فایل کانفیگ، بیلد کنیم. استفاده عملی از آن می تواند در یک رپو واحد که شامل چندین microservice است باشد، همچنین بیلد هایی بر روی چندین پلتفرم مختلف. دستور docker wasm به ما این امکان را می دهد تا wasm ها و یا web assembly module ها را به عنوان کانتینر با داکر ران کنیم. حال wasm چیست؟ wasm یک فرمت باینری است که در محیط sandbox اجرا می شود. استفاده عملی از این قابلیت می تواند در یک تابع سبک و بدون سرور مانند(FaaS ها) باشد و یا همچنین در IoT و یا لبه شبکه به این دلیل که بسیار سایز کمی دارند.

3. RUN برای اجرا کردن دستورات بیلد است که یک لایه image اضافه می کند در سایز image اثر گذار است، COPY فایل ها را به image کپی می کند و یک لایه image اضافه می کند و در سایز image اثر گذار است، ADD فایل ها را هم می تواند کپی کند و هم دانلود و unpack که باز هم به مانند دو دستور قبل هم لایه اضافه می کند و هم در سایز اثر گذار است(منظور از سایز image در اینجا، سایز نهایی image می باشد)، CMD کامند دیفالت کانتینر می باشد و نه لایه ای اضافه می کند و نه در حجم اثری دارد، ENTRYPOINT دستور استارتاپ فیکس می باشد که باز هم رفتارش به مانند CMD است، WORKDIR ورکینگ دایرککتوری را تنظیم می کند و یک لایه اضافه می کند اما اندکی در سایز اثر دارد.

# لینک های استفاده شده

# 1. استفاده از فشرده سازی برای داکرفایل go:

# [https://www.dockerfile.app/dockerfile/2dfedcd8-87ad-4edd-924e-098f41fad00c](https://www.dockerfile.app/dockerfile/2dfedcd8-87ad-4edd-924e-098f41fad00c%20)