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:

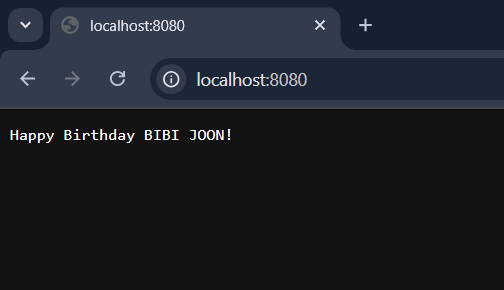
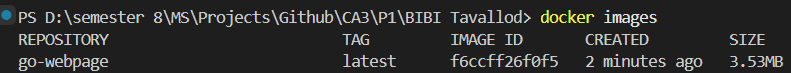
****

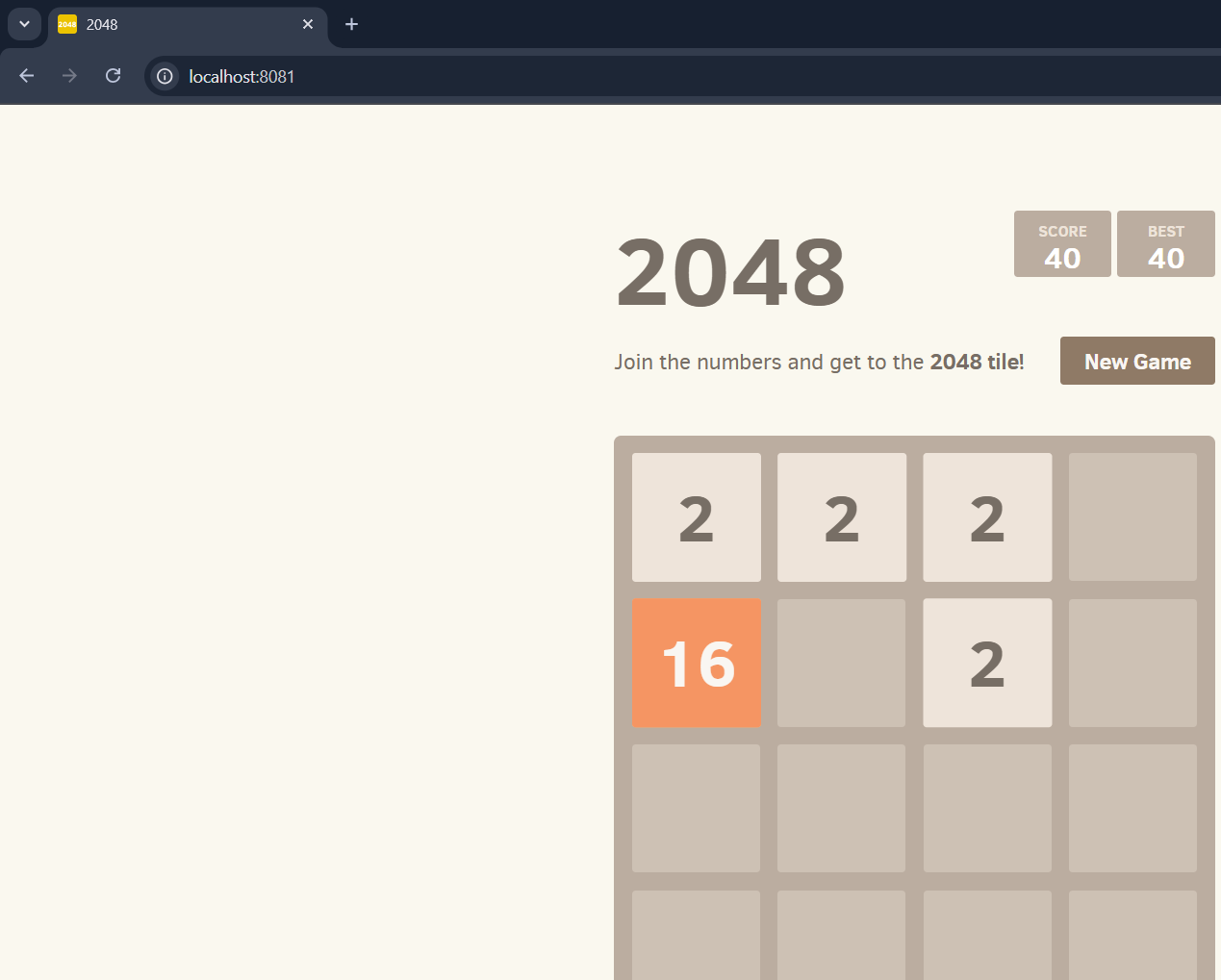
Image اولیه ساخته شده حجمش برابر با 7.88MB بود که به سراغ کمتر کردن آن رفتیم و به نتیجه زیر رسیدیم:

****

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

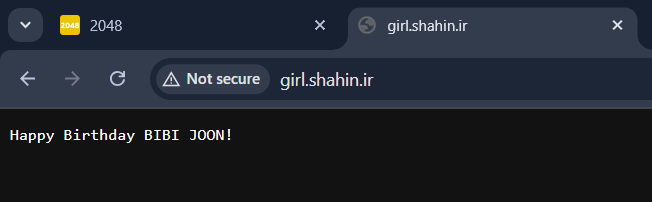
COPY . /usr/share/nginx/html چون کد بازی فقط html/css بود. اجرای کانتینر:

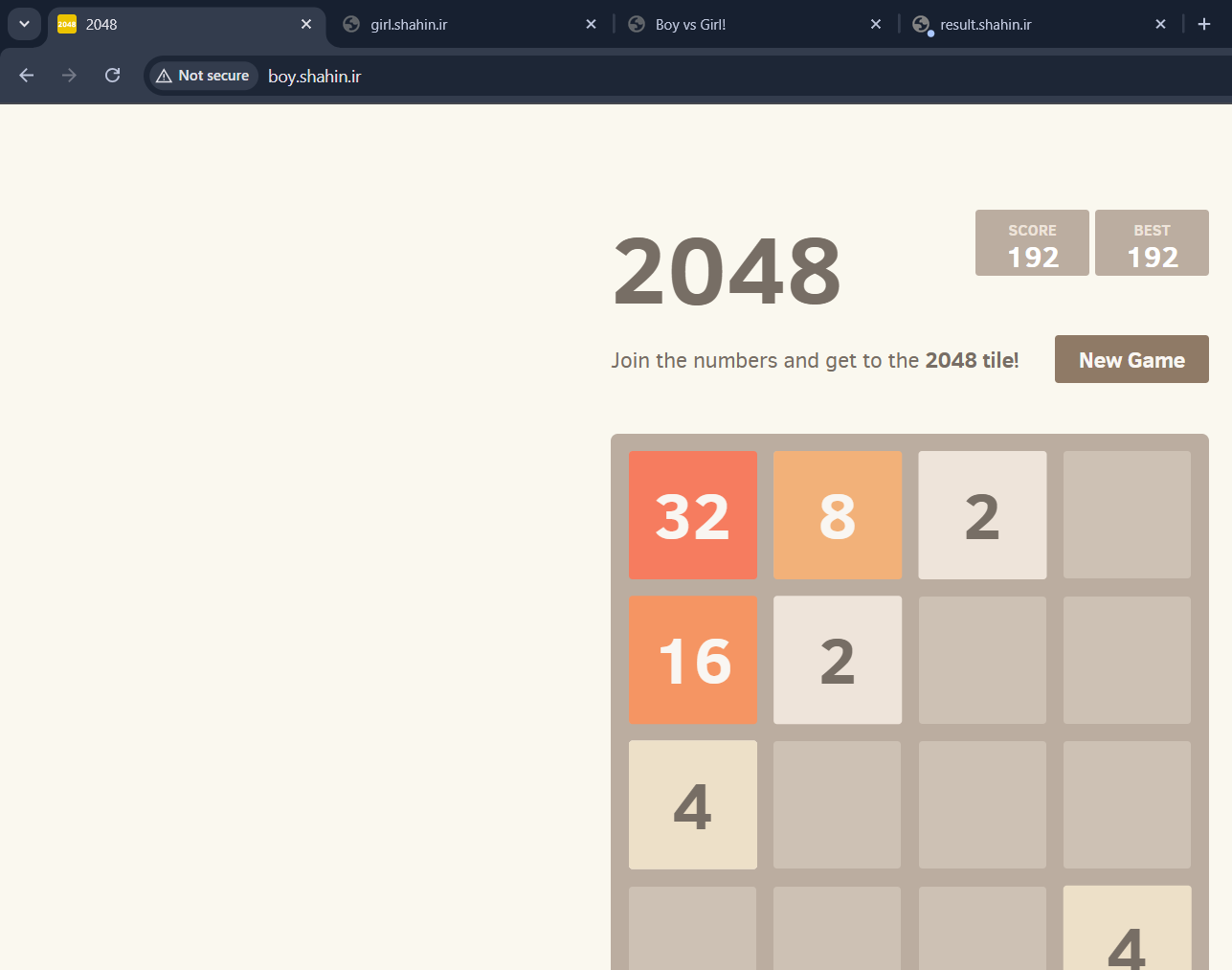


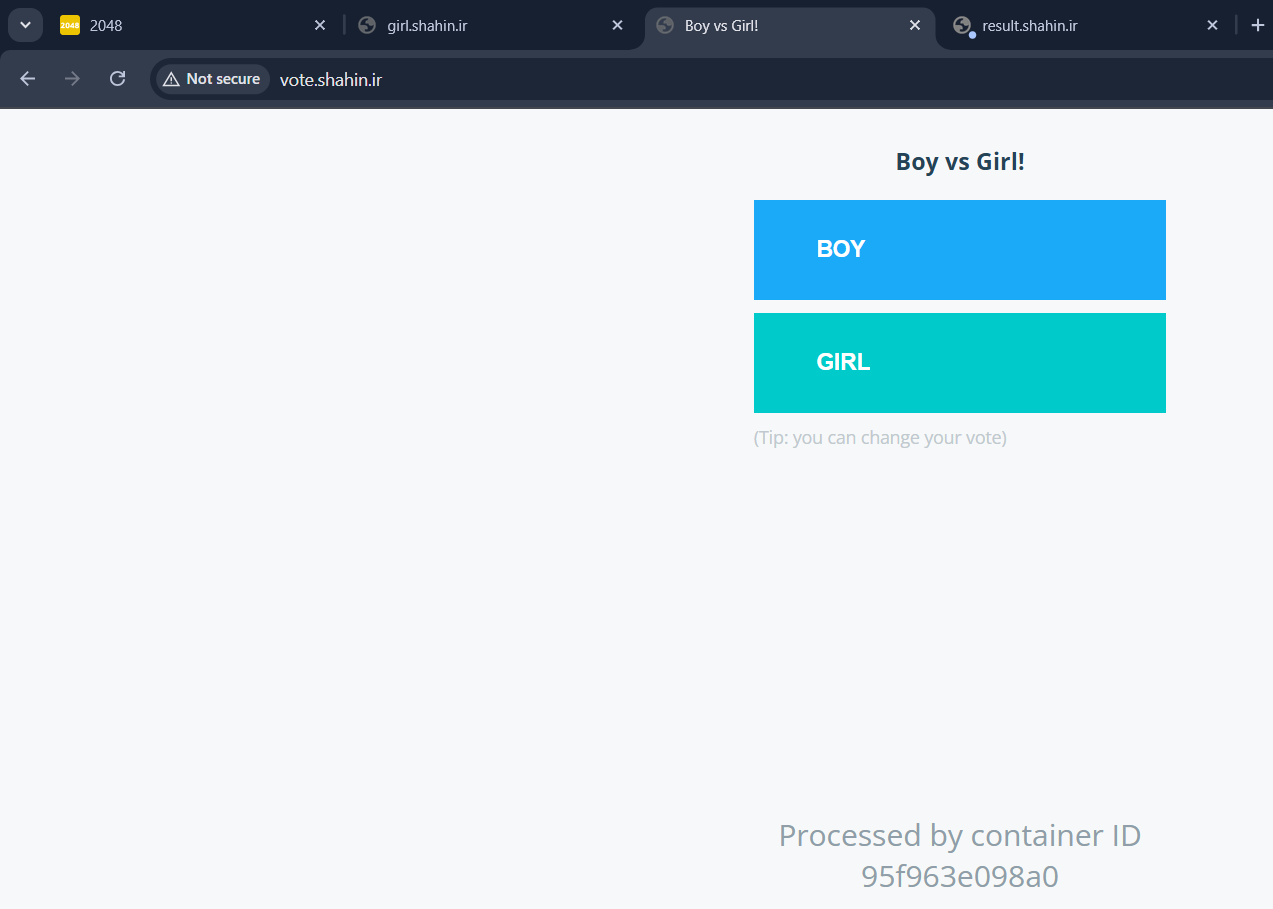
**داکرکامپوز:**

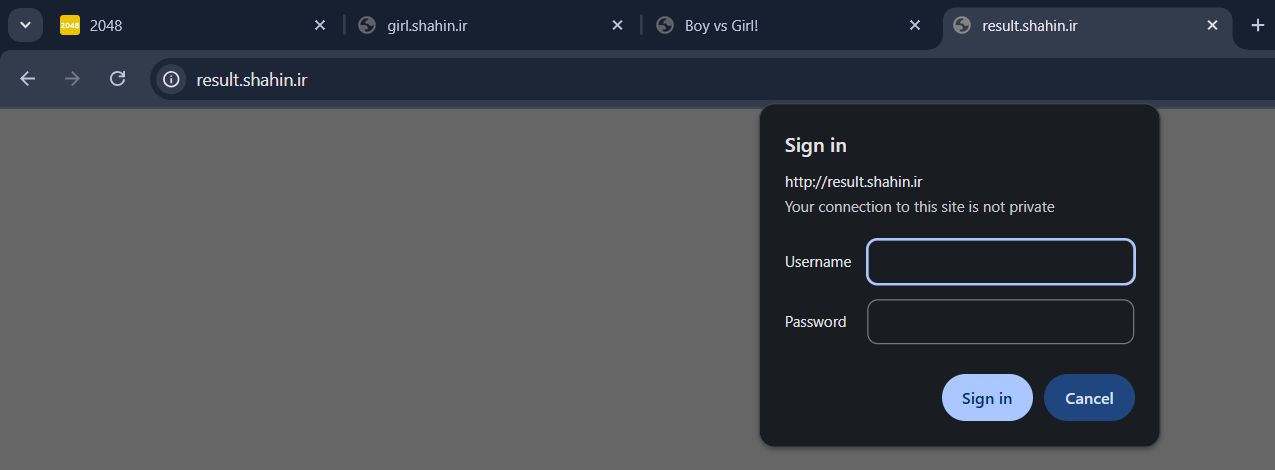
در این بخش با مشکلات بسیار بسیار زیادی بر خوردیم 😊 که رفع شدن آن ها را می توانیم در github مشاهده کنیم.

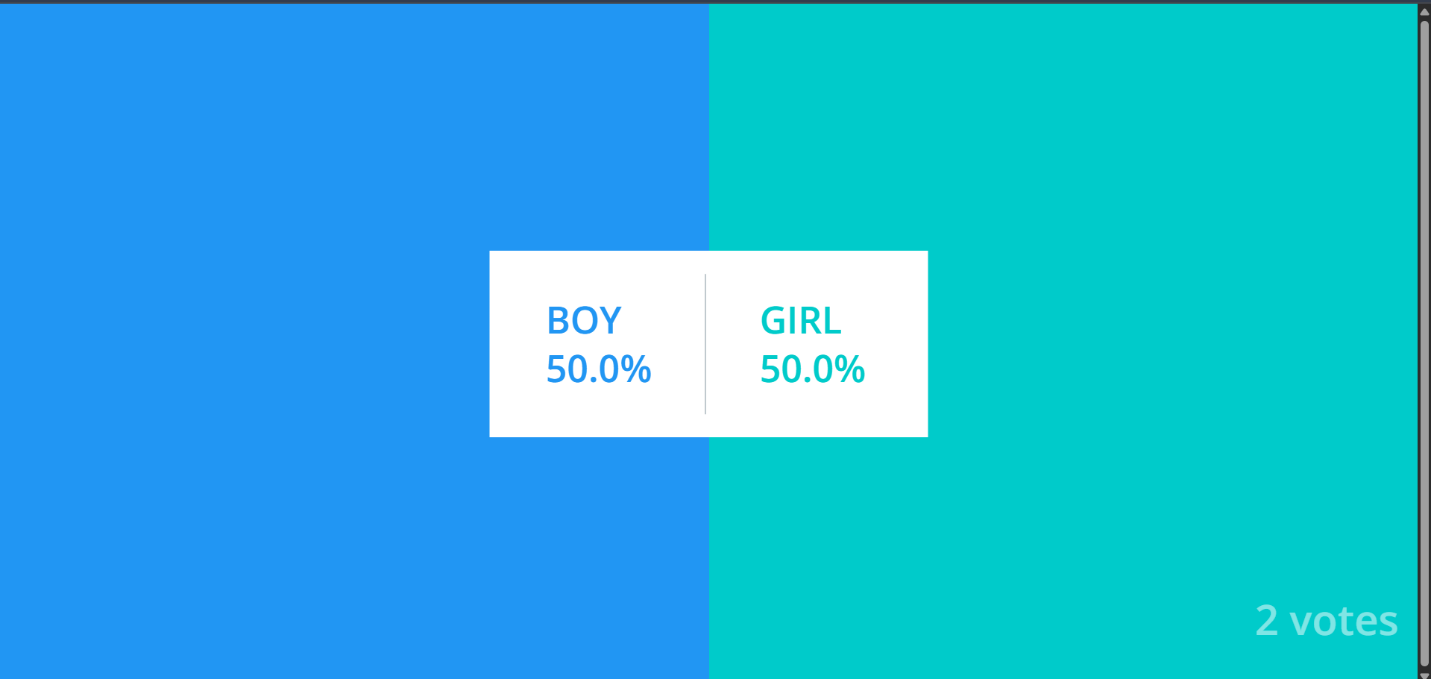
با استفاده از توضیحات داده شده، تغییرات داده شد و نتیجه را می بینیم:









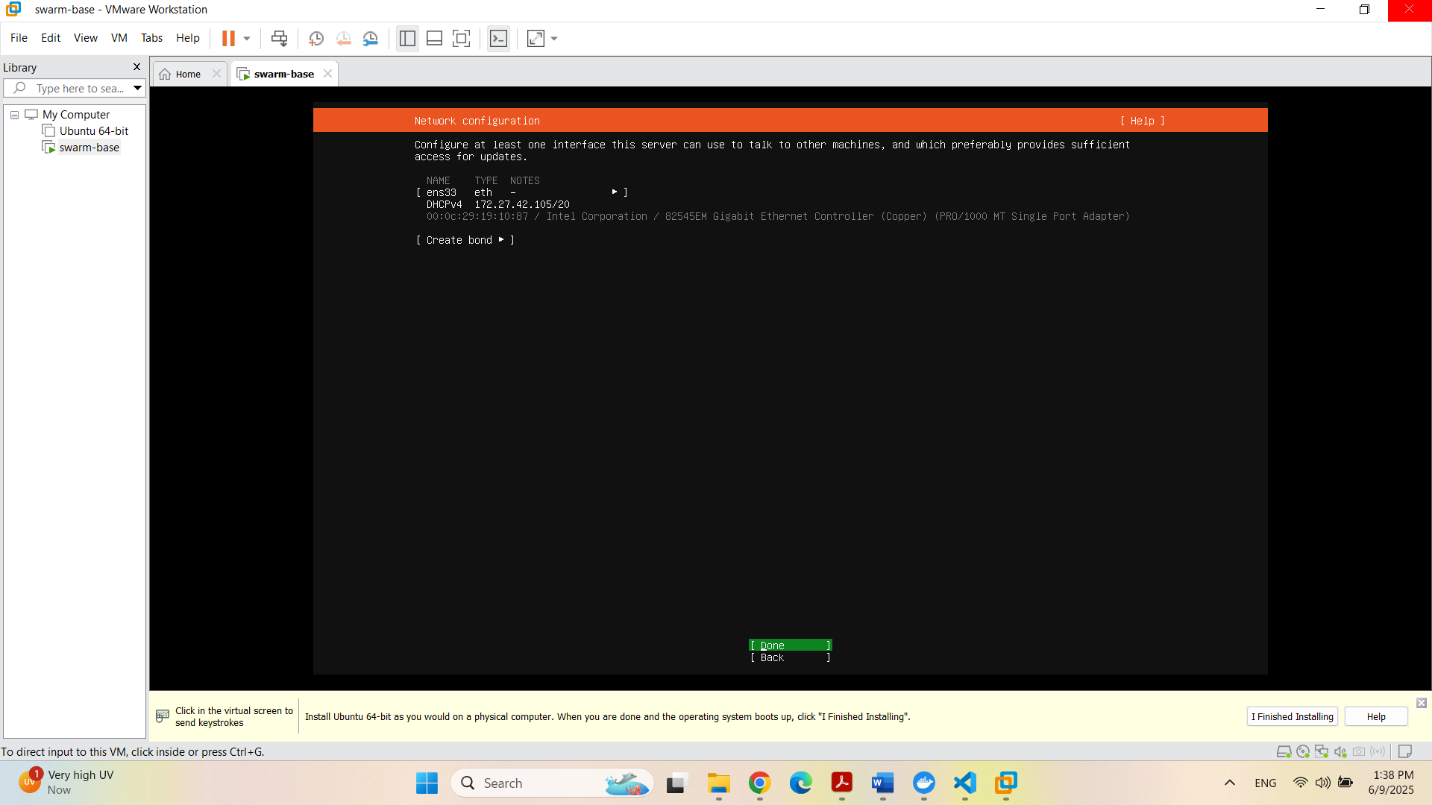
****

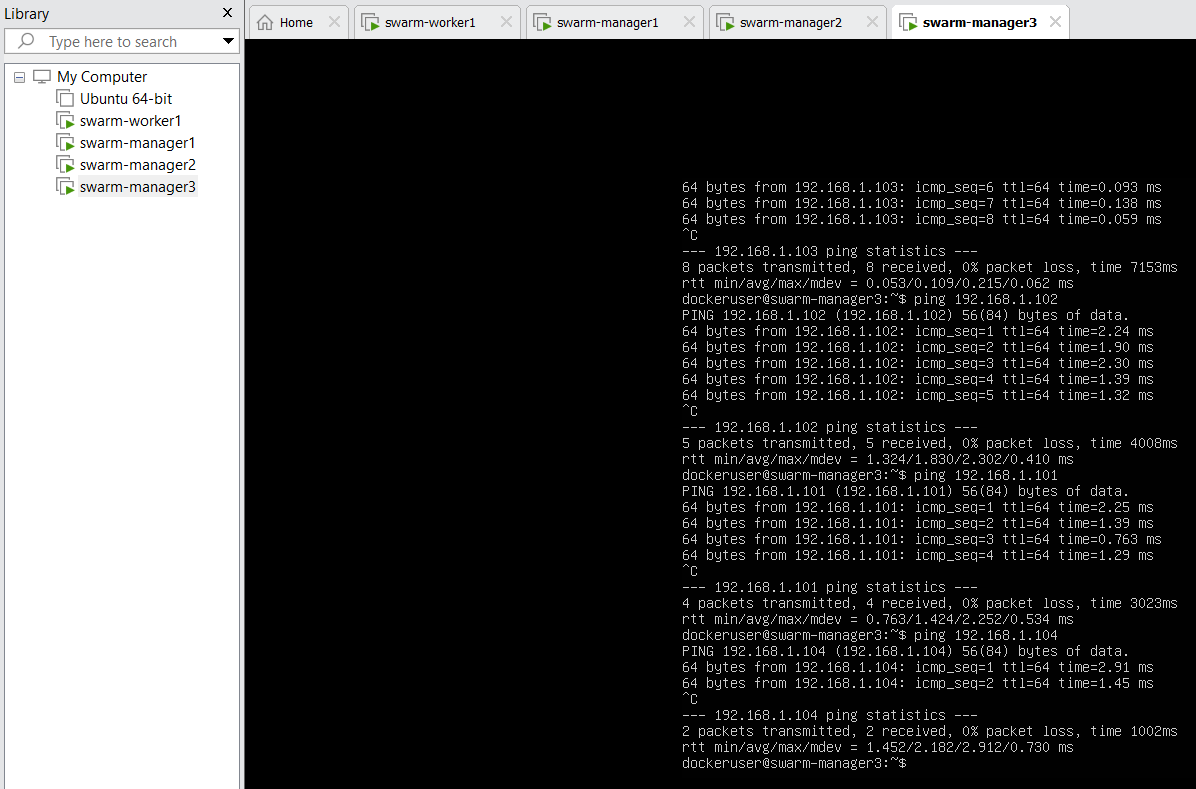
**2. سوآرم!**

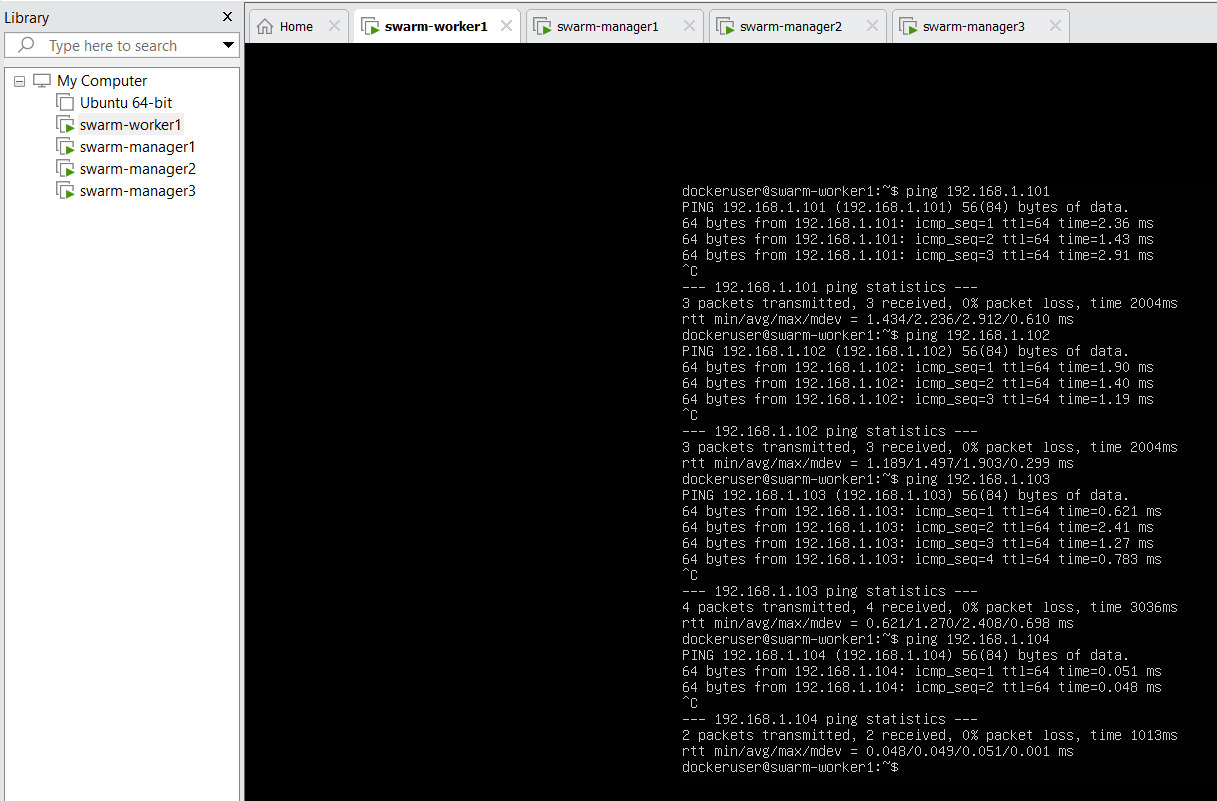
برای این بخش هم از ایزو اوبونتو سرور 2022 استفاده کردیم و بعد از نصب و راه اندازی کامل آن به همراه docker ، از آن کلون گرفتیم و دستورات لازم را زدیم و در آخر آخر هم با استفاده از دستور زیر اپ را بر آن دپلوی کردیم:

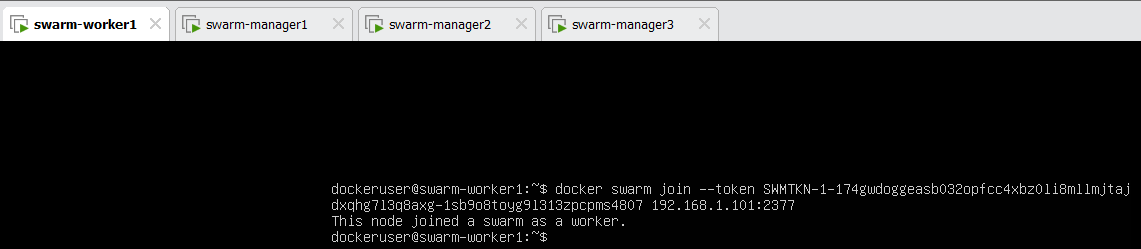
docker service create --name nginx --replicas 5 --publish published=80,target=80 nginx

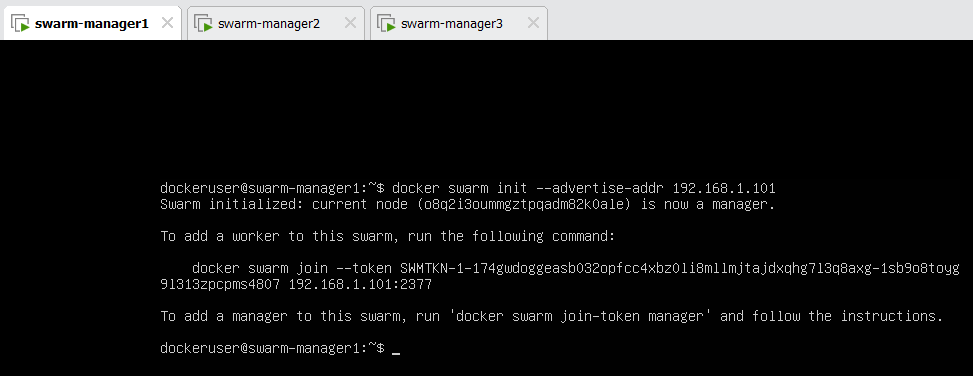
در زیر می توانیم مراحل را از قبیل نصب، اضافه شدن و ساخت swarm و غیره را ببینیم

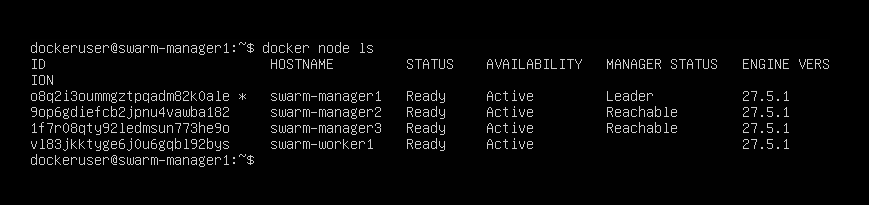


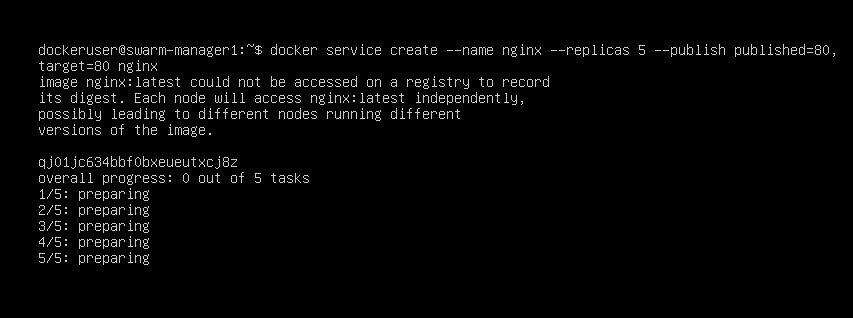












**امتیازی**

برای دسترسی از بیرون می توانیم از آیپی هایی که پیش تر به هر کدام از ورکر و یا مستر ها دادیم استفاده بکنیم، اما بهتر است برای مستر باشد. چون داکر سوارم از روتینگ مش استفاده می کند و تمامی نود ها ترافیک را بر روی پورت های شناخته شده قبول می کنند. درخواست ها هم به صورت اتوماتیک به کانتینر های available روت می شوند و مهم نیست بر روی کدام نود هستند. برای dns هم باید domain مون رو پوینت کنیم به Ip آدرس یکی یا بیشتر از مستر نود ها. بیشترش برای availability بیشتر هست.

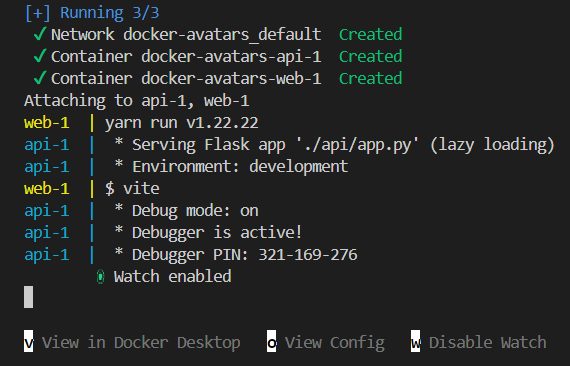
**سوالات تشریحی**

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:

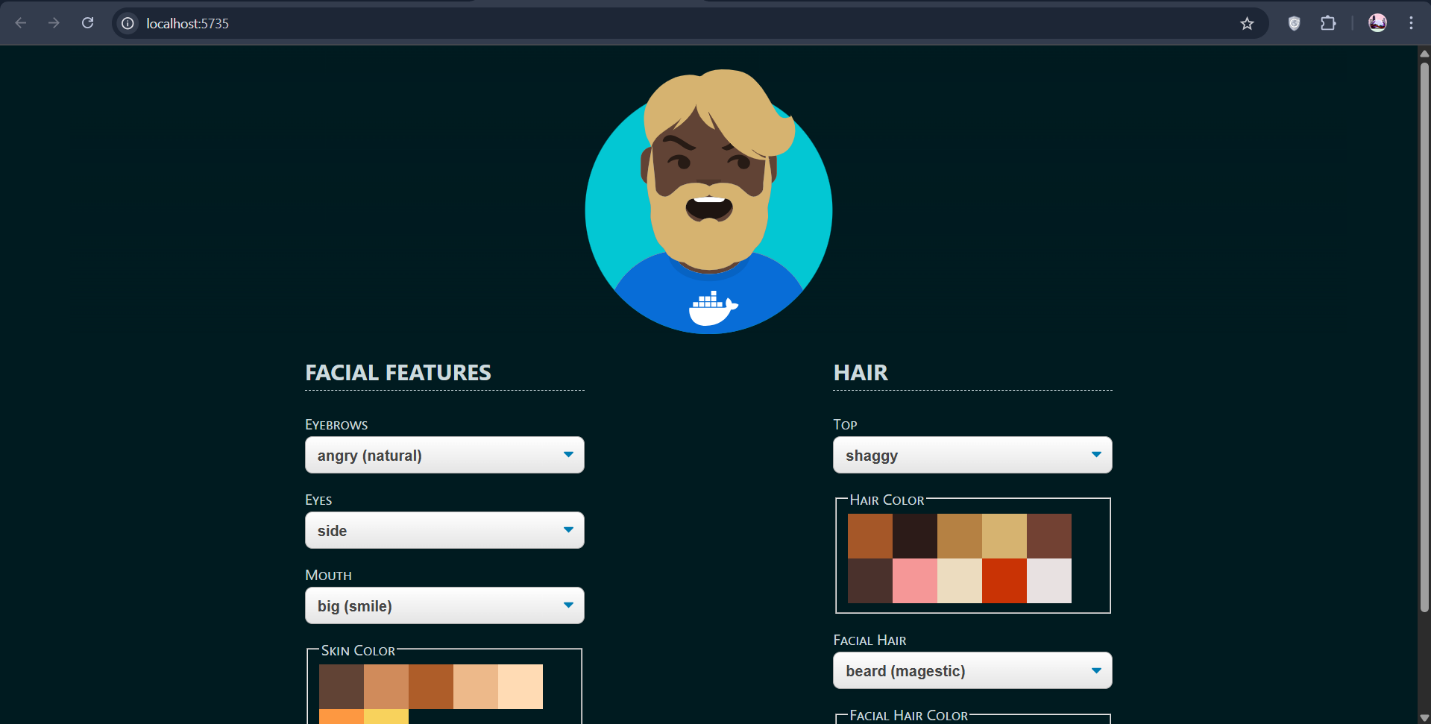
docker compose up –build پروژه را بیلد کردیم و بالا آوردیم، در فایل compose.yaml دیدیم که مسیر web واچ شده است پس به داخل مسیر web/src/ رفتیم تا css موجود در App.css را تغییر دهیم. در عکس زیر می توانیم فایلی که می خواهیم در آن تغییر دهیم و همچنین مراحل build شدن را مشاهده کنیم:



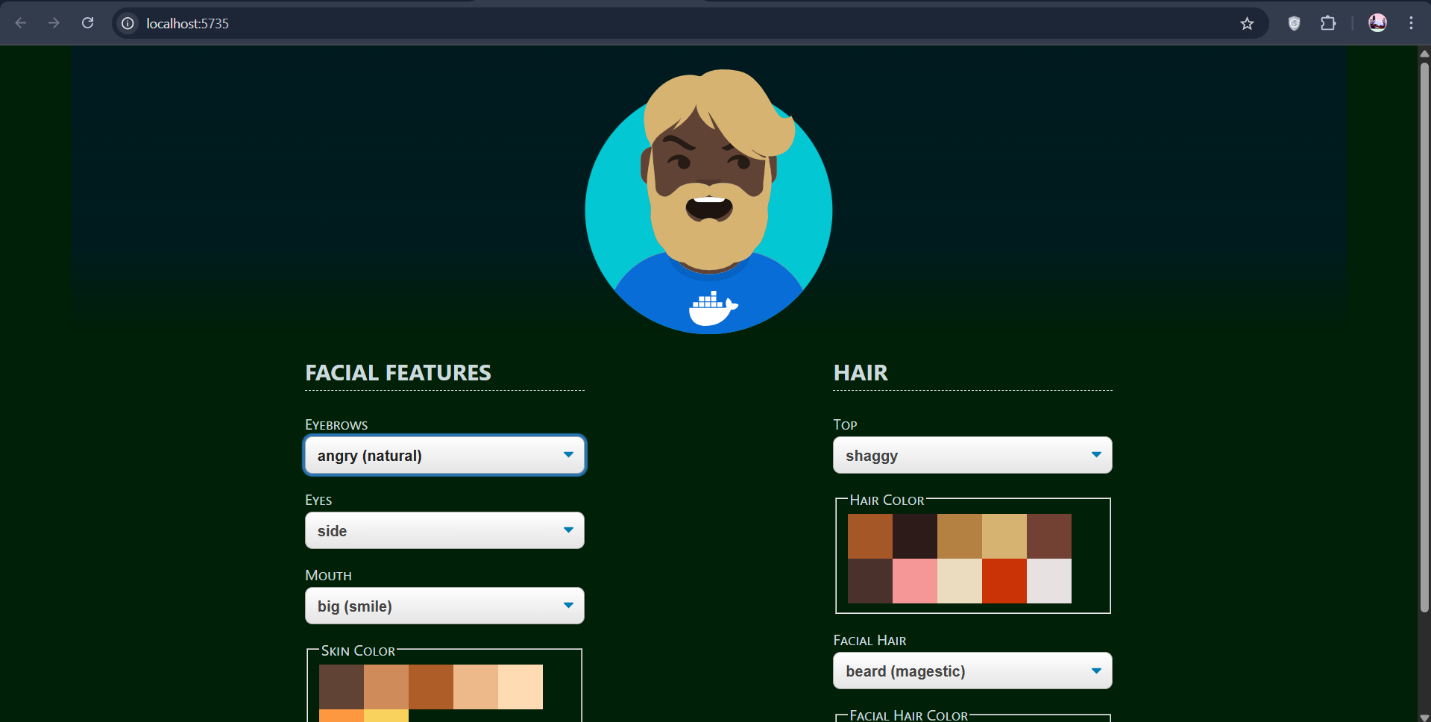
بعد از بیلد موفقیت آمیز با زدن w توانستیم watch را فعال کنیم:



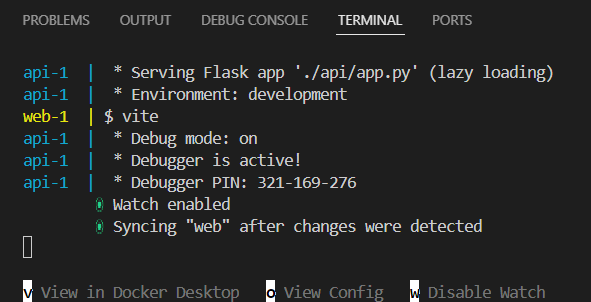
سایتی که بعد از اجرای پروژه بالا آمد:



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)

# 2. تمامی لینک های گفته شده در صورت پروژه

# 3. برای بخش هایی از داکرفایل ها از ChatGPT کمک گرفته شد:

<https://chatgpt.com/share/68452852-43f0-800d-93a6-d51755237d1e>

چت:

We have a simple golang code for a webpage also the https://github.com/gabrielecirulli/2048 for a 2048 game. We want to have both on Shahin.ir that we cant! We want to solve this using the reverse proxy(like nginx) and send these 2 webpages back to reverse proxy and run them up on different sub domains. We also have this voting app project: https://github.com/dockersamples/example-voting-app that we want to deploy and have a voting but not for now. I will tell you step-by-step what I want to do in the next message.

**ChatGPT said:**

Great! You're on the right track thinking about using a **reverse proxy like Nginx** to serve multiple web applications under different **subdomains** (e.g., game.shahin.ir, webpage.shahin.ir, etc.).

To summarize your setup and intentions:

**You have:**

1. A **simple Go-based webpage**.
2. The **2048 game** from GitHub (gabrielecirulli/2048).
3. A **voting app** (dockersamples/example-voting-app) — to be deployed later.
4. A domain: shahin.ir.

**You want to:**

* Host the **Go webpage** and the **2048 game** on subdomains like:
  + webpage.shahin.ir → Go app
  + game.shahin.ir → 2048 game
* Use **Nginx as a reverse proxy** to route these subdomains to the correct backends.

I’m ready for your step-by-step instructions. Go ahead and tell me what you want to do first, and I’ll guide you through it.

**You said:**

So, I am using docker desktop on windows 11, I want to have a dockerfile for bulding the codes and having a docker image for the https://github.com/gabrielecirulli/2048 game and the go webpage I said. This is the go code: package main import ( "fmt" "log" "net/http" ) func handler(w http.ResponseWriter, r \*http.Request) { fmt.Fprintf(w, "Happy Birthday BIBI JOON!") } func main() { http.HandleFunc("/", handler) log.Println("Starting server on :80") err := http.ListenAndServe(":80", nil) if err != nil { log.Fatal("Error starting server: ", err) } }. I want the minimum possible dockerfile for the go code to reduce the size of the output image as possible.

**ChatGPT said:**

Perfect — here's how we’ll proceed:

**✅ Part 1: Dockerfile for the Go Webpage**

You want a minimal Docker image. The **best practice** for that is to use **multi-stage builds**, compiling your Go code statically and using scratch or alpine in the final image.

**🔧 Dockerfile for Go Webpage (minimal)**

dockerfile

CopyEdit

# Build stage

FROM golang:1.22-alpine AS builder

WORKDIR /app

COPY . .

RUN go build -ldflags="-s -w" -o main

# Final stage: minimal image

FROM scratch

COPY --from=builder /app/main /main

EXPOSE 80

ENTRYPOINT ["/main"]

**👉 Instructions:**

1. Place your main.go in a folder.
2. Save the above Dockerfile in the same folder.
3. Build the image:

sh

CopyEdit

docker build -t go-webpage .

1. Run the container:

sh

CopyEdit

docker run -d -p 8080:80 --name go-web go-webpage

Now visiting http://localhost:8080 should show:  
**“Happy Birthday BIBI JOON!”**

**✅ Part 2: Dockerfile for 2048 Game**

This game is just static HTML/JS/CSS. You can serve it from a tiny web server like **Nginx**.

**🔧 Dockerfile for 2048**

dockerfile

CopyEdit

FROM nginx:alpine

COPY . /usr/share/nginx/html

**👉 Instructions:**

1. Clone the repo:

sh

CopyEdit

git clone https://github.com/gabrielecirulli/2048.git

cd 2048

1. Save the above Dockerfile in the root of the cloned repo.
2. Build it:

sh

CopyEdit

docker build -t game-2048 .

1. Run it:

sh

CopyEdit

docker run -d -p 8081:80 --name game game-2048

Now http://localhost:8081 will load the game.

Let me know when you're ready to set up the **reverse proxy** (Nginx) with **Docker** to map:

* webpage.shahin.ir → Go container
* game.shahin.ir → 2048 container

I'll walk you through that too.

**You said:**

So now I want to change the docker compose file of the voting app: https://github.com/dockersamples/example-voting-app, and add 3 services related to nginx reverse proxy, go webpage, and 2048 game to it. all services must deploy from a single docker compose, if we have dependencies we must control the order of run in compose, all services like database and ... that do not need to be seen from outside must be on a different network, just port 80 must be published and no other ports, with setting dns of the laptop, subdomains of vote and result for the voting app and subdomains of boy(2048 game) and girl(go web page) from Shahin.ir must be configured in the reverse proxy. Also we must change the voting app code to have the boy and girl for the options to vote. We must add a config in reverse proxy such that result page needs user and pass for showing. users are arbitary but passwords must be minimum 30 random characters. All user and pass and environment variables and configs in the compose file must be read from .env. At the end the compose file runs with docker compose up -d and all 4 domains tell above must be accessible from browser.

**ChatGPT said:**

Great — here’s the full plan to create a **single unified Docker Compose setup** that:

**🔧 Serves 4 Subdomains:**

* girl.shahin.ir → Go Webpage
* boy.shahin.ir → 2048 Game
* vote.shahin.ir → Voting App Frontend
* result.shahin.ir → Voting App Results (protected with basic auth)

**📋 Key Requirements:**

1. **Only port 80 exposed**.
2. Internal services like Redis/DB on a **private network**.
3. Use **Nginx** as a reverse proxy, reading subdomain config.
4. Use **.env file** for all environment variables.
5. Change voting app to have options boy and girl.
6. Use **basic auth** for results, password ≥ 30 random characters.

**✅ Directory Structure**

css

CopyEdit

project-root/

├── .env

├── docker-compose.yml

├── nginx/

│ ├── default.conf

│ └── htpasswd

├── go-web/

│ ├── main.go

│ └── Dockerfile

├── 2048/

│ ├── [2048 game files]

│ └── Dockerfile

└── voting-app/

└── [cloned and modified voting app]

**📄 Step 1: .env Example**

env

CopyEdit

# .env

# Reverse Proxy Basic Auth

BASIC\_AUTH\_USER=admin

BASIC\_AUTH\_PASSWORD=$apr1$Z7v...G5f73oH/zLn1 # generated via htpasswd

# Domain names

DOMAIN=shahin.ir

SUBDOMAIN\_GIRL=girl

SUBDOMAIN\_BOY=boy

SUBDOMAIN\_VOTE=vote

SUBDOMAIN\_RESULT=result

# Docker networks

INTERNAL\_NETWORK=internal\_net

PUBLIC\_NETWORK=proxy\_net

Generate .htpasswd entry using:

bash

CopyEdit

htpasswd -nb admin 'YOUR-30-CHAR-RANDOM-PASSWORD'

**📄 Step 2: nginx/default.conf**

nginx

CopyEdit

server {

listen 80;

server\_name ${SUBDOMAIN\_GIRL}.${DOMAIN};

location / {

proxy\_pass http://go-web:80;

}

}

server {

listen 80;

server\_name ${SUBDOMAIN\_BOY}.${DOMAIN};

location / {

proxy\_pass http://game-2048:80;

}

}

server {

listen 80;

server\_name ${SUBDOMAIN\_VOTE}.${DOMAIN};

location / {

proxy\_pass http://voting-app:80;

}

}

server {

listen 80;

server\_name ${SUBDOMAIN\_RESULT}.${DOMAIN};

auth\_basic "Restricted";

auth\_basic\_user\_file /etc/nginx/.htpasswd;

location / {

proxy\_pass http://result-app:80;

}

}

**📄 Step 3: docker-compose.yml**

yaml

CopyEdit

version: '3.9'

services:

reverse-proxy:

image: nginx:alpine

ports:

- "80:80"

volumes:

- ./nginx/default.conf:/etc/nginx/conf.d/default.conf

- ./nginx/htpasswd:/etc/nginx/.htpasswd

networks:

- ${PUBLIC\_NETWORK}

go-web:

build:

context: ./go-web

networks:

- ${PUBLIC\_NETWORK}

game-2048:

build:

context: ./2048

networks:

- ${PUBLIC\_NETWORK}

voting-app:

build:

context: ./voting-app/vote

networks:

- ${INTERNAL\_NETWORK}

- ${PUBLIC\_NETWORK}

depends\_on:

- redis

result-app:

build:

context: ./voting-app/result

networks:

- ${INTERNAL\_NETWORK}

- ${PUBLIC\_NETWORK}

depends\_on:

- db

worker:

build:

context: ./voting-app/worker

networks:

- ${INTERNAL\_NETWORK}

depends\_on:

- redis

- db

db:

image: postgres:15

environment:

POSTGRES\_USER: postgres

POSTGRES\_PASSWORD: example

networks:

- ${INTERNAL\_NETWORK}

redis:

image: redis:alpine

networks:

- ${INTERNAL\_NETWORK}

networks:

${PUBLIC\_NETWORK}:

${INTERNAL\_NETWORK}:

internal: true

**📄 Step 4: Modify Voting App to Use boy and girl**

In voting-app/vote/app.py or wherever the options are hardcoded, change:

python

CopyEdit

options = ["boy", "girl"]

Rebuild image after this change.

**🟢 Final Step: Launching It**

bash

CopyEdit

docker compose up -d

Then point your laptop’s DNS (e.g., via /etc/hosts on WSL or C:\Windows\System32\drivers\etc\hosts on Windows) to:

CopyEdit

127.0.0.1 girl.shahin.ir boy.shahin.ir vote.shahin.ir result.shahin.ir

Visit in browser:

* http://girl.shahin.ir
* http://boy.shahin.ir
* http://vote.shahin.ir
* http://result.shahin.ir → prompts for auth

**✅ Summary**

You now have:

* All services running from **one compose file**.
* **Nginx** routing traffic based on subdomain.
* **Only port 80** exposed.
* Clean environment via .env.
* Reverse proxy auth-secured.

Let me know if you want me to generate any of these files, or if you’re ready to move to DNS and deployment!

**You said:**

I ran this under nginx directory that has a .conf file but this happend: PS D:\semester 8\MS\Projects\Github\CA3\P1\nginx> htpasswd -nb admin 'YOUR-30-CHAR-RANDOM-PASSWORD' htpasswd : The term 'htpasswd' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the spelling of the name, or if a path was included, verify that the path is correct and try again. At line:1 char:1 + htpasswd -nb admin 'YOUR-30-CHAR-RANDOM-PASSWORD' + ~~~~~~~~ + CategoryInfo : ObjectNotFound: (htpasswd:String) [], CommandNotFoundException + FullyQualifiedErrorId : CommandNotFoundException

**ChatGPT said:**

You're seeing that error because htpasswd is not available by default on Windows. It's part of the **Apache HTTP Server utilities**, specifically from the apache2-utils package on Linux or httpd-tools.

**✅ Solution: Generate .htpasswd Without Installing Anything**

**Option 1: Use Docker to Generate It**

You can run a one-time Docker command to generate the .htpasswd file:

powershell

CopyEdit

docker run --rm httpd:alpine htpasswd -nb admin "YOUR-30-CHAR-RANDOM-PASSWORD"

**Example:**

powershell

CopyEdit

docker run --rm httpd:alpine htpasswd -nb admin "a8df9fads7f6as7dfg89adgfasd7f6a8sd7f6a"

You'll get output like:

perl

CopyEdit

admin:$apr1$tWEr4...oA0tG7

Copy that whole line and place it into a file:

**📄 nginx/htpasswd**

perl

CopyEdit

admin:$apr1$tWEr4...oA0tG7

Make sure this file is readable by the Nginx container and mounted in your docker-compose.yml.

Let me know if you want help generating a strong password or automating this!

**You said:**

In the previous response naming's are not like my project structure, I have the 2048 game in the Bazi 2048 directory, the go web page in BIBI Tavallod directory and the voting app in the voting app directory and nginx directory for nginx.

**ChatGPT said:**

Thanks for the clarification! Based on your actual directory structure:

bash

CopyEdit

project-root/

├── Bazi 2048/ # 2048 game

├── BIBI Tavallod/ # Go webpage

├── voting app/ # Voting app

├── nginx/ # Nginx reverse proxy config

├── .env # Environment variables

└── docker-compose.yml # Unified compose file

I'll update everything accordingly — including:

* build paths in the docker-compose.yml
* reverse proxy routes
* comments and volumes

Let me know when you’re ready, and I’ll generate the full **updated docker-compose.yml**, nginx/default.conf, .env, and instructions tailored to these directory names.

**You said:**

yes update all the previous responses in regard to this

**ChatGPT said:**

Perfect. Here’s the **complete updated setup** tailored to your actual directory structure:

**✅ Your Directory Structure:**

bash

CopyEdit

project-root/

├── Bazi 2048/ # 2048 game

│ └── Dockerfile

├── BIBI Tavallod/ # Go webpage

│ └── Dockerfile

├── voting app/ # Voting app

│ ├── vote/

│ ├── result/

│ ├── worker/

│ ├── Dockerfile (optional per service)

│ └── [modified to use "boy" and "girl"]

├── nginx/

│ ├── default.conf

│ └── htpasswd

├── .env

└── docker-compose.yml

**📄 1. .env File (Example)**

env

CopyEdit

# .env

DOMAIN=shahin.ir

SUBDOMAIN\_GIRL=girl

SUBDOMAIN\_BOY=boy

SUBDOMAIN\_VOTE=vote

SUBDOMAIN\_RESULT=result

# Basic Auth for result

BASIC\_AUTH\_USER=admin

BASIC\_AUTH\_PASSWORD=$apr1$Z7v...G5f73oH/zLn1

# Docker networks

PUBLIC\_NETWORK=proxy\_net

INTERNAL\_NETWORK=internal\_net

🔐 Replace BASIC\_AUTH\_PASSWORD with a 30-character+ htpasswd hash. You can generate it with:

bash

CopyEdit

docker run --rm httpd:alpine htpasswd -nb admin "your-very-long-random-password"

**📄 2. nginx/default.conf**

nginx

CopyEdit

server {

listen 80;

server\_name ${SUBDOMAIN\_GIRL}.${DOMAIN};

location / {

proxy\_pass http://bibi-tavallod:80;

}

}

server {

listen 80;

server\_name ${SUBDOMAIN\_BOY}.${DOMAIN};

location / {

proxy\_pass http://bazi-2048:80;

}

}

server {

listen 80;

server\_name ${SUBDOMAIN\_VOTE}.${DOMAIN};

location / {

proxy\_pass http://vote-app:80;

}

}

server {

listen 80;

server\_name ${SUBDOMAIN\_RESULT}.${DOMAIN};

auth\_basic "Restricted";

auth\_basic\_user\_file /etc/nginx/.htpasswd;

location / {

proxy\_pass http://result-app:80;

}

}

**📄 3. docker-compose.yml (Updated)**

yaml

CopyEdit

version: '3.9'

services:

reverse-proxy:

image: nginx:alpine

ports:

- "80:80"

volumes:

- ./nginx/default.conf:/etc/nginx/conf.d/default.conf

- ./nginx/htpasswd:/etc/nginx/.htpasswd

networks:

- ${PUBLIC\_NETWORK}

bibi-tavallod:

build:

context: ./BIBI Tavallod

networks:

- ${PUBLIC\_NETWORK}

bazi-2048:

build:

context: ./Bazi 2048

networks:

- ${PUBLIC\_NETWORK}

vote-app:

build:

context: ./voting app/vote

networks:

- ${PUBLIC\_NETWORK}

- ${INTERNAL\_NETWORK}

depends\_on:

- redis

result-app:

build:

context: ./voting app/result

networks:

- ${PUBLIC\_NETWORK}

- ${INTERNAL\_NETWORK}

depends\_on:

- db

worker:

build:

context: ./voting app/worker

networks:

- ${INTERNAL\_NETWORK}

depends\_on:

- redis

- db

redis:

image: redis:alpine

networks:

- ${INTERNAL\_NETWORK}

db:

image: postgres:15

environment:

POSTGRES\_USER: postgres

POSTGRES\_PASSWORD: example

networks:

- ${INTERNAL\_NETWORK}

networks:

${PUBLIC\_NETWORK}:

${INTERNAL\_NETWORK}:

internal: true

**📄 4. Modify Voting App Options**

In voting app/vote/app.py (or relevant backend script), change:

python

CopyEdit

options = ["boy", "girl"]

Rebuild after modifying the code.

**🟢 Final Step: Run Everything**

**1. Add to your hosts file:**

Edit:

* **Windows:** C:\Windows\System32\drivers\etc\hosts
* Add:

CopyEdit

127.0.0.1 girl.shahin.ir

127.0.0.1 boy.shahin.ir

127.0.0.1 vote.shahin.ir

127.0.0.1 result.shahin.ir

**2. From project-root, run:**

bash

CopyEdit

docker compose up -d

**✅ You Now Have:**

* girl.shahin.ir → your Go webpage
* boy.shahin.ir → 2048 game
* vote.shahin.ir → vote page (with "boy" and "girl")
* result.shahin.ir → result page (with basic auth)

Would you like me to generate:

* Dockerfile for BIBI Tavallod (Go)
* Dockerfile for Bazi 2048 (Nginx static)

Or review your current ones?

**You said:**

This happend: PS D:\semester 8\MS\Projects\Github\CA3\P1> docker compose up -d time="2025-05-27T18:57:40+03:30" level=warning msg="The \"apr1\" variable is not set. Defaulting to a blank string." time="2025-05-27T18:57:40+03:30" level=warning msg="The \"GHFbM09S\" variable is not set. Defaulting to a blank string." time="2025-05-27T18:57:40+03:30" level=warning msg="The \"LdsK7hQsFq1x7OkLXysJr\" variable is not set. Defaulting to a blank string." time="2025-05-27T18:57:40+03:30" level=warning msg="The \"apr1\" variable is not set. Defaulting to a blank string." time="2025-05-27T18:57:40+03:30" level=warning msg="The \"GHFbM09S\" variable is not set. Defaulting to a blank string." time="2025-05-27T18:57:40+03:30" level=warning msg="The \"LdsK7hQsFq1x7OkLXysJr\" variable is not set. Defaulting to a blank string." time="2025-05-27T18:57:40+03:30" level=warning msg="The \"apr1\" variable is not set. Defaulting to a blank string." time="2025-05-27T18:57:40+03:30" level=warning msg="The \"GHFbM09S\" variable is not set. Defaulting to a blank string." time="2025-05-27T18:57:40+03:30" level=warning msg="The \"LdsK7hQsFq1x7OkLXysJr\" variable is not set. Defaulting to a blank string." time="2025-05-27T18:57:40+03:30" level=warning msg="The \"apr1\" variable is not set. Defaulting to a blank string." time="2025-05-27T18:57:40+03:30" level=warning msg="The \"GHFbM09S\" variable is not set. Defaulting to a blank string." time="2025-05-27T18:57:40+03:30" level=warning msg="The \"LdsK7hQsFq1x7OkLXysJr\" variable is not set. Defaulting to a blank string." time="2025-05-27T18:57:40+03:30" level=warning msg="D:\\semester 8\\MS\\Projects\\Github\\CA3\\P1\\docker-compose.yml: the attribute version is obsolete, it will be ignored, please remove it to avoid potential confusion" service "worker" refers to undefined network internal\_net: invalid compose project

**ChatGPT said:**

Thanks for the output — this tells us two clear issues:

**❗ Issue 1: .env file contains characters Docker tries to interpret as environment variables**

You used this in your .env:

env

CopyEdit

BASIC\_AUTH\_PASSWORD=$apr1$GHFbM09S$LdsK7hQsFq1x7OkLXysJr

But Docker thinks $apr1, $GHFbM09S, and $LdsK7hQsFq1x7OkLXysJr are variables.

**✅ Fix 1: Escape the dollar signs in .env**

In your .env file, change this:

env

CopyEdit

BASIC\_AUTH\_PASSWORD=$apr1$GHFbM09S$LdsK7hQsFq1x7OkLXysJr

To this:

env

CopyEdit

BASIC\_AUTH\_PASSWORD=\$apr1\$GHFbM09S\$LdsK7hQsFq1x7OkLXysJr

This tells Docker Compose to treat it **as a literal string**, not variable references.

**❗ Issue 2: internal\_net and proxy\_net were used as environment variables, but you defined them in networks: like this:**

yaml

CopyEdit

networks:

${PUBLIC\_NETWORK}:

${INTERNAL\_NETWORK}:

internal: true

But Compose doesn't substitute variable **names** for **keys** in YAML.

**✅ Fix 2: Replace variables in networks block with static names**

Replace this:

yaml

CopyEdit

networks:

${PUBLIC\_NETWORK}:

${INTERNAL\_NETWORK}:

internal: true

With:

yaml

CopyEdit

networks:

proxy\_net:

internal\_net:

internal: true

Then update all services that use ${PUBLIC\_NETWORK} and ${INTERNAL\_NETWORK} to use proxy\_net and internal\_net directly.

**✅ Updated Snippet (Final networks: section)**

yaml

CopyEdit

networks:

proxy\_net:

internal\_net:

internal: true

And for example, change:

yaml

CopyEdit

networks:

- ${INTERNAL\_NETWORK}

To:

yaml

CopyEdit

networks:

- internal\_net

**✅ Final Steps**

1. Fix .env to escape $ signs in password
2. Replace ${PUBLIC\_NETWORK} and ${INTERNAL\_NETWORK} with static names
3. Run again:

bash

CopyEdit

docker compose down -v

docker compose up -d --build

Let me know if you'd like me to post the fully corrected .env and docker-compose.yml content!

**You said:**

we had this before running compose up again: PS D:\semester 8\MS\Projects\Github\CA3\P1> docker compose down -v time="2025-05-27T19:00:30+03:30" level=warning msg="The \"apr1\" variable is not set. Defaulting to a blank string." time="2025-05-27T19:00:30+03:30" level=warning msg="The \"GHFbM09S\" variable is not set. Defaulting to a blank string." time="2025-05-27T19:00:30+03:30" level=warning msg="The \"LdsK7hQsFq1x7OkLXysJr\" variable is not set. Defaulting to a blank string." time="2025-05-27T19:00:30+03:30" level=warning msg="The \"apr1\" variable is not set. Defaulting to a blank string." time="2025-05-27T19:00:30+03:30" level=warning msg="The \"GHFbM09S\" variable is not set. Defaulting to a blank string." time="2025-05-27T19:00:30+03:30" level=warning msg="The \"LdsK7hQsFq1x7OkLXysJr\" variable is not set. Defaulting to a blank string." time="2025-05-27T19:00:30+03:30" level=warning msg="The \"apr1\" variable is not set. Defaulting to a blank string." time="2025-05-27T19:00:30+03:30" level=warning msg="The \"GHFbM09S\" variable is not set. Defaulting to a blank string." time="2025-05-27T19:00:30+03:30" level=warning msg="The \"LdsK7hQsFq1x7OkLXysJr\" variable is not set. Defaulting to a blank string." time="2025-05-27T19:00:30+03:30" level=warning msg="The \"apr1\" variable is not set. Defaulting to a blank string." time="2025-05-27T19:00:30+03:30" level=warning msg="The \"GHFbM09S\" variable is not set. Defaulting to a blank string." time="2025-05-27T19:00:30+03:30" level=warning msg="The \"LdsK7hQsFq1x7OkLXysJr\" variable is not set. Defaulting to a blank string." time="2025-05-27T19:00:30+03:30" level=warning msg="D:\\semester 8\\MS\\Projects\\Github\\CA3\\P1\\docker-compose.yml: the attribute version is obsolete, it will be ignored, please remove it to avoid potential confusion" error during connect: Get "http://%2F%2F.%2Fpipe%2FdockerDesktopLinuxEngine/v1.46/containers/json?all=1&filters=%7B%22label%22%3A%7B%22com.docker.compose.config-hash%22%3Atrue%2C%22com.docker.compose.oneoff%3DFalse%22%3Atrue%2C%22com.docker.compose.project%3Dp1%22%3Atrue%7D%7D": open //./pipe/dockerDesktopLinuxEngine: The system cannot find the file specified.

**ChatGPT said:**

Thanks for the update. This new error is **not related to your Docker Compose files or .env variables**, but rather to Docker Desktop itself.

**❗ Error Summary:**

arduino

CopyEdit

error during connect: ... open //./pipe/dockerDesktopLinuxEngine: The system cannot find the file specified.

This means: **Docker Desktop's Linux backend is not running or not accessible.**

**✅ Fix Steps:**

**1. Restart Docker Desktop**

* Open Docker Desktop (search in Start menu)
* If it’s not running, **start it**
* If it’s already open, **click the menu and select "Restart Docker Desktop"**

This resolves 90% of these pipe/socket issues on Windows.

**2. Verify Docker is Working**

After restart, open PowerShell or CMD and run:

bash

CopyEdit

docker info

You should see detailed output (not a connection error). If you do, you're good to proceed.

**3. Try docker compose down -v again**

Once Docker is up:

bash

CopyEdit

docker compose down -v

Then run:

bash

CopyEdit

docker compose up -d --build

**⚠ Optional Cleanup**

If restart doesn't fix it:

* **Quit Docker Desktop completely** (from system tray → Quit)
* Reopen it
* **Ensure WSL 2 backend is healthy** if you're using it
  + Open WSL terminal (wsl)
  + Run: docker info from there too

Let me know once Docker is back up and Compose is running — I’ll help you test all subdomains next.

همچنین برای قسمت compose از: <https://chat.qwen.ai/s/fa9c7203-fdba-4bb4-b1bd-5d8263640f56?fev=0.0.110>

بخش از چت(کپی کردن به شدت درد آور بود)

**🔄 4. Restart Services**

After fixing the **.htpasswd** file, restart everything:

bash

1

2

docker compose down

docker compose up -d --build

**📦 5. Verify Inside NGINX Container**

Enter the NGINX container:

bash

1

docker exec -it reverse-proxy sh

Check that the **.htpasswd** file exists and has the correct content:

bash

1

cat /etc/nginx/.htpasswd

It should show:

1

admin:$apr1$slv0827u$mTnJtFnJ/OUiHzEiy2VL..

**🔎 6. Test Authentication in Browser**

Visit **http://result.shahin.ir** in your browser. You should now see a basic authentication prompt.

Enter:

* Username: **admin**
* Password: **1234**

If authentication works, you’ll see the results page without errors.

**🚀 Final Verification**

| **Task** | **Command** |
| --- | --- |
| Check local**.htpasswd** | **cat nginx-auth/.htpasswd** |
| Delete incorrect**.htpasswd**directory | **rm -rf nginx-auth/.htpasswd**(if it's a directory) |
| Restart services | **docker compose down && docker compose up -d** |
| Verify inside NGINX | **docker exec -it reverse-proxy cat /etc/nginx/.htpasswd** |
| Test in browser | Visit**http://result.shahin.ir** |

**✅ Expected Outcome**

Once fixed:

* ✅ All domains work: **vote.shahin.ir**, **result.shahin.ir**, **boy.shahin.ir**, **girl.shahin.ir**
* ✅ **result.shahin.ir** prompts for username/password
* ✅ Basic auth works with **admin:1234**
* ✅ No more **403 Forbidden** errors

**📝 Additional Notes**

**1. Stronger Passwords**

For production, use a stronger password than **1234**. Generate one with:

bash

1

openssl rand -base64 30 | tr -d '/+' | cut -c -30

Then regenerate **.htpasswd**:

bash

1

htpasswd -b ./nginx-auth/.htpasswd admin <strong\_password>

**2. Prevent Automatic Directory Creation**

To avoid Docker creating directories automatically:

* Always ensure the **.htpasswd** file exists before running **docker compose up**.
* Use a clear naming convention (e.g., **.htpasswd.txt**) locally and mount it as **.htpasswd** in NGINX.

**🎉 Next Steps**

Please confirm:

1. ✅ Did you delete the **.htpasswd** directory ?
2. ✅ Does **cat nginx-auth/.htpasswd** show the correct content?
3. ✅ After restarting, does **docker exec -it reverse-proxy cat /etc/nginx/.htpasswd** show the correct content?

Let me know how it goes! If you still encounter issues, I’ll help troubleshoot further.

برای بخش swarm: <https://chat.qwen.ai/s/3c00ef91-bd01-4a25-b917-dfffda1f1381?fev=0.0.111>

**4. Access the Nginx Service**

Since you've mapped port **80** on the host to port **80** in the container, you can access the Nginx service by pointing your browser or using **curl** to any of the nodes' IP addresses:

* From Host Machine : Open a web browser or use **curl**:

bash

1

curl http://<any-node-ip>

Replace **<any-node-ip>** with the IP address of any node (e.g., **192.168.1.101**).

* From Another Node : If you're inside another VM or container, you can also access it via the node's IP.

**5. Ensure One Container per Node**

Docker Swarm uses a scheduling strategy called global placement constraints to ensure that containers are distributed evenly across nodes. However, if you specifically want to enforce one container per node, you can use the **--mode global** option when creating the service:

**Create a Global Service:**

bash

1

2

3

4

docker service create \

--name nginx-global \

--mode global \

nginx

This ensures exactly one instance of the Nginx service runs on each node.

**🚀 Final Output**

After deploying and scaling the Nginx service, you should see:

1. Service Status :

bash

1

docker service ls

1

2

ID NAME MODE REPLICAS IMAGE PORTS

qweasdzxc nginx replicated 5/5 nginx \*:80->80/tcp

1. Container Distribution :

bash

1

docker service ps nginx

1

2

3

4

5

6

NAME IMAGE NODE DESIRED STATE CURRENT STATE ERROR PORTS

nginx.1 nginx swarm-manager1 Running Running 2 minutes ago

nginx.2 nginx swarm-manager2 Running Running 2 minutes ago

nginx.3 nginx swarm-manager3 Running Running 2 minutes ago

nginx.4 nginx swarm-worker1 Running Running 2 minutes ago

nginx.5 nginx swarm-manager1 Running Running 2 minutes ago

* + At least one container is running on each node.
  + The remaining containers are distributed across the nodes.

**✅ Testing Access**

* Open a web browser or use **curl** to test the Nginx service:

bash

1

curl http://192.168.1.101

You should see the default Nginx welcome page:

1

2

3

4

5

<!DOCTYPE html>

<html>

<head>

<title>Welcome to nginx!</title>

...

**🧩 Next Steps**

1. Monitor the Service : Use **docker service ps nginx** to track the health and distribution of containers.
2. Scale Further : If needed, scale the service up or down:

bash

1

docker service scale nginx=10

1. Add More Nodes : If you add more nodes to the Swarm, Docker will automatically distribute new containers to them.
2. Explore Advanced Features :
   * Use placement constraints to control where containers run.
   * Configure load balancing for external access.

Let me know if you encounter any issues during deployment or scaling! 😊

Copy

Ask

Explain

On which machine should I enter the command?



**Qwen3-235B-A22B**

Great question!

"On which machine should I enter the Docker Swarm commands like **docker service create**?"

**✅ Short Answer:**

You should run all management commands (like deploying and scaling services) on a manager node — preferably the swarm leader.

In your case, that’s:

1

swarm-manager1

This is the node where you ran **docker swarm init**, and it's marked as the Leader in the Swarm.

**🧠 Why Manager Nodes?**

Docker Swarm uses a manager-worker architecture :

| **Node Type** | **Purpose** |
| --- | --- |
| Manager | Manages the cluster: schedules tasks, maintains state, handles updates |
| Worker | Only runs containers, doesn't manage the cluster |

Only manager nodes can:

* Create or update services
* Scale services
* Remove nodes or services
* View full cluster state with **docker node ls**

So to deploy Nginx and scale it across the cluster, always use one of the managers — ideally the leader (**swarm-manager1**), so you avoid any potential sync issues.

**🔁 What If the Leader Goes Down?**

Swarm is fault-tolerant:

* If **swarm-manager1** goes down, another manager (e.g., **swarm-manager2**) will become the new leader.
* You can then safely run management commands from the new leader.

But for now, stick with **swarm-manager1**.

**✅ Summary**

| **Task** | **Where to Run It** |
| --- | --- |
| Initialize Swarm | **swarm-manager1** |
| Deploy services (**docker service create**) | **swarm-manager1** |
| Scale services (**docker service scale**) | Any manager node |
| Check node list (**docker node ls**) | Any manager node |
| Access running containers | Any node (use its IP to access exposed services like Nginx) |

Let me know when you're ready to proceed with the deployment! 😊

tell about accessing this app from outside. how requests are serviced and ip of which machine should we put in dns server. answer short



**Qwen3-235B-A22B**

To access your Nginx service from outside:

**✅ Use the IP of any Swarm node (manager or worker) — preferably a manager like swarm-manager1.**

**🌐 Why?**

* Docker Swarm uses routing mesh : any node accepts traffic on published ports (e.g., port 80).
* Requests are automatically routed to available containers, no matter which node they're on.

**📌 For DNS:**

Point your domain (e.g., **example.com**) to the IP of one or more manager nodes , like:

1

A record: example.com → 192.168.1.101

You can add multiple A records for high availability.

**🔄 Result:**

* Any incoming HTTP request hits the Swarm cluster.
* Routing mesh directs it to a running Nginx container.
* You get load balancing and fault tolerance across all nodes.