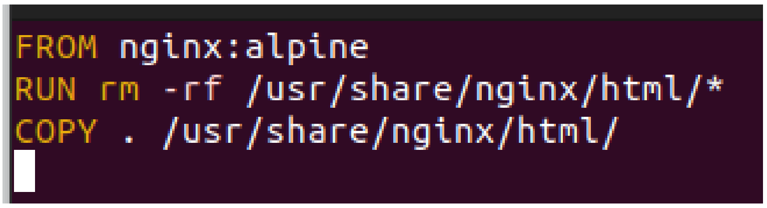
Dockerfile:

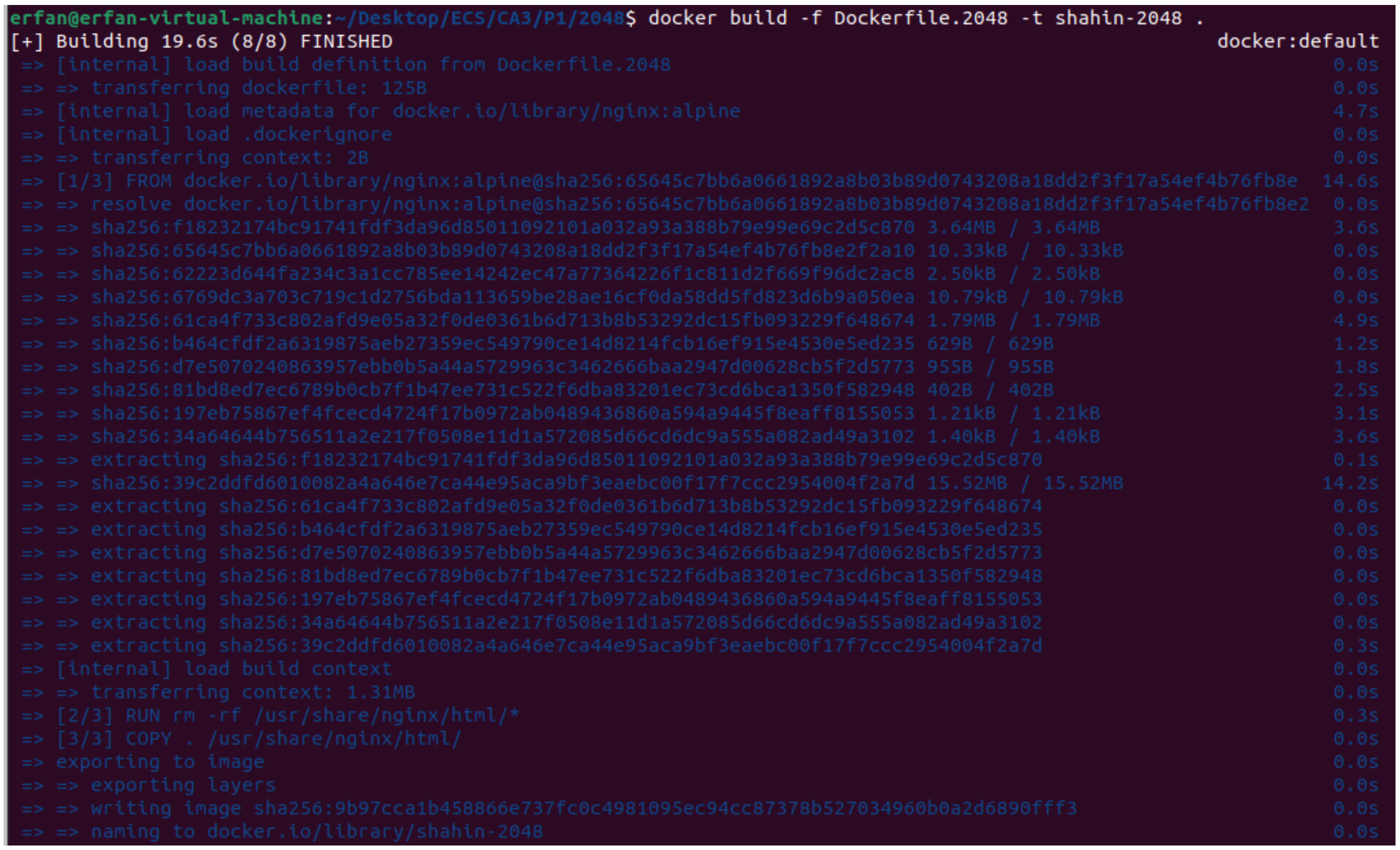
At first, we create Dockerfile.2048:



```
FROM nginx:alpine
RUN rm -rf /usr/share/nginx/html/*
COPY . /usr/share/nginx/html/
```

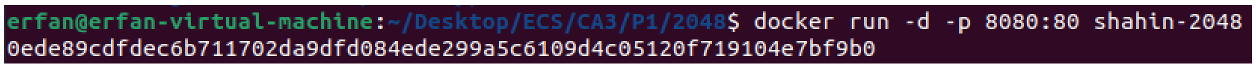
- Start with a lightweight version of the NGINX web server.
- Delete the default "Welcome to NGINX" page and all its files so that they don't get in the way.
- Copy our own website files for 2048 game into the folder NGINX uses to serve web pages.

After that, we use the instructions in Dockerfile.2048 to build a new image called shahin-2048:



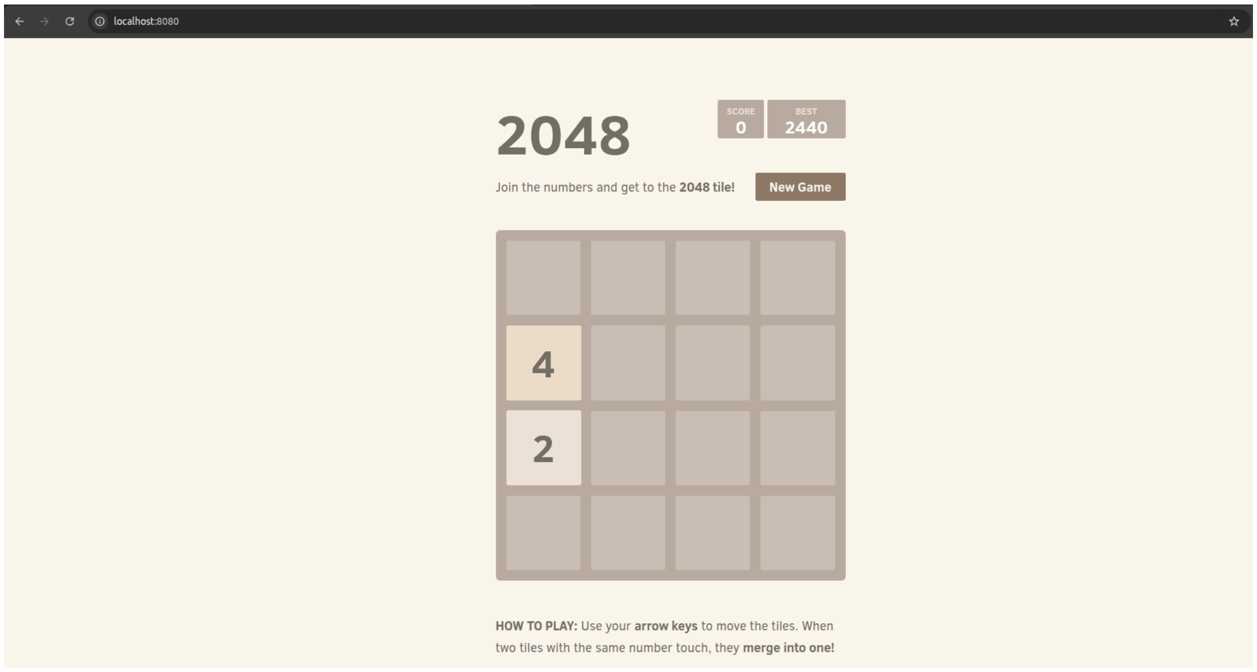
```
erfan@erfan-virtual-machine:~/Desktop/ECS/CA3/P1/2048$ docker build -f Dockerfile.2048 -t shahin-2048 .
[+] Building 19.6s (8/8) FINISHED                                                                     docker:default
 => [internal] load build definition from Dockerfile.2048                                                       0.0s
 => => transferring dockerfile: 125B                                                                            0.0s
 => [internal] load metadata for docker.io/library/nginx:alpine                                                 4.7s
 => [internal] load .dockerignore                                                                               0.0s
 => => transferring context: 2B                                                                                 0.0s
 => [1/3] FROM docker.io/library/nginx:alpine@sha256:65645c7bb6a0661892a8b03b89d0743208a18dd2f3f17a54ef4b76fb8e  14.6s
 => => resolve docker.io/library/nginx:alpine@sha256:65645c7bb6a0661892a8b03b89d0743208a18dd2f3f17a54ef4b76fb8e2  0.0s
 => => sha256:f18232174bc91741fdf3da96d85011092101a032a93a388b79e99e69c2d5c870 3.64MB / 3.64MB                  3.6s
 => => sha256:65645c7bb6a0661892a8b03b89d0743208a18dd2f3f17a54ef4b76fb8e2f2a10 10.33kB / 10.33kB                0.0s
 => => sha256:62223d644fa234c3a1cc785ee14242ec47a77364226f1c811d2f669f96dc2ac8 2.50kB / 2.50kB                  0.0s
 => => sha256:6769dc3a703c719c1d2756bda113659be28ae16cf0da58dd5fd823d6b9a050ea 10.79kB / 10.79kB                0.0s
 => => sha256:61ca4f733c802afd9e05a32f0de0361b6d713b8b53292dc15fb093229f648674 1.79MB / 1.79MB                  4.9s
 => => sha256:b464cfdf2a6319875aeb27359ec549790ce14d8214fcb16ef915e4530e5ed235 629B / 629B                      1.2s
 => => sha256:d7e5070240863957ebb0b5a44a5729963c3462666baa2947d00628cb5f2d5773 955B / 955B                      1.8s
 => => sha256:81bd8ed7ec6789b0cb7f1b47ee731c522f6dba83201ec73cd6bca1350f582948 402B / 402B                      2.5s
 => => sha256:197eb75867ef4fcecd4724f17b0972ab0489436860a594a9445f8eaff8155053 1.21kB / 1.21kB                  3.1s
 => => sha256:34a64644b756511a2e217f0508e11d1a572085d66cd6dc9a555a082ad49a3102 1.40kB / 1.40kB                  3.6s
 => => extracting sha256:f18232174bc91741fdf3da96d85011092101a032a93a388b79e99e69c2d5c870                       0.1s
 => => sha256:39c2ddfd6010082a4a646e7ca44e95aca9bf3eaebc00f17f7ccc2954004f2a7d 15.52MB / 15.52MB               14.2s
 => => extracting sha256:61ca4f733c802afd9e05a32f0de0361b6d713b8b53292dc15fb093229f648674                       0.0s
 => => extracting sha256:b464cfdf2a6319875aeb27359ec549790ce14d8214fcb16ef915e4530e5ed235                       0.0s
 => => extracting sha256:d7e5070240863957ebb0b5a44a5729963c3462666baa2947d00628cb5f2d5773                       0.0s
 => => extracting sha256:81bd8ed7ec6789b0cb7f1b47ee731c522f6dba83201ec73cd6bca1350f582948                       0.0s
 => => extracting sha256:197eb75867ef4fcecd4724f17b0972ab0489436860a594a9445f8eaff8155053                       0.0s
 => => extracting sha256:34a64644b756511a2e217f0508e11d1a572085d66cd6dc9a555a082ad49a3102                       0.0s
 => => extracting sha256:39c2ddfd6010082a4a646e7ca44e95aca9bf3eaebc00f17f7ccc2954004f2a7d                       0.3s
 => [internal] load build context                                                                               0.0s
 => => transferring context: 1.31MB                                                                             0.0s
 => [2/3] RUN rm -rf /usr/share/nginx/html/*                                                                    0.3s
 => [3/3] COPY . /usr/share/nginx/html/                                                                         0.0s
 => exporting to image                                                                                          0.0s
 => => exporting layers                                                                                         0.0s
 => => writing image sha256:9b97cca1b458866e737fc0c4981095ec94cc87378b527034960b0a2d6890fff3                    0.0s
 => => naming to docker.io/library/shahin-2048                                                                  0.0s
```

Then we start a new container for the game 2048 in the background and map port 8080 on our computer to port 80 inside the container:



```
erfan@erfan-virtual-machine:~/Desktop/ECS/CA3/P1/2048$ docker run -d -p 8080:80 shahin-2048
0ede89cdfdec6b711702da9dfd084ede299a5c6109d4c05120f719104e7bf9b0
```

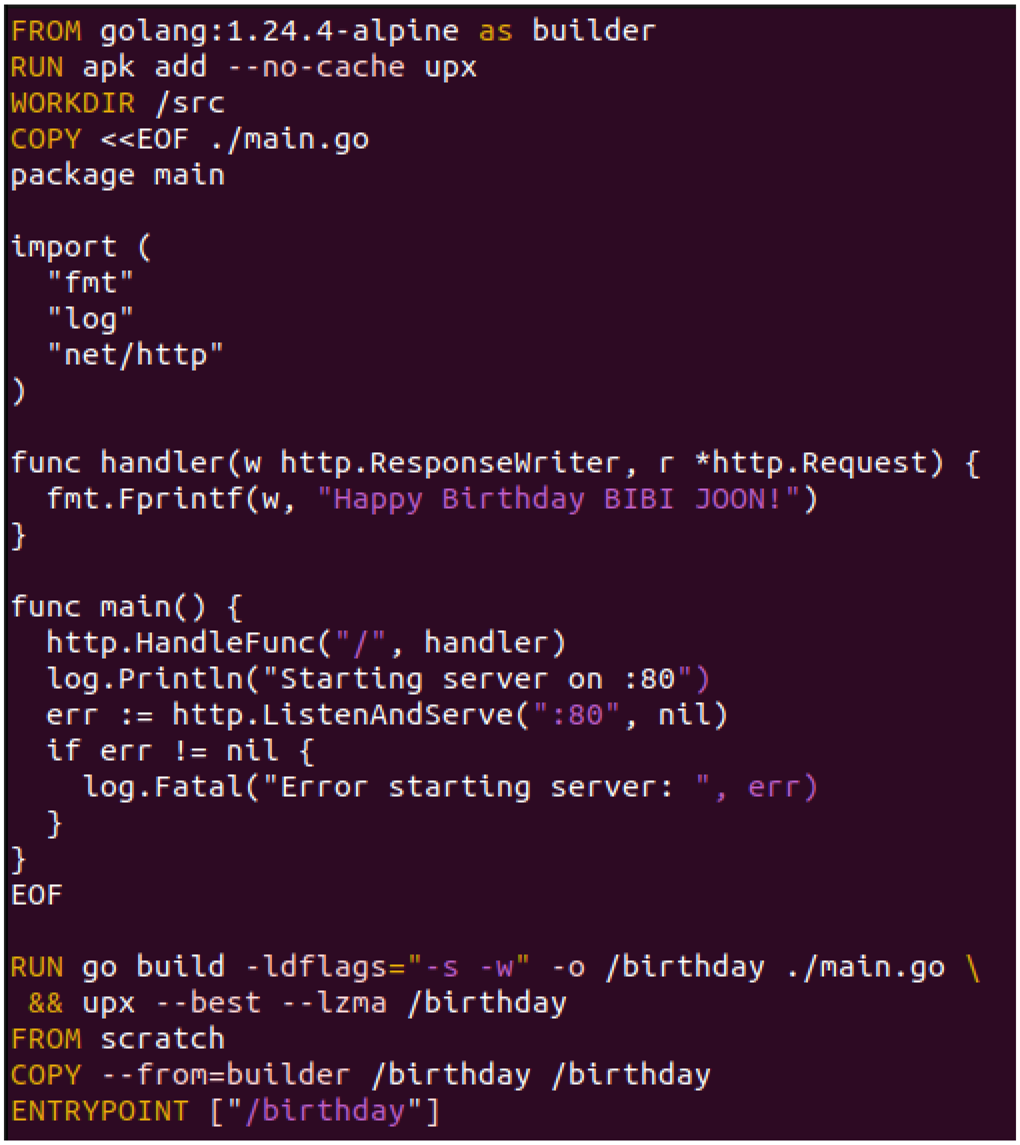
We can view it at: http://localhost:8080



Join the numbers and get to the **2048 tile!**

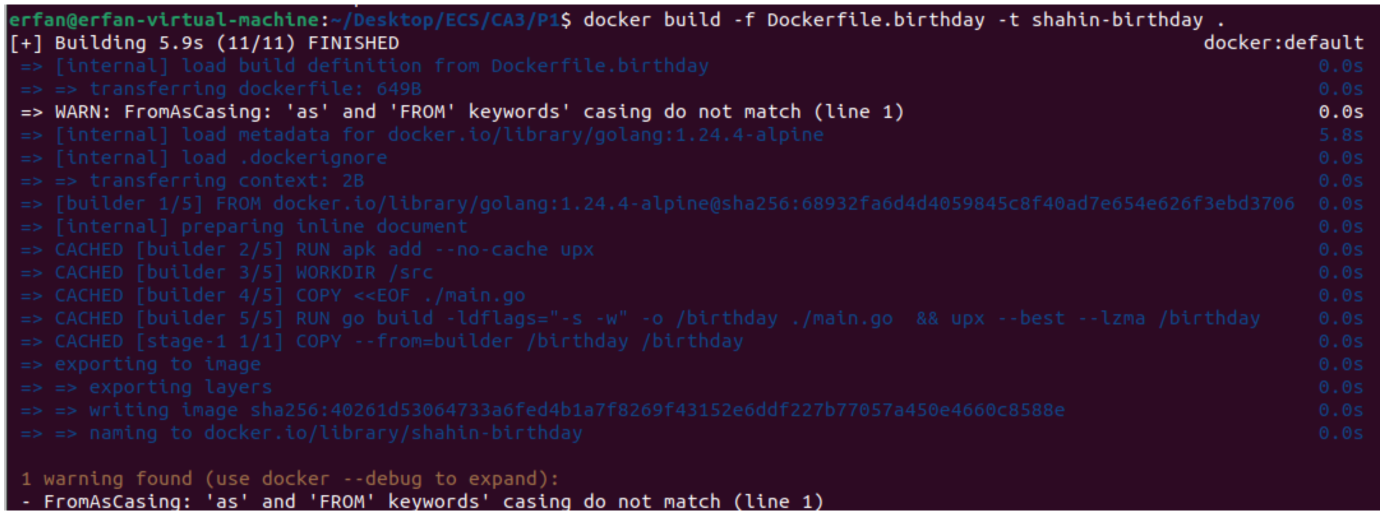
HOW TO PLAY: Use your **arrow keys** to move the tiles. When two tiles with the same number touch, they **merge into one!**

Now we create Dockerfile.birthday for the Go code:

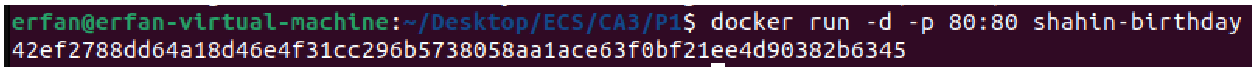


```
FROM golang:1.24.4-alpine as builder
RUN apk add --no-cache upx
WORKDIR /src
COPY <<EOF ./main.go
package main

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)
  }
}
EOF

RUN go build -ldflags="-s -w" -o /birthday ./main.go \
 && upx --best --lzma /birthday
FROM scratch
COPY --from=builder /birthday /birthday
ENTRYPOINT ["/birthday"]
```

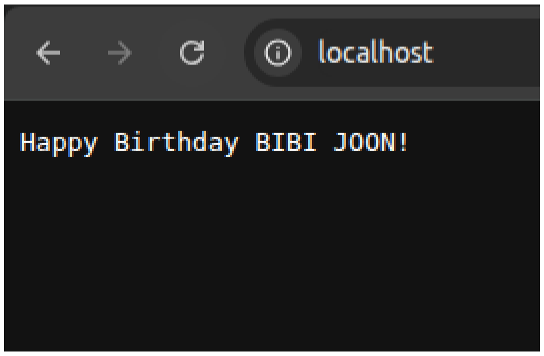
- Start with a small Go environment.
- Install a tool that compresses binaries called *upx*.
- Create a folder called /src to work in.
- Paste the Go code directly into the Dockerfile.
- Compile the Go program.
- Remove debug info to make it smaller and compress the binary even more.
- switch to a minimal empty image.



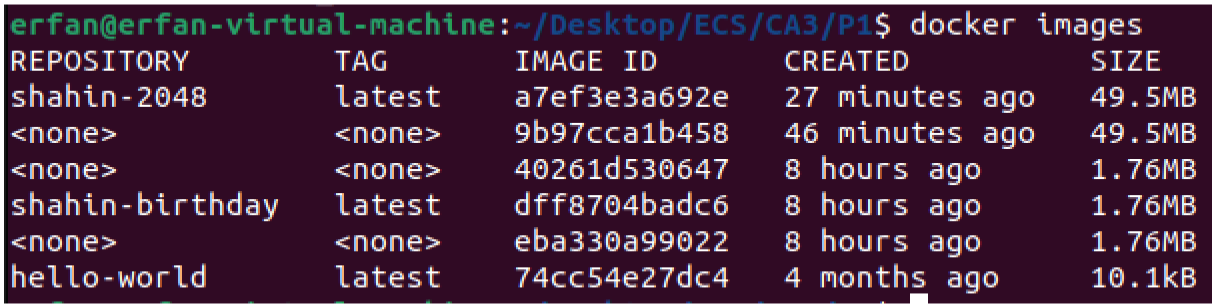
```
erfan@erfan-virtual-machine:~/Desktop/ECS/CA3/P1$ docker build -f Dockerfile.birthday -t shahin-birthday .
[+] Building 5.9s (11/11) FINISHED                                                                    docker:default
 => [internal] load build definition from Dockerfile.birthday                                                   0.0s
 => => transferring dockerfile: 649B                                                                            0.0s
 => WARN: FromAsCasing: 'as' and 'FROM' keywords' casing do not match (line 1)                                  0.0s
 => [internal] load metadata for docker.io/library/golang:1.24.4-alpine                                         5.8s
 => [internal] load .dockerignore                                                                               0.0s
 => => transferring context: 2B                                                                                 0.0s
 => [builder 1/5] FROM docker.io/library/golang:1.24.4-alpine@sha256:68932fa6d4d4059845c8f40ad7e654e626f3ebd3706  0.0s
 => [internal] preparing inline document                                                                        0.0s
 => CACHED [builder 2/5] RUN apk add --no-cache upx                                                             0.0s
 => CACHED [builder 3/5] WORKDIR /src                                                                           0.0s
 => CACHED [builder 4/5] COPY <<EOF ./main.go                                                                   0.0s
 => CACHED [builder 5/5] RUN go build -ldflags="-s -w" -o /birthday ./main.go  && upx --best --lzma /birthday    0.0s
 => CACHED [stage-1 1/1] COPY --from=builder /birthday /birthday                                                0.0s
 => exporting to image                                                                                          0.0s
 => => exporting layers                                                                                         0.0s
 => => writing image sha256:40261d53064733a6fed4b1a7f8269f43152e6ddf227b77057a450e4660c8588e                     0.0s
 => => naming to docker.io/library/shahin-birthday                                                              0.0s

 1 warning found (use docker --debug to expand):
 - FromAsCasing: 'as' and 'FROM' keywords' casing do not match (line 1)
```



```
erfan@erfan-virtual-machine:~/Desktop/ECS/CA3/P1$ docker run -d -p 80:80 shahin-birthday
42ef2788dd64a18d46e4f31cc296b5738058aa1ace63f0bf21ee4d90382b6345
```



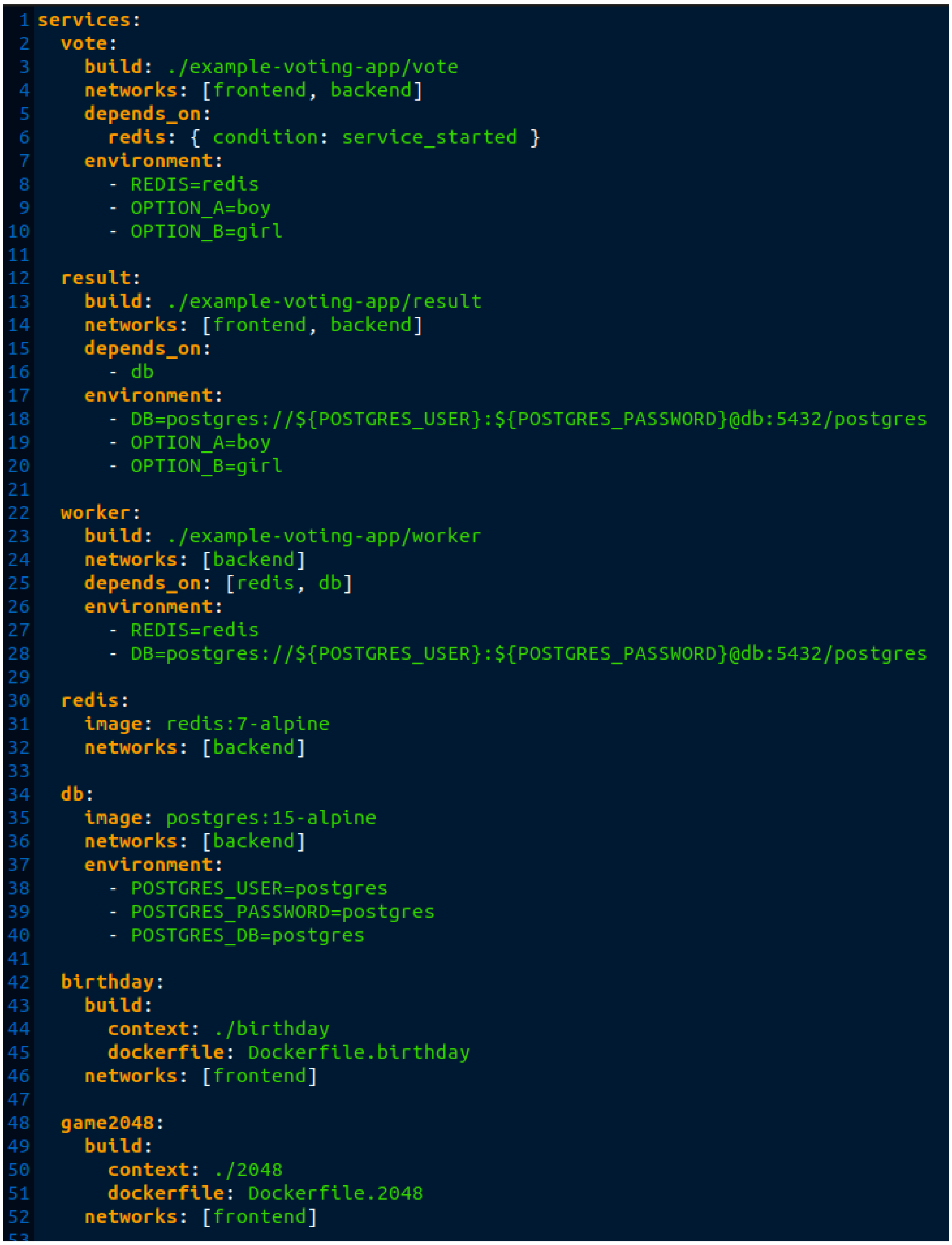
```
Happy Birthday BIBI JOON!
```



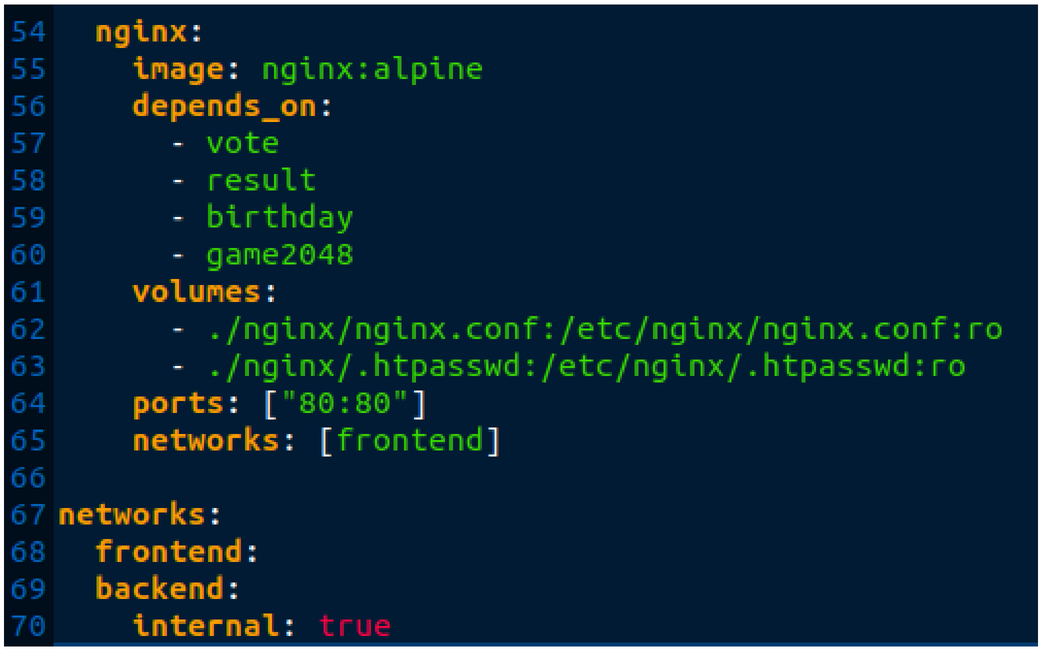
```
erfan@erfan-virtual-machine:~/Desktop/ECS/CA3/P1$ docker images
REPOSITORY        TAG       IMAGE ID       CREATED          SIZE
shahin-2048       latest    a7ef3e3a692e   27 minutes ago   49.5MB
<none>            <none>    9b97cca1b458   46 minutes ago   49.5MB
<none>            <none>    40261d530647   8 hours ago      1.76MB
shahin-birthday   latest    dff8704badc6   8 hours ago      1.76MB
<none>            <none>    eba330a99022   8 hours ago      1.76MB
hello-world       latest    74cc54e27dc4   4 months ago     10.1kB
```

Docker compose:

We have Dockerfile.2048 in the 2048 folder, Dockerfile.birthday in the birthday folder and now the example-voting-app folder. Now we add docker-compose.yml file:



```yaml
services:
  vote:
    build: ./example-voting-app/vote
    networks: [frontend, backend]
    depends_on:
      redis: { condition: service_started }
    environment:
      - REDIS=redis
      - OPTION_A=boy
      - OPTION_B=girl

  result:
    build: ./example-voting-app/result
    networks: [frontend, backend]
    depends_on:
      - db
    environment:
      - DB=postgres://${POSTGRES_USER}:${POSTGRES_PASSWORD}@db:5432/postgres
      - OPTION_A=boy
      - OPTION_B=girl

  worker:
    build: ./example-voting-app/worker
    networks: [backend]
    depends_on: [redis, db]
    environment:
      - REDIS=redis
      - DB=postgres://${POSTGRES_USER}:${POSTGRES_PASSWORD}@db:5432/postgres

  redis:
    image: redis:7-alpine
    networks: [backend]

  db:
    image: postgres:15-alpine
    networks: [backend]
    environment:
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=postgres
      - POSTGRES_DB=postgres

  birthday:
    build:
      context: ./birthday
      dockerfile: Dockerfile.birthday
    networks: [frontend]

  game2048:
    build:
      context: ./2048
      dockerfile: Dockerfile.2048
    networks: [frontend]
```



```yaml
  nginx:
    image: nginx:alpine
    depends_on:
      - vote
      - result
      - birthday
      - game2048
    volumes:
      - ./nginx/nginx.conf:/etc/nginx/nginx.conf:ro
      - ./nginx/.htpasswd:/etc/nginx/.htpasswd:ro
    ports: ["80:80"]
    networks: [frontend]

networks:
  frontend:
  backend:
    internal: true
```

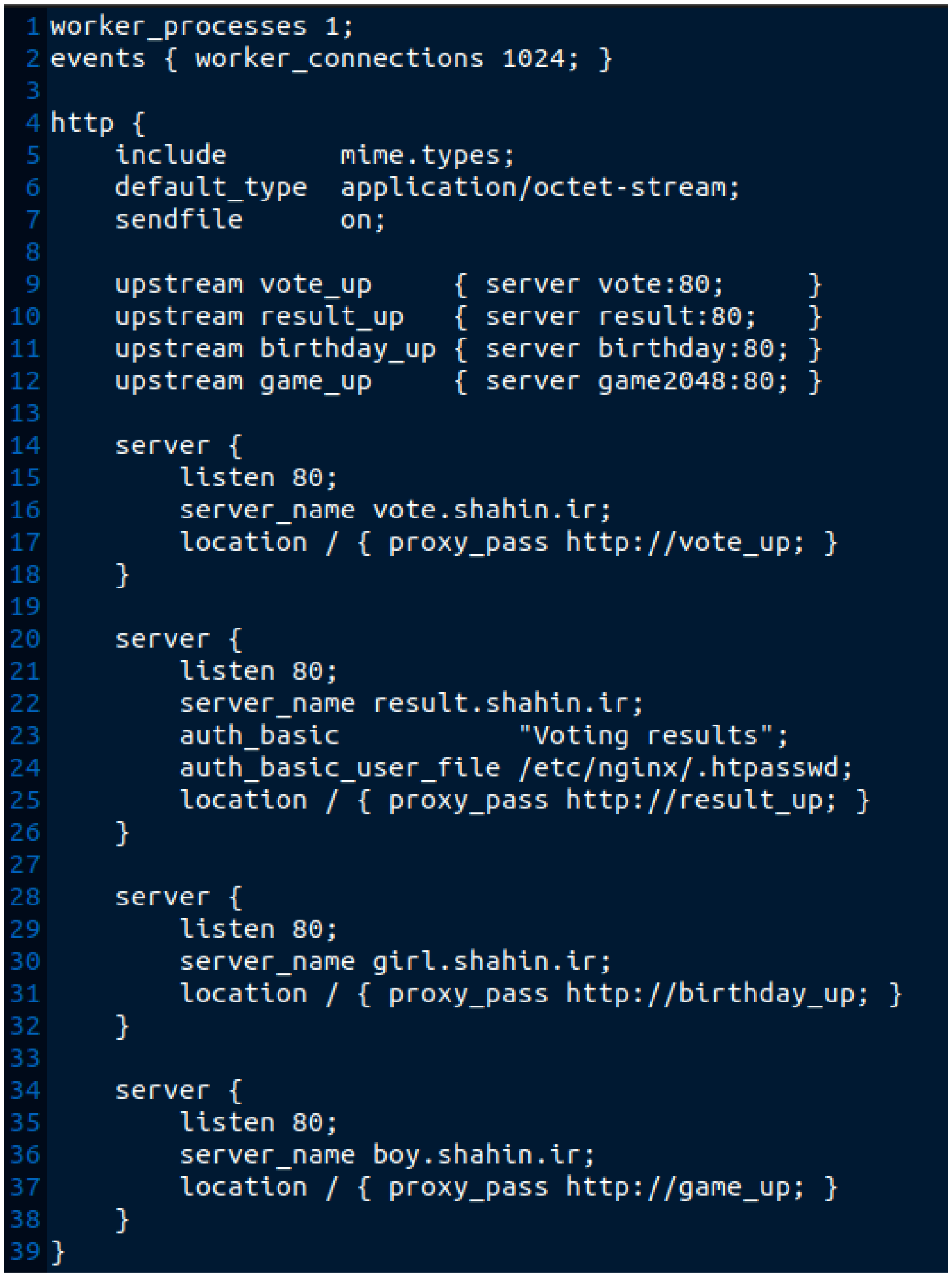
This Docker Compose file sets up a multi-service application using containers. The main application is a voting system, where users can choose between “boy" or "girl" option. The vote service handles the voting interface, relying on a Redis server to store vote data temporarily. There's also a result service that reads the stored data from a PostgreSQL database and displays the current vote count. Behind the scenes, a worker service pulls the vote data from Redis and writes it to the database, keeping the system updated and in sync.

Supporting these services are Redis and PostgreSQL. Both are crucial for data handling but are not directly accessed by users.

In addition to the voting system, there is a birthday app, and also the 2048 game.

To bring it all together, an Nginx server acts as a reverse proxy. It routes web traffic to the appropriate app and uses basic HTTP authentication (via an .htpasswd file) to control access. Everything runs on a shared frontend network.

We then create nginx folder and nginx.conf in that folder:



```
worker_processes 1;
events { worker_connections 1024; }

http {
    include       mime.types;
    default_type  application/octet-stream;
    sendfile      on;

    upstream vote_up     { server vote:80;     }
    upstream result_up   { server result:80;   }
    upstream birthday_up { server birthday:80; }
    upstream game_up     { server game2048:80; }

    server {
        listen 80;
        server_name vote.shahin.ir;
        location / { proxy_pass http://vote_up; }
    }

    server {
        listen 80;
        server_name result.shahin.ir;
        auth_basic           "Voting results";
        auth_basic_user_file /etc/nginx/.htpasswd;
        location / { proxy_pass http://result_up; }
    }

    server {
        listen 80;
        server_name girl.shahin.ir;
        location / { proxy_pass http://birthday_up; }
    }

    server {
        listen 80;
        server_name boy.shahin.ir;
        location / { proxy_pass http://game_up; }
    }
}
```

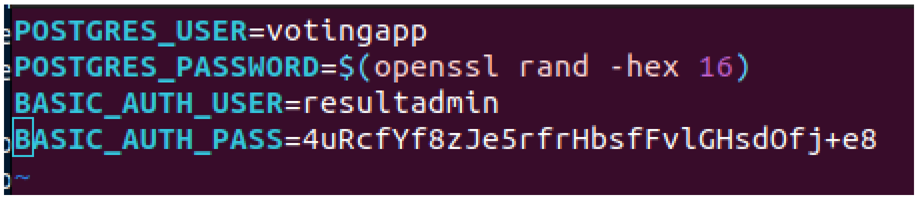
This Nginx configuration file is responsible for directing web traffic to the right parts of our application, based on the domain name a user enters.

At the top, it sets up some basic performance rules. It only uses one worker process (which is enough for small projects) and can handle up to 1024 simultaneous connections. It also includes some helpful defaults like how to handle file types and enables efficient file serving.

The `http` section defines shortcuts (called *upstreams*) to four different backend services. These shortcuts make it easier for Nginx to forward users to the right place.

Then we have server blocks. For example, if someone visits vote.shahin.ir, Nginx forwards them to the voting app. If they visit result.shahin.ir, they'll be shown the voting results. The other two domains, girl.shahin.ir and boy.shahin.ir, route to the birthday app and the 2048 game.

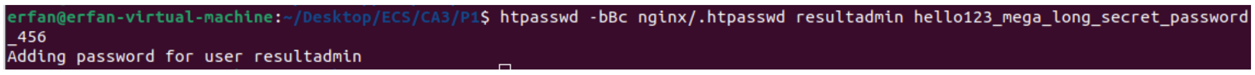
Then we have .env file:



```
POSTGRES_USER=votingapp
POSTGRES_PASSWORD=$(openssl rand -hex 16)
BASIC_AUTH_USER=resultadmin
BASIC_AUTH_PASS=4uRcfYf8zJe5rfrHbsfFvlGHsdOfj+e8
~
```

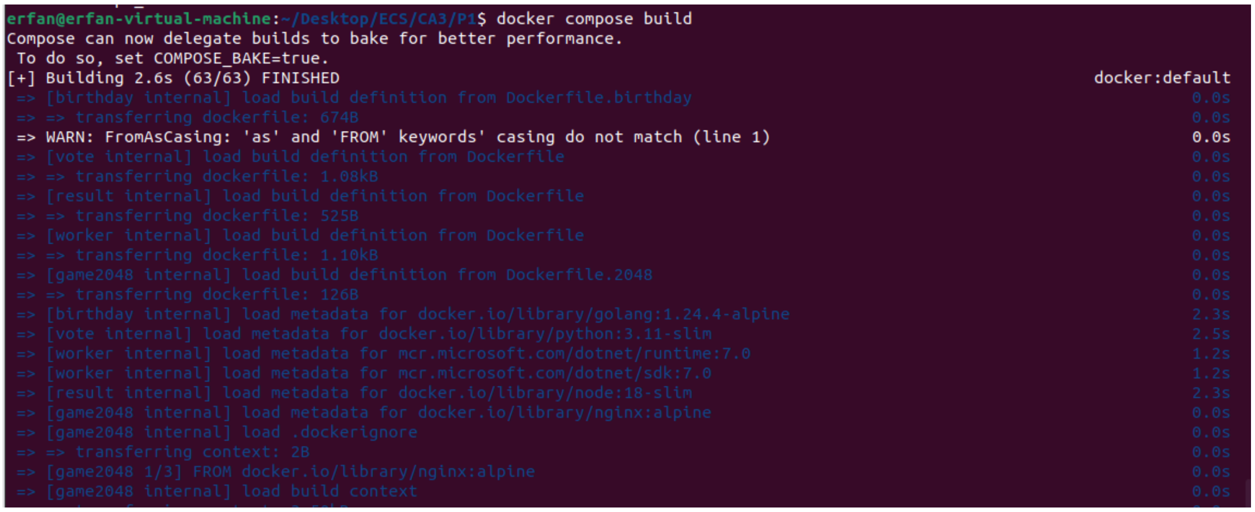
This file sets secure usernames and strong, randomly generated passwords for both the database and the admin login. It helps keep the app safe by protecting the database and restricting access to the results page.

Now we generate password hash and save it in .htpasswd:



```
erfan@erfan-virtual-machine:~/Desktop/ECS/CA3/P1$ htpasswd -bBc nginx/.htpasswd resultadmin hello123_mega_long_secret_password
_456
Adding password for user resultadmin
```

After changing cats and dogs to girl and boy in voting app, it’s time to build and run composer:

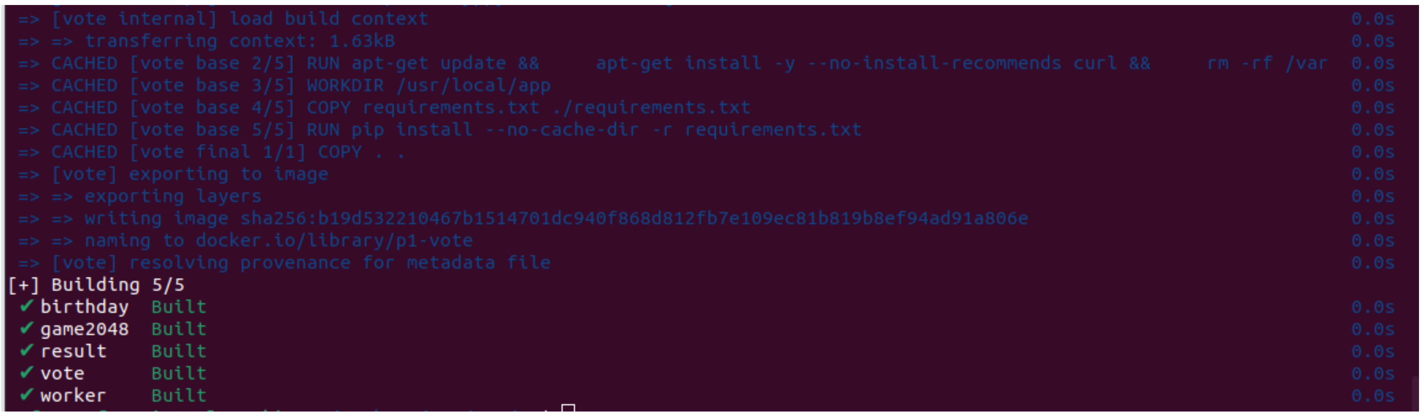


```
erfan@erfan-virtual-machine:~/Desktop/ECS/CA3/P1$ docker compose build
Compose can now delegate builds to bake for better performance.
 To do so, set COMPOSE_BAKE=true.
[+] Building 2.6s (63/63) FINISHED                                                                          docker:default
 => [birthday internal] load build definition from Dockerfile.birthday                                               0.0s
 => => transferring dockerfile: 674B                                                                                 0.0s
 => WARN: FromAsCasing: 'as' and 'FROM' keywords' casing do not match (line 1)                                       0.0s
 => [vote internal] load build definition from Dockerfile                                                            0.0s
 => => transferring dockerfile: 1.08kB                                                                               0.0s
 => [result internal] load build definition from Dockerfile                                                          0.0s
 => => transferring dockerfile: 525B                                                                                 0.0s
 => [worker internal] load build definition from Dockerfile                                                          0.0s
 => => transferring dockerfile: 1.10kB                                                                               0.0s
 => [game2048 internal] load build definition from Dockerfile.2048                                                   0.0s
 => => transferring dockerfile: 126B                                                                                 0.0s
 => [birthday internal] load metadata for docker.io/library/golang:1.24.4-alpine                                     2.3s
 => [vote internal] load metadata for docker.io/library/python:3.11-slim                                             2.5s
 => [worker internal] load metadata for mcr.microsoft.com/dotnet/runtime:7.0                                         1.2s
 => [worker internal] load metadata for mcr.microsoft.com/dotnet/sdk:7.0                                             1.2s
 => [result internal] load metadata for docker.io/library/node:18-slim                                               2.3s
 => [game2048 internal] load metadata for docker.io/library/nginx:alpine                                             0.0s
 => [game2048 internal] load .dockerignore                                                                           0.0s
 => => transferring context: 2B                                                                                      0.0s
 => [game2048 1/3] FROM docker.io/library/nginx:alpine                                                               0.0s
 => [game2048 internal] load build context                                                                           0.0s
```

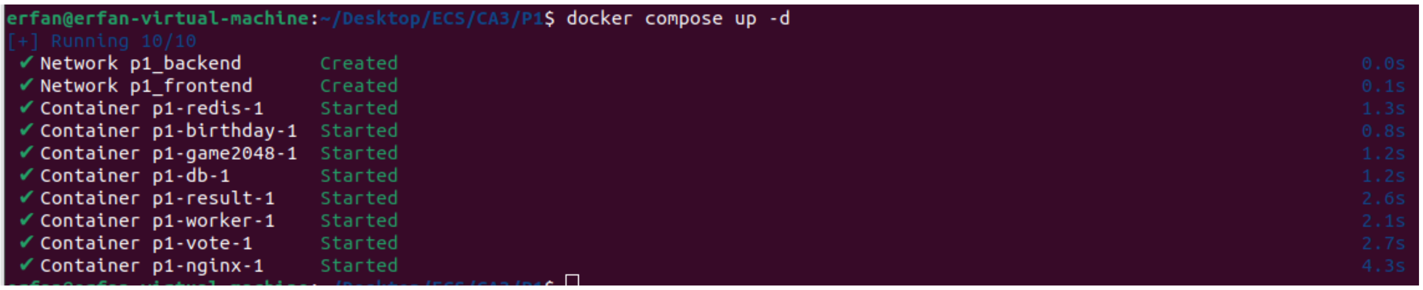
.

.

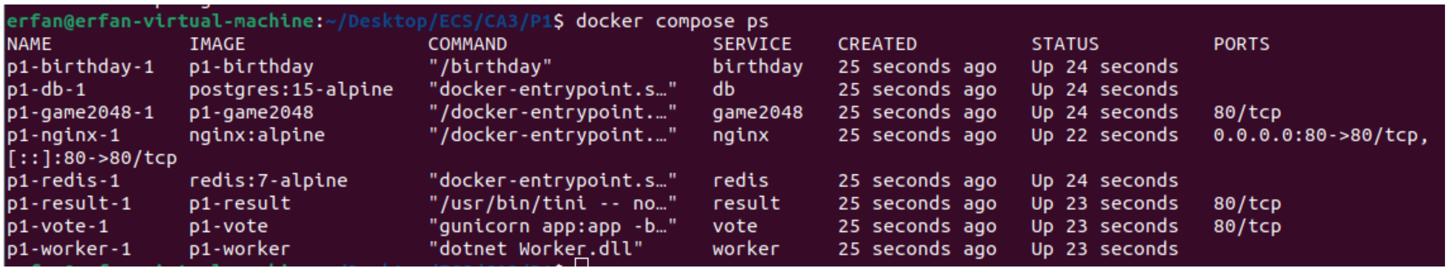
.



```
 => [vote internal] load build context                                                                               0.0s
 => => transferring context: 1.63kB                                                                                  0.0s
 => CACHED [vote base 2/5] RUN apt-get update &&     apt-get install -y --no-install-recommends curl &&     rm -rf /var  0.0s
 => CACHED [vote base 3/5] WORKDIR /usr/local/app                                                                    0.0s
 => CACHED [vote base 4/5] COPY requirements.txt ./requirements.txt                                                  0.0s
 => CACHED [vote base 5/5] RUN pip install --no-cache-dir -r requirements.txt                                        0.0s
 => CACHED [vote final 1/1] COPY . .                                                                                 0.0s
 => [vote] exporting to image                                                                                        0.0s
 => => exporting layers                                                                                              0.0s
 => => writing image sha256:b19d532210467b1514701dc940f868d812fb7e109ec81b819b8ef94ad91a806e                         0.0s
 => => naming to docker.io/library/p1-vote                                                                           0.0s
 => [vote] resolving provenance for metadata file                                                                    0.0s
[+] Building 5/5
 ✔ birthday  Built                                                                                                   0.0s
 ✔ game2048  Built                                                                                                   0.0s
 ✔ result    Built                                                                                                   0.0s
 ✔ vote      Built                                                                                                   0.0s
 ✔ worker    Built                                                                                                   0.0s
```



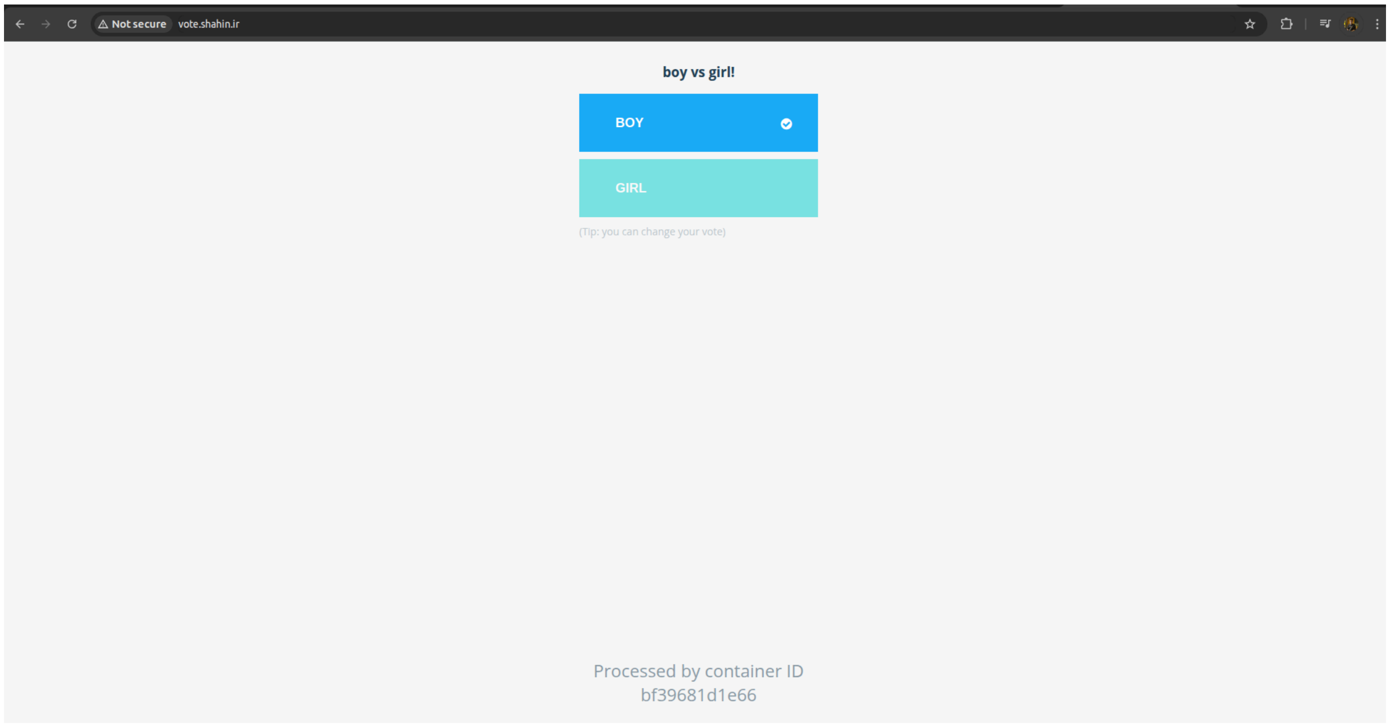
```
erfan@erfan-virtual-machine:~/Desktop/ECS/CA3/P1$ docker compose up -d
[+] Running 10/10
 ✔ Network p1_backend       Created                                                                                  0.0s
 ✔ Network p1_frontend      Created                                                                                  0.1s
 ✔ Container p1-redis-1     Started                                                                                  1.3s
 ✔ Container p1-birthday-1  Started                                                                                  0.8s
 ✔ Container p1-game2048-1  Started                                                                                  1.2s
 ✔ Container p1-db-1        Started                                                                                  1.2s
 ✔ Container p1-result-1    Started                                                                                  2.6s
 ✔ Container p1-worker-1    Started                                                                                  2.1s
 ✔ Container p1-vote-1      Started                                                                                  2.7s
 ✔ Container p1-nginx-1     Started                                                                                  4.3s
```



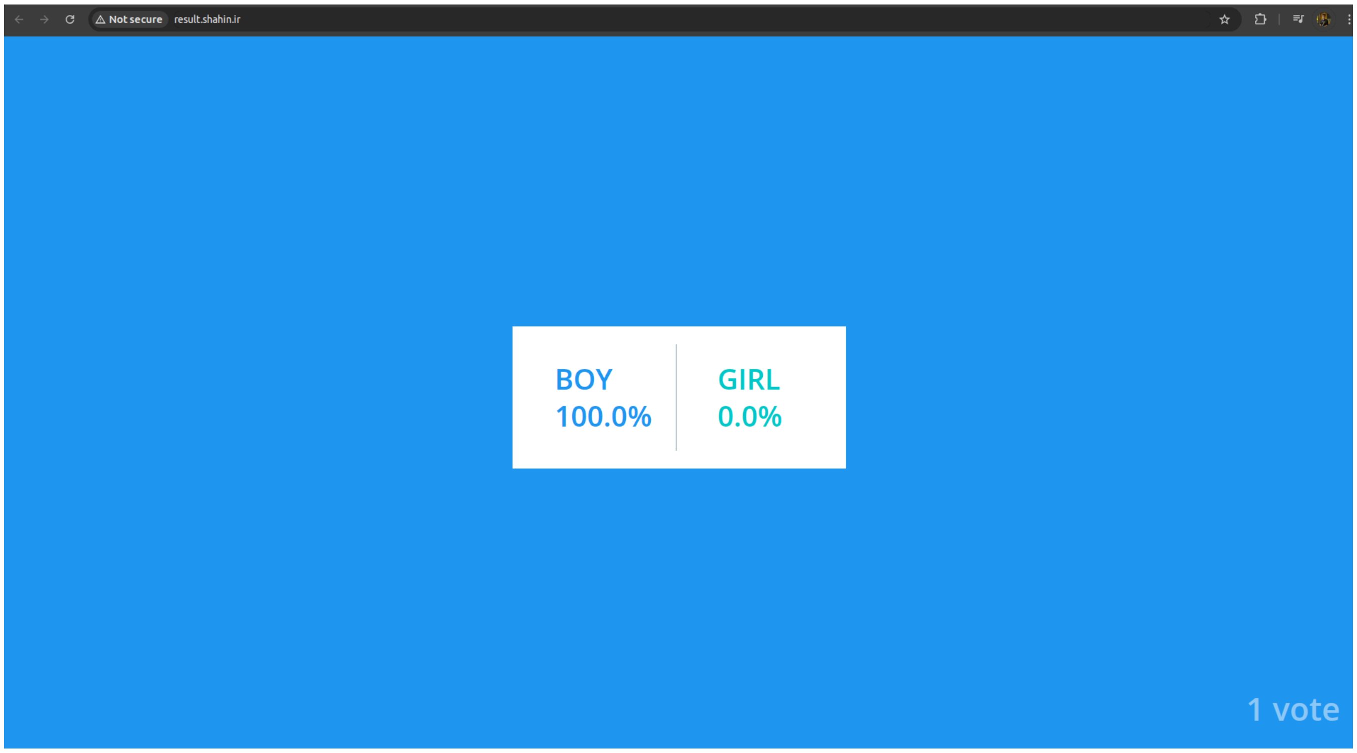
```
erfan@erfan-virtual-machine:~/Desktop/ECS/CA3/P1$ docker compose ps
NAME              IMAGE                COMMAND                  SERVICE     CREATED          STATUS          PORTS
p1-birthday-1     p1-birthday          "/birthday"              birthday    25 seconds ago   Up 24 seconds
p1-db-1           postgres:15-alpine   "docker-entrypoint.s…"   db          25 seconds ago   Up 24 seconds
p1-game2048-1     p1-game2048          "/docker-entrypoint.…"   game2048    25 seconds ago   Up 24 seconds   80/tcp
p1-nginx-1        nginx:alpine         "/docker-entrypoint.…"   nginx       25 seconds ago   Up 22 seconds   0.0.0.0:80->80/tcp,
[::]:80->80/tcp
p1-redis-1        redis:7-alpine       "docker-entrypoint.s…"   redis       25 seconds ago   Up 24 seconds
p1-result-1       p1-result            "/usr/bin/tini -- no…"   result      25 seconds ago   Up 23 seconds   80/tcp
p1-vote-1         p1-vote              "gunicorn app:app -b…"   vote        25 seconds ago   Up 23 seconds   80/tcp
p1-worker-1       p1-worker            "dotnet Worker.dll"      worker      25 seconds ago   Up 23 seconds
```

At last we can visit:

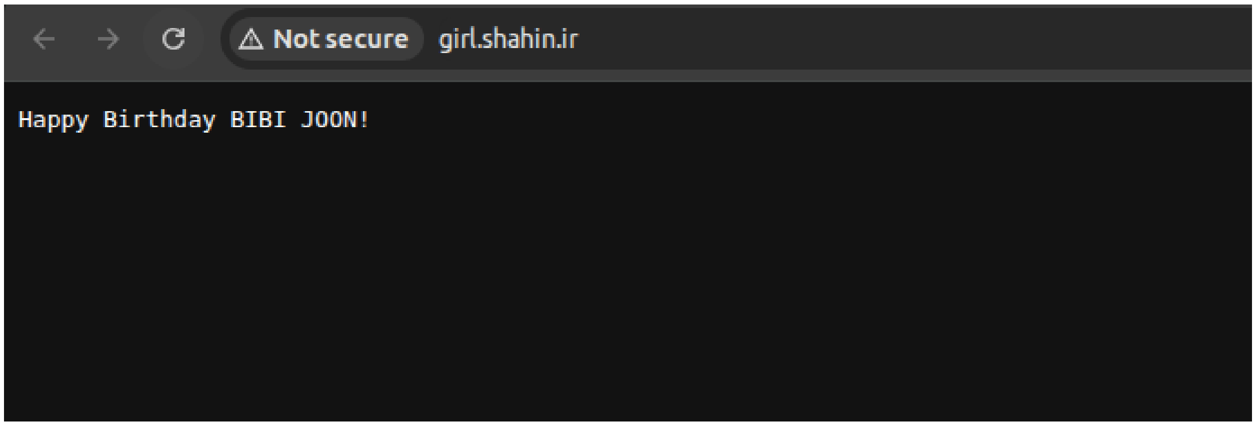
- http://vote.shahin.ir : voting form (choices: girl / boy)



- http://result.shahin.ir : result of voting

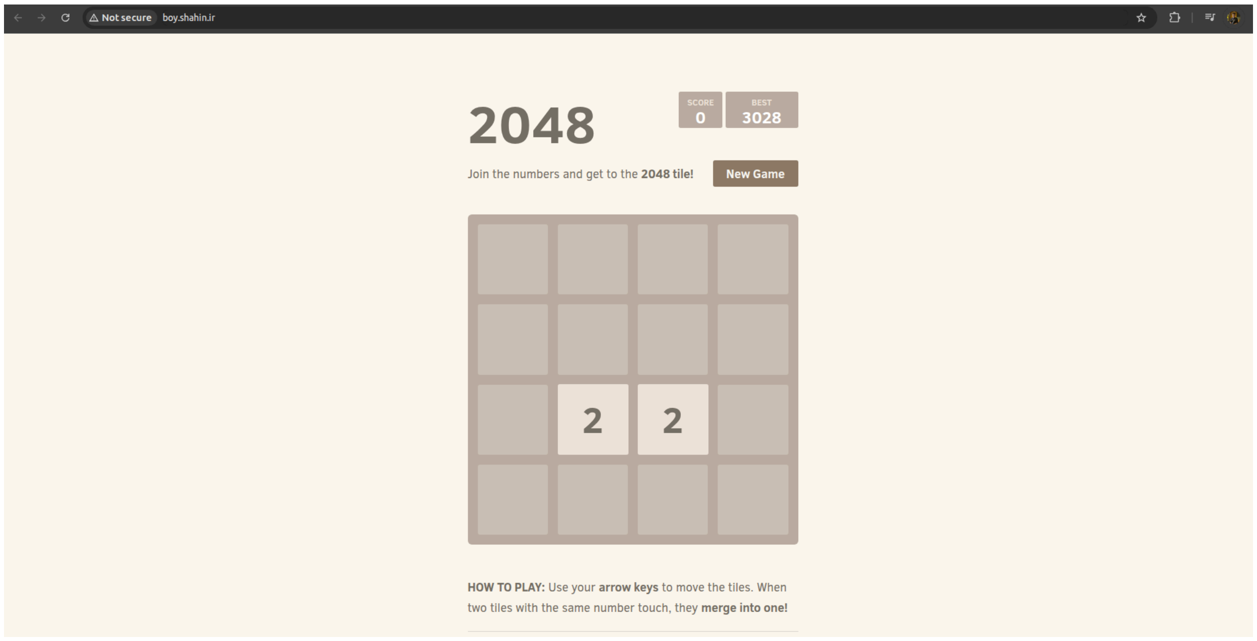


- http://girl.shahin.ir : Go “Happy Birthday BIBI JOON!” page



```
Happy Birthday BIBI JOON!
```

- http://boy.shahin.ir : 2048 game



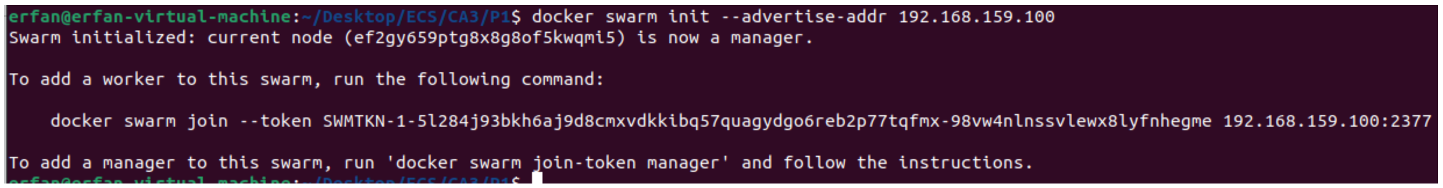
Join the numbers and get to the **2048 tile!**

HOW TO PLAY: Use your **arrow keys** to move the tiles. When two tiles with the same number touch, they **merge into one!**

Swarm:

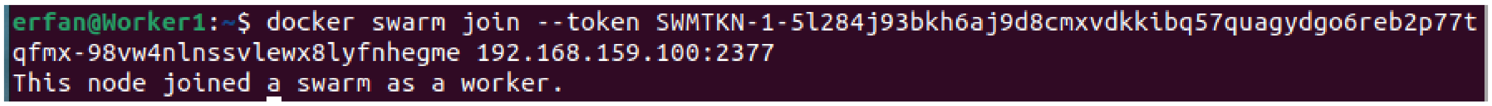
First, we created full clones of the main VM in VMware Workstation to set up the worker machines. Then, we enabled the Host-Only Network (VMnet1) for each machine so they could ping each other. After that, we assigned static IP addresses to each of them.

After setting up 4 machines, we initialized Docker Swarm on the main machine:

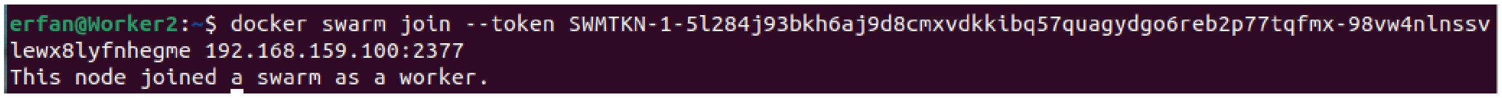


```
erfan@erfan-virtual-machine:~/Desktop/ECS/CA3/P1$ docker swarm init --advertise-addr 192.168.159.100
Swarm initialized: current node (ef2gy659ptg8x8g8of5kwqmi5) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-5l284j93bkh6aj9d8cmxvdkkibq57quagydgo6reb2p77tqfmx-98vw4nlnssvlewx8lyfnhegme 192.168.159.100:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.
```

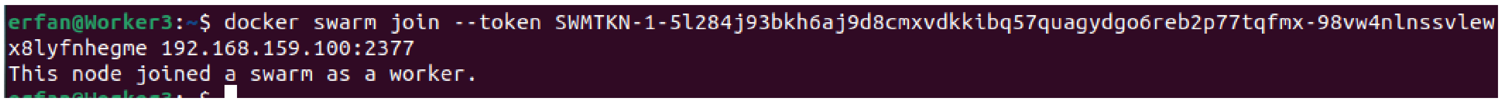
Then, we ran the join command copied from the manager on each worker:



```
erfan@Worker1:~$ docker swarm join --token SWMTKN-1-5l284j93bkh6aj9d8cmxvdkkibq57quagydgo6reb2p77t
qfmx-98vw4nlnssvlewx8lyfnhegme 192.168.159.100:2377
This node joined a swarm as a worker.
```

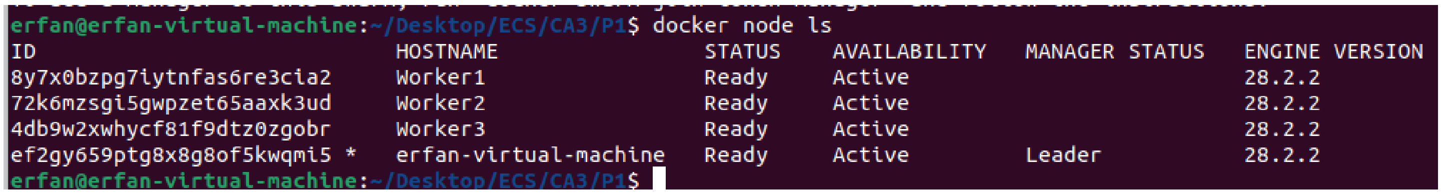


```
erfan@Worker2:~$ docker swarm join --token SWMTKN-1-5l284j93bkh6aj9d8cmxvdkkibq57quagydgo6reb2p77tqfmx-98vw4nlnssv
lewx8lyfnhegme 192.168.159.100:2377
This node joined a swarm as a worker.
```



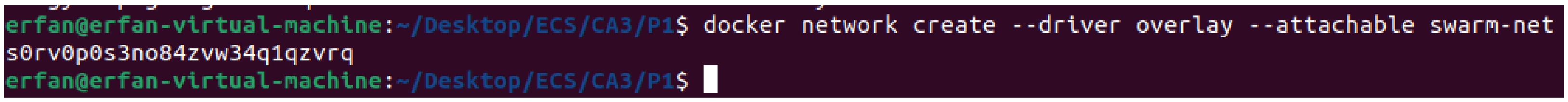
```
erfan@Worker3:~$ docker swarm join --token SWMTKN-1-5l284j93bkh6aj9d8cmxvdkkibq57quagydgo6reb2p77tqfmx-98vw4nlnssvlew
x8lyfnhegme 192.168.159.100:2377
This node joined a swarm as a worker.
```

￼￼￼￼￼￼￼￼￼￼To verify cluster is working:



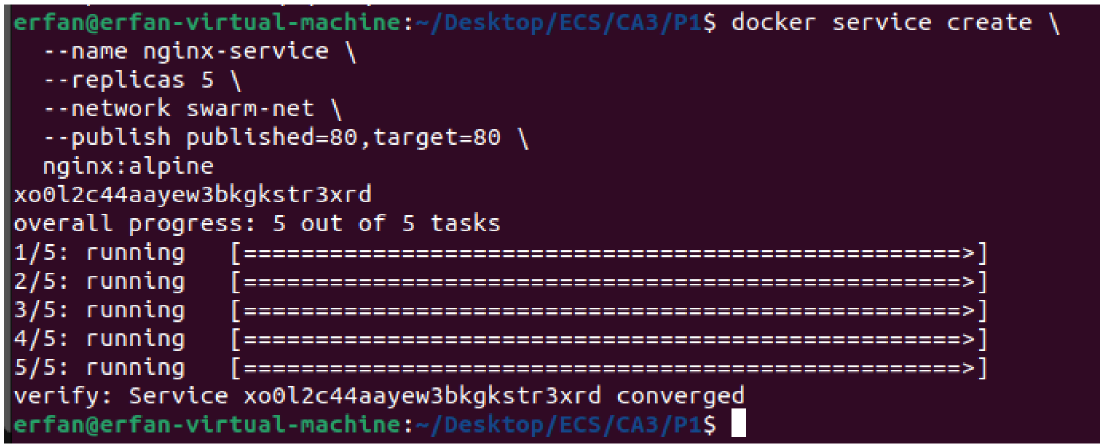
```
erfan@erfan-virtual-machine:~/Desktop/ECS/CA3/P1$ docker node ls
ID                            HOSTNAME                 STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
8y7x0bzpg7iytnfas6re3cia2     Worker1                  Ready     Active                          28.2.2
72k6mzsgi5gwpzet65aaxk3ud     Worker2                  Ready     Active                          28.2.2
4db9w2xwhycf81f9dtz0zgobr     Worker3                  Ready     Active                          28.2.2
ef2gy659ptg8x8g8of5kwqmi5 *   erfan-virtual-machine    Ready     Active         Leader           28.2.2
erfan@erfan-virtual-machine:~/Desktop/ECS/CA3/P1$
```

Then, below command is run to let containers talk to each other across nodes:



```
erfan@erfan-virtual-machine:~/Desktop/ECS/CA3/P1$ docker network create --driver overlay --attachable swarm-net
s0rv0p0s3no84zvw34q1qzvrq
erfan@erfan-virtual-machine:~/Desktop/ECS/CA3/P1$
```

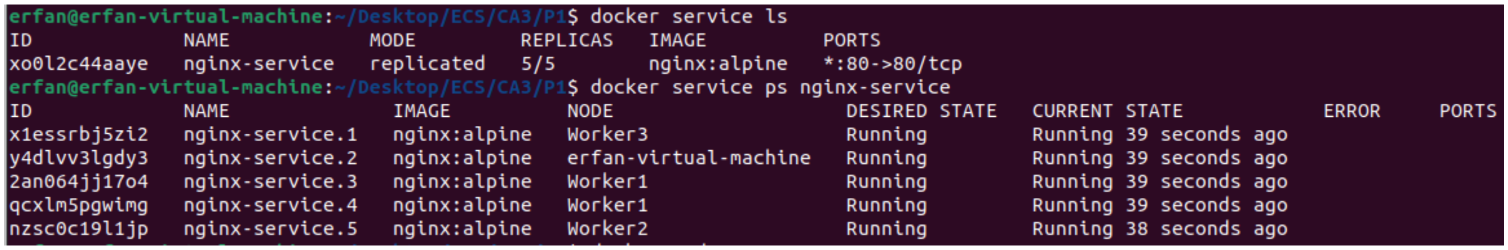
Now on main machine, we deploy Nginx with 5 replicas:



```
erfan@erfan-virtual-machine:~/Desktop/ECS/CA3/P1$ docker service create \
  --name nginx-service \
  --replicas 5 \
  --network swarm-net \
  --publish published=80,target=80 \
  nginx:alpine
xo0l2c44aayew3bkgkstr3xrd
overall progress: 5 out of 5 tasks
1/5: running   [==================================================>]
2/5: running   [==================================================>]
3/5: running   [==================================================>]
4/5: running   [==================================================>]
5/5: running   [==================================================>]
verify: Service xo0l2c44aayew3bkgkstr3xrd converged
erfan@erfan-virtual-machine:~/Desktop/ECS/CA3/P1$
```

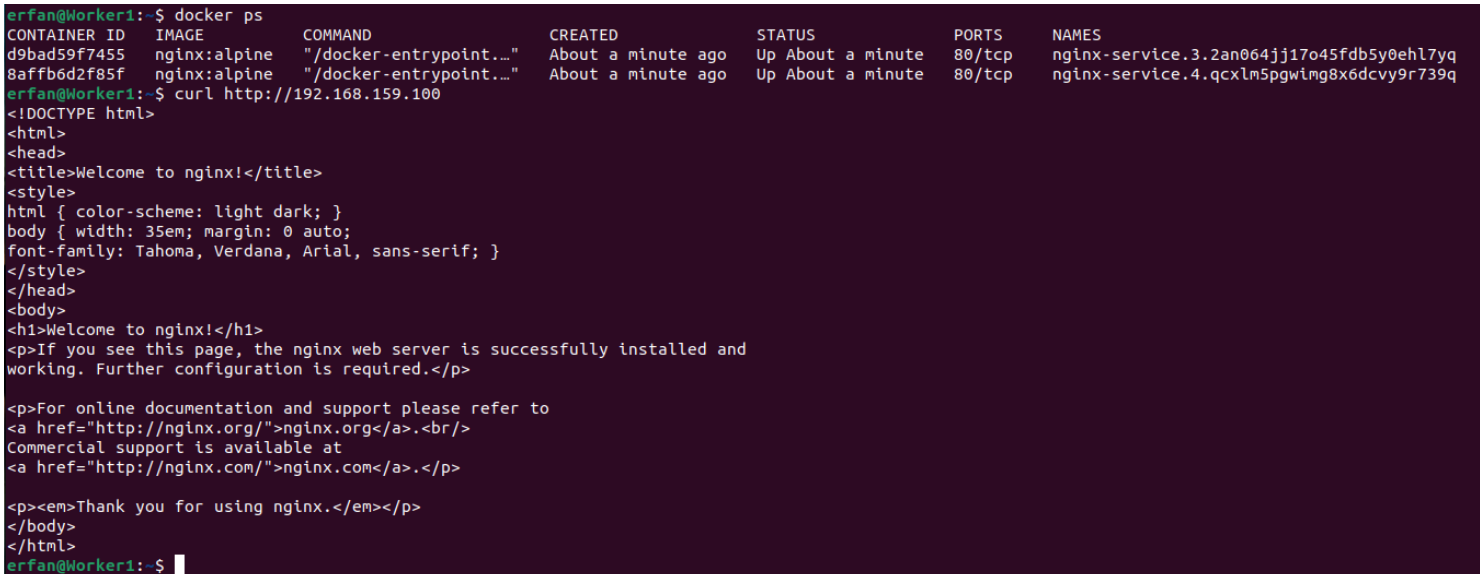
- **--replicas 5:** run 5 containers across all nodes
  --publish: exposes port 80 on the manager to the outside
- **--network swarm-net:** attaches containers to our overlay network

Validating deployment:

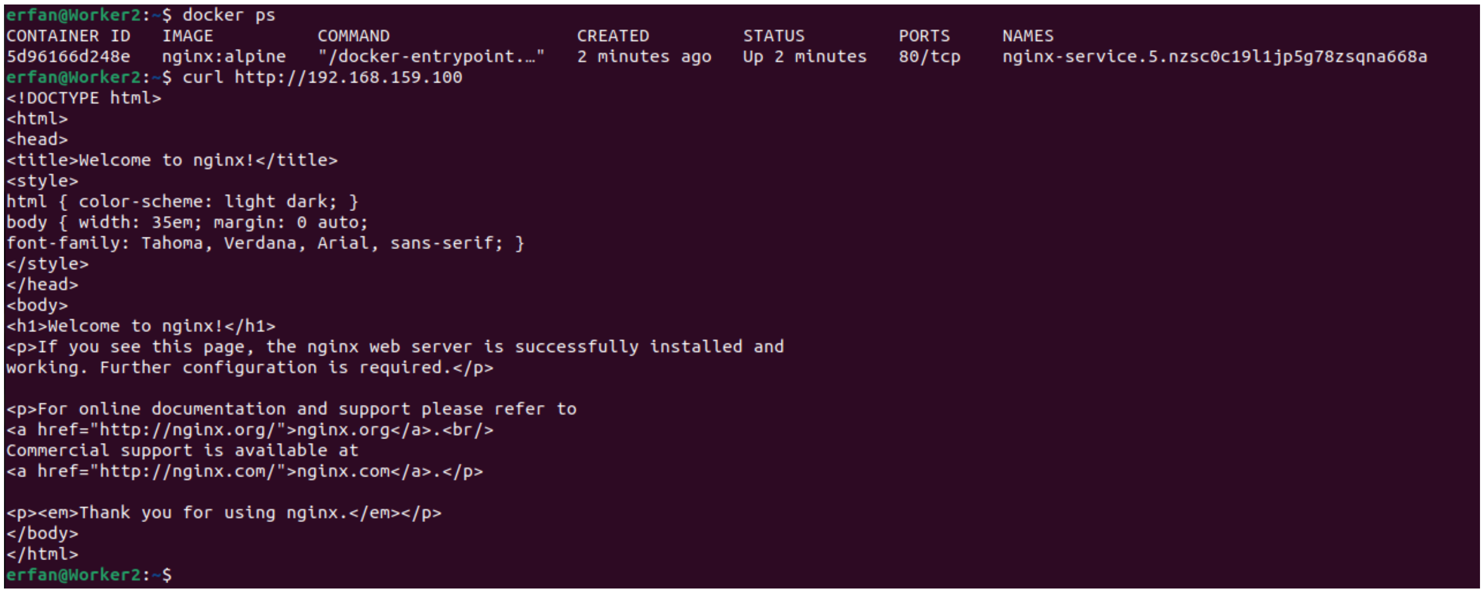


```
erfan@erfan-virtual-machine:~/Desktop/ECS/CA3/P1$ docker service ls
ID             NAME            MODE         REPLICAS   IMAGE          PORTS
xo0l2c44aaye   nginx-service   replicated   5/5        nginx:alpine   *:80->80/tcp
erfan@erfan-virtual-machine:~/Desktop/ECS/CA3/P1$ docker service ps nginx-service
ID             NAME              IMAGE          NODE                      DESIRED STATE   CURRENT STATE            ERROR     PORTS
x1essrbj5zi2   nginx-service.1   nginx:alpine   Worker3                   Running         Running 39 seconds ago
y4dlvv3lgdy3   nginx-service.2   nginx:alpine   erfan-virtual-machine     Running         Running 39 seconds ago
2an064jj17o4   nginx-service.3   nginx:alpine   Worker1                   Running         Running 39 seconds ago
qcxlm5pgwimg   nginx-service.4   nginx:alpine   Worker1                   Running         Running 39 seconds ago
nzsc0c19l1jp   nginx-service.5   nginx:alpine   Worker2                   Running         Running 38 seconds ago
```

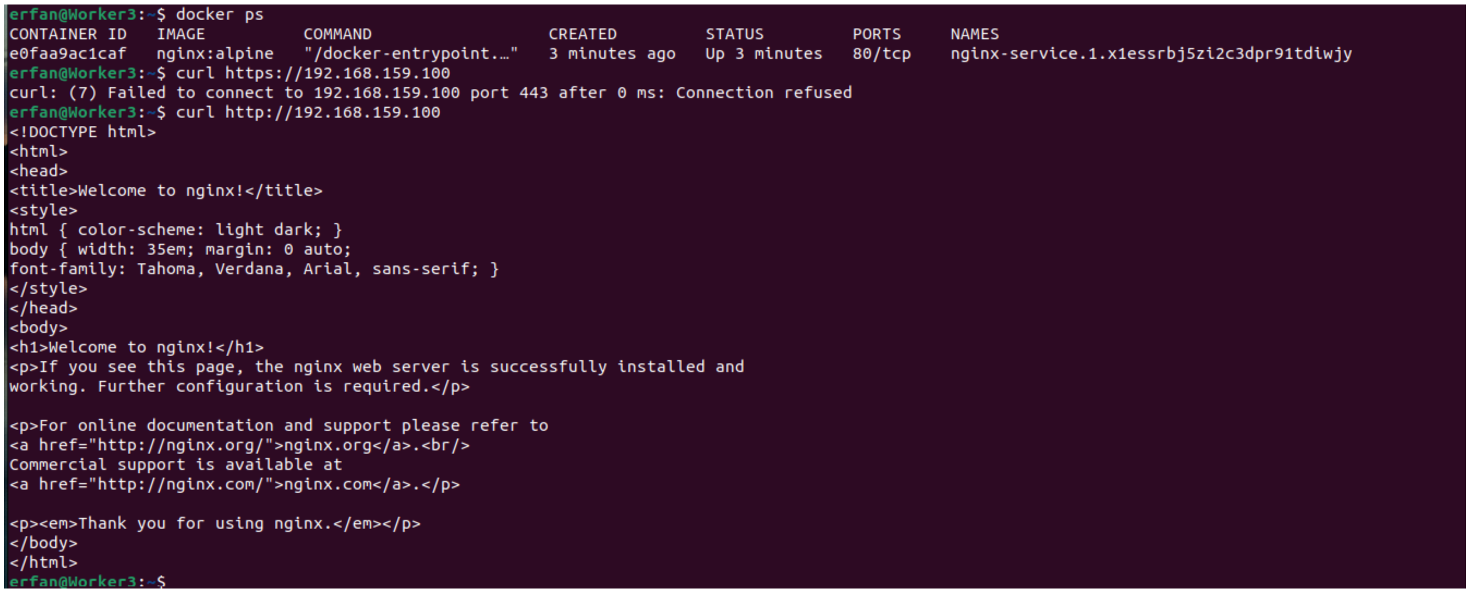
Test workers:



```
erfan@Worker1:~$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED              STATUS              PORTS     NAMES
d9bad59f7455   nginx:alpine   "/docker-entrypoint.…"   About a minute ago   Up About a minute   80/tcp    nginx-service.3.2an064jj17o45fdb5y0ehl7yq
8affb6d2f85f   nginx:alpine   "/docker-entrypoint.…"   About a minute ago   Up About a minute   80/tcp    nginx-service.4.qcxlm5pgwimg8x6dcvy9r739q
erfan@Worker1:~$ curl http://192.168.159.100
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
erfan@Worker1:~$
```



```
erfan@Worker2:~$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS     NAMES
5d96166d248e   nginx:alpine   "/docker-entrypoint.…"   2 minutes ago   Up 2 minutes   80/tcp    nginx-service.5.nzsc0c19l1jp5g78zsqna668a
erfan@Worker2:~$ curl http://192.168.159.100
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
erfan@Worker2:~$
```

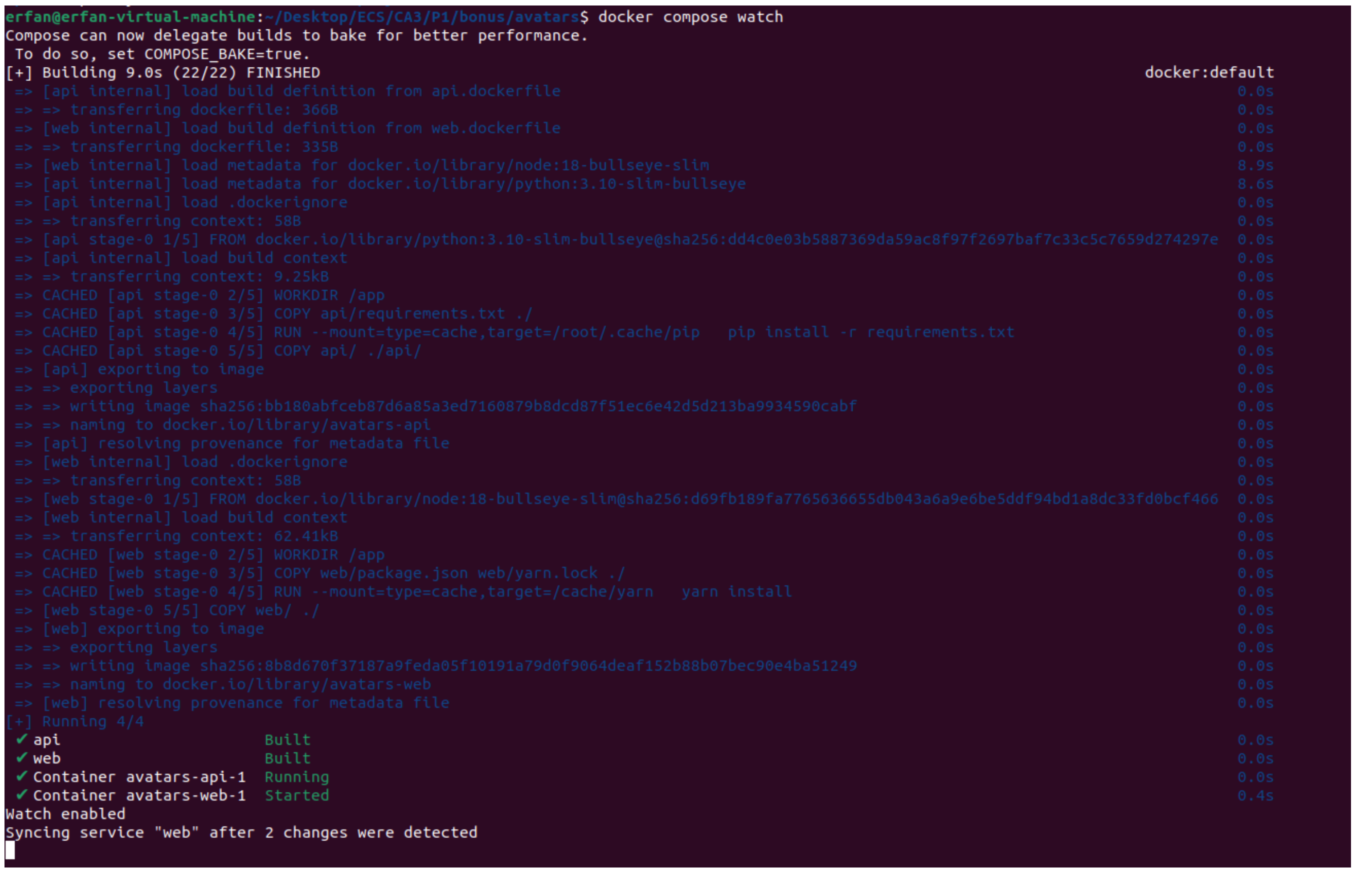


```
erfan@Worker3:~$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS     NAMES
e0faa9ac1caf   nginx:alpine   "/docker-entrypoint.…"   3 minutes ago   Up 3 minutes   80/tcp    nginx-service.1.x1essrbj5zi2c3dpr91tdiwjy
erfan@Worker3:~$ curl https://192.168.159.100
curl: (7) Failed to connect to 192.168.159.100 port 443 after 0 ms: Connection refused
erfan@Worker3:~$ curl http://192.168.159.100
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
erfan@Worker3:~$
```

برای دسترسی به اپلیکیشن دیپلوی‌شده روی کلاستر داکر سوآرم از بیرون، ما معمولاً یکی از نودها، مثلاً نود main، را به عنوان درگاه ورودی انتخاب می‌کنیم و فقط پورت مشخصی مثل پورت 80 را روی آن باز می‌گذاریم. در این حالت، تمام درخواست‌هایی که از بیرون (مثلاً مرورگر کاربر یا curl) به IP این ماشین ارسال می‌شود، توسط خود داکر بین سرویس‌ها و کانتینرهای مختلف پشت صحنه load balance و مدیریت می‌شود.
بنابراین از نگاه کاربر، فقط با یک IP و یک پورت طرف هستیم، ولی در پشت صحنه ممکن است درخواست‌ها بین چند ماشین پخش شوند. برای اینکه کاربران بتوانند به راحتی با نام دامنه به این اپ دسترسی داشته باشند، باید آدرس IP نود manager را به عنوان رکورد A در DNS ثبت کنیم؛ یعنی وقتی کاربر مثلاً به app.example.com وصل می‌شود، DNS باید آدرس IP همان نودی را بدهد که پورت 80 آن پابلیش شده است و درخواست را به Swarm منتقل می‌کند.

Questions:

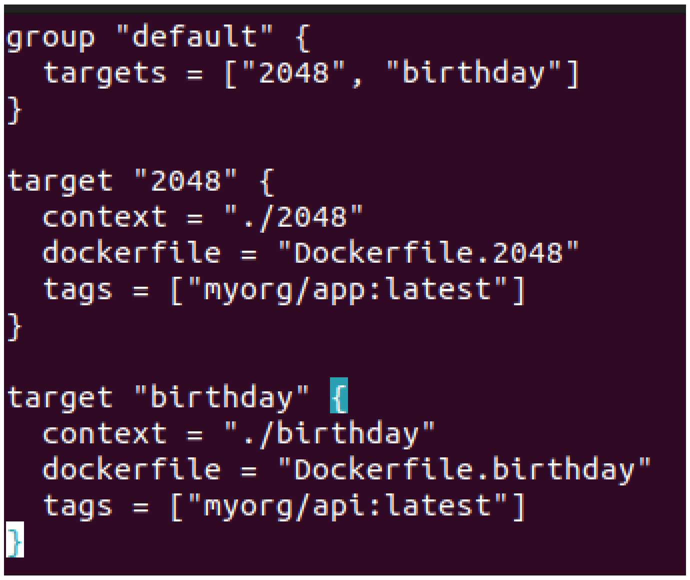
1) قابلیت docker compose watch به ما اجازه می‌دهد تغییرات لحظه‌ای در فایل‌های پروژه را بدون نیاز به build مجدد یا restart کردن کانتینرها مشاهده کنیم. برای اجرای عملی این قابلیت، ابتدا پروژه Avatars را کلون می‌کنیم و با اجرای docker compose watch به جای docker compose up، سرویس‌ها را بالا می‌آوریم. سپس می‌توانیم مثلاً رنگ یک بخش در فایل public/styles.css را تغییر دهیم. بعد از ذخیره فایل، تغییر به صورت زنده روی اپلیکیشن اعمال می‌شود و نیاز به هیچ دستور اضافی نیست. این ویژگی روند توسعه را بسیار سریع‌تر می‌کند.



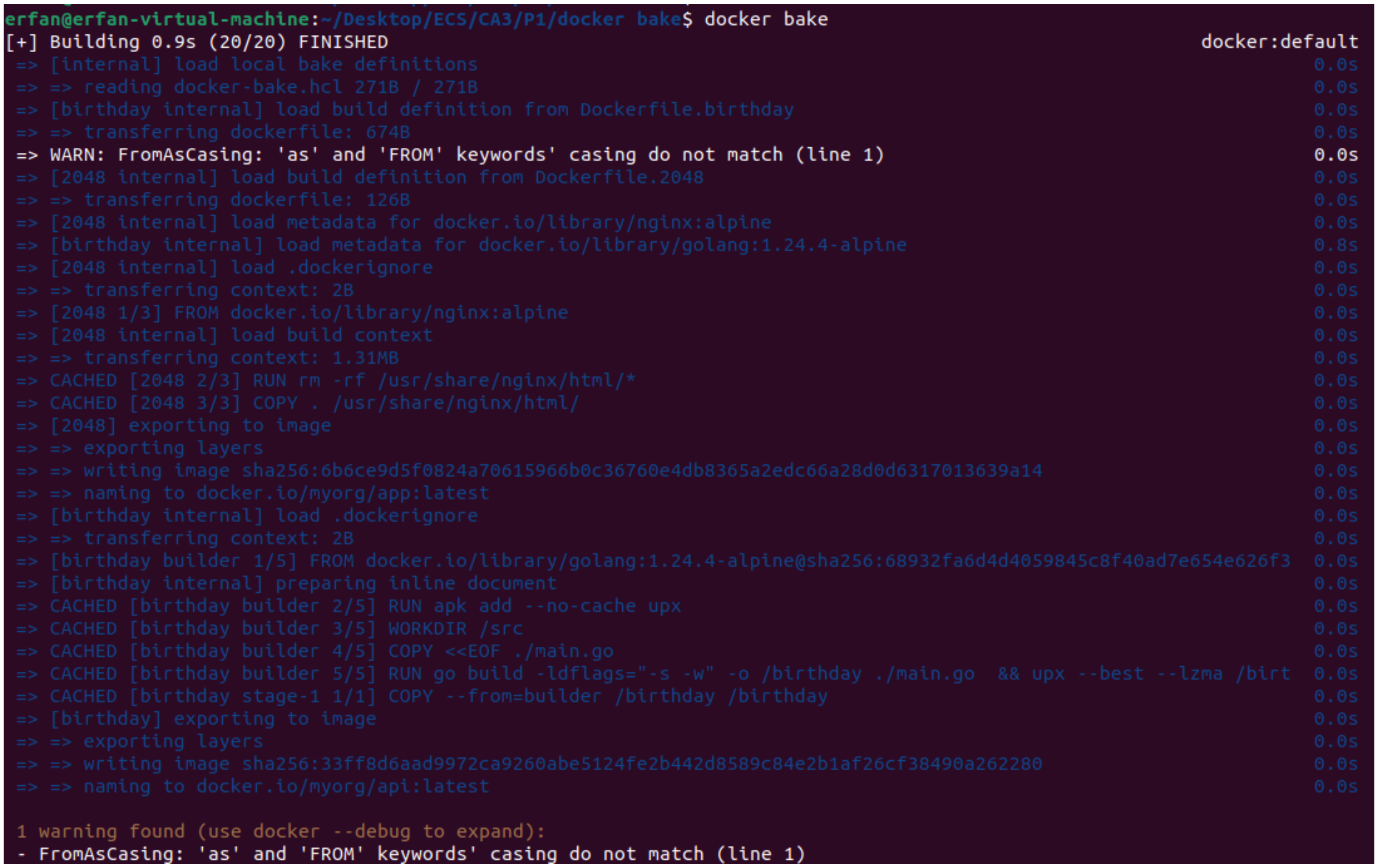
```
erfan@erfan-virtual-machine:~/Desktop/ECS/CA3/P1/bonus/avatars$ docker compose watch
Compose can now delegate builds to bake for better performance.
 To do so, set COMPOSE_BAKE=true.
[+] Building 9.0s (22/22) FINISHED                                                                                         docker:default
 => [api internal] load build definition from api.dockerfile                                                                          0.0s
 => => transferring dockerfile: 366B                                                                                                  0.0s
 => [web internal] load build definition from web.dockerfile                                                                          0.0s
 => => transferring dockerfile: 335B                                                                                                  0.0s
 => [web internal] load metadata for docker.io/library/node:18-bullseye-slim                                                          8.9s
 => [api internal] load metadata for docker.io/library/python:3.10-slim-bullseye                                                      8.6s
 => [api internal] load .dockerignore                                                                                                 0.0s
 => => transferring context: 58B                                                                                                      0.0s
 => [api stage-0 1/5] FROM docker.io/library/python:3.10-slim-bullseye@sha256:dd4c0e03b5887369da59ac8f97f2697baf7c33c5c7659d274297e  0.0s
 => [api internal] load build context                                                                                                 0.0s
 => => transferring context: 9.25kB                                                                                                   0.0s
 => CACHED [api stage-0 2/5] WORKDIR /app                                                                                             0.0s
 => CACHED [api stage-0 3/5] COPY api/requirements.txt ./                                                                             0.0s
 => CACHED [api stage-0 4/5] RUN --mount=type=cache,target=/root/.cache/pip   pip install -r requirements.txt                         0.0s
 => CACHED [api stage-0 5/5] COPY api/ ./api/                                                                                         0.0s
 => [api] exporting to image                                                                                                          0.0s
 => => exporting layers                                                                                                               0.0s
 => => writing image sha256:bb180abfceb87d6a85a3ed7160879b8dcd87f51ec6e42d5d213ba9934590cabf                                          0.0s
 => => naming to docker.io/library/avatars-api                                                                                        0.0s
 => [api] resolving provenance for metadata file                                                                                      0.0s
 => [web internal] load .dockerignore                                                                                                 0.0s
 => => transferring context: 58B                                                                                                      0.0s
 => [web stage-0 1/5] FROM docker.io/library/node:18-bullseye-slim@sha256:d69fb189fa7765636655db043a6a9e6be5ddf94bd1a8dc33fd0bcf466  0.0s
 => [web internal] load build context                                                                                                 0.0s
 => => transferring context: 62.41kB                                                                                                  0.0s
 => CACHED [web stage-0 2/5] WORKDIR /app                                                                                             0.0s
 => CACHED [web stage-0 3/5] COPY web/package.json web/yarn.lock ./                                                                   0.0s
 => CACHED [web stage-0 4/5] RUN --mount=type=cache,target=/cache/yarn   yarn install                                                 0.0s
 => [web stage-0 5/5] COPY web/ ./                                                                                                    0.0s
 => [web] exporting to image                                                                                                          0.0s
 => => exporting layers                                                                                                               0.0s
 => => writing image sha256:8b8d670f37187a9feda05f10191a79d0f9064deaf152b88b07bec90e4ba51249                                          0.0s
 => => naming to docker.io/library/avatars-web                                                                                        0.0s
 => [web] resolving provenance for metadata file                                                                                      0.0s
[+] Running 4/4
 ✔ api                        Built                                                                                                   0.0s
 ✔ web                        Built                                                                                                   0.0s
 ✔ Container avatars-api-1  Running                                                                                                   0.0s
 ✔ Container avatars-web-1  Started                                                                                                   0.4s
Watch enabled
Syncing service "web" after 2 changes were detected
```

2) docker wasm قابلیت جدیدی است که اجرای ماژول‌های WebAssembly را در محیط Docker ممکن می‌کند. این ویژگی برای اجرای برنامه‌های سبک، سریع و امن بسیار مفید است و به ما اجازه می‌دهد اپلیکیشن‌هایی را که به زبان‌هایی مثل Rust یا Go نوشته شده‌اند، با سرعت بالا روی کانتینر اجرا کنیم. در مقابل، docker bake ابزاری برای build موازی چندین نسخه از ایمیج است که با استفاده از فایل docker-bake.hcl می‌توان buildهای پیچیده را تعریف و مدیریت کرد. این ابزار مخصوصاً زمانی مفید است که بخواهیم یک پروژه را برای چند معماری یا پلتفرم مختلف بسازیم، مثل amd64 و arm64 به‌صورت همزمان.

Docker bake:

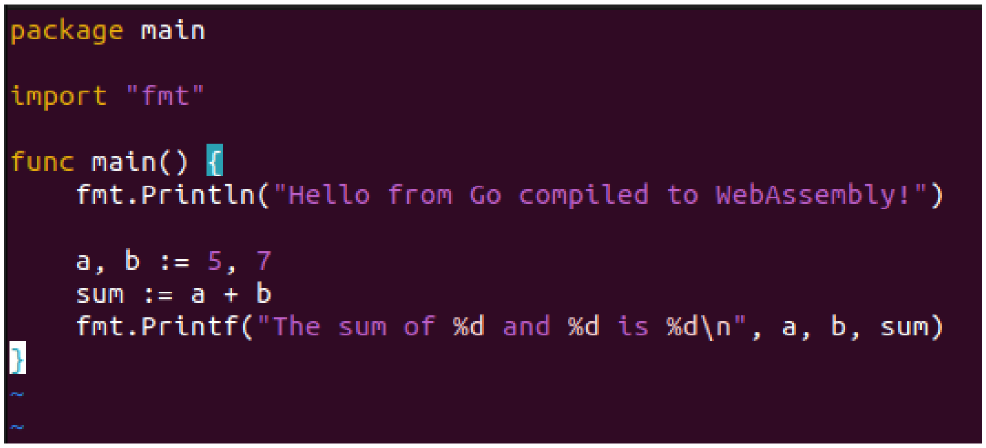


```
group "default" {
  targets = ["2048", "birthday"]
}

target "2048" {
  context = "./2048"
  dockerfile = "Dockerfile.2048"
  tags = ["myorg/app:latest"]
}

target "birthday" {
  context = "./birthday"
  dockerfile = "Dockerfile.birthday"
  tags = ["myorg/api:latest"]
}
```

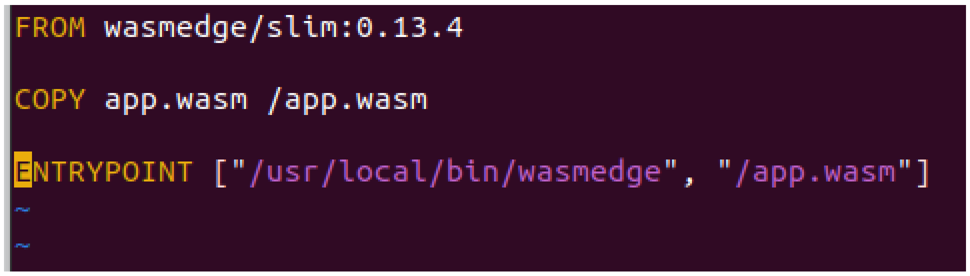


```
erfan@erfan-virtual-machine:~/Desktop/ECS/CA3/P1/docker bake$ docker bake
[+] Building 0.9s (20/20) FINISHED                                                                        docker:default
 => [internal] load local bake definitions                                                                          0.0s
 => => reading docker-bake.hcl 271B / 271B                                                                          0.0s
 => [birthday internal] load build definition from Dockerfile.birthday                                              0.0s
 => => transferring dockerfile: 674B                                                                                0.0s
 => WARN: FromAsCasing: 'as' and 'FROM' keywords' casing do not match (line 1)                                      0.0s
 => [2048 internal] load build definition from Dockerfile.2048                                                      0.0s
 => => transferring dockerfile: 126B                                                                                0.0s
 => [2048 internal] load metadata for docker.io/library/nginx:alpine                                                0.0s
 => [birthday internal] load metadata for docker.io/library/golang:1.24.4-alpine                                    0.8s
 => [2048 internal] load .dockerignore                                                                              0.0s
 => => transferring context: 2B                                                                                     0.0s
 => [2048 1/3] FROM docker.io/library/nginx:alpine                                                                  0.0s
 => [2048 internal] load build context                                                                              0.0s
 => => transferring context: 1.31MB                                                                                 0.0s
 => CACHED [2048 2/3] RUN rm -rf /usr/share/nginx/html/*                                                            0.0s
 => CACHED [2048 3/3] COPY . /usr/share/nginx/html/                                                                 0.0s
 => [2048] exporting to image                                                                                       0.0s
 => => exporting layers                                                                                             0.0s
 => => writing image sha256:6b6ce9d5f0824a70615966b0c36760e4db8365a2edc66a28d0d6317013639a14                        0.0s
 => => naming to docker.io/myorg/app:latest                                                                         0.0s
 => [birthday internal] load .dockerignore                                                                          0.0s
 => => transferring context: 2B                                                                                     0.0s
 => [birthday builder 1/5] FROM docker.io/library/golang:1.24.4-alpine@sha256:68932fa6d4d4059845c8f40ad7e654e626f3  0.0s
 => [birthday internal] preparing inline document                                                                   0.0s
 => CACHED [birthday builder 2/5] RUN apk add --no-cache upx                                                        0.0s
 => CACHED [birthday builder 3/5] WORKDIR /src                                                                      0.0s
 => CACHED [birthday builder 4/5] COPY <<EOF ./main.go                                                              0.0s
 => CACHED [birthday builder 5/5] RUN go build -ldflags="-s -w" -o /birthday ./main.go  && upx --best --lzma /birt  0.0s
 => CACHED [birthday stage-1 1/1] COPY --from=builder /birthday /birthday                                           0.0s
 => [birthday] exporting to image                                                                                   0.0s
 => => exporting layers                                                                                             0.0s
 => => writing image sha256:33ff8d6aad9972ca9260abe5124fe2b442d8589c84e2b1af26cf38490a262280                        0.0s
 => => naming to docker.io/myorg/api:latest                                                                         0.0s

 1 warning found (use docker --debug to expand):
 - FromAsCasing: 'as' and 'FROM' keywords' casing do not match (line 1)
```

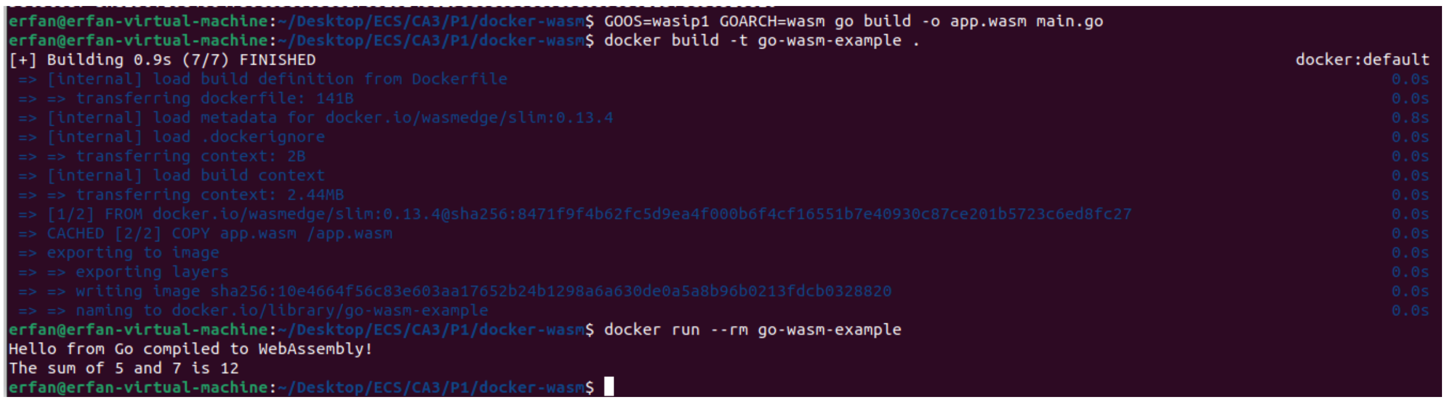
Docker wasm:



```go
package main

import "fmt"

func main() {
	fmt.Println("Hello from Go compiled to WebAssembly!")

	a, b := 5, 7
	sum := a + b
	fmt.Printf("The sum of %d and %d is %d\n", a, b, sum)
}
~
~
```



```dockerfile
FROM wasmedge/slim:0.13.4

COPY app.wasm /app.wasm

ENTRYPOINT ["/usr/local/bin/wasmedge", "/app.wasm"]
~
~
```



```
erfan@erfan-virtual-machine:~/Desktop/ECS/CA3/P1/docker-wasm$ GOOS=wasip1 GOARCH=wasm go build -o app.wasm main.go
erfan@erfan-virtual-machine:~/Desktop/ECS/CA3/P1/docker-wasm$ docker build -t go-wasm-example .
[+] Building 0.9s (7/7) FINISHED                                                                          docker:default
 => [internal] load build definition from Dockerfile                                                                0.0s
 => => transferring dockerfile: 141B                                                                                0.0s
 => [internal] load metadata for docker.io/wasmedge/slim:0.13.4                                                     0.8s
 => [internal] load .dockerignore                                                                                   0.0s
 => => transferring context: 2B                                                                                     0.0s
 => [internal] load build context                                                                                   0.0s
 => => transferring context: 2.44MB                                                                                 0.0s
 => [1/2] FROM docker.io/wasmedge/slim:0.13.4@sha256:8471f9f4b62fc5d9ea4f000b6f4cf16551b7e40930c87ce201b5723c6ed8fc27  0.0s
 => CACHED [2/2] COPY app.wasm /app.wasm                                                                            0.0s
 => exporting to image                                                                                              0.0s
 => => exporting layers                                                                                             0.0s
 => => writing image sha256:10e4664f56c83e603aa17652b24b1298a6a630de0a5a8b96b0213fdcb0328820                        0.0s
 => => naming to docker.io/library/go-wasm-example                                                                  0.0s
erfan@erfan-virtual-machine:~/Desktop/ECS/CA3/P1/docker-wasm$ docker run --rm go-wasm-example
Hello from Go compiled to WebAssembly!
The sum of 5 and 7 is 12
erfan@erfan-virtual-machine:~/Desktop/ECS/CA3/P1/docker-wasm$
```

3) RUN برای اجرای دستورات هنگام build استفاده می‌شود و باعث ایجاد یک لایه جدید می‌شود؛ مثلاً نصب پکیج یا ساخت فایل. COPY و ADD هر دو برای انتقال فایل از سیستم میزبان به ایمیج استفاده می‌شوند، با این تفاوت که ADD قابلیت‌های بیشتری مثل unpack کردن فایل‌های فشرده دارد. دستور ENTRYPOINT مشخص می‌کند که کانتینر همیشه با چه برنامه‌ای شروع شود و COMMAND مشخص می‌کند که پارامترهای پیش‌فرض اجرای کانتینر چه باشد. WORKSPACE یا WORKDIR مسیر کاری پیش‌فرض داخل کانتینر را مشخص می‌کند. از بین این دستورات، RUN، COPY و ADD باعث ایجاد لایه جدید در ایمیج می‌شوند و به طور مستقیم بر حجم و اندازه خروجی تأثیر می‌گذارند، در حالی که ENTRYPOINT، COMMAND و WORKDIR فقط تنظیمات مربوط به اجرای کانتینر هستند و لایه‌ای به ایمیج اضافه نمی‌کنند.