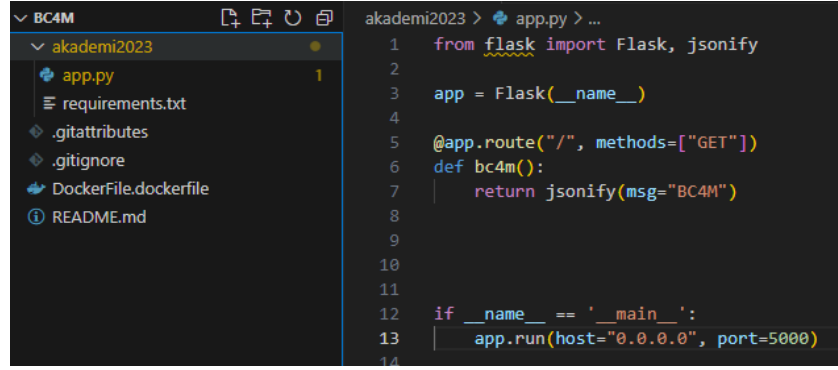


GitHub’da oluşturduğum Git reposuna örnek uygulama kodlarını yüklemek için browser üzerinden repo oluşturdum. İnternette araştırdığım sırada “Visual Studio Code” kurarak yazılı kodları da daha detaylı görerek çalışmayı amaçladım.

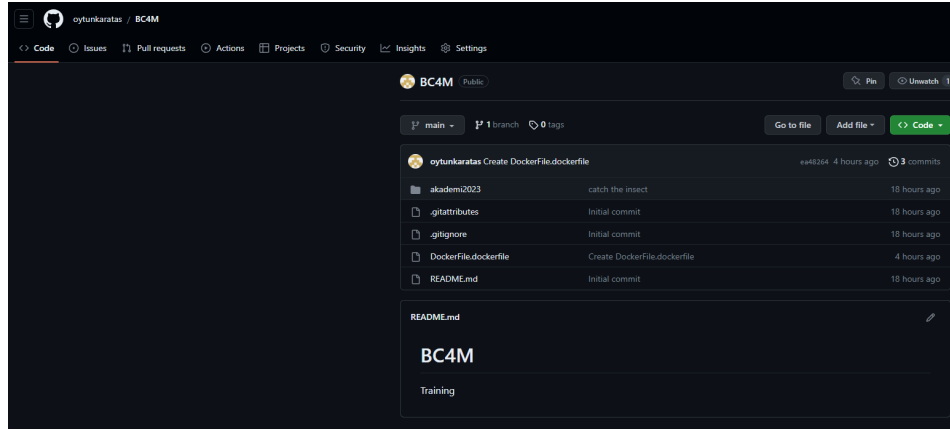


```
1 from flask import Flask, jsonify
2
3 app = Flask(__name__)
4
5 @app.route("/", methods=["GET"])
6 def bc4m():
7     return jsonify(msg="BC4M")
8
9
10
11
12 if __name__ == '__main__':
13     app.run(host="0.0.0.0", port=5000)
```

Resim-1

Bu sayede örnek kodları daha detaylı şekilde görebildim.

Daha sonra araştırmamı devam ettirirken GitHub uygulamasının daha verimli olacağını öğrenip bilgisayarına yükleyip kodları onun aracılığı ile oluşturduğum Git reposunun içine aktarma işlemine başladım.



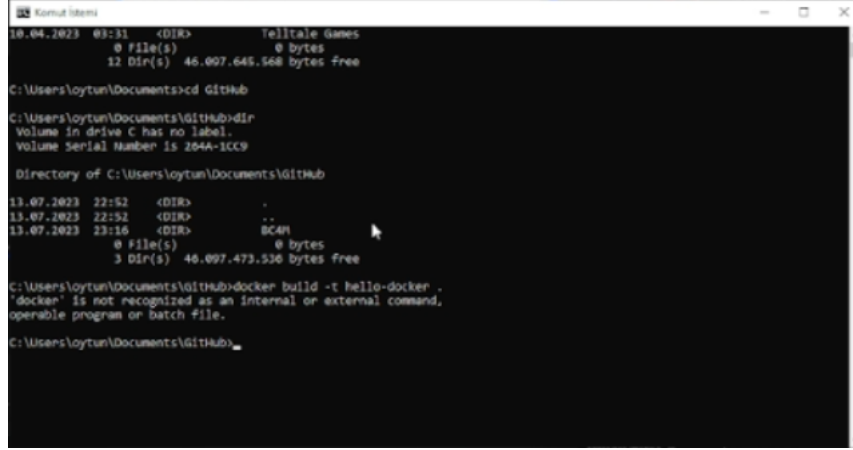
Resim-2

Örnek kodları oluşturduğum repoya yüklemek için ilk başta hata aldığım için farklı bir yol izleyerek “C:\Users\oytun\Documents\GitHub\BC4M” dizinin içine klasörü attıktan sonra GitHub Desktop uygulamasında direkt görebildim.

“Commit to main” demek için uygulama üzerinde yer alan “Summary” ve altında yer alan “Description” alanlarını doldurup main’e yolladım. Ardından adımları izleyip “Publish repository” diyerek GitHub.com’da da görünür hale getirdim. Repoyu ilk açtığım anda private olarak açılan halini de ayarlarından public hale getirdikten sonra “< > Code” sekmesi içindeki HTTPS linkini kopyalayarak kodu repoya import ettiğimi görmüş oldum.

Ve bu sayede görevlerden ilkinin tamamlanmış oldum.

Görevlerden ikincisi olan Dockerfile oluşturma işleminde, uygulamanın docker üzerinde çalışması için içinde komutların yazacağı uzantısız Docker'ı oluşturdum.



```
Komut İstemi
10.04.2023 09:31 <DIR> Telltale Games
0 File(s) 0 bytes
12 Dir(s) 46.007.645.568 bytes free

C:\Users\oytun\Documents>cd GitHub

C:\Users\oytun\Documents\GitHub>dir
Volume in drive C has no label.
Volume Serial Number is 204A-1CC9

Directory of C:\Users\oytun\Documents\GitHub

11.07.2023 22:52 <DIR> .
11.07.2023 22:52 <DIR> ..
11.07.2023 23:16 <DIR> BC4M
0 File(s) 0 bytes
3 Dir(s) 46.007.473.536 bytes free

C:\Users\oytun\Documents\GitHub>docker build -t hello-docker .
'docker' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\oytun\Documents\GitHub>
```

Resim-3

İlk denemelerimde command prompt'ta “docker build –t hello docker” örneğini kullanarak imajı görmeye çalıştım fakat başarılı olamadım. Dockerfile oluşturma esnasında içine eklediğim komutların ve Dockerfile'ın adının düzgün yazılması gerekliliğini göz önüne alarak çeşitli denemelerde bulundum.

```
# Temel olarak hangi görüntüden (image) başlamak istediğinizi belirtin
FROM base_image

# Gerekli bağımlılıkları kurun
RUN command

# Çalışma dizininizi belirleyin
WORKDIR /path/to/directory

# Gerekli dosyaları kopyalayın (local --> container)
COPY /path/to/source/path/to/destination

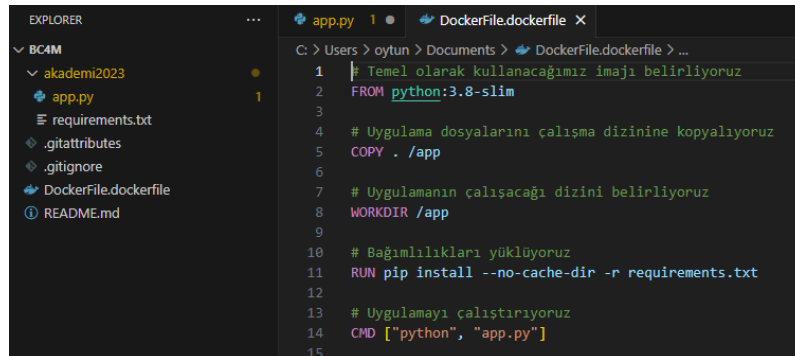
# Port numaralarını belirtin
EXPOSE port_number
```

Resim-4

Bulduğum çeşitli örnek ve entegre etmeye çalıştığım şekliyle Docker için elzem olan; “FROM, COPY, WORKDIR, RUN ve CMD” komutlarını düzenledim.

Başka bir MacOS işletim sistemli bir bilgisayarımdan “git clone <https://github.com/oytunkaratas/BC4M.git>” komutuyla kopyalama işlemini yapmaya çalıştım.

Tekrar GitHub Desktop uygulaması üzerinde Dockerfile.dockerfile uzantılı olarak görebildim ve ilk görevde yaptığım adımdaki gibi “commit to main” diyerek Git repoma import edebilmiş oldum.



```
EXPLORER
...
BC4M
  akademia2023
    app.py
    requirements.txt
    .gitattributes
    .gitignore
    DockerFile.dockerfile
    README.md

app.py
DockerFile.dockerfile X

C: > Users > oytun > Documents > DockerFile.dockerfile > ...
1 | # Temel olarak kullanacağımız imajı belirliyoruz
2 | FROM python:3.8-slim
3 |
4 | # Uygulama dosyalarını çalışma dizinine kopyalıyoruz
5 | COPY . /app
6 |
7 | # Uygulamanın çalışacağı dizini belirliyoruz
8 | WORKDIR /app
9 |
10 | # Bağımlılıkları yüklüyoruz
11 | RUN pip install --no-cache-dir -r requirements.txt
12 |
13 | # Uygulamayı çalıştırıyoruz
14 | CMD ["python", "app.py"]
15 |
```

Resim-5

Bu sayede ikinci görev olan Dockerfile dosyasını Git reposuna yükleme işlemini tamamladım.

Üçüncü görev olan “Docker Image oluşturulması ve Docker Image Registry’de saklanması” adımında Docker Image’ını daha rahat oluşturmayı görmek adına Docker Desktop uygulamasını indirip orayı inceledim.

Görevin ilk adımı olan DockerHub üzerinde bir image deposu açmak adına siteye giriş yaptıktan sonra repoyu oluşturmuş oldum. DockerDesktop üzerinde oluşturduğum Dockerfile ile bir docker image oluşturamadığım için DockerHub’da gördüğüm uluslararası çok kullanılan hazır image lar ile deneme yaptım. Bunlardan bazıları; helloworld, wordpress gibi herkesin daha aşına olduğu çok indirilen örnekler.

Command prompt’ta bunların görünürlüğüne test etmek adına “docker image command” yazarak gerekli komutları listeledim. Daha sonra bu genel, ünlü örnekleri “docker pull <isim>” kullanarak sistemime image ları indirdim. Daha sonra “docker image ls” yazarak sistemimde olan image ları listeyebildim.

```
Komut İstemi
Microsoft Windows [Version 10.0.19045.3208]
(c) Microsoft Corporation. Tüm hakları saklıdır.

C:\Users\oytun>docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
golang               1.19-alpine        f234544d6572       20 hours ago       354MB
wordpress            latest             043bd1522bd8       9 days ago         664MB
python               latest             c0e63845ae98       4 weeks ago        1.01GB
docker/extension-npm-installer latest             d99dccc73a01       5 weeks ago        195MB
hello-world          latest             9c7a54a9a43c       2 months ago       13.3kB
alpine/git            v2.34.2            5255164e42a7       14 months ago      38.2MB

C:\Users\oytun>
```

Resim-6

```
C:\Users\oytun>docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
719385e32844: Pull complete
Digest: sha256:926fac19d22aa2d60f1a276b66a20eb765fbeca2db5dbdaafeb456ad8ce81598
Status: Downloaded newer image for hello-world:latest
docker.io/library/hello-world:latest

What's Next?
  View summary of image vulnerabilities and recommendations + docker scout quickview hello-world

C:\Users\oytun>docker pull wordpress
Using default tag: latest
latest: Pulling from library/wordpress
9d21b12d5fab: Pull complete
1d2f438b3058: Pull complete
f04ed8a9ea49: Pull complete
10478cdf9e5da: Pull complete
81a7d8cd4ad8: Pull complete
0f62c4a728c4: Pull complete
49c7e2158665: Pull complete
4b4197cf218: Pull complete
3c74c30f4cf1: Pull complete
72993f5e6f4f: Pull complete
4b6baf92ab5f: Pull complete
4d48d4cf37c3: Pull complete
bb572f9b1334: Pull complete
f1c16814b1c4: Pull complete
9da0248a1537: Pull complete
7597c03f0937: Pull complete
1abb118899ad: Pull complete
19d448ef6f6d: Pull complete
8da944021c69: Pull complete
1d3c25173155: Pull complete
4b3f22984209: Pull complete
Digest: sha256:9bdcaf49c07ab8433b1de25761524948e57e89c039915d0372e9ab86bc64354c
Status: Downloaded newer image for wordpress:latest
docker.io/library/wordpress:latest

What's Next?
  View summary of image vulnerabilities and recommendations + docker scout quickview wordpress
```

Resim-7

Örneklerden biri; “docker pull hello world” komutuyla indirilen docker image ve katmanları.

Oluşturduğum dockerfile ı docker image olarak oluşturmaya çalışırken kullandığım –ve DockerHub üzerinden aldığım linki command prompt’a yazdıktan sonra” komut sonrası tekrar bir hata aldım ve bunu anlamaya çalıştım.

```
Komut İstemi

operable program or batch file.

C:\Users\oytun>cd Documents

C:\Users\oytun\Documents>docker build -t oytunkaratas/bc4m:stagname .
10/23/2018 13:17:41 https: server: error reading preface from client ../pipe/docker_engine: file has already been closed
[*] Building 0.2s (2/2) FINISHED                                docker:default
-> [internal] load .dockerignore                                0.1s
-> => transferring context: 2B                                    0.0s
-> [internal] load build definition from Dockerfile              0.0s
-> => transferring dockerfile: 2B                                0.0s
ERROR: failed to solve: failed to read dockerfile: open /var/lib/docker/tmp/buildkit-mount2273886466/Dockerfile: no such file or directory

C:\Users\oytun\Documents>docker push oytunkaratas/bc4m:stagname
The push refers to repository [docker.io/oytunkaratas/bc4m]
An image does not exist locally with the tag: oytunkaratas/bc4m

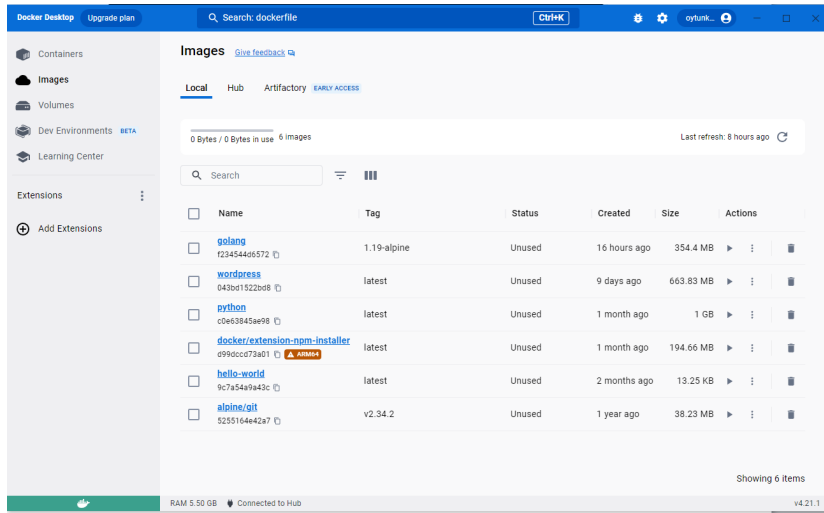
C:\Users\oytun\Documents>docker pull oytunkaratas/bc4m
Using default tag: latest
Error response from daemon: manifest for oytunkaratas/bc4m:latest not found: manifest unknown: manifest unknown

C:\Users\oytun\Documents>docker build -t oytunkaratas/bc4m:stagnam .
[*] Building 0.2s (2/2) FINISHED                                docker:default
-> [internal] load build definition from Dockerfile              0.0s
-> => transferring dockerfile: 2B                                    0.0s
-> [internal] load .dockerignore                                0.1s
-> => transferring context: 2B                                    0.0s
ERROR: failed to solve: failed to read dockerfile: open /var/lib/docker/tmp/buildkit-mount3329232819/Dockerfile: no such file or directory

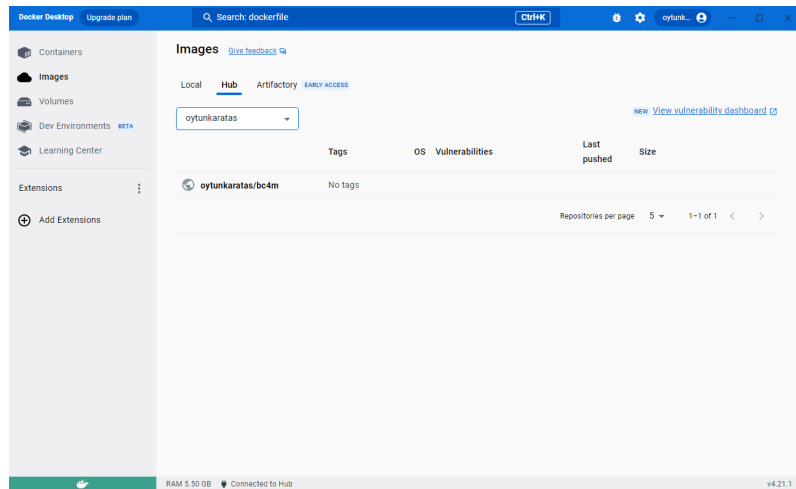
C:\Users\oytun\Documents>
```

Resim-8

Docker Desktop uygulamasında indirdiğim docker image ları görüntüleyebiliyor fakat Git reposunda da bulunan Dockerfile’ımı docker image’a çeviremediğim için halen uygulama üzerinde HUB sekmesi dışında farklı bir yerde göremedim.



Resim-9



Resim-10

Daha sonrasında internette benzer hataları almış kişilerden, forumlardan sorunun kökünü anlamaya çalıştım ve öncelikle Dockerfile dosyasının adı ve uzantısını tekrar doğru hale getirdikten emin olduktan sonra “akademi2023” klasörü içinde yollanan örnek kod olan “app.py” ve “requirements.txt” dosyasının da klasörde değil, direkt olarak Dockerfile dosyası ile aynı klasörde olması gerektiğini öğrenerek “akademi2023” klasörü dışına çıkardım.

Akabinde hazırladığım Dockerfile ile docker image oluşturmak için command prompt’ta kodların bulunduğu dizine gittim. Bunu yapmak için cd komutunu kullanıyoruz. Daha sonra “ docker build -t <uygulama adı> .” - yani isimlendirdiğim oytun-bc4m-test- komutunu kullanarak docker image’ı oluşturabildim.

```
Komut İstemi
Microsoft Windows [Version 10.0.19045.3208]
(c) Microsoft Corporation. Tüm hakları saklıdır.

C:\Users\oytun>cd documents

C:\Users\oytun\Documents>cd github

C:\Users\oytun\Documents\GitHub>cd bc4m

C:\Users\oytun\Documents\GitHub\BC4M>docker build -t oytun-bc4m-test .
[+] Building 12.8s (11/11) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 472B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/python:3.9
=> [auth] library/python:pull token for registry-1.docker.io
=> [1/5] FROM docker.io/library/python:3.9@sha256:ba10a2af9d6c3bd0d20c46ecbf866dabcbad4e6a3dd7b82e2dfb1a9bed479d87
=> => resolve docker.io/library/python:3.9@sha256:ba10a2af9d6c3bd0d20c46ecbf866dabcbad4e6a3dd7b82e2dfb1a9bed479d87
=> => sha256:ba10a2af9d6c3bd0d20c46ecbf866dabcbad4e6a3dd7b82e2dfb1a9bed479d87 1.86kB / 1.86kB
=> => sha256:46d99870b9c25e64c5583a59fec411df04191ab6b14cf81a/2b9e4a20a78b659 2.01kB / 2.01kB
=> => sha256:1d7821476b679b633b88afbb713dc4ac6b0857d1b4195fca1bc49d4400813e54 7.51kB / 7.51kB
=> => sha256:9421d62bef941e63ffebd2e6843ba2a8b8b91f5faf640d21f72450177f1ee824 15.82MB / 15.82MB
=> => sha256:67f28346abde952b467c778c7138e4883c52e680b599efea6e44575adb332f3f 2.85MB / 2.85MB
=> => sha256:10d7d75bfda3514a5f4aeb3bc75fb533830fb68086bad3260e60f4017805d48 243B / 243B
=> => extracting sha256:9421d62bef941e63ffebd2e6843ba2a8b8b91f5faf640d21f72450177f1ee824
=> => extracting sha256:19d7d75bfda3514a5f4aeb3bc75fb533830fb68086bad3260e60f4017805d48
=> => extracting sha256:67f28346abde952b467c778c7138e4883c52e680b599efea6e44575adb332f3f
=> [internal] load build context
=> => transferring context: 44.53kB
=> [2/5] WORKDIR /app
=> [3/5] COPY requirements.txt .
=> [4/5] RUN pip install --no-cache-dir -r requirements.txt
=> [5/5] COPY . .
=> => exporting to image
=> => exporting layers
=> => writing image sha256:0a37efb757783056f0c70050c1d5548b1c4e049c57654306ef4012809cb0529e
=> => naming to docker.io/library/oytun-bc4m-test

What's Next?
View summary of image vulnerabilities and recommendations -> docker scout quickview
```

Resim-11

Sonraki adım olan Docker Hub’da bir image reposu oluşturmak için “docker build -t oytunkaratas/oytun-bc4m-test” yazarak ilerleyebileceğimi de öğrendim. Bir başka seçenek; direkt browser üzerinden manuel olarak bu repoyu oluşturmak.

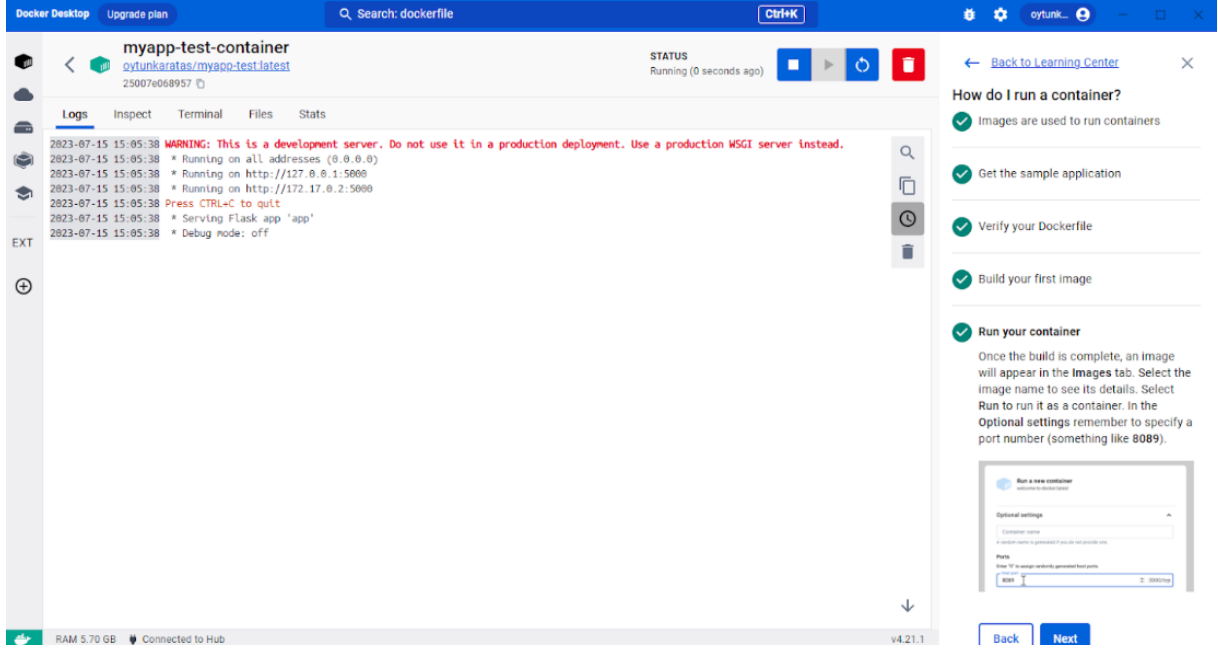
Elimde oluşturduğum docker image ı Docker Hub daki repoma yüklemek adına; “docker push oytunkaratas/oytun-bc4m-test” yani docker push komutu sonrası DockerHub’daki kullanıcı adım/repo adı .

```
C:\Users\oytun\Documents\GitHub\BC4M>docker push oytunkaratas/myapp-test
Using default tag: latest
The push refers to repository [docker.io/oytunkaratas/myapp-test]
c75f1e758254: Layer already exists
d6d5d1ec3ac8: Layer already exists
f307eebeb764: Layer already exists
957acc796e6d: Layer already exists
40b826ecd0e2: Layer already exists
97f54b85ff57: Layer already exists
52c358a45793: Layer already exists
f1acaab90728: Layer already exists
28218ecd8008: Layer already exists
2f66f3254105: Layer already exists
a72216901005: Layer already exists
61581d479298: Layer already exists
latest: digest: sha256:132302e596ba4e13eadf4afbde2074c4ccf9b41412b4fb57b314184de8f7f74d size: 2841
C:\Users\oytun\Documents\GitHub\BC4M>
```

Resim-12

Ayrıca local’de oluşturduğum Docker image ları rahatça listeleyebilsem de herhangi bir Docker image’ı bilgisayara yüklemek için bu sefer “docker pull kullanıcı adı/repo adı” komutunu kullanabileceğimi öğrendim.

Bu adımları da bitirdikten sonra üçüncü görevi de başarıyla yapabildim. Sırada son adım olan Docker Container’ın çalıştırılması adımı kaldı.



Resim-13

Bu adımda ilk olarak Docker Desktop uygulaması üzerinden ilerledim. Manuel olarak oluşturduğum Docker image’a run komutuyla çalıştırdım ve bu sonucu aldım. Command prompt üzerinden yapmak içinse internetteki gerek yerel, gerek yabancı kaynaklardan öğrendiğim kadarıyla “docker run -p <host-port>:<container-port> kullanıcı adı/repo adı” komutuyla yapabildiğimi öğrendim ve komutu kendi bilgilerimle düzenleyerek command prompt/terminal üzerinden de çalıştırdım.

Bu işlemi de yaptıktan sonra “<http://127.0.0.1:5000>” linki üzerinden mesajımı görüntüleyemedim ve bu sorunun ne olduğunu anlamaya çalışırken çokça kez yeniden Docker image oluşturup, ardında Docker Hub’a push layıp ardından docker run komutuyla çalıştırıp denemelerde bulundum.

Forumlarda benzer hataları alan insanların çözümlerini anlamaya çalıştım ve host port:container portlarının bilgisinin yanlışlığından olabileceğini, bir başka çözüm olabilecek öneri olan Flask uygulamasını Gunicorn ile çalıştırmanın yararlı olabileceğini yazanlar vardı. Bu yüzden Dockerfile dosyasında değişiklik yaparak bağımlılıkların altına “RUN pip install gunicorn” satırını ek olarak koyarak denemelerde bulundum.

Daha sonra aynı adımlar defalarca kez sıfırdan alarak yaptığım halde yine de sonucuma ulaşamıyordum.

Terminale “docker image ls” yazarak bilgisayardaki image’ları görebiliyor, Docker Container’larımı “docker run ...” komutuyla çalıştırdıktan sonra bilgisayarım üzerindeki çalışan Docker Container’ları görebilmek için “docker ps -a” komutunu kullanarak listeleyebiliyordum. Ek olarak; Docker Desktop uygulamasında görüntüleyebildiğim gibi, container loglarını terminal üzerinden görüntülemek için bir önceki kullandığım komut sonrası çıkan bilgilerden container ID’sini alıp “docker logs <container ID>” komutunda kullanarak listeleyebildiğimi öğrendim.

Almadığım sonuçlar neticesinde tüm adımları sil baştan yapmaya karar verdim ve oluşturduğum tüm Docker image'ları, image repolarını ve çalıştırdığım container'ların hepsini silerek işe koyuldum. Araştırmam sırasında işlemleri devam ettirirken Dockerfile dosyasında yaptığım değişikliğin uygulamamın çalışmasında sorun teşkil ettiği bilgisiyle verilen case'teki tüm görevleri adım adım bir daha yapmaya başladım.

```
Komut İstemi
Microsoft Windows [Version 10.0.19045.3208]
(c) Microsoft Corporation. Tüm hakları saklıdır.

C:\Users\oytun>docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
golang               1.19-alpine        f234544d6572       44 hours ago       354MB
wordpress            latest             043bd1522bd8       10 days ago        664MB
python               latest             c0e63845ae98       4 weeks ago        1.01GB
docker/extension-npm-installer latest             d99dccc73a01       5 weeks ago        195MB
hello-world          latest             9c7a54a9a43c       2 months ago       13.3kB
alpine/git            v2.34.2            5255164e42a7       14 months ago      38.2MB

C:\Users\oytun>docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES

C:\Users\oytun>cd documents
C:\Users\oytun\Documents>cd github
C:\Users\oytun\Documents\GitHub>cd bc4m
```

Resim-14

Silme işlemi sonrası docker image ls komutuyla bilgisayardaki image'ların görüntülenmesi.

```
C:\Users\oytun\Documents\GitHub\BC4M>docker build -t myapp-deneme:0.1 .
[+] Building 1.0s (10/10) FINISHED
=> [internal] load .dockerignore                                0.0s
=> => transferring context: 2B                                   0.0s
=> [internal] load build definition from Dockerfile              0.1s
=> => transferring dockerfile: 486B                               0.0s
=> [internal] load metadata for docker.io/library/python:3.9    0.7s
=> [1/5] FROM docker.io/library/python:3.9@sha256:ba10a2af9d6c3bd020c46ecbf866dabcbad4e6a3dd7b82e2dfb1a9b6d479d 0.0s
=> [internal] load build context                                0.1s
=> => transferring context: 6.60kB                                0.0s
< CACHED [2/5] WORKDIR /app                                     0.0s
=> CACHED [3/5] COPY requirements.txt .                          0.0s
=> CACHED [4/5] RUN pip install --no-cache-dir -r requirements.txt 0.0s
=> CACHED [5/5] COPY . .                                        0.0s
=> exporting to image                                           0.0s
=> => exporting layers                                           0.0s
=> => writing image sha256:0301bb84e901579345f2ce5eb28970c63dc58c50add371d83c38db202f6b52be 0.0s
=> => naming to docker.io/library/myapp-deneme:0.1             0.0s

What's Next?
View summary of image vulnerabilities and recommendations -> docker scout quickview
```

Resim-15

Dockerfile dosyasından yeni Docker image oluşturmak için “docker build -t myapp-deneme:0.1 .” komutunu kullanma.


```

C:\Users\oytun\Documents\GitHub\BC4M>docker run -d -p 5000:5000 myapp-deneme:0.1
d04d27f2bbf77691f98b720b2ad064282c248829d03173a440716ad037af1f97

C:\Users\oytun\Documents\GitHub\BC4M>docker ps -a
"docker ps" accepts no arguments.
See 'docker ps --help'.

Usage:  docker ps [OPTIONS]

List containers

C < sers\oytun\Documents\GitHub\BC4M>docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                    NAMES
d04d27f2bbf7   myapp-deneme:0.1  "python app.py"         26 seconds ago Up 26 seconds  0.0.0.0:5000->5000/tcp    jolly_kalam

C:\Users\oytun\Documents\GitHub\BC4M>docker logs d04d27f2bbf7
* Serving Flask app 'app'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://172.17.0.2:5000
Press CTRL+C to quit

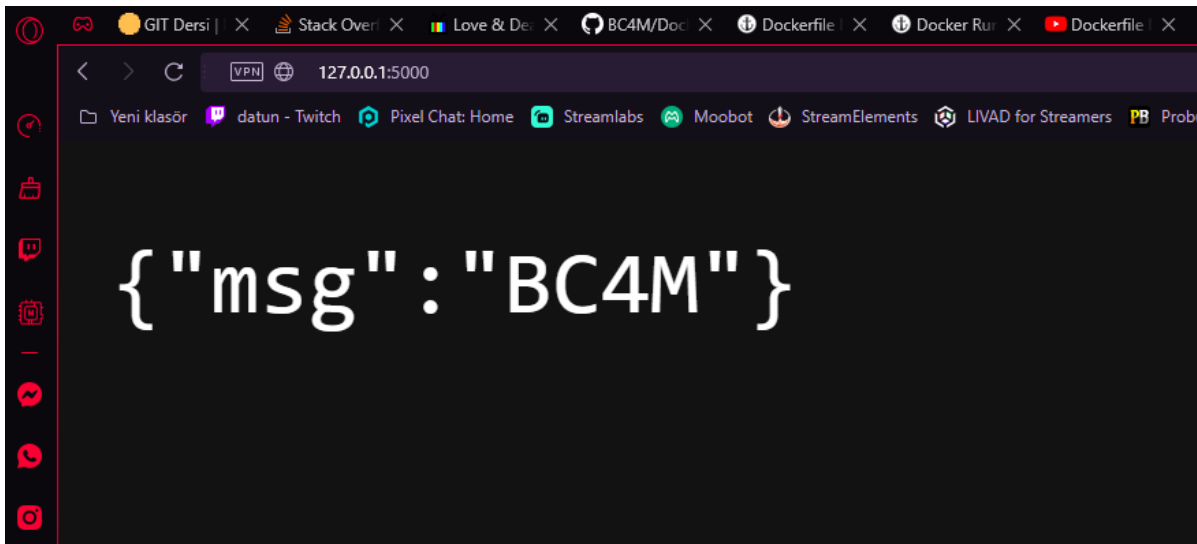
```

Resim-16

Docker image oluşturma işlemim sonrası “docker run” komutuyla bu sefer farklı olarak uygulamanın çalışabilirliğini görmek için farklı bir yol izledim. “docker run -d -p 5000:5000 myapp-deneme:0.1” diyerek docker container olarak uygulamayı çalıştırıp host port ve container port bilgisine “app.py” içinde belirtilen ve Dockerfile’da da “EXPOSE 5000” komutunu ekleyerek container’ın hangi portlardan erişilebileceğini belirttim.

Bu sefer “docker ps -a” komutuyla container’ları listelediğimde ports bilgisinin olduğunu görüntüledim ve uygulamanın çalıştığını görüntüleyebildim.

Çözüm için yollar ararken bir önceki yaptığımdan farklı olarak “docker run -d -p” komutunun amacının Docker Container’ı arka planda çalıştırmak ve container’a port yönlendirmesi yapmak olduğunu öğrendim. Ve bu sayede uygulamamı başarılı bir şekilde çalıştırabildim.



Resim-17

Başarılı şekilde çalışan uygulama ve görüntülenen mesaj.


```
C:\Users\oytun\Documents\GitHub\BC4D>docker run -d -p 5000:5000 myapp-official
4e8505ceb7dc29744526435d7285e427df3ab31b5d9c1a084aca4b31ceci1bb
docker: Error response from daemon: driver failed programming external connectivity on endpoint ecstatic_haslett (22485c43eb34f8622db7f773adeedf45c508650075bd0f88cb03e342c223020): Bind for 0.0.0.0:5000 failed: port is already allocated.

C:\Users\oytun\Documents\GitHub\BC4D>docker run -d -p 5000:5000 myapp-official:latest
c3e020474cee716d29cc0d0a7ada160ed59c912b480991c344960c1e295e6e
docker: Error response from daemon: driver failed programming external connectivity on endpoint busy_stonebraker (f8c678146b0b468ca9541ba2c8516e1c734cfa7c984d0cddc7384425dd2b5e43): Bind for 0.0.0.0:5000 failed: port is already allocated.

C:\Users\oytun\Documents\GitHub\BC4D>docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
2007c46a92c4       myapp-official     "python app.py"     About a minute ago   Created            5000/tcp           exciting_shirley
4e8505ceb7dc       myapp-official     "python app.py"     17 minutes ago      Created            5000/tcp           ecstatic_haslett
804d27f2bbf7       myapp-deneme:0.1   "python app.py"     46 minutes ago      Up 46 minutes      0.0.0.0:5000->5000/tcp   jolly_kalam
```

Resim-20

Bir önceki başarılı denememde kullandığım komut ile bir Docker Container olarak uygulamamı çalıştırdım (docker run -d -p 5000:5000 myapp-official) ve ardından container'ları command prompt'ta listeledim (docker ps -a) .

Burada yaptığım hatayı daha sonra farkettim ki hali hazırda 5000:5000 olarak yönlendirdiğim port değerlerini bir daha kullanmıştım, bu yüzden bu container'ın port bilgisi çıkmıyordu. Tekrar yaptığım araştırmalar sonucu <host port:container port> bilgisini 8080:8080 olacak şekilde değiştirerek bir daha “docker run -d -p 8080:8080 myapp-official” komutunu yineledim. Bu port bilgilerini kullanmamın sebebi; insanların daha genel olarak bu port ile işlem yaptığı bilgisini görmem oldu ve denemem sonrasında port bilgileri bu kez karşıma çıktı.

```
C:\Users\oytun\Documents\GitHub\BC4D>docker run -d -p 8080:8080 myapp-official
2007c46a92c473b4ab28eb17a195e2c84f56160a07a019fed6753432f644e308

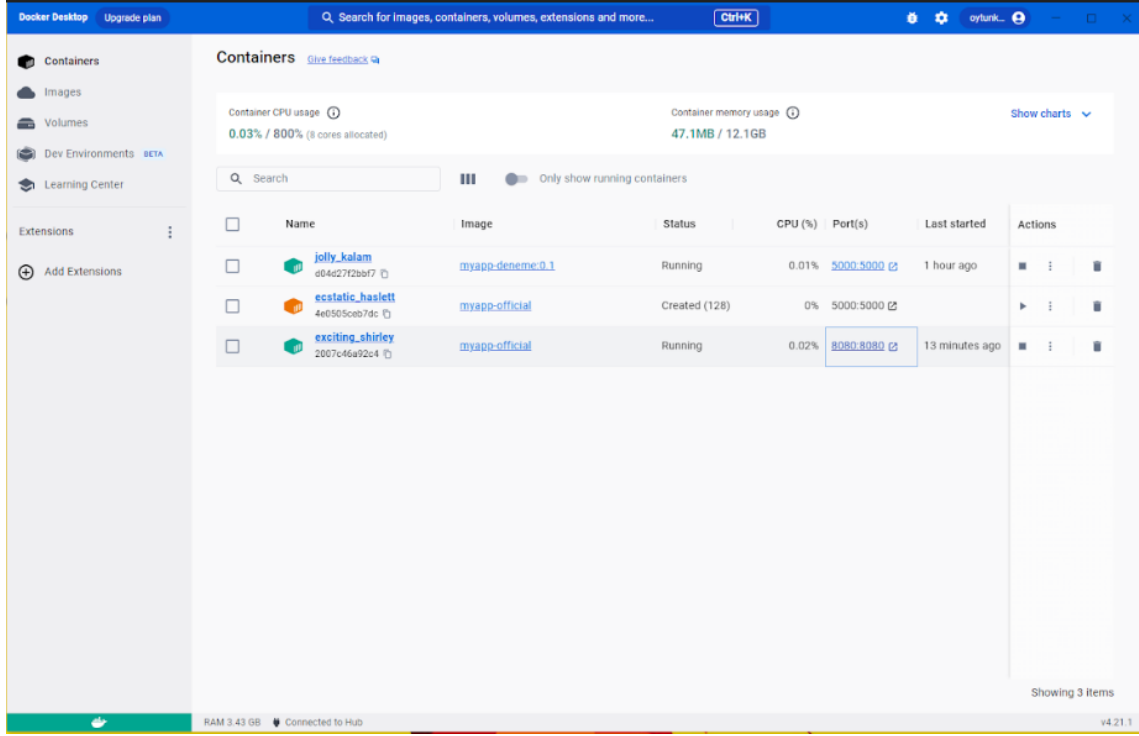
C:\Users\oytun\Documents\GitHub\BC4D>docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
2007c46a92c4       myapp-official     "python app.py"     19 seconds ago     Up 18 seconds      5000/tcp, 0.0.0.0:8080->8080/tcp   exciting_shirley
4e8505ceb7dc       myapp-official     "python app.py"     17 minutes ago     Created            5000/tcp           ecstatic_haslett
804d27f2bbf7       myapp-deneme:0.1   "python app.py"     About an hour ago   Up About an hour   0.0.0.0:5000->5000/tcp   jolly_kalam

C:\Users\oytun\Documents\GitHub\BC4D>docker logs 2007c46a92c4
* Serving Flask app 'app'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://172.17.0.3:5000
Press CTRL+C to quit

C:\Users\oytun\Documents\GitHub\BC4D>
```

Resim-21

Ardından bu container'ın loglarını listeledim (docker logs 2007c46a92c4) .



Resim-22

Docker Desktop uygulamasında Container'ların listesi.

Çalışmamın sonunda iki farklı yol izlemiş halde, bir Docker Container olarak uygulamamı çalıştırıp “{“msg”:“BC4M”}” mesajını başarılı bir şekilde görebildim.