

Cuprins

Terminal.....	1
Browser.....	3
Composer in browser.....	4

Din rezultatele căutării pe web, am găsit un tutorial de la Microsoft Learn care vă poate ajuta să începeți cu aplicațiile Docker în VS Code 1. Acest tutorial face parte dintr-o serie de trei părți care vă introduce în utilizarea Docker cu Visual Studio Code (VS Code). Veți învăța cum să creați și să rulați containere Docker, să persistați date și să implementați aplicația containerizată pe Azure. VS Code oferă o extensie care vă permite să lucrați cu un serviciu local Docker Desktop 1.

De asemenea, există o extensie Docker pentru Visual Studio Code care facilitează construirea, gestionarea și implementarea aplicațiilor containerizate în Visual Studio Code 2. Această extensie oferă IntelliSense atunci când editați fișierele Dockerfile și docker-compose.yml și vă permite să examinați și să gestionați activele Docker: containere, imagini, volume, rețele și registre de containere 2.

Sper că aceste informații vă vor ajuta să începeți cu utilizarea Docker în VS Code.

Puteți genera fișiere Docker în spațiul de lucru deschizând Command Palette (Ctrl+Shift+P) și utilizând comanda Docker: Add Docker Files to Workspace. Comanda va genera fișierele Dockerfile și .dockerignore și le va adăuga la spațiul de lucru. Comanda vă va întreba, de asemenea, dacă doriți să adăugați fișiere Docker Compose, dar acest lucru este opțional [webpage].

Terminal

Pentru a afișa mesajul “hello-world”, puteți edita fișierul Dockerfile generat și adăugați următoarele linii:

```
FROM ubuntu:latest
```

```
CMD echo "hello-world"
```

Aceasta va crea o imagine bazată pe cea mai recentă versiune Ubuntu și va rula comanda echo “hello-world” atunci când containerul este pornit.

Puteți utiliza următorul fișier docker-compose.yml pentru a afișa mesajul “hello world” în terminalul Ubuntu:

```
version: '3.8'
```

```
services:
```

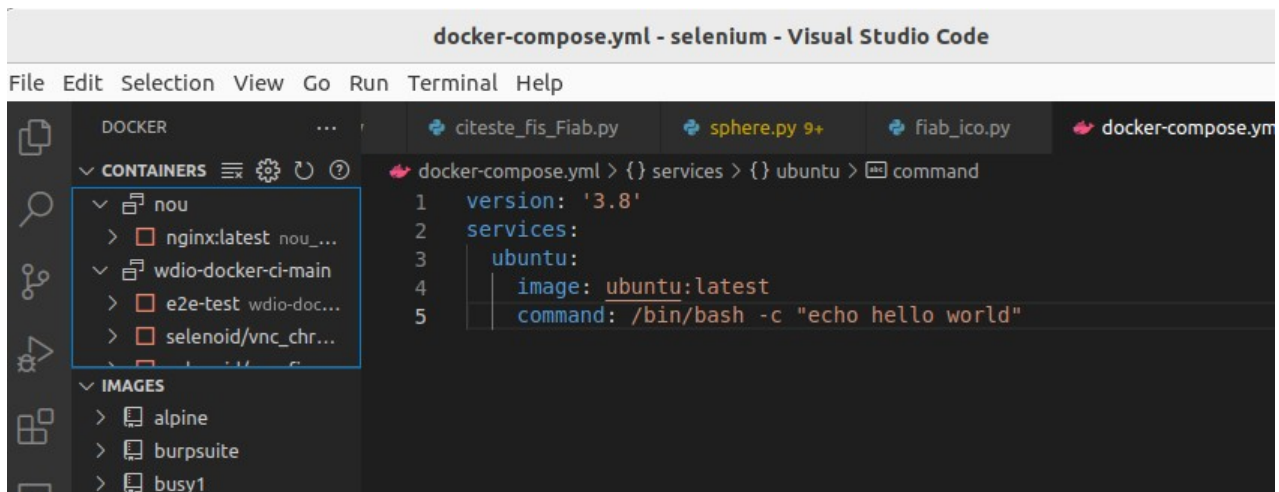
```
  ubuntu:
```

```
    image: ubuntu:latest
```

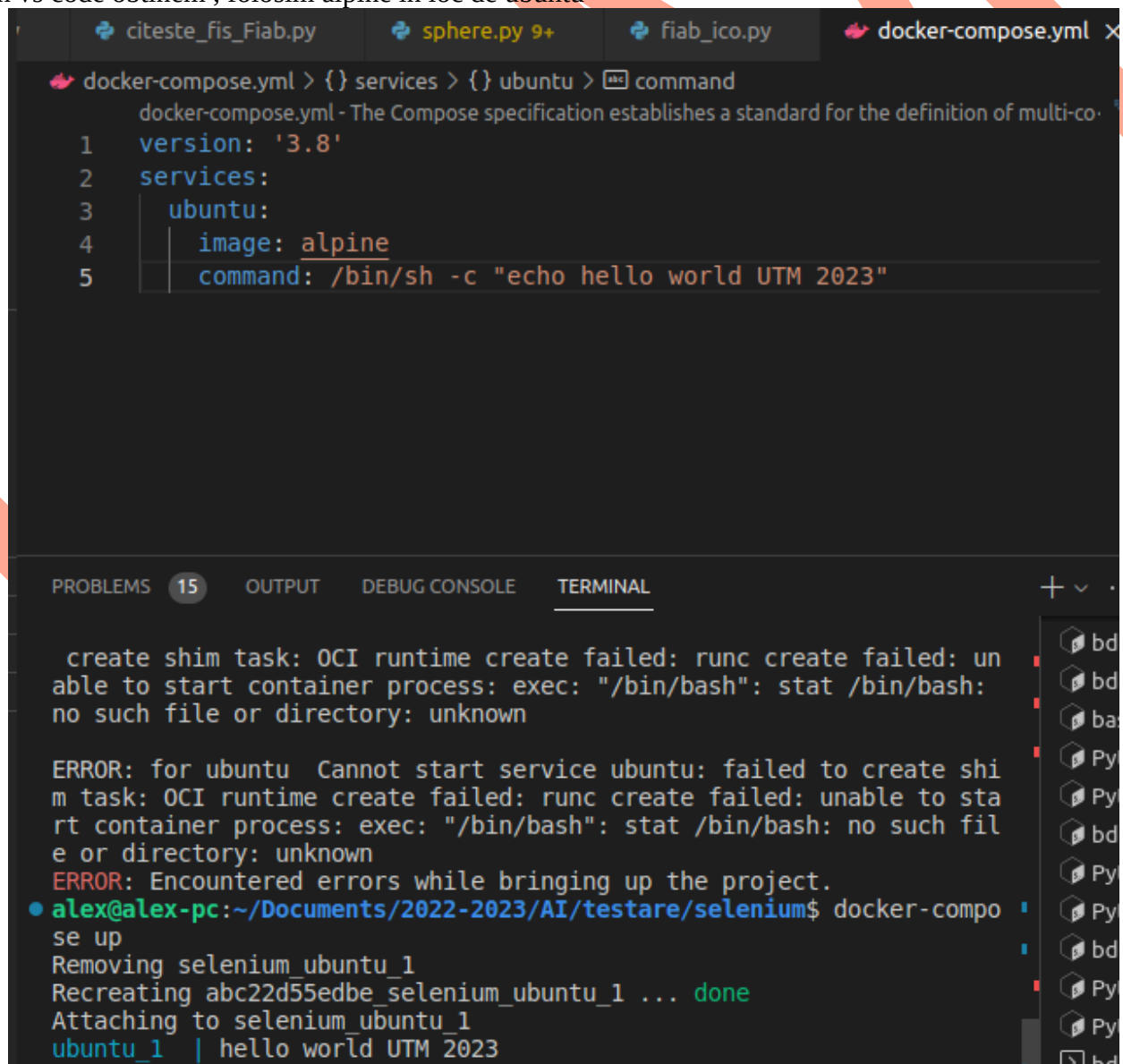
```
    command: /bin/bash -c "echo hello world"
```

Acest fișier definește un serviciu numit ubuntu care utilizează cea mai recentă imagine Ubuntu. Comanda specificată în secțiunea command va fi executată atunci când containerul este pornit și va afișa mesajul “hello world” în terminal.

Pentru a rula acest serviciu, salvați fișierul docker-compose.yml într-un director și navigați la acel director în terminal. Apoi, executați comanda docker-compose up pentru a porni serviciul și a afișa mesajul “hello world”.



in vs code obținem , folosim alpine in loc de ubuntu



Browser

Sent message.

Puteți utiliza următorul fișier docker-compose.yml pentru a afișa mesajul “hello world” într-un browser la portul 5000 folosind Flask:

```
version: '3.8'
services:
  web:
    build: .
    ports:
      - "5000:5000"
```

În același director cu fișierul docker-compose.yml, creați un fișier Dockerfile cu următorul conținut:

```
FROM python:3.8-slim-buster
```

```
WORKDIR /app
```

```
COPY requirements.txt requirements.txt
RUN pip install -r requirements.txt
```

```
COPY . .
```

```
CMD [ "python", "app.py" ]
```

Acest Dockerfile definește o imagine care se bazează pe imaginea Python 3.8 și instalează dependențele necesare pentru a rula o aplicație Flask.

De asemenea, va trebui să creați un fișier requirements.txt cu următorul conținut:

```
Flask==2.1.1
```

Acest fișier specifică dependențele necesare pentru a rula aplicația Flask.

În cele din urmă, va trebui să creați un fișier app.py cu următorul conținut:

```
from flask import Flask
```

```
app = Flask(__name__)
```

```
@app.route('/')
def hello_world():
    return 'Hello, World!'
```

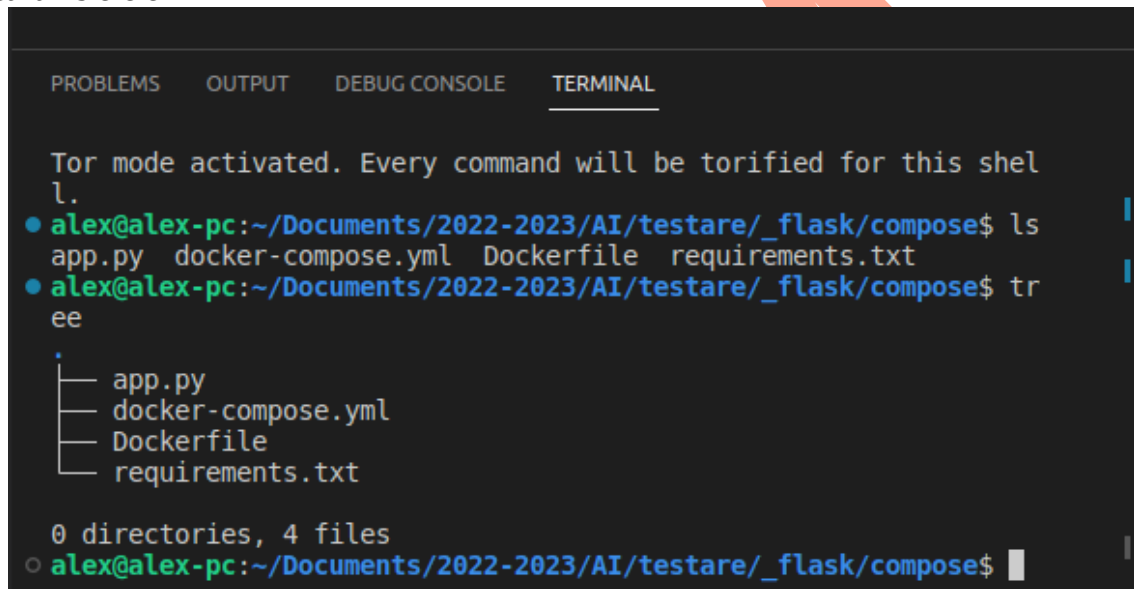
```
if __name__ == '__main__':
    app.run(host='0.0.0.0')
```

Acest fișier definește o aplicație Flask simplă care afișează mesajul “Hello, World!” atunci când este accesată ruta rădăcină.

Pentru a rula acest serviciu, salvați toate fișierele în același director și navigați la acel director în terminal. Apoi, executați comanda `docker-compose up --build` pentru a construi imaginea și a porni serviciul. Odată ce serviciul este pornit, puteți accesa `http://localhost:5000` în browser-ul dvs. pentru a vedea mesajul "Hello, World!".

Composer in browser

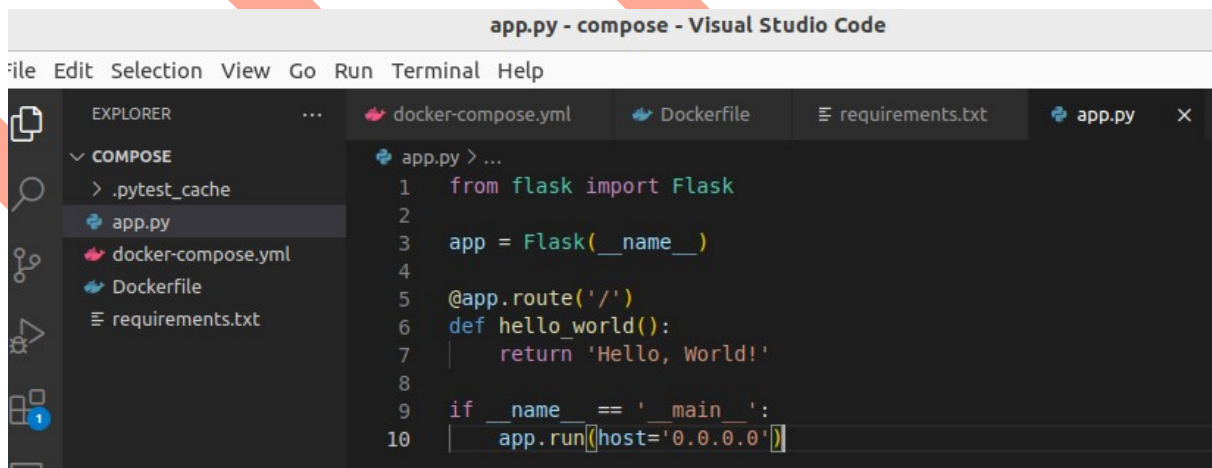
se creaza fisierele



```
Tor mode activated. Every command will be torified for this shell
alex@alex-pc:~/Documents/2022-2023/AI/testare/_flask/compose$ ls
app.py  docker-compose.yml  Dockerfile  requirements.txt
alex@alex-pc:~/Documents/2022-2023/AI/testare/_flask/compose$ tree
.
├── app.py
├── docker-compose.yml
├── Dockerfile
└── requirements.txt

0 directories, 4 files
alex@alex-pc:~/Documents/2022-2023/AI/testare/_flask/compose$
```

sau



```
app.py - compose - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
└─ COMPOSE
  └─ .pytest_cache
  └─ app.py
  └─ docker-compose.yml
  └─ Dockerfile
  └─ requirements.txt
app.py
1 from flask import Flask
2
3 app = Flask(__name__)
4
5 @app.route('/')
6 def hello_world():
7     return 'Hello, World!'
8
9 if __name__ == '__main__':
10     app.run(host='0.0.0.0')
```

fișierul **docker-compose.yml**

```
version: '3.8'
services:
  web:
    build: .
    ports:
      - "5000:5000"
```

fisierul **Dockerfile**

```
FROM python:3.8-slim-buster

WORKDIR /app

COPY requirements.txt requirements.txt
RUN pip install -r requirements.txt

COPY . .

CMD [ "python", "app.py" ]
```

fisierul **requirements.txt**

```
Flask==2.1.1
```

fisierul app.py

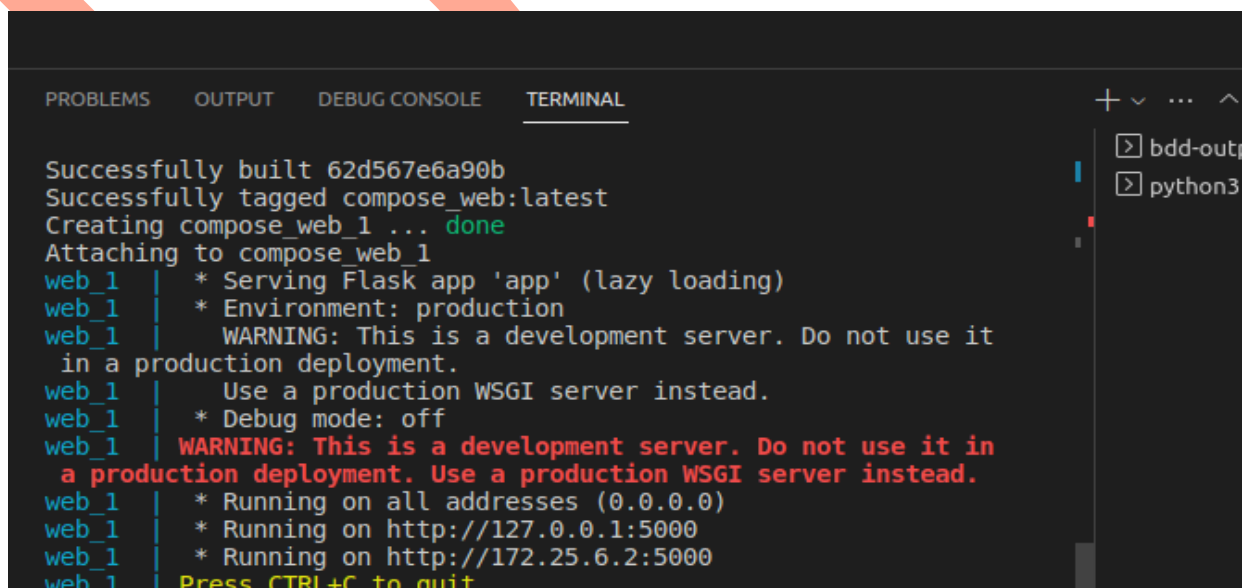
```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, World!'

if __name__ == '__main__':
    app.run(host='0.0.0.0')
```

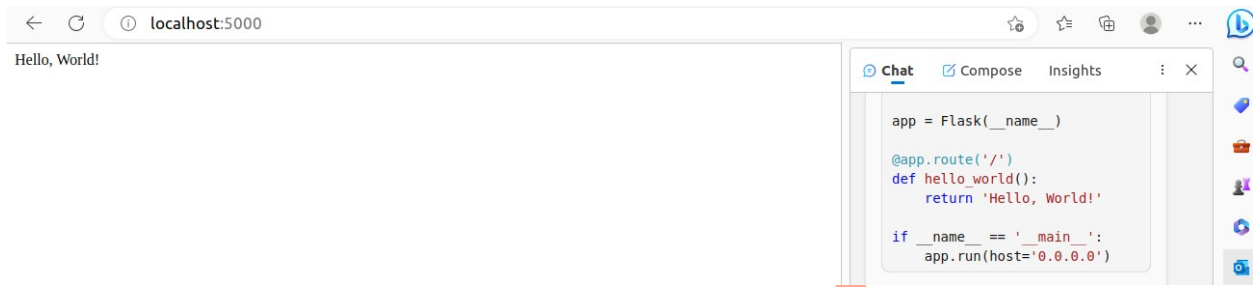
docker-compose up -build



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

Successfully built 62d567e6a90b
Successfully tagged compose_web:latest
Creating compose_web_1 ... done
Attaching to compose_web_1
web_1 | * Serving Flask app 'app' (lazy loading)
web_1 | * Environment: production
web_1 | WARNING: This is a development server. Do not use it
web_1 | in a production deployment.
web_1 | Use a production WSGI server instead.
web_1 | * Debug mode: off
web_1 | WARNING: This is a development server. Do not use it in
web_1 | a production deployment. Use a production WSGI server instead.
web_1 | * Running on all addresses (0.0.0.0)
web_1 | * Running on http://127.0.0.1:5000
web_1 | * Running on http://172.25.6.2:5000
web_1 | Press CTRL+C to quit
```

si in browser port 5000:5000



UML