باسمه تعالی

درس: مبانی رایانش ابری

نام و نام خانوادگی: تینا توکلی ،پارسا هدایت نیا، محمد هادی امینی

شماره دانشجویی: 9912762370،9822762039،9922762220

نام تمرین : تمرین عملی داکر

نصب wsl:

دستورات به ترتیب در عکس مشخص است.

```
PS C:\Windows\system32> wsl --update
Installing: Windows Subsystem for Linux
Catastrophic failure
PS C:\Windows\system32> wsl --update
Installing: Windows Subsystem for Linux
Windows Subsystem for Linux has been installed.
PS C:\Windows\system32> wsl --list -online
The parameter is incorrect.
Error code: Wsl/E_INVALIDARG
PS C:\Windows\system32> wsl --list --online
The following is a list of valid distributions that can be installed.
Install using 'wsl.exe --install <Distro>'.

NAME                                   FRIENDLY NAME
Ubuntu                                 Ubuntu
Debian                                 Debian GNU/Linux
kali-linux                             Kali Linux Rolling
Ubuntu-18.04                           Ubuntu 18.04 LTS
Ubuntu-20.04                           Ubuntu 20.04 LTS
Ubuntu-22.04                           Ubuntu 22.04 LTS
Ubuntu-24.04                           Ubuntu 24.04 LTS
OracleLinux_7_9                        Oracle Linux 7.9
OracleLinux_8_7                        Oracle Linux 8.7
OracleLinux_9_1                        Oracle Linux 9.1
openSUSE-Leap-15.5                     openSUSE Leap 15.5
SUSE-Linux-Enterprise-Server-15-SP4    SUSE Linux Enterprise Server 15 SP4
SUSE-Linux-Enterprise-15-SP5           SUSE Linux Enterprise 15 SP5
openSUSE-Tumbleweed                    openSUSE Tumbleweed
```

تصویر 1-نصب wsl

```
openSUSE-Tumbleweed                    openSUSE Tumbleweed
PS C:\Windows\system32> wsl --install -d Debian
Installing Windows optional component: VirtualMachinePlatform

Deployment Image Servicing and Management tool
Version: 10.0.19041.3636

Image Version: 10.0.19045.4412

Enabling feature(s)
[==========================100.0%==========================]
The operation completed successfully.
Installing: Debian GNU/Linux
Debian GNU/Linux has been installed.
The requested operation is successful. Changes will not be effective until the system is rebooted.
```

تصویر 2-نصب Debian

```
PS C:\Windows\system32> wsl -l -v
  NAME      STATE           VERSION
* Debian    Stopped         1
PS C:\Windows\system32> wsl --set-version Debian 2
For information on key differences with WSL 2 please visit https://aka.ms/wsl2
Conversion in progress, this may take a few minutes.
The operation completed successfully.
PS C:\Windows\system32>
PS C:\Windows\system32> wsl -l -v
  NAME      STATE           VERSION
* Debian    Stopped         2
PS C:\Windows\system32>
```

تصویر 3تغییر version wsl

به منظور نصب docker از لینک زیر استفاده شده است:

[Install Docker Engine on Debian | Docker Docs]

```
Setting up docker-compose-plugin (2.27.1-1~debian.12~bookworm) ...
Setting up libltdl7:amd64 (2.4.7-5) ...
Setting up docker-ce-cli (5:26.1.4-1~debian.12~bookworm) ...
Setting up libslirp0:amd64 (4.7.0-1) ...
Setting up pigz (2.6-1) ...
Setting up libnfnetlink0:amd64 (1.0.2-2) ...
Setting up dbus-session-bus-common (1.14.10-1~deb12u1) ...
Setting up git-man (1:2.39.2-1.1) ...
Setting up libx11-6:amd64 (2:1.8.4-2+deb12u2) ...
Setting up dbus-system-bus-common (1.14.10-1~deb12u1) ...
Setting up libfido2-1:amd64 (1.12.0-2+b1) ...
Setting up libxml2:amd64 (2.9.14+dfsg-1.3~deb12u1) ...
Setting up libxmuu1:amd64 (2:1.1.3-3) ...
Setting up dbus-bin (1.14.10-1~deb12u1) ...
Setting up libgdbm6:amd64 (1.23-3) ...
Setting up slirp4netns (1.2.0-1) ...
Setting up openssh-client (1:9.2p1-2+deb12u2) ...
Setting up libxext6:amd64 (2:1.3.4-1+b1) ...
Setting up dbus-daemon (1.14.10-1~deb12u1) ...
Setting up dbus (1.14.10-1~deb12u1) ...
invoke-rc.d: could not determine current runlevel
Setting up shared-mime-info (2.2-1) ...
Setting up libgdbm-compat4:amd64 (1.23-3) ...
Setting up xauth (1:1.1.2-1) ...
Setting up libnetfilter-conntrack3:amd64 (1.0.9-3) ...
Setting up libpam-systemd:amd64 (252.22-1~deb12u1) ...
Setting up libperl5.36:amd64 (5.36.0-7+deb12u1) ...
Setting up iptables (1.8.9-2) ...
update-alternatives: using /usr/sbin/iptables-legacy to provide /usr/sbin/iptables (iptables) in auto mode
update-alternatives: using /usr/sbin/ip6tables-legacy to provide /usr/sbin/ip6tables (ip6tables) in auto mode
update-alternatives: using /usr/sbin/iptables-nft to provide /usr/sbin/iptables (iptables) in auto mode
update-alternatives: using /usr/sbin/ip6tables-nft to provide /usr/sbin/ip6tables (ip6tables) in auto mode
update-alternatives: using /usr/sbin/arptables-nft to provide /usr/sbin/arptables (arptables) in auto mode
update-alternatives: using /usr/sbin/ebtables-nft to provide /usr/sbin/ebtables (ebtables) in auto mode
Setting up dbus-user-session (1.14.10-1~deb12u1) ...
Setting up perl (5.36.0-7+deb12u1) ...
Setting up docker-ce (5:26.1.4-1~debian.12~bookworm) ...
invoke-rc.d: could not determine current runlevel
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /lib/systemd/system/docker.socket.
Setting up docker-ce-rootless-extras (5:26.1.4-1~debian.12~bookworm) ...
Setting up liberror-perl (0.17029-2) ...
Setting up git (1:2.39.2-1.1) ...
Processing triggers for libc-bin (2.36-9+deb12u4) ...
tina@DESKTOP-7MQQ1G3:/mnt/c/Windows/system32$
```

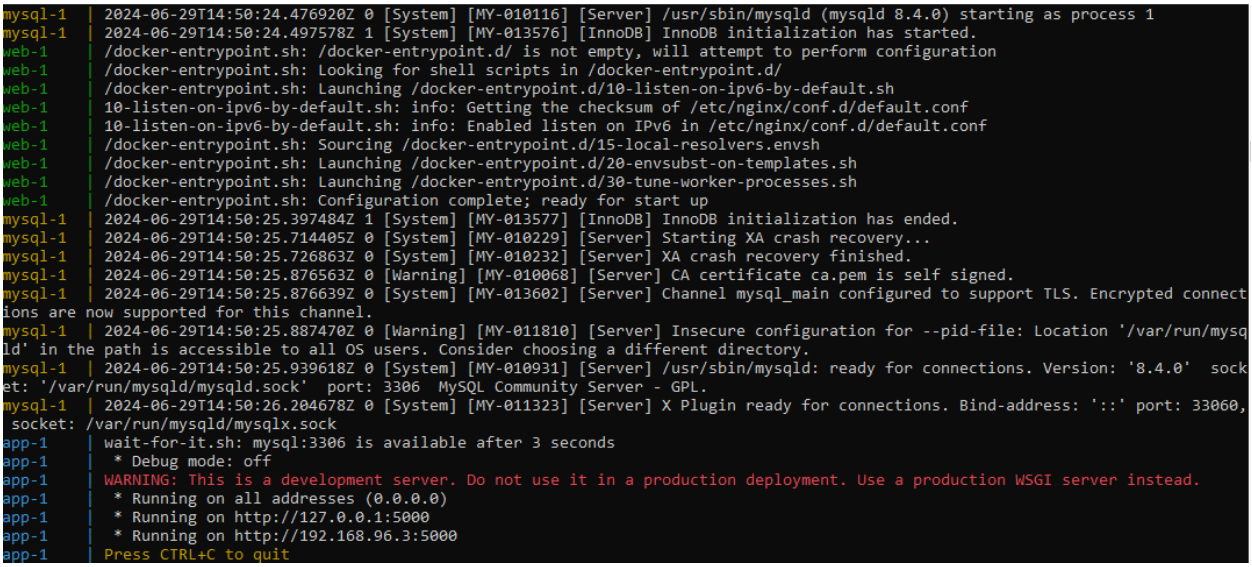
تصویر 4 -مراحل پایانی نصب docker

برای ران کردن کانتینرها در داکر از دستورات زیر استفاده شده است:

sudo service docker start

sudo docker login --username tina1381

sudo docker compose up --build



به منظور اجرای مجدد برنامه بعد از هربار ریستارت شدن :

- ساخت فایل nano ~/start-docker-compose.sh
  - محتوای فایل:
    - #!/bash/bin
    - cd /mnt/d/code/other/cloud-exercise
    - sudo docker compose up -d
- crontab -e
- @reboot /home/hadi/start-docker-compose.sh
- sudo service cron start

فایل docker-compose.yml:

سرویس‌ها:

- بخش `services` برنامه‌های جداگانه‌ای را که برنامه چند کانتینری را تشکیل می‌دهند، تعریف می‌کند. هر سرویس با تنظیمات خاص خود پیکربندی می‌شود. ما سه سرویس داریم: `web`، `app` و `mysql`.

سرویس web:

- `web:`
  - این خط بلوک پیکربندی برای سرویس `web` را شروع می‌کند.
- `build: ./nginx`

- این خط به Docker Compose دستور می‌دهد تا یک تصویر Docker برای سرویس `web` از Dockerfile واقع در دایرکتوری `nginx/.` در پروژه بسازد. Dockerfile مراحل ایجاد image، از جمله image پایه و هر پیکربندی اضافی را تعریف می‌کند.

- `image: tina1381/nginx-web:latest`
  - این خط به ما امکان می‌دهد تا یک `image` از پیش ساخته شده را از یک رجیستری Docker (مانند Docker Hub) دریافت کنیم.
- `ports:`
  - `"80:80"`
  - این خط پورت 80 (پورت HTTP پیش‌فرض) container را به پورت 80 روی دستگاه میزبان map می‌کند. این به ما امکان می‌دهد با بازدید از `http://localhost:80` در مرورگر وب خود به برنامه وب که در container اجرا می‌شود دسترسی پیدا کنیم.
- `depends_on:`
  - `app`
  - این خط یک وابستگی برای سرویس `web` تعریف می‌کند. مشخص می‌کند که سرویس `web` نمی‌تواند تا زمانی که سرویس `app` راه‌اندازی و اجرا شود، شروع شود.

```
deploy:
      resources:
        limits:
          cpus: '0.50'
          memory: '256M'
        reservations:
          cpus: '0.25'
          memory: '128M'
```

  - این بخش اختصاص منابع را برای container سرویس `web` در Docker کنترل می‌کند. دو مجموعه از مقادیر را تعریف می‌کند:
    - `limits`: این حداکثر میزان منابع CPU و حافظه را که container می‌تواند استفاده کند، تعیین می‌کند. در اینجا، به 0.5 CPU و 256 مگابایت حافظه محدود می‌شود.
    - `reservations`: این حداقل میزان تضمین شده منابع CPU و حافظه را که container دریافت می‌کند، تعیین می‌کند. در اینجا، 0.25 CPU و 128 مگابایت حافظه رزرو شده است.

سرویس app:

- `build: ./flask`
- به Docker Compose دستور می‌دهد تا `image` را برای سرویس `app` از Dockerfile واقع در دایرکتوری `flask/.` بسازد.
- `volumes:`
- data/:data/.

- این خط یک volume به نام data را تعریف می‌کند که دایرکتوری ./data را در host machine به دایرکتوری /data در داخل container متصل می‌کند. این محل ذخیره مناسبی برای داده‌های برنامه (مانند فایل‌های پایگاه داده یا فایل‌های آپلود شده) است که باید بین اجراهای container حفظ شود.

- ./wait-for-it.sh:/wait-for-it.sh -
  - این خط یک volume به نام wait-for-it.sh را تعریف می‌کند که اسکریپت ./wait-for-it.sh را از host machine به مسیر /wait-for-it.sh در داخل container متصل می‌کند. این اسکریپت برای صبر کردن تا وابستگی‌هایی مانند پایگاه داده MySQL در دسترس باشند، استفاده می‌شود.
- ./static:/app/static -
  - این خط یک volume به نام static را تعریف می‌کند که دایرکتوری ./static را در host machine به مسیر /app/static در داخل container متصل می‌کند. این مسیر معمولاً برای فایل‌های استاتیک مانند `image` ، CSS و جاوا اسکریپت استفاده می‌شود که توسط برنامه ما استفاده می‌شوند.
- entrypoint:
  - ["wait-for-it.sh", "mysql:3306", "--", "flask", "run", "--host=0.0.0.0", "--/."
    "port=5000"]:
  - این خط دستوری را که هنگام راه اندازی container سرویس app اجرا می‌شود، مشخص می‌کند.
  - wait-for-it.sh mysql:3306/.: این قسمت از دستور از اسکریپت wait-for-it.sh برای صبر کردن تا پایگاه داده MySQL قبل از راه اندازی برنامه Flask در دسترس باشد، استفاده می‌کند.
  - flask run --host=0.0.0.0 --port=5000: این قسمت از دستور فرمان flask run را برای راه اندازی برنامه Flask اجرا می‌کند و مشخص می‌کند که باید به آدرس 0.0.0.0 و پورت 5000 گوش دهد.
- environment:
  - FLASK_ENV=development -:
  - این خط متغیر محیطی FLASK_ENV را با مقدار development تنظیم می‌کند. به برنامه Flask می‌گوید که در حالت توسعه اجرا شود، ممکن است پیکربندی‌های خاصی را برای توسعه اعمال کند.
  - -
    DATABASE_URL=mysql+pymysql://tina:${MYSQL_PASSWORD}@mysql/my
    database:
  - این خط متغیر محیطی DATABASE_URL را با رشته اتصال به پایگاه داده MySQL تنظیم می‌کند.

- depends_on`:
- mysql`
  - مشابه سرویس web، این خط وابستگی به سرویس mysql`

- `Mysql
- `image: tina1381/mysql:latest`
  - این خط به Docker Compose می‌گوید که از یک `image` Docker از پیش ساخته شده به نام `tina1381/mysql` با آخرین تگ برای اجرای سرویس MySQL استفاده کند.
- `restart: always`
  - این خط خط مشی راه اندازی مجدد را برای سرویس `mysql` تعیین می‌کند. با مقدار `always`، container MySQL حتی پس از اینکه به طور ناگهانی متوقف شود یا کرش کند، به طور خودکار دوباره راه اندازی می‌شود. این برای اطمینان از در دسترس بودن پایگاه داده برای برنامه‌های دیگر (مانند سرویس `app`) که به آن وابسته هستند، مهم است.

```
environment:
      MYSQL_ROOT_PASSWORD: tinatvk81
      MYSQL_DATABASE: mydatabase
      MYSQL_USER: tina
      MYSQL_PASSWORD: tinatvk81
```

  - این خط متغیرهای محیطی را برای container `mysql` تعریف می‌کند:
    - `MYSQL_ROOT_PASSWORD`: رمز عبور کاربر root MySQL را برای ورود به پایگاه داده تنظیم می‌کند.
    - `MYSQL_DATABASE`: نام پایگاه داده‌ای که باید ایجاد شود را مشخص می‌کند.
    - `MYSQL_USER`: نام کاربری را برای ایجاد در پایگاه داده MySQL با سطح دسترسی مشخص شده در `MYSQL_USER` تنظیم می‌کند.
    - `MYSQL_PASSWORD`: رمز عبور کاربر MySQL را با نام کاربری `MYSQL_USER` تعیین می‌کند.
- `volumes:
  - var/lib/mysql/:mysql-data
  - `sql.d/init_db.docker-entrypoint-initdb/:sql.init_db/.
  - این خط volumeهایی را تعریف می‌کند که داده‌ها را برای پایگاه داده MySQL بین container و دستگاه میزبان حفظ می‌کنند:
    - اولین volume دایرکتوری `mysql-data` را در دستگاه میزبان به دایرکتوری `var/lib/mysql/` در داخل container متصل می‌کند. این محل ذخیره سازی پیش فرض داده های پایگاه داده MySQL است.
    - دومین volume اسکریپت `init_db.sql/.` را از دستگاه میزبان به مسیر `docker-/entrypoint-initdb.d/init_db.sql` در داخل container متصل می‌کند. این اسکریپت SQL در هنگام راه اندازی اولیه container برای ایجاد پایگاه داده `mydb`، ایجاد کاربر `tina` و اعطای مجوزهای لازم به آن، و اجرای هرگونه دستورات SQL اضافی که در اسکریپت موجود است، اجرا می‌شود.

```
deploy:
```

```
    resources:
      limits:
        cpus: '0.50'
        memory: '512M'
      reservations:
        cpus: '0.25'
        memory: '256M'
```

- این بخش اختصاص منابع را برای container سرویس `mysql` در Docker کنترل می‌کند. مشابه با سرویس‌های `web` و `app`، دو مجموعه از مقادیر را برای CPU و حافظه تعریف می‌کند: `limits` و `reservations`.

volumeها:

- `mysql-data:`
  - این خط یک volume به نام `mysql-data` را تعریف می‌کند. این volume هیچ دایرکتوری پیش‌فرضی در host machine ندارد، زیرا فقط توسط containerهای `mysql` و `web` از طریق mount points مشخص شده در پیکربندی‌های سرویس آنها قابل دسترسی است.

توضیحات Nginx.conf:

بخش server { }:

- `server { ... }`
  - این بلوک یک سرور مجازی را در Nginx تعریف می کند. یک سرور مجازی مجموعه‌ای از تنظیمات است که نحوه مدیریت درخواست‌های HTTP را برای یک دامنه یا مسیر خاص تعیین می‌کند.

listen 80;

- این خط به Nginx می‌گوید که روی پورت 80 (پورت پیش‌فرض HTTP) گوش دهد.

Serve static files:

- location / { ... }
  - این بلوک نحوه مدیریت درخواست هایی که به root(/) ارسال می شوند را تعریف می کند.
- root /usr/share/nginx/html;
  - این خط به Nginx می‌گوید که برای یافتن فایل‌های staticبه دایرکتوری /usr/share/nginx/html در host machine مراجعه کند. این دایرکتوری محل پیش‌فرض برای فایل‌های استاتیک Nginx است.
- index index.html;
  - این خط به Nginx می‌گوید که به دنبال فایل index.html در دایرکتوری root (usr/share/nginx/html/) بگردد و آن را به عنوان صفحه پیش‌فرض نمایش دهد.

Proxy requests to the Flask application:

- location /app/ { ... }

- این بلوک نحوه مدیریت درخواست هایی که به مسیر /app/ (فرض بر این است که این مسیر پیشوند URL برنامه Flask شما است) ارسال می شوند را تعریف می کند.

- rewrite ^/app/(.*)$ /$1 break;
  - این خط برای اطمینان از اینکه URL های درخواستی به درستی به برنامه Flask منتقل می شوند، مسیر را بازنویسی میکند.
  - الگوی ^/app/(.*)$ مطابقت هر درخواستی را که با /app/ شروع می شود، با گروهی به نام $1 که باقیمانده مسیر است، انجام می دهد.
  - $1 سپس به مسیر اصلی درخواست در دستور proxy_pass بعدی منتقل می شود.
  - کلمه کلیدی break به Nginx می گوید که دیگر قوانین location را برای این درخواست بررسی نکند.
- proxy_pass http://app:5000/;
  - این خط به Nginx دستور می دهد تا هر درخواستی که به مسیر /app/ ارسال می شود را به برنامه Flask در حال اجرا در container با نام app روی پورت 5000 منتقل کند .
- خطوط بعدی (proxy_set_header ...) برای تنظیم هدرهای خاصی در درخواست های منتقل شده استفاده می شود تا برنامه Flask اطلاعات صحیحی درباره درخواست اصلی دریافت کند.

: Proxy static files to the Flask application

- location /static/ { ... }
  - این بلوک نحوه مدیریت درخواست هایی که به مسیر /static/ (مسیر پیش فرض برای فایل های استاتیک برنامه Flask) ارسال می شوند را تعریف می کند.
- خطوط بعدی (proxy_pass ... و proxy_set_header ...) مشابه بخش /app/ عمل می کنند و درخواست ها را به برنامه Flask در app container با تنظیم هدرهای مناسب برای اطمینان از عملکرد صحیح برنامه ارسال می کنند.

لینک image های ساخته شده:

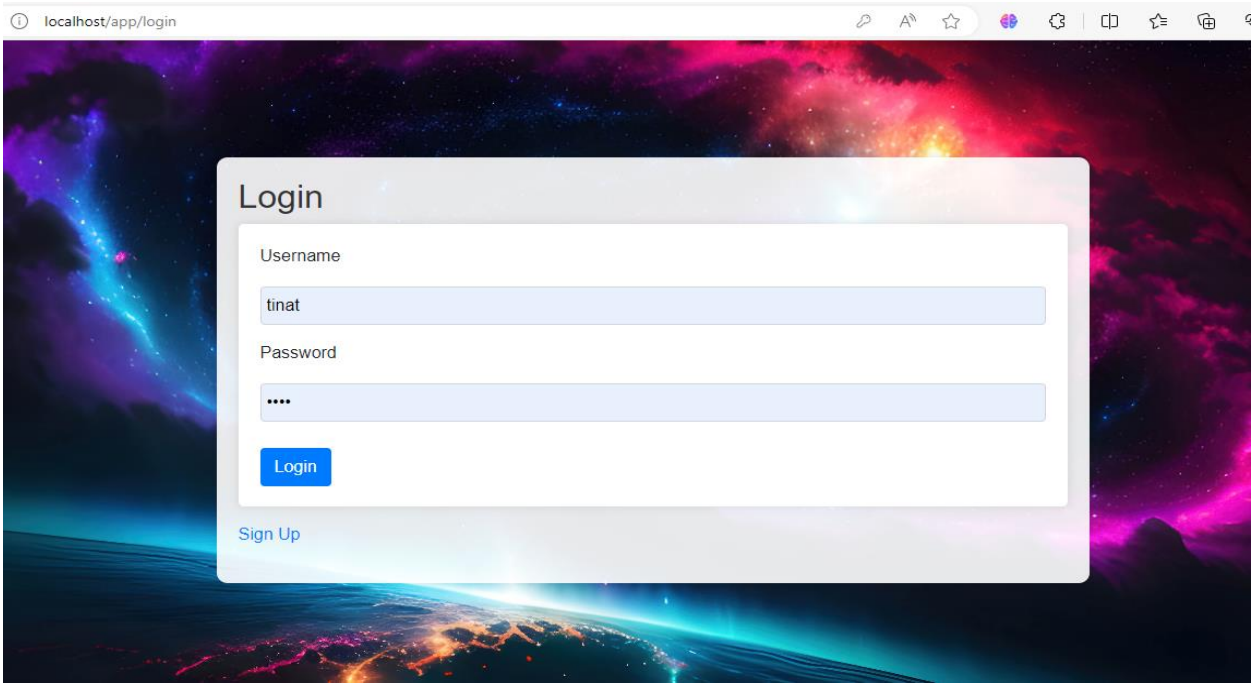
https://hub.docker.com/repository/docker/tina1381/nginx-web

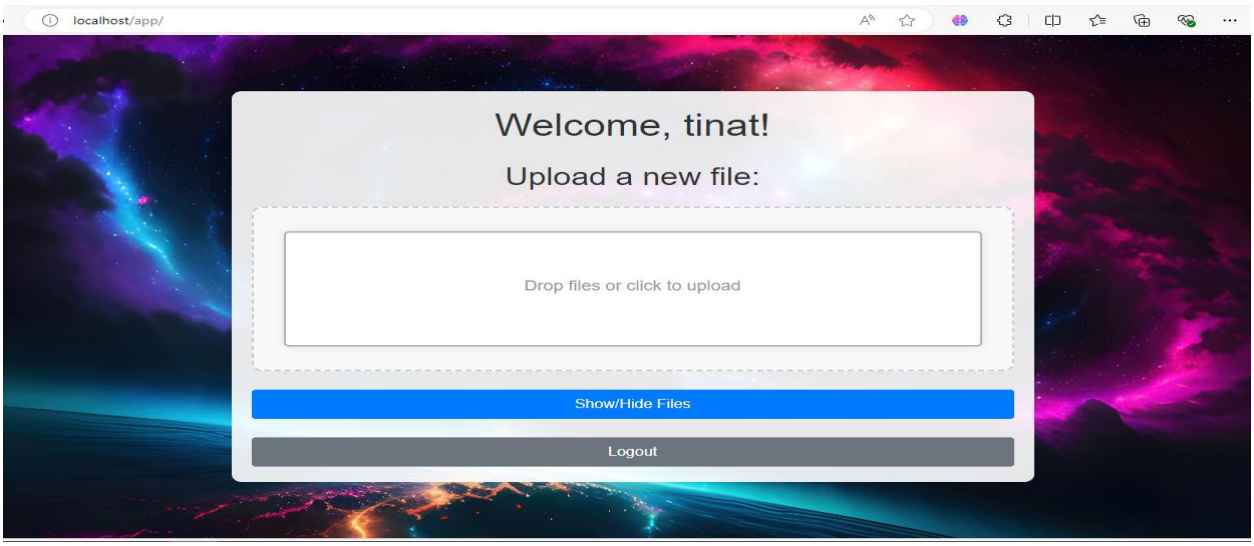
https://hub.docker.com/repository/docker/tina1381/flask-app

https://hub.docker.com/repository/docker/tina1381/mysql

خروجی:


Group Information
تینا توکلی
شماره دانشجویی: 9922762220
هادی امینی
شماره دانشجویی: 9912762370
پارسا هدایت نیا
شماره دانشجویی: 9822762039


localhost/app/login
Login
Username
tinat
Password
Login
Sign Up


localhost/app/
Welcome, tinat!
Upload a new file:
Drop files or click to upload
Show/Hide Files
Logout