

# Dockerize Applications

TimeSeries Team

Ngày 30 tháng 9 năm 2025

## Mục lục

<b>I. Docker là gì?</b>	<b>2</b>
1 Khái niệm	2
2 Tại sao bạn nên sử dụng Dockerizing?	2
<b>II. Giới thiệu</b>	<b>4</b>
1 Kiến trúc Docker	4
2 Những khái niệm cơ bản về Docker	5
2.1 Image là gì?	5
2.2 Container là gì?	5
2.3 Registry là gì?	5
2.4 Docker Compose là gì?	5
<b>III. Docker thực chiến</b>	<b>6</b>
1 Cài đặt Docker	6
2 Tạo Project đơn giản đầu tiên	6
2.1 Chuẩn bị môi trường	7
2.2 Tạo cấu trúc dự án	7
2.3 Build & Run bằng Docker Compose	9

# I. Docker là gì?

## 1 Khái niệm

Hãy bắt đầu bằng cách giải thích ngắn gọn về Docker. Đây là một công cụ mã nguồn mở, đóng gói ứng dụng của bạn với tất cả các chức năng cần thiết thành một gói duy nhất. Bạn có thể sử dụng Docker để đóng gói ứng dụng với mọi thứ cần thiết để chạy ứng dụng (chẳng hạn như thư viện) và đóng gói nó thành một gói duy nhất — một container. Container được tạo từ các hình ảnh chỉ định chính xác nội dung của chúng. Trong bài viết này, chúng ta sẽ tìm hiểu sâu hơn về Docker và cách để Dockerize dự án của mình.



Hình 1: Dockerize

**Docker** là một nền tảng mở để phát triển, phân phối và chạy ứng dụng. Docker cho phép bạn tách biệt ứng dụng của mình khỏi hạ tầng, nhờ đó bạn có thể triển khai phần mềm nhanh chóng hơn. Với Docker, bạn có thể quản lý hạ tầng của mình theo cùng một cách mà bạn quản lý ứng dụng.

Docker cung cấp khả năng đóng gói và chạy ứng dụng trong một môi trường gọi là container. Tính cô lập và bảo mật cho phép bạn chạy nhiều container cùng lúc trên một máy chủ nhất định. Container rất nhẹ và chứa mọi thứ cần thiết để chạy ứng dụng, vì vậy bạn không cần phải phụ thuộc vào những gì được cài đặt trên máy chủ. Bạn có thể chia sẻ container trong khi làm việc và đảm bảo mọi người cùng chia sẻ đều có cùng một container hoạt động theo cùng một cách.

## 2 Tại sao bạn nên sử dụng Dockerizing?

Việc Docker hóa các ứng dụng hoặc dịch vụ của bạn mang lại nhiều lợi ích:

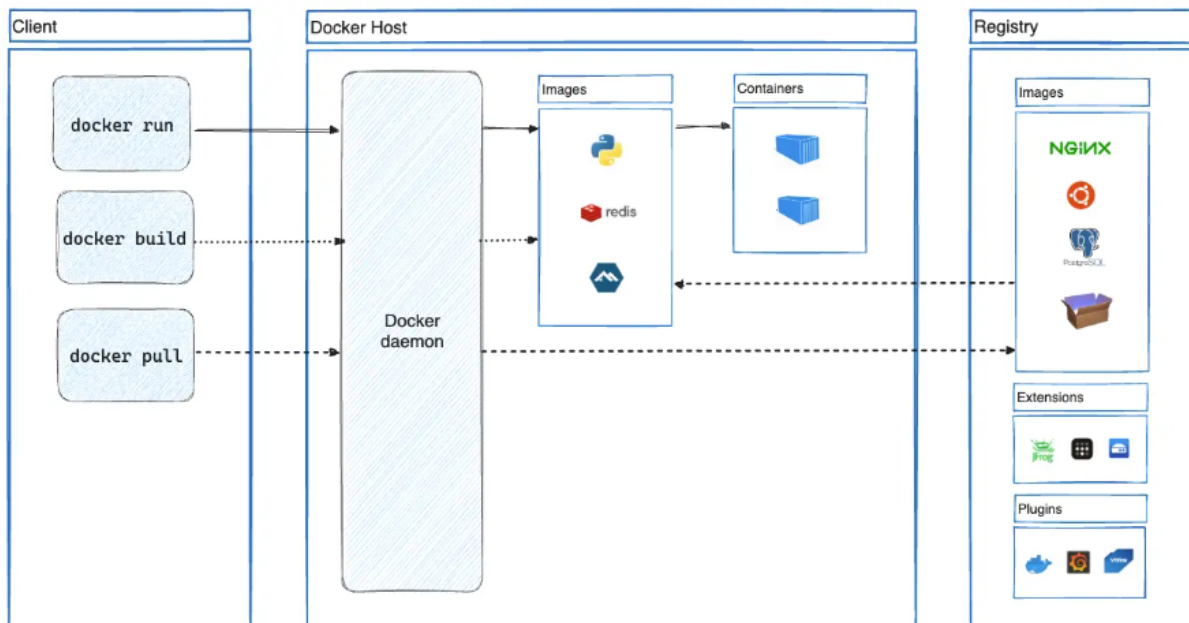
- **Tính nhất quán** : Docker container đóng gói các ứng dụng và các phần phụ thuộc của chúng, đảm bảo chúng chạy nhất quán trên các môi trường khác nhau. Điều này loại bỏ vấn đề "nó hoạt động trên máy của tôi nhưng không hoạt động trên máy của bạn" và duy trì sự nhất quán giữa các môi trường phát triển, thử nghiệm và sản xuất.
- **Tính di động** : Container Docker có tính di động cao và có thể chạy trên bất kỳ hệ thống nào hỗ trợ Docker. Tính di động này giúp đơn giản hóa quá trình di chuyển ứng dụng giữa các môi

trường khác nhau, cho dù bạn đang phát triển cục bộ, triển khai lên máy chủ đám mây hay di chuyển sang cơ sở hạ tầng khác.

- **Hiệu quả :** Container Docker nhẹ và chia sẻ nhân của hệ điều hành máy chủ, giúp tiết kiệm tài nguyên hơn so với ảo hóa truyền thống. Bạn có thể chạy nhiều container hơn trên cùng một phần cứng, cải thiện việc sử dụng tài nguyên.
- **Cô lập :** Container cung cấp khả năng cô lập cho các ứng dụng của bạn, nghĩa là chúng không gây ảnh hưởng lẫn nhau hoặc ảnh hưởng đến hệ thống máy chủ. Tính cô lập này tăng cường bảo mật và ổn định, giúp việc chạy nhiều ứng dụng trên một máy chủ an toàn hơn.
- **Khả năng mở rộng:** Docker giúp việc mở rộng ứng dụng theo chiều ngang dễ dàng hơn bằng cách thêm hoặc bớt các phiên bản container. Điều này đặc biệt hữu ích đối với các kiến trúc microservice, nơi các thành phần có thể được mở rộng độc lập.
- **Kiểm soát phiên bản :** Docker image có thể được quản lý phiên bản, cho phép bạn khôi phục lại trạng thái trước đó của ứng dụng nếu có sự cố phát sinh trong quá trình triển khai. Điều này giúp việc quản lý và bảo trì các phiên bản phần mềm khác nhau trở nên dễ dàng hơn.
- **Quản lý cấu hình đơn giản:** Docker image có thể bao gồm các biến môi trường và tệp cấu hình, giúp quản lý cài đặt ứng dụng dễ dàng hơn theo cách nhất quán và có thể tái tạo.
- **Tối ưu hóa tài nguyên:** Docker cho phép bạn tối ưu hóa việc phân bổ tài nguyên bằng cách tạo các container riêng biệt cho các thành phần khác nhau của ứng dụng. Điều này có thể giúp sử dụng tài nguyên tốt hơn và quản lý các ứng dụng phức tạp dễ dàng hơn.
- **DevOps và CI/CD** Docker là công nghệ nền tảng trong các hoạt động DevOps hiện đại. Nó đơn giản hóa quy trình xây dựng, thử nghiệm và triển khai ứng dụng, cho phép chu kỳ phát hành nhanh hơn. Container cũng có thể được điều phối để triển khai và mở rộng tự động trong các quy trình CI/CD.

## II. Giới thiệu

### 1 Kiến trúc Docker



Docker sử dụng kiến trúc máy khách-máy chủ. Máy khách Docker giao tiếp với daemon Docker, thực hiện các công việc nặng nhọc như xây dựng, chạy và phân phối các container Docker của bạn. Máy khách Docker và daemon có thể chạy trên cùng một hệ thống, hoặc bạn có thể kết nối máy khách Docker với một daemon Docker từ xa. Máy khách Docker và daemon giao tiếp bằng API REST, qua socket UNIX hoặc giao diện mạng. Một máy khách Docker khác là Docker Compose, cho phép bạn làm việc với các ứng dụng bao gồm một tập hợp các container.

- **Daemon Docker:** nhận yêu cầu API Docker và quản lý các đối tượng Docker như images, containers, networks, and volumes. Một trình nền cũng có thể giao tiếp với các trình nền khác để quản lý các dịch vụ Docker.
- **Docker Client:** là cách chính mà nhiều người dùng Docker tương tác với Docker. Khi bạn sử dụng các lệnh như docker run, máy khách sẽ gửi các lệnh này đến dockerd, để thực thi chúng. dockerLệnh này sử dụng API Docker. Máy khách Docker có thể giao tiếp với nhiều daemon.
- **Docker Desktop:** Docker Desktop là một ứng dụng dễ cài đặt cho môi trường Mac, Windows hoặc Linux, cho phép bạn xây dựng và chia sẻ các ứng dụng và dịch vụ vi mô được đóng gói trong container. Docker Desktop bao gồm Docker daemon ( dockerd), Docker client ( docker), Docker Compose, Docker Content Trust, Kubernetes và Credential Helper.
- **Docker registries:** Docker registry lưu trữ các hình ảnh Docker. Docker Hub là một registry công khai mà bất kỳ ai cũng có thể sử dụng, và Docker sẽ tìm kiếm các hình ảnh trên Docker Hub theo mặc định. Khi bạn sử dụng lệnh docker pullhoặc docker run, Docker sẽ kéo các hình ảnh cần thiết từ sổ đăng ký đã cấu hình của bạn. Khi bạn sử dụng docker pushlệnh, Docker sẽ đẩy hình ảnh của bạn vào sổ đăng ký đã cấu hình.

- **Docker objects:** Khi sử dụng Docker, bạn đang tạo và sử dụng hình ảnh, vùng chứa, mạng, ổ đĩa, plugin và các đối tượng khác. Phần tiếp theo sẽ cung cấp cụ thể hơn về một số đối tượng đó.

## 2 Những khái niệm cơ bản về Docker

### 2.1 Image là gì?

Image là một mẫu chỉ đọc kèm theo hướng dẫn tạo vùng chứa Docker. Thông thường, một image được xây dựng dựa trên một image khác, với một số tùy chỉnh bổ sung. Ví dụ: bạn có thể xây dựng một image dựa trên ubuntu image đó, nhưng cài đặt máy chủ web Apache và ứng dụng của bạn, cũng như các chi tiết cấu hình cần thiết để ứng dụng chạy.

Bạn có thể tự tạo image của riêng mình hoặc chỉ sử dụng image do người khác tạo và xuất bản trong registry. Để xây dựng image của riêng mình, bạn tạo một Dockerfile với cú pháp đơn giản để xác định các bước cần thiết để tạo và chạy image. Mỗi lệnh trong Dockerfile sẽ tạo ra một layer trong image. Khi bạn thay đổi Dockerfile và xây dựng lại image, chỉ những layer đã thay đổi mới được xây dựng lại. Đây là một phần lý do khiến image trở nên nhẹ, nhỏ gọn và nhanh hơn so với các công nghệ ảo hóa khác.

### 2.2 Container là gì?

Container là một phiên bản có thể chạy được của một image. Bạn có thể tạo, khởi động, dừng, di chuyển hoặc xóa một container bằng Docker API hoặc CLI. Bạn có thể kết nối một container với một hoặc nhiều mạng, gắn bộ lưu trữ vào container, hoặc thậm chí tạo một image mới dựa trên trạng thái hiện tại của nó.

Theo mặc định, một container được cô lập tương đối tốt với các container khác và máy chủ của nó. Bạn có thể kiểm soát mức độ cô lập của mạng, bộ lưu trữ hoặc các hệ thống con cơ bản khác của container với các container khác hoặc với máy chủ.

Một container được định nghĩa bởi image của nó cũng như bất kỳ tùy chọn cấu hình nào bạn cung cấp khi tạo hoặc khởi động nó. Khi một container bị xóa, mọi thay đổi về trạng thái của nó mà không được lưu trữ trong bộ nhớ liên tục sẽ biến mất.

### 2.3 Registry là gì?

Registry là một hệ thống lưu trữ nơi bạn có thể đẩy (push) các hình ảnh container lên và kéo (pull) các hình ảnh đó xuống để sử dụng. Cụ thể, registry chứa các repository – nơi lưu trữ các hình ảnh container. Điều này giúp các thành viên trong nhóm có thể truy cập và sử dụng các hình ảnh mới nhất cho dự án của họ một cách thuận tiện.

Các lựa chọn phổ biến để tạo repository trong registry:

- Docker Hub (mặc định và phổ biến nhất)
- Amazon Elastic Container Registry (ECR)
- GitHub Container Registry
- Azure Container Registry
- Registry nội bộ (tự xây dựng cho tổ chức)

### 2.4 Docker Compose là gì?

Docker Compose là một công cụ giúp chúng ta dễ dàng định nghĩa và vận hành các ứng dụng đa container trên Docker. Thay vì phải chạy nhiều lệnh docker run riêng biệt cho từng dịch vụ trong ứng dụng phức tạp (ví dụ như một web server và một cơ sở dữ liệu), Docker Compose cho phép bạn quản lý mọi thứ chỉ trong một file cấu hình duy nhất.

Vậy thực hiện bằng cách nào?

Đầu tiên, Định nghĩa toàn bộ cấu hình đa container trong một file `compose.yml` sử dụng cú pháp YAML. Trong file này, bạn xác định từng “service” (dịch vụ) với các thông tin như: image nào sẽ dùng hoặc build, cấu hình cổng, volume, biến môi trường, và các phụ thuộc giữa các dịch vụ. Sau đó, Chỉ cần một lệnh duy nhất `docker compose up -d --build` là Docker Compose sẽ tự động dựng và chạy toàn bộ ứng dụng đa container. Khi muốn dừng và xóa toàn bộ ứng dụng, chỉ cần chạy lệnh `docker compose down`.

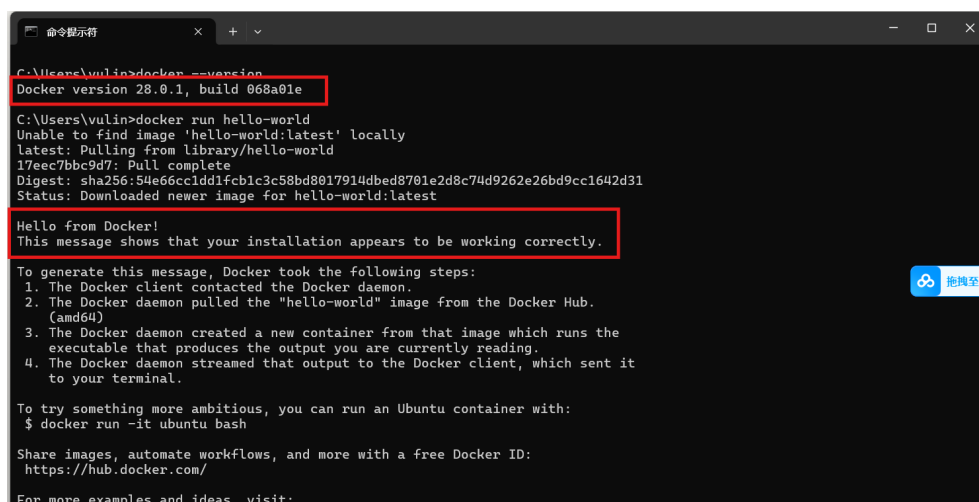
## III. Docker thực chiến

### 1 Cài đặt Docker

- Cài đặt Docker Desktop (Windows/Mac) hoặc Docker Engine (Linux)[tải tại link](#)
- Sau khi tải về và cài đặt xong, chạy lệnh sau ở bash

```
1 # check version
2 docker --version
3 # check docker run
4 docker run hello-world
5
```

- Nếu kết quả hiển thị như hình thì chúc mừng, bạn đã cài đặt thành công!



```
C:\Users\vulin>docker --version
Docker version 28.0.1, build 068a01e

C:\Users\vulin>docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
17eec7bbc9d7: Pull complete
Digest: sha256:54e66c1dd1fcb1c3c58bd8017914dbed8701e2d8c74d9262e26bd9cc1642d31
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
```

Hình 2: Kiểm tra cài đặt ở CMD

- Xem các image đã tải bằng lệnh: `docker images`

### 2 Tạo Project đơn giản đầu tiên

Trong phần này mình sẽ cùng bạn làm một project nhỏ để hiểu cách dockerize mô hình machine learning. Chúng ta sẽ:

1. Train một mô hình ML đơn giản (Logistic Regression trên dataset Iris).
2. Đóng gói mô hình vào một API (Flask).

3. Viết Dockerfile để chạy API trong container.
4. Dùng docker-compose để khởi động hệ thống.
5. Test API ngay tại local.

## 2.1 Chuẩn bị môi trường

Kiểm tra xem Docker đã ok chưa

```
1 docker --version
2 docker compose version
```

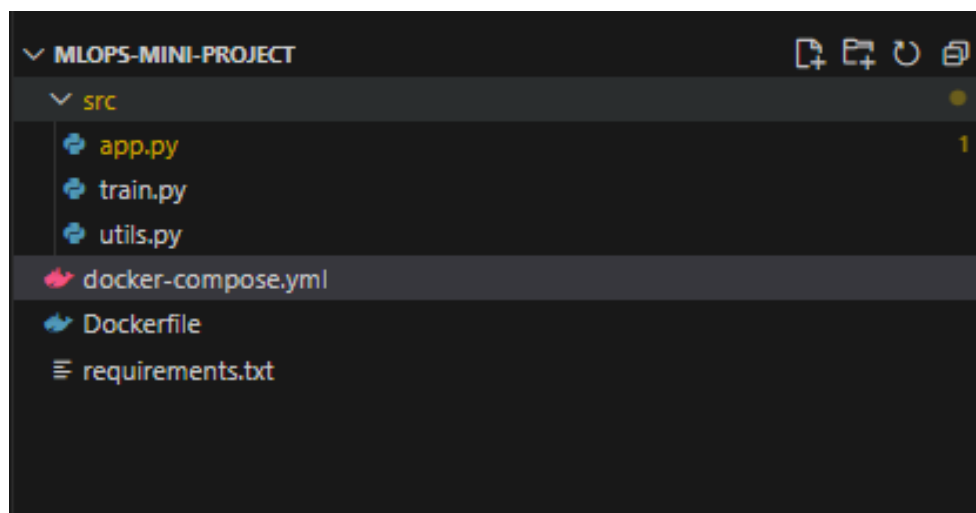
Nếu hiển thị version thì là ok

```
C:\Users\vulin>docker --version
Docker version 28.0.1, build 068a01e

C:\Users\vulin>docker compose version
Docker Compose version v2.33.1-desktop.1
```

## 2.2 Tạo cấu trúc dự án

Tạo folder mlops-mini-project với cấu trúc như hình:



Trong thư mục có tạo các file gồm

- train.py

```
1 import pickle
2 from sklearn.datasets import load_iris
3 from sklearn.linear_model import LogisticRegression
4
5 # Load dataset Iris
6 X, y = load_iris(return_X_y=True)
7
8 # Train Logistic Regression
9 clf = LogisticRegression(max_iter=200)
```

```
10 clf.fit(X, y)
11
12 # Save model
13 with open("model.pkl", "wb") as f:
14     pickle.dump(clf, f)
15
16 print("    Model trained and saved as model.pkl")
```

- app.py

```
1 from flask import Flask, request, jsonify, render_template_string
2 import pickle
3 import numpy as np
4
5 app = Flask(__name__)
6
7 # Load model
8 with open("model.pkl", "rb") as f:
9     model = pickle.load(f)
10
11 # Trang home c    form HTML
12 @app.route("/", methods=["GET", "POST"])
13 def home():
14     html = """
15     <h2> Iris Classifier Demo</h2>
16     <form method="POST" action="/">
17         <label>Sepal Length:</label>
18         <input type="text" name="f1"><br><br>
19         <label>Sepal Width:</label>
20         <input type="text" name="f2"><br><br>
21         <label>Petal Length:</label>
22         <input type="text" name="f3"><br><br>
23         <label>Petal Width:</label>
24         <input type="text" name="f4"><br><br>
25         <button type="submit">Predict</button>
26     </form>
27     {% if prediction is not none %}
28         <h3> Prediction: {{ prediction }}</h3>
29     {% endif %}
30     """
31     if request.method == "POST":
32         try:
33             f1, f2, f3, f4 = float(request.form["f1"]), float(request.form["f2"])
34             , float(request.form["f3"]), float(request.form["f4"])
35             X = np.array([f1, f2, f3, f4]).reshape(1, -1)
36             pred = model.predict(X)[0]
37             return render_template_string(html, prediction=int(pred))
38         except:
39             return render_template_string(html, prediction=" Invalid input!")
40     return render_template_string(html, prediction=None)
41
42 @app.route("/predict", methods=["POST"])
43 def predict():
44     data = request.json["features"]
45     X = np.array(data).reshape(1, -1)
46     pred = model.predict(X)[0]
47     return jsonify({"prediction": int(pred)})
48
49 if __name__ == "__main__":
50     app.run(host="0.0.0.0", port=5000)
```



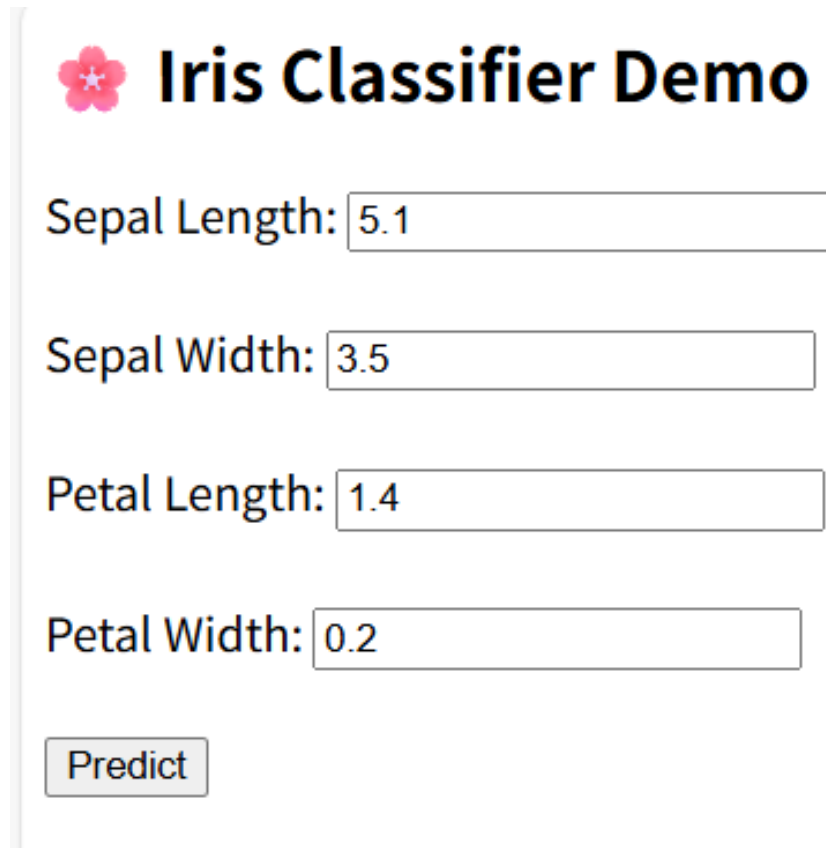
## 2.3 Build & Run bằng Docker Compose


```
1 docker compose build
2 docker compose up
```

Nếu thành công, bạn sẽ thấy dòng logs:

```
1 * Running on http://0.0.0.0:5000
```

Và khi truy cập vào link bạn sẽ thấy giao diện:



 **Iris Classifier Demo**

Sepal Length:

Sepal Width:

Petal Length:

Petal Width:

Như vậy là bạn đã thành công dockerize project của bạn rồi đó!